Oracle Financial Services
Behavior Detection Platform
Administration Guide

Release 6.1.3
December 2013

ORACLE
FINANCIAL SERVICES

Oracle Financial Services Behavior Detection Platform **Administration Guide**

Release 6.1.3
December 2013

Document Control Number: 9MN12-0002
Document Number: AG-12-OFS-0002-6.1.3-01

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

Document Number: AG-12-OFS-0002-6.1.3-01
Second Edition (December 2013)

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

Contents

List of Figures

List of Tables

About this Guide

This guide explains the concept behind the Oracle Financial Services Behavior Detection Platform, and provides comprehensive instructions for proper system administration, as well as daily operations and maintenance. 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 Behavior Detection Platform 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 Behavior Detection Platform, include the following:

- **Oracle Financial Services Installer:** Installs and configures the Oracle Financial Services application at a specific deployment site. The Oracle Financial Services Installer also installs and upgrades any additional Oracle Financial Services solution sets, and requires access to deployment-specific configuration information (for example, machine names and port numbers).
- **System Administrator:** Configures, maintains, and adjusts the system, and is usually an employee of a specific Oracle Financial Services customer. The System Administrator maintains user accounts and roles, monitors data ingestion and alert management, archives data, loads data feeds, and performs post-processing tasks. In addition, the System Administrator can reload cache. However, the scenario description is not visible to the System Administrator.

Scope of this Guide

This guide describes the physical and logical architecture of the Oracle Financial Services application. It also provides instructions for installing and configuring the application, its subsystem components, and required third-party software for operation.

The Oracle Financial Services application provides the foundation for all its products. Advanced data mining algorithms and sophisticated pattern recognition technologies power the application. It provides an open and scalable infrastructure that supports rich, end-to-end functionality across all Oracle Financial Services solution sets. The application's' extensible, modular architecture enables a customer to deploy new solution sets readily as the need arises.

How this Guide is Organized

The *Oracle Financial Services Behavior Detection Platform Administration Guide*, includes the following chapters:

- Chapter *1, The Oracle Financial Services Behavior Detection Platform,* provides a brief overview of the application and its components.
- Chapter *2, Oracle Financial Services Jobs,* provides an overview of the Oracle Financial Services Job Protocol and procedures for performing various tasks that relate to starting, stopping, and recovering jobs.
- Chapter *3, Security Configuration,* covers the required day-to-day operations and maintenance of users, groups, and organizational units.
- Chapter *4, Data Ingestion,* describes the operation and process flow of Data Ingestion subsystem components.
- Chapter *5, Informatica Workflows,* describes the derivation and aggregation of data through workflows in Informatica, after the data ingestion process completes.
- Chapter *6, Post-Processing Tasks,* explains how to customize features that affect presentation of user information on the desktop.
- Chapter *7, Batch Processing Utilities,* provides information about the database utilities related to the batch process.
- Chapter *8, Administrative Utilities,* provides information about the database utilities that are independent of the batch process.
- Appendix A, *Logging,* describes the logging feature.
- Appendix B, *Oracle Financial Services Software Updates,* describes the application of software updates (hotfixes) and their impact on customization.
- Appendix C, *Report Management,*, describes the configuration and integration details of the Report Framework within user interface.
- Appendix D, *Informatica Workflow Details,* lists how the Informatica Workflows are used within the application.
- Appendix E, *Moving Oracle Financial Services System Environment,* explains the steps to move the system from one environment to another.
- The *Index* provides an alphabetized cross-reference list that helps you locate information quickly.

Where to Find More Information

For more information about Oracle Financial Services Behavior Detection Platform, refer to the following documents:

- *Oracle Financial Services Behavior Detection Platform Scenario Manager User Guide*
- *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*
- *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide,* Vol.1, *Business Data*
- *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide,* Vol.2, *Oracle Financial Services Data*
- *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide,* Vol.3, *Case Management Data*
- *Oracle Financial Services Behavior Detection Platform Oracle Financial Services Service Guide*
- *Oracle Financial Services Behavior Detection Platform Configuration Guide*
- *Oracle Financial Services Analytical Applications Infrastructure DeFI User Manual.doc*
- *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3*
- *Oracle Financial Services Behavior Detection Platform Stage 1 Installation Guide*
- *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration Release 7.3*
- *Oracle Financial Services Enterprise Case Management Installation Guide - Stage 3*

For installation and configuration information about Altio software, refer to the following documents:

- AltioLive Deployment Guide
- AltioLive Developers Guide

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 1 lists the conventions used in this guide.

Table 1. 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, Oracle, and Informatica. 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 application's architecture:

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

The following sections describe these views.

Component View

Figure 1 illustrates the concept that a series of tiers and subsystems compose the application's architecture.

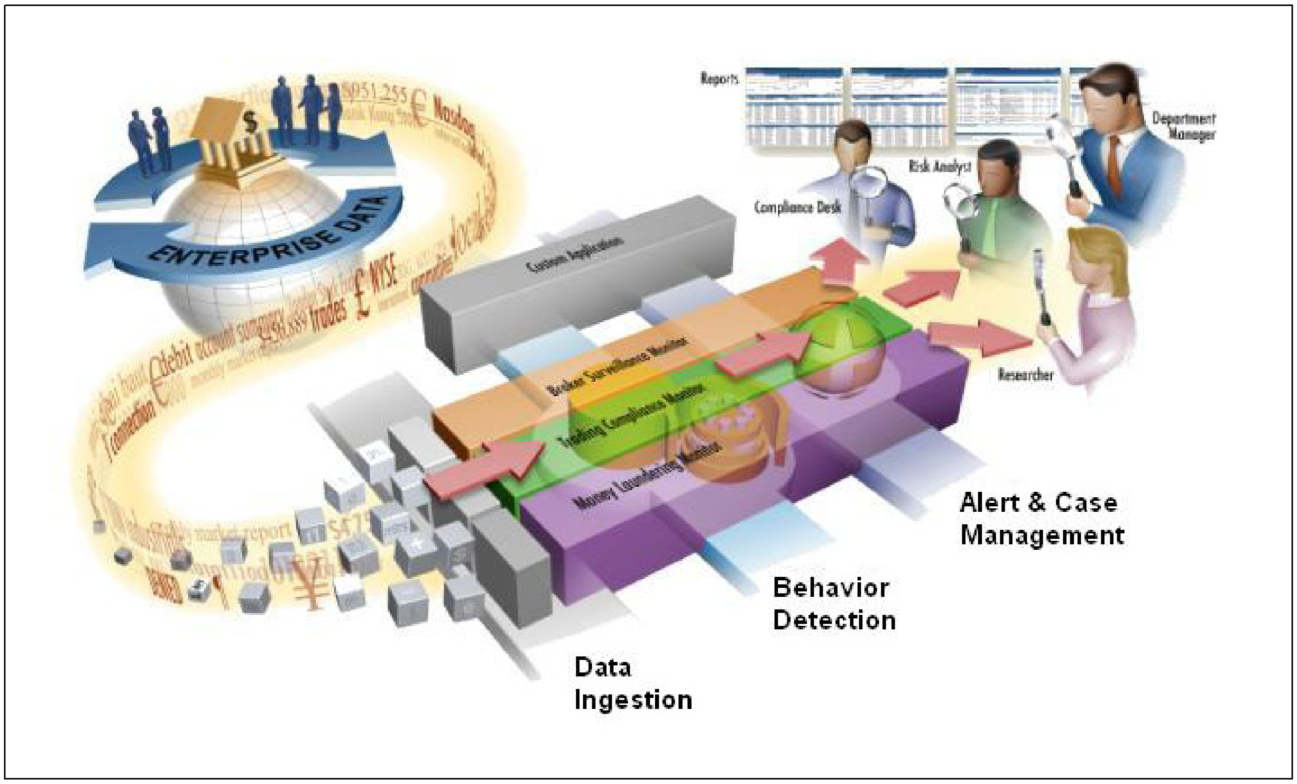


Figure 1. Oracle Financial Services Architecture—Overview

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

Tiers

The 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. The application 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 installation.
- Each **Oracle Financial Services solution set** (Anti-Money Laundering and Fraud Detection) extends the Oracle Financial Services framework. Each adds domain-specific content to provide the required services for addressing a specific business problem. It includes reusable domain artifacts such as scenarios, input data transformation code, and profiling scripts. A solution set also provides the required presentation packages and custom application objects for supporting user-interface functionality specific to the business domain.

Subsystems

The application is composed of the following four subsystems:

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

A set of components further divides each subsystem. Components are units of a subsystem that can be installed separately onto a different server. Table 2 outlines the definition for the subsystems and components. In some cases, however, individual deployments can add subsystems or components to meet a client's custom requirements.

Table 2. Subsystems and their Components

Common Name	Directory Name	Contents
Administration Tools	admin_tools	Web-enabled Administration Tools
Case Management UI	ftpshare\<case info-dom>\erwin\forms	Xmls for rendering the UI
Alert Management UI	ftpshare\< alert info-dom>\erwin\forms	Xmls for rendering the UI
Case Management Web	solution\cm	JSPs used in Case Management
Alert Management Web	solution\am	JSPs used in Alert Management
Behavior Detection	behavior_detection	(Subsystem)
Data Ingestion	ingestion_manager	Java components, Informatica components, scripts, and stored procedures
Database Tools	database/db_tools	For DB tools directory
Detection Algorithms	algorithms	C++ behavior detection algorithms
Scenario Manager	toolkit	Job and scenario editors
Financial Services Data Model	database	Database utilities and database creation scripts
Web Services	services	Web services for watch list scanning and for the alert management supervisor (used when posting alerts to Behavior Detection)

Deployment View

Oracle Financial Services architecture from the perspective of its deployment is illustrated in Figure 2. This deployment view illustrates deployment of the major components of each subsystem across servers. Additionally, the deployment view shows the primary communications links and protocols between the processing nodes.

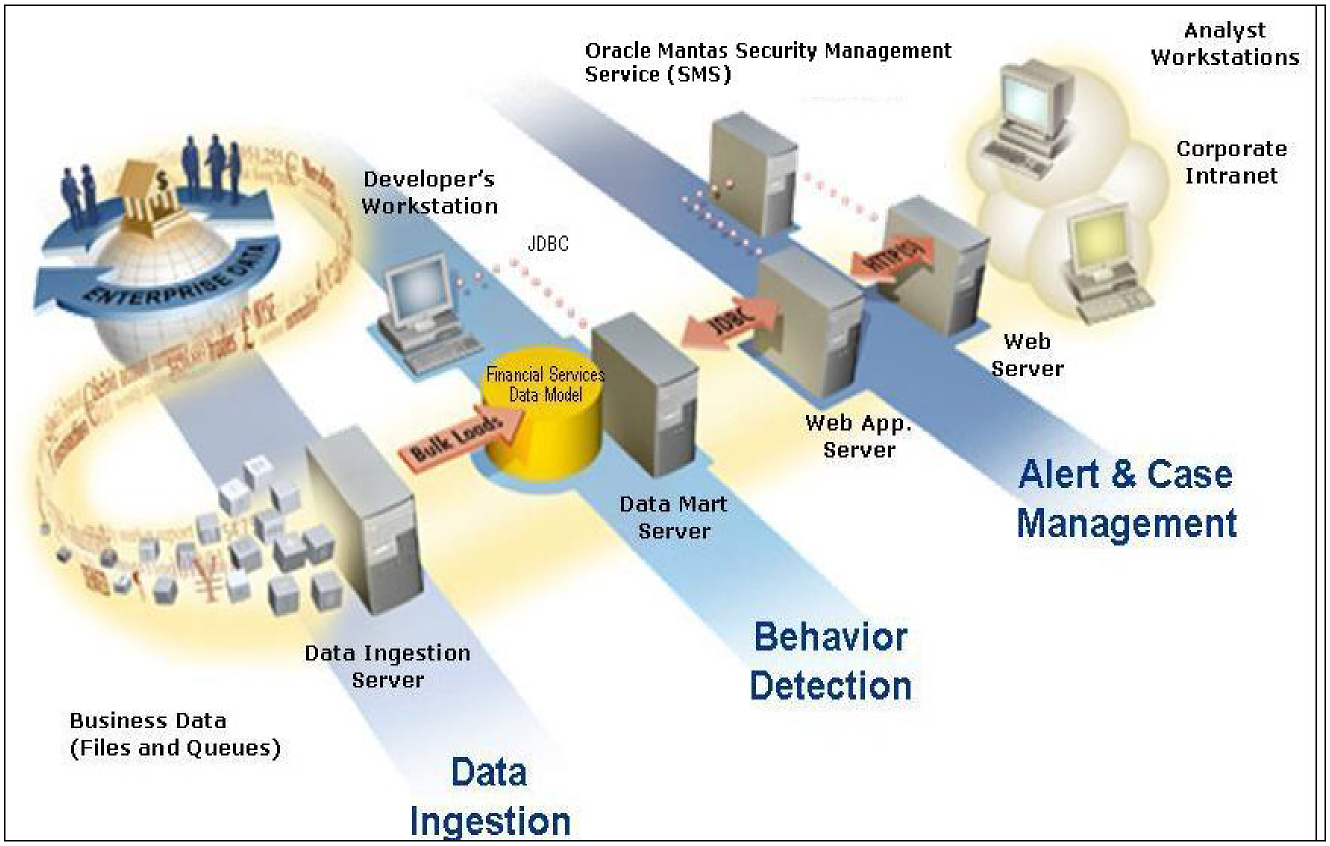


Figure 2. Oracle Financial Services Architecture—Deployment View

The complex interactions between the components of the Alert & Case Management subsystem become apparent in the deployment view. The Alert & Case Management subsystem requires the following:

- Web browser
- Web server
- Web Application server
- Security Management Service (SMS)

Oracle Financial Services Alert Management and Case Management use inbuilt SMS (Security Management Service) for handling both authentication and authorization. The Alert & Case Management subsystem also supports the use of an External Authentication Management (EAM) tool to perform user authentication at the Web server, if a customer requires it.

These components can operate when deployed on a single computer or when distributed across multiple computers (Figure 2). In addition to being horizontally scalable, the application is vertically scalable in that replication of each of the components can occur across multiple servers.

Security View

The security view of the architecture and use of security features of the network in a Behavior Detection architecture deployment is illustrated in Figure 3. 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.

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

.

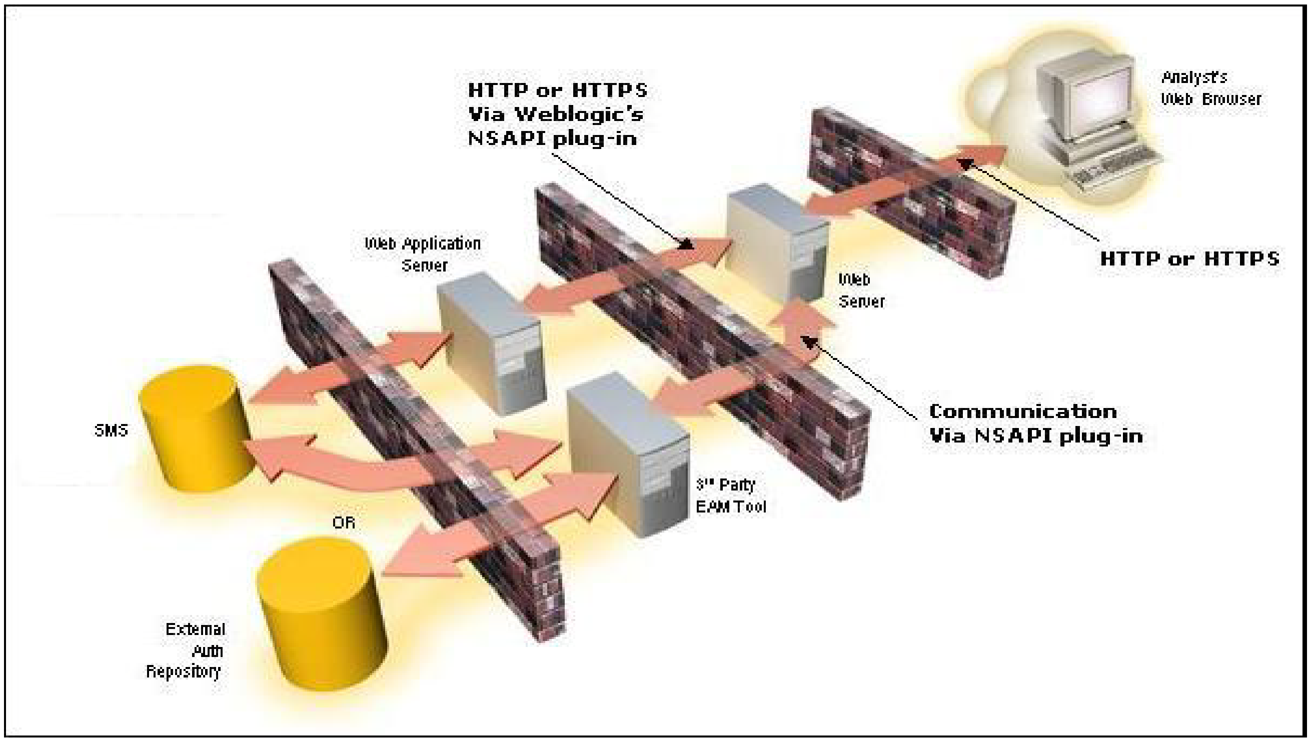


Figure 3. Oracle Financial Services Architecture—Security View

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

About Oracle Financial Services Operations

As the Oracle Financial Services administrator, you can coordinate the overall operations of the application Data Ingestion, Behavior Detection, and Post-Processing (Figure 4).

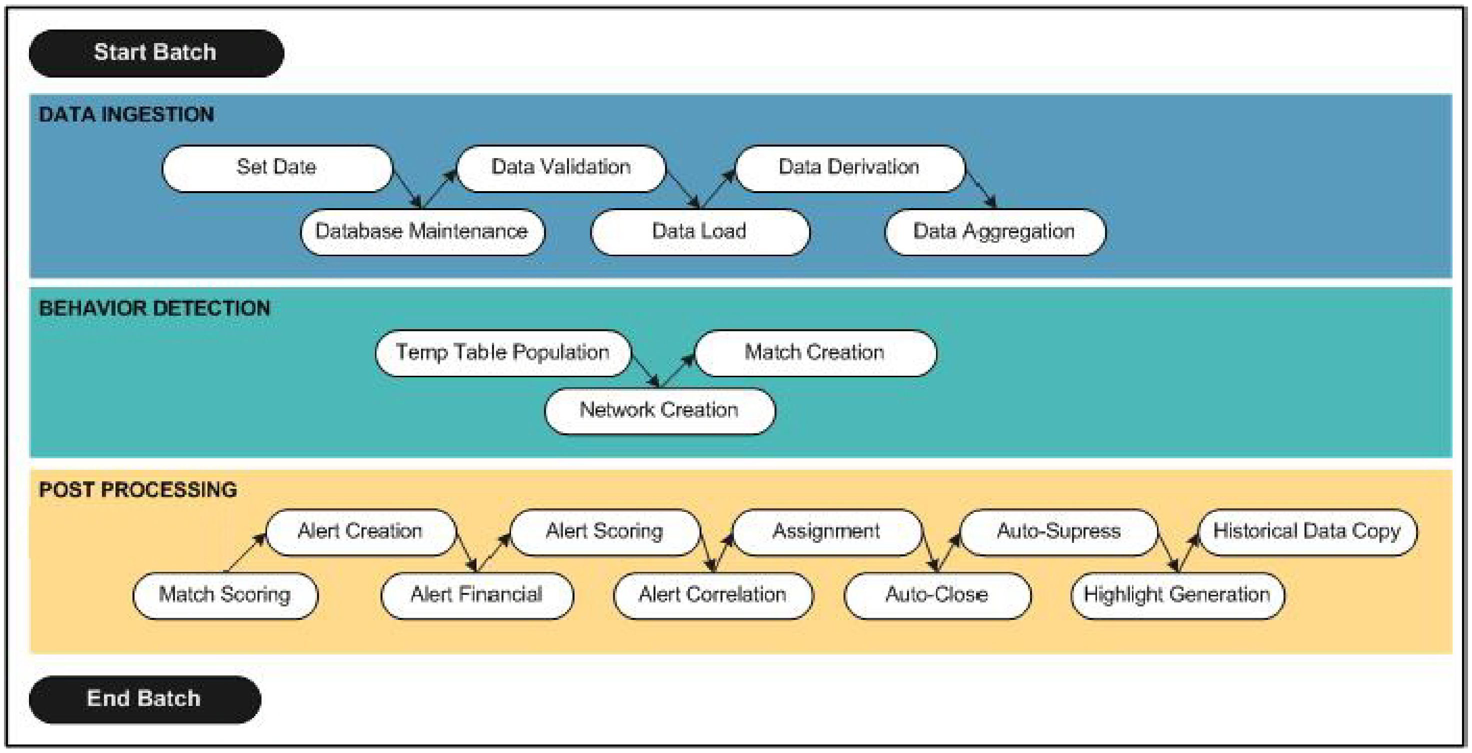


Figure 4. Oracle Financial Services Architecture—Oracle Financial Services Processing

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

As Figure 4 illustrates, each cycle of the 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 7, *Batch Control Utility,* on page 223, for more information).

Data Ingestion

The Oracle Financial Services Ingestion Manager controls the data ingestion process. The *Oracle Financial Services Behavior Detection Platform Data Interface Specification (DIS)* 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, applies required derivations and aggregations, and populates the database with the results (Refer to Chapter 4, *Data Ingestion,* on page 51, 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.

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

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

Post-Processing

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

1. A match scoring process computes a ranking for scenario matches indicating a degree of risk associated with the detected event or behavior (Refer to section *Match Scoring*, on page 173, for more information).
2. 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 section *Update Alert Financial Data*, on page 176, for more information).
3. An alert financial data update process records additional data for alerts such as the related Investment Advisor or Security involved in the alert (Refer to section *Update Alert Financial Data,* on page 176, for more information).
4. An alert scoring process ranks the alerts (including each match within the alerts) to indicate the degree of risk associated with the detected event or behavior (Refer to section *Alert Scoring*, on page 176, for more information).
5. An assignment process determines the user or set of users responsible for handling each alert. The process assigns an alert to an owner based on customer-specific assignment instructions (Refer to section *Assignment*, on page 177, for more information).
6. *Optional:* An auto-close algorithm closes alerts that are of a lower priority to the business. The application automatically suppresses alerts according to customer-specific auto-close instructions (Refer to section *Auto-Close*, on page 178, for more information).
7. *Optional:* An auto-suppression algorithm suppresses alerts that share specific scenario and focal entity attributes for a particular time frame. The application automatically suppresses alerts according to customer-specific auto-suppression instructions (Refer to section *Automatic Alert Suppression*, on page 183, for more information).
8. The system generates highlights for alerts that appear in the alert list in the Alert & Case Management subsystem and stores them to the database (Refer to section *Highlight Generation*, on page 184, for more information).
9. A historical data copy process copies alert-related data that the Ingestion Manager provides, from the source data tables to the historical data tables (Refer to section *Historical Data Copy*, on page 185, for more information).
10. An alert correlation process correlates alerts to business entities and optionally to each other based on configurable rule sets. (Refer to section *Assignment*, on page 177, for more information).

End Batch

The system ends batch processing when processing of data from the Oracle client is complete (Refer to section *Ending a Batch Process*, on page 229, for more information). The Alert & Case Management subsystem then controls alert and case management processes. Refer to the *Oracle Financial Services Alert Management User Guide,* Release 6.1.3, 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 the Oracle Financial Services administrator, you invoke the `dispatcher` and `mantas` processes by running the shell scripts in Table 3.

Table 3. 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 `dis-patcher` 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.

Refer to Chapter 2 of the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide*, Volume 2, Release 6.1.1, for more information.

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

Understanding the Oracle Financial Services Job Protocol

Oracle Financial Services Behavior Detection Job templates are maintained through the Scenario Manager. 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 4 provides an example of a job template group with two job templates.

Table 4. `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 16, and *Stopping the Dispatcher*, on page 17, 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 5 describes environment variables in the `system.env` file.

Table 5. 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 6 describes database environment variables in the `system.env` file.

Table 6. 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 7 shows operating system variables in the `system.env` file.

Table 7. 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 the steps:

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

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

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

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

   Optional parameters include the following:

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

CAUTION: For 32-bit Linux configurations, Oracle 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 Oracle Financial Services 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 12, 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.

Note: Before you run Network jobs, change the batch size of Network related counter to 10,000. By default, the counters is set to 100. The network counters in `KDD_COUNTER` are `NTWRK_ID_SEQ`, `NODE_ID_SEQ`, `LINK_ID_SEQ`, `LINK_SUMMARY_ID_SEQ`, and `LINK_TYPE_SUMMARY_ID_SEQ`. Refer to the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide,* Vol.2, *Oracle Financial Services Data*, for more details.

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 Oracle Financial Services 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 19, 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 17, 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 17, 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 user authentication and authorization. It also contains instructions for setting up user accounts in the database to access the Scenario Manager. This chapter focuses on the following topics:

- About the Oracle Financial Services User Authentication
- Setting up a User
- About Configuring Access Control Metadata
- Mapping Users To Access Control Metadata
- About Scenario Manager Login Accounts
- About Changing Passwords for System Accounts
- About Configuring File Type Extensions

About the Oracle Financial Services User Authentication

The primary way to access information is through a Web browser that accesses the Alert Management, Case Management, Account Approval/Pre-Trade Approval, and Administration Tools. The application offers SMS for authentication of web browser clients.

The Scenario Manager authenticates use of the Behavior Detection database only.

Behavior Detection offers two authentication mechanisms 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 to section *Understanding SMS*, on page 26, for more information).
- Web server authentication for Oracle clients who want to utilize their own External Authentication Management (EAM) tool.

Understanding SMS

Oracle Financial Servicesapplication 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 Behavior Detection based on the following:

- Authentication of a unique user ID and password that enables access to Alert Management, Case Management, Account Approval/Pre-Trade Approval, and Administration Tool.

For accessing Alert Management:

- Set of policies that associate functional role with access to specific system functions.
- One or more associated organizational affiliations that control the user's access to alerts.
- Relationship to one or more scenario groups.
- Access to one or more jurisdictions.
- Access to one or more business domains.

For accessing Case Management:

- Set of policies that associate functional roles with access to specific system functions.
- Access to one or more case types/subtypes.
- One or more associated organizational affiliations that control the user's access to cases.
- Access to one or more jurisdictions.
- Access to one or more business domains.

For accessing Account Approval/Pre-Trade Approval:

- Set of policies that associate functional roles with access to specific system functions.
- One or more associated organizational affiliations that control the user's access to Account Approval and Pre-Trade Approval requests.
- Access to one or more jurisdictions.
- Access to one or more business domains.

For accessing Administration Tools:

- Set of policies that associate admin functional role with access to specific system functions.

Setting up a User

To set up a user and provide the user access to the application, 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 to section *User Group and User Roles*, on page 27, for more information on how to map User Roles to User Groups.

Note: For the above sections, Refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3* for further information.

User Group and User Roles

User Roles are predefined in the 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 more information about creating a new user group and mapping it to an existing role, refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3*:

Note: All User Groups should be mapped to Alert Management Infodom.

Actions to Role mappings are done through Database tables. Sample action to role mappings are included in the application. Refer to the *Oracle Financial Services Behavior Detection Platform Configuration Guide*, for more information on changing the mapping of roles to actions.

Actions are primarily associated with a User Role, not an individual user. However, the ability to Reassign To All when taking a Reassign action is associated at the individual user level. Reassign To All means that a user is allowed to assign to users and organizations that may not be within their normal viewing privileges.

Table 8 describes the predefined User Roles and corresponding User Groups.

Table 8. Alert Management Roles and User Groups

Role	Group Name	User group Code
AM Analyst I	AM Analyst I User Group	AMANALYST1GRP
AM Analyst II	AM Analyst II User Group	AMANALYST2GRP
AM Analyst III	AM Analyst III User Group	AMANALYST3GRP
AM Supervisor	AM Supervisor User Group	AMSUPVISRGRP
AM Executive	AM Executive User Group	AMEXCUTIVEGRP
AM Internal Auditor	AM Internal Auditor User Group	AMINAUDITRGRP
AM External Auditor	AM External Auditor User Group	AMEXAUDITRGRP
AM Data Miner	AM Data Miner User Group	AMDATAMNRGRP
AM mantas Administrator	mantas Administrator User Group	AMMANADMNGR

Table 9 describes the Case Management Roles and corresponding User Groups.

Table 9. Case Management Roles and User Groups

Role	Group Name	User group Code
Case Analyst1	Case Analyst1 UserGroup	CMANALYST1UG
Case Analyst2	Case Analyst2 UserGroup	CMANALYST2UG
Case Supervisor	Case Supervisor UserGroup	CMSUPERVISORUG
Case Executive	Case Executive UserGroup	CMEXECUTIVEUG
Case Internal Auditor	Case Internal Auditor UserGroup	CMINAUDITORUG
Case External Auditor	Case External Auditor UserGroup	CMEXAUDITORUG
Case Viewer	Case Viewer UserGroup	CMVIEWERUG
Case Initiator	Case Initiator User Group	CMINITIATRUG
Case Administrator	Case Administrator User Group	CMMANADMNUG

Table 10 describes the Account Approval and Pre-Trade Approval Roles and corresponding User Groups.

Table 10. Account Approval/Pre-Trade Approval Roles and User Groups

Role	Group Name	User group Code
Employee	Employee User Group	CREMPLOYEEUG
Control Room Analyst	Control Room Analyst User Group	CRANALYSTUG
Control Room Supervisor	Control Room Supervisor User Group	CRSUPVISRUG
IP Manager	IP Manager User Group	CRIPMANAGERUG
IP Manager Supervisor	IP Manager Supervisor User Group	CRIPMGRSUPVSRUG

While it is possible for a user to have more than one role within an application, care must be taken with the combination of roles. Specifically a user should be assigned only one role from among the following:

- AM Analyst I
- AM Analyst II
- AM Analyst III
- AM Supervisor
- AM Executive
- AM Internal and External Auditors

A user with one of these AM users can otherwise be assigned additional role(s) from among AM Scenario Group, AM Administrator and a Case Management role. Similarly, a user with a Case Management role should be assigned only one role from among the following:

- Case Analyst1
- Case Analyst2
- Case Supervisor
- Case Executive
- Case Internal and External Auditors
- Case Viewer
- Case Initiator

A user with one of these Case Management roles can be assigned additional roles from among the AM roles or the Case Administrator role .

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 the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3* for more information about User to User Group mapping.

Mapping a user to Multiple User Groups within Alert Management and Case Management

If a user needs to have more than one role within Behavior Detection, then the user needs to be mapped to the different User Groups associated with the corresponding role. When the user logs in, user access permissions would be the union of access and permissions across all roles.

Mapping a user to Multiple User Groups across Alert Management, Case Management, and Other Applications

If a user needs to have different roles in both Alert and Case Management 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. In the Behavior Detection Start page, there are two menu links, one for Alert Management and one for Case Management. For any other platform applications the user is mapped to, clicking each link opens the selected application in a new window.

Mapping a Function to a Role

The following list of functions need to be mapped to appropriate Alert and Case User Roles through Function-Role Map function, which is, available in Security Management System, by logging in as the System Administrator in OFSAAI toolkit.

AMACCESS

All Alert Management user roles should be mapped to the function AMACCESS in order to access an alert. Users of roles that are not mapped to this function cannot access the details of the Alerts.

CMACCESS

All Case Management user roles should be mapped to the function CMACCESS in order to access a Case. Users of roles that are not mapped to this function cannot access the details of the Case.

RSGNTALL

This function should be mapped to Case Analyst1, Case Analyst2 and Case Supervisor Roles to assign ownership of a case without applying restriction on the Organization associated with the Case.

If the ownership assignment is required to be restricted based on Organization associated with the Case for any of these user roles, then the RSGNTALL function need not be mapped to the above roles.

Note: The Function to Role mappings are pre-packaged with the solution.

Defining the User Access Properties and Relationships

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

- **Business Domain(s):** Property that enables an Oracle client to model client data along operational business lines and practices.
- **Jurisdiction(s):** Property that enables an Oracle client to model client data across such attributes as geographic location or type or category of a business entity.

- **Organization(s):** Department or organization to which an individual user belongs.
- **Role(s):** Permissions or authorizations assigned to a user in the system (such as, Oracle Financial Services administrator or Auditor).
- **Scenario Group(s):** Group of scenarios in Oracle Financial Services that identifies a set of scenario permissions and to which a user has access rights applicable to Alert Management only.
- **Case Type/Subtype(s):** Case type/subtypes combinations to which, a user has access rights applicable to Case Management only.

Figure 5 illustrates the Oracle Financial Services user authorization model.

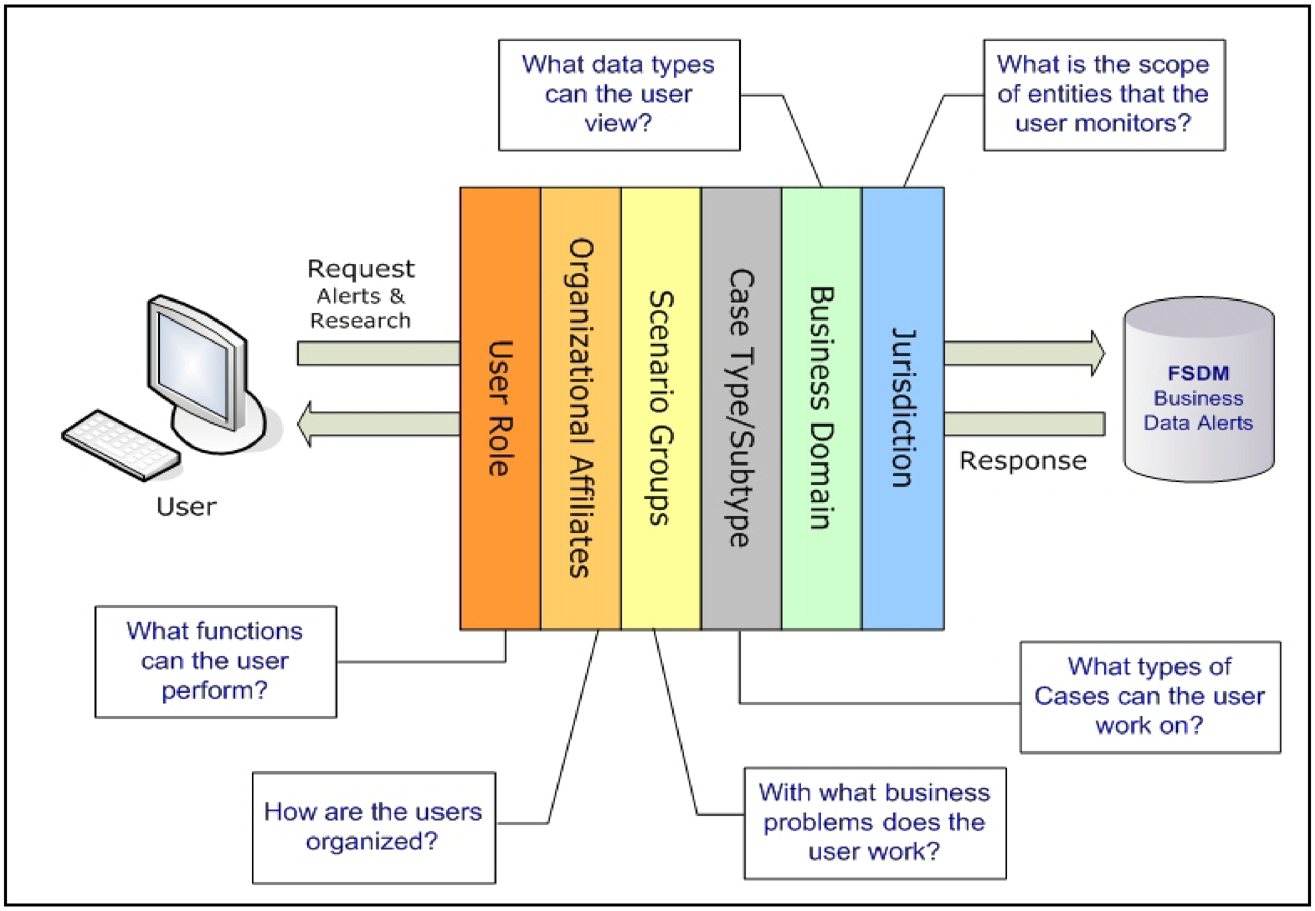

What data types can the user view?
What is the scope of entities that the user monitors?
Request Alerts & Research
User
User Role
Organizational Affiliates
Scenario Groups
Case Type/Subtype
Business Domain
Jurisdiction
FSDM Business Data Alerts
Response
What functions can the user perform?
What types of Cases can the user work on?
How are the users organized?
With what business problems does the user work?

Figure 5. Oracle Financial Services User Authorization Model

Table 11 provides the relationships between the data points that Figure 5 illustrates.

Table 11. Relationships between Data Points

Data Point	Relationship
Organization	Root of an Oracle client's organization hierarchy
	Associated with 0..n users as a line organization
	Associated with 0..n users for view access to the organization
	Associated with 1..n Business Domains
	Associated with 1..n Scenario Groups
	Associated with 1..n Case Type/Subtypes
	Associated with 1..n Jurisdictions
	Has no direct relationship with a Role
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`)
Users (Admin Tools)	Should be mapped only to mantas Admin Role.
Scenario Class	Associated to 0..n users
	Associated with Scenarios referenced in `KDD_SCNRO` table.
Case Type/Subtype	Associated to 0..n users
	Group name identifies the case type/subtype, matching a case `CASE_TYPE_SUBTYPE_CD` in the `KDD_CASE_TYPE_SUBTYPE` table.
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 12.

Table 12. Access Control Items and Locations

Data Item	Location
User Name	`KDD_REVIEW_OWNER`
User ID	`KDD_REVIEW_OWNER` (in the database)
Primary Organization	`KDD_REVIEW_OWNER`
Viewable Organizations	`KDD_REVIEW_OWNER_ORG`
Role	`CSSMS_ROLE_MAST`
User to Scenario Group	`KDD_SCNRO_GRP_ACCESS`
Scenario Group	`KDD_SCNRO_GRP_ACCESS`
Scenario to Scenario Group	`KDD_SCNRO_GRP_MEMBERSHIP`
Case Type/Subtype	`KDD_CASE_TYPE_SUBTYPE`
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 and Account Approval/Pre-Trade Approval. 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 (Refer to *Creating Jurisdiction in the Database*, on page 35, for more information).
- **Business Domain:** Operational line of business (Refer to *Creating Business Domain in the Database*, on page 36, for more information).
- **Scenario Group:** Grouping of scenarios to control user access to scenarios.
- **Role:** Permissions or authorizations assigned to a user.
- **Organization:** User group to which a user belongs.

Some data types such as Scenario Group, Role, Business Domain, Case Type, and Case Subtype which compose the user security configuration are predefined with sample values which are available through the installer. Clients can change or add new values for these data types (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 and Account Approval/Pre-Trade Approval uses Jurisdictions to limit user access to data in the database. Records from the Oracle 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 the Alert, Case Management, and Account Approval/Pre-Trade Approval system, users can view only data 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.

There can be two approaches to create a jurisdiction in the database:

- *Creating Jurisdiction in the Database through Scripts*
- *Creating Jurisdiction in Database through Excel Upload*

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 13 describes.

Table 13. `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 similar to the sample script in Figure 6.

```
INSERT INTO KDD_JRSDCN (JRSDCN_CD, JRSDCN_NM, JRSDCN_DSPLY_NM,
JRSDCN_DESC_TX) VALUES ('N', 'North', 'North', 'Northern US');
```

Figure 6. 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 Jurisdiction in Database through Excel Upload

The Excel upload process inserts the data into the appropriate dimension tables based on the pre-configured Excel upload definitions installed as part of the application installation. Data already existing should not be loaded again, as this would result in failure of upload. When uploading additional records, only the incremental records should be maintained in the Excel template with the correct unique identifier key.

1. All template Excel files for Excel upload are available in `ftpshare/STAGE/Excelupload/AMCMLookupFiles`.
2. All date values should be provided in `MM/DD/YYYY` format in the Excel worksheet.
3. Whenever a record is deleted from the Excel, the complete row should be deleted. In other words, no blank active record should exist in the Excel.
4. After selecting the Excel template, preview it before uploading.

The Excel Upload screen can be accessed by logging in as Admin user.

The Excel template to be used is KDD_JURISDICTION.xls.

Creating Business Domain in the Database

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.

The system also stores business domains in the KDD_CENTRICITY table to control access to Research against different types of entities. Derived External Entities and Addresses inherit the business domain set that is configured in KDD_CENTRICITY for those focus types.

'There can be two approaches to creating a Business Domain in the database:

- *Creating Business Domain in the Database through Scripts*
- *Creating Business Domain in the Database through Excel Upload*

Creating Business Domain in the Database through Scripts

To create a business domain, follow these steps:

1. Add the appropriate user record to the KDD_BUS_DMN database table, which Table 14 describes.

Table 14. 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 Oracle client specified the business domain, you should set the flag to N.

   The KDD_BUS_DMN table already contains predefined business domains for the Oracle client.

2. Add more records to the table by using a SQL script similar to the sample script in Figure 7.

```
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 7. 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'
```

Creating Business Domain in the Database through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 35, to perform the Excel Upload for Business Domain. The Excel template to be used is KDD_BUS_DMN.xls.

Creating Case Type/Subtype

If your firm has implemented Oracle Financial Services Enterprise Case Management, you will need to establish access permissions associated with the available Case Types and Subtypes. Case Type/Subtype is used for data access controls similar to business domain but have a different objective. The case type/subtype can be used to identify records of different case types or to provide more granular restrictions to data such as case data.

The following tables are involved in the display of the Case type, Subtype, SubClass1, and SubClass2 in the Case Management UI and are specific to the Case Management implementation.

- `KDD_CASE_TYPE_SUBTYPE`- Each record in the Case Type Subtype table represents a case type subtype available in the Oracle Financial Services Case Management system. Cases are logically grouped to a certain type and subtypes based on their behavior of interest and purpose of investigation like AML, Fraud, etc. When generated a case should be mandatorily assigned to one of the case types for further investigation. For a case type, subtype is may or may not exist.
- `KDD_SUBCLASS1`- Each record in the Case Subclass 1 table represents a subclass based on which the cases of a particular type and subtype can be grouped. On categorizing the cases based on type and subtype they can further be grouped based on these subclasses. Case Subclass 1 provides the list of subclasses for first level grouping. Subclasses are not mandatory information for a case.
- `KDD_SUBCLASS2`- Each record in the Case Subclass 2 table represents a subclass based on which the cases of a particular type and subtype can be grouped. On categorizing the cases based on type and subtype they can further be grouped based on these subclasses. Case Subclass 2 provides the list of subclasses for second level grouping. Subclasses are not mandatory information for a case.
- `KDD_TYPE_CLASS_MAP`- Each record in the Case Type and Class Map table represents the set of valid combinations of case type/subtype, subclass1 and subclass2 values which can be used to group the cases for proper investigation.

Refer to the *Oracle Financial Services Behavior Detection Platform Financial Services Data Model Volume 3: Case Management* for more information about these tables.

Creating Case Type/Subtype in Investigation Schema

You can create a Case Type/Subtype in the investigation schema in the following ways:

- *Adding Entries directly in the Table Using Script*
- *Adding Entries through Excel Upload*

Adding Entries directly in the Table Using Script

To add entries in the table using scripts, follow these steps:

1. Add the appropriate record to the `KDD_CASE_TYPE_SUBTYPE` database table.
2. Add records to the table by using a SQL script similar to the following sample script.

```
insert into KDD_CASE_TYPE_SUBTYPE (CASE_TYPE_SUBTYPE_CD,
CASE_TYPE_CD, CASE_TYPE_NM, CASE_TYPE_DESC, CASE_SUBTYPE_CD,
CASE_SUBTYPE_NM, CASE_SUBTYPE_DESC, CASE_CLASSIFICATION_CD,
LAST_UPDATED_BY, LAST_UPDATED_DT, COMMENTS)
values ('AML_SURV', 'AML', 'Anti-Money Laundering', 'Anti-Money
Laundering', 'SURV', 'AML Surveillance', 'AML Surveillance',
'AML', null, null, null);
```

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 35, for the steps to perform the Excel Upload of Case Subtype.

The Excel template to be used is `KDD_CASE_TYPE_SUBTYPE.xls`

Creating Case Subclass1 in Investigation Schema

You can create a Case Subclass1 in the database in the following ways:

- Adding Entries directly in the Table using script.
- Add the appropriate record to the `KDD_CASE_SUBCLASS1` database table.

Add records to the table by following SQL script, similar to the sample script.

```
insert into KDD_SUBCLASS1 (CASE_SUBCLASS1_CD,
CASE_SUBCLASS1_NM, CASE_SUBCLASS1_DESC, LAST_UPDATED_DT,
LAST_UPDATED_BY, COMMENTS)
values ('BSA', 'Bank Secrecy Act', 'Bank Secrecy Act', null,
null, null)
```

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 35, for the steps to perform the Excel Upload of Case Subclass1.

The Excel template to be used is `KDD_CASE_SUBCLASS1.xls`

Creating Case Subclass2 in Investigation Schema

You can create a Case Subclass2 in the database in the following ways:

- Adding Entries directly in the Table using scripts.
- Adding the appropriate record to the `KDD_CASE_SUBCLASS2` database table.
- Adding records to the table by using a SQL script similar to the following sample script.

```
insert into KDD_SUBCLASS2 (CASE_SUBCLASS2_CD,
CASE_SUBCLASS2_NM, CASE_SUBCLASS1_DESC, LAST_UPDATED_DT,
LAST_UPDATED_BY, COMMENTS)
values ('BSA', 'Bank Secrecy Act', 'Bank Secrecy Act', null,
null, null)
```

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 35, for the steps to perform the Excel Upload of Case Subclass2.

The Excel template to be used is `KDD_CASE_SUBCLASS2.xls`

Creating Case Type and Class Map in Investigation Schema

You can create a Case Type and Class Map in the database in the following ways:

Adding Entries directly in the Table using script

- Add the appropriate record to the `KDD_TYPE_CLASS_MAP` database table
- Add records to the table by using a SQL script similar to the following sample script.

```
insert into KDD_TYPE_CLASS_MAP (CASE_TYPE_CLASS_SEQ_ID,
CASE_TYPE_SUBTYPE_CD, CASE_SUBCLASS1_CD, CASE_SUBCLASS2_CD)
values (1, 'AML_SURV', 'BSA', 'CMIR')

insert into KDD_TYPE_CLASS_MAP (CASE_TYPE_CLASS_SEQ_ID,
CASE_TYPE_SUBTYPE_CD, CASE_SUBCLASS1_CD, CASE_SUBCLASS2_CD)
values (2, 'AML_SURV', 'BSA', 'FBAR');
```

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 35, for the steps to perform the Excel Upload of Case Type and Class Map.

The Excel template to be used is `KDD_TYPE_CLASS_MAP.xls`

Note: All template Excel files for Excel upload will be available in the following location: `ftpshare/STAGE/Excelupload/AMCMLookupFiles`

Creating Organizations in the Database

There can be two approaches to create an Organization in the database:

- Creating Organization in the Database through scripts
- Creating Organization in the Database through Excel Upload

Creating Organization in the Database through scripts

Add entries directly to the `KDD_ORG` table using a script.

1. Add the appropriate record to the `KDD_ORG` database table, which Table 15 describes.

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

Table 15. `KDD_ORG` Table Attributes

Business Field	Column Name	Date Type	Definition	Null
Organization	`ORG_CD`	CHAR(20)	Unique identifier for this organization.	No
Organization Display Name	`ORG_NM`	CHAR(60)	Short name for the organization that is used for display purposes.	Yes
Organization description	`ORG_DESC_TX`	CHAR(100)	Description of this organization	Yes
Line Organization	`PRNT_ORG_CD`	CHAR(20)	Identifies the parent organization of which this organization is considered to be a child	Yes
Modification Date	`MODFY_DT`	DATE	Identifies the last modified Date and time.	Yes
Modified User	`MODFY_ID`	NUMBER(10)	Identifies the user id of the user who last modified the data	Yes
Comment	`COMMENT_TX`	CHAR(4000)	Comment	

2. Add records to the table by using a SQL script similar to the sample script in Figure 8. Sample SQL Script for Loading `KDD_ORG`.

```
insert into KDD_ORG (ORG_CD, ORG_NM, ORG_DESC_TX, PRNT_ORG_CD,
MODFY_DT, MODFY_ID, COMMENT_TX)
values ('TestOrgA', 'TestOrgA', 'TestOrgA', null, null, null, null);
insert into KDD_ORG (ORG_CD, ORG_NM, ORG_DESC_TX, PRNT_ORG_CD,
MODFY_DT, MODFY_ID, COMMENT_TX)
values ('TestOrgB', 'TestOrgB', 'TestOrgB', 'TestOrgA', null, null,
null);
insert into KDD_ORG (ORG_CD, ORG_NM, ORG_DESC_TX,
PRNT_ORG_CD,MODFY_DT, MODFY_ID, COMMENT_TX) values ('TestOrgC',
'TestOrgC', 'TestOrgC', 'TestOrgA', null, null,null); MODFY_DT,
MODFY_ID, COMMENT_TX)
```

Figure 8. Sample SQL Script for Loading KDD_ORG

Creating Organization in the Database through Excel Upload

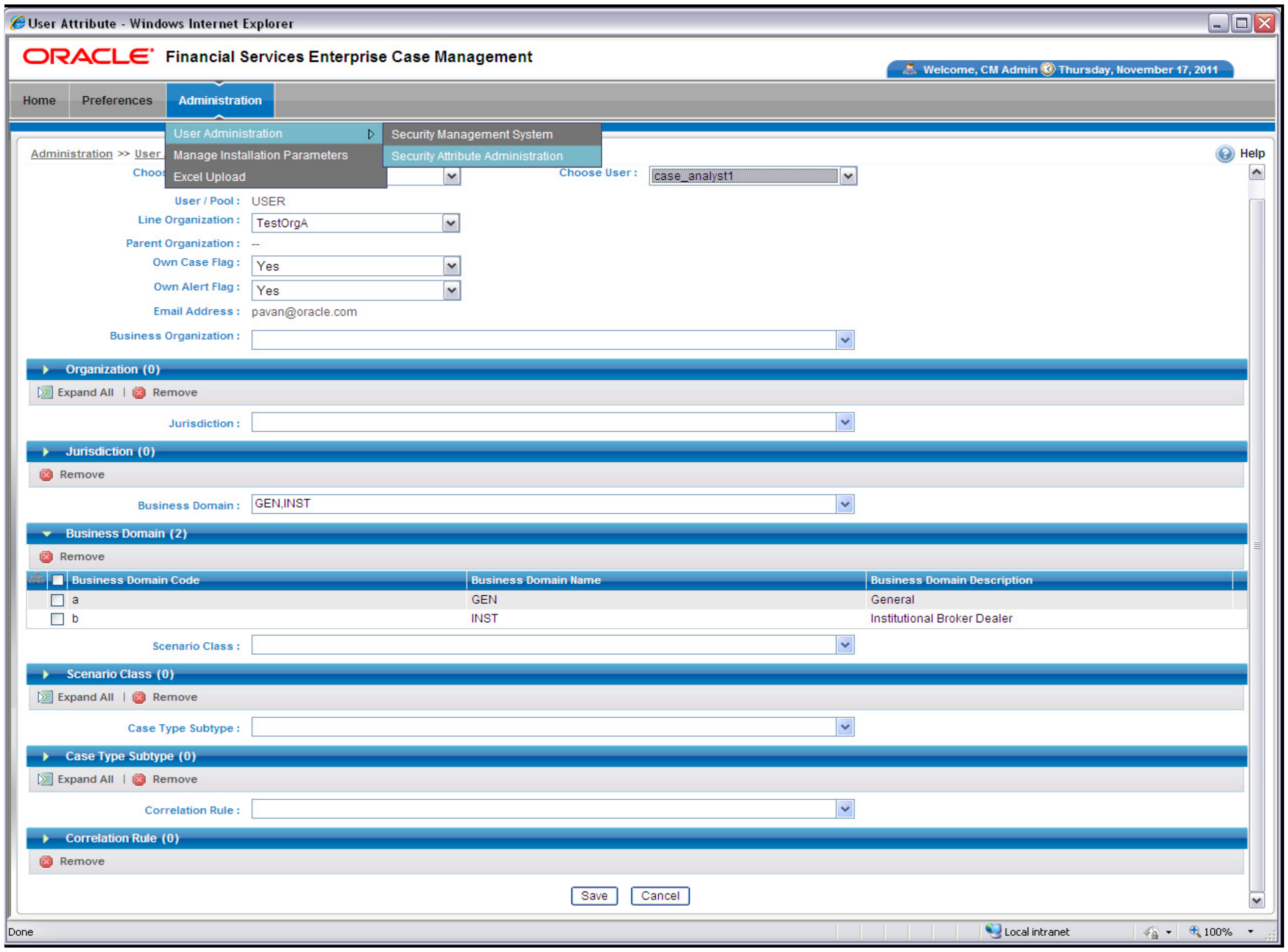
Refer to *Creating Jurisdiction in the Database*, on page 35, to perform the Excel Upload of organization.

The Excel template to be used is `KDD_ORG.xls`.

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. The Security Attribute Administration can be accessed from the Administration menu (Figure 9).

Note: Before proceeding with providing a user access through this UI, all necessary data should be available in the appropriate database tables and the user must be created.


User Attribute - Windows Internet Explorer
ORACLE Financial Services Enterprise Case Management
Welcome, CM Admin
Thursday, November 17, 2011
Home
Preferences
Administration
User Administration
Manage Installation Parameters
Excel Upload
Security Management System
Security Attribute Administration
Administration >> User
Help
Expand All
Remove
Save
Cancel
Done
Local intranet
100%

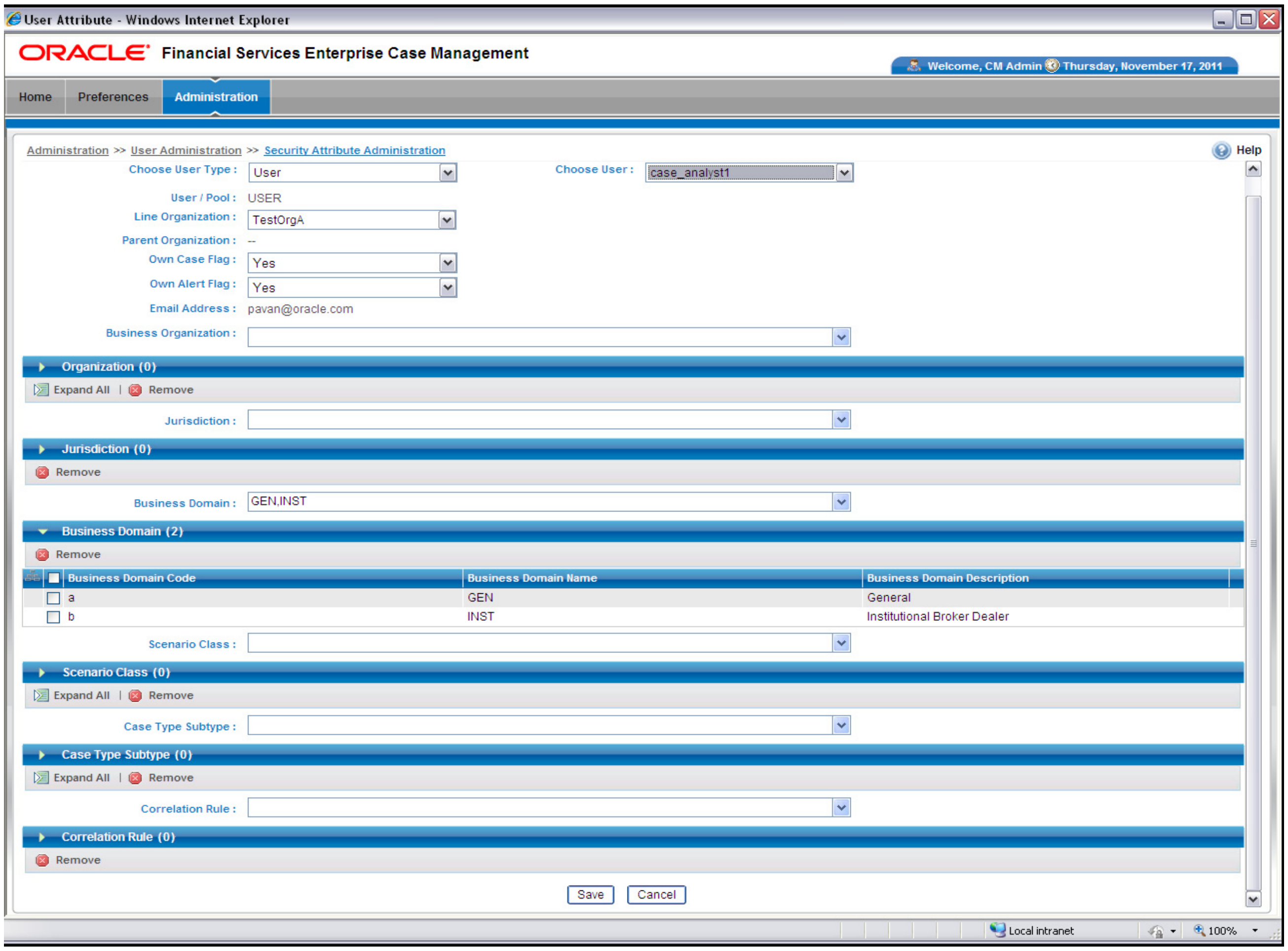
Choose User :	case_analyst1
User / Pool :	USER
Line Organization :	TestOrgA
Parent Organization :	--
Own Case Flag :	Yes
Own Alert Flag :	Yes
Email Address :	pavan@oracle.com
Business Organization :	
Organization (0)	
Jurisdiction :	
Jurisdiction (0)	
Business Domain :	GEN,INST
Business Domain (2)	

Business Domain Code	Business Domain Name	Business Domain Description
a	GEN	General
b	INST	Institutional Broker Dealer

Scenario Class :	
Scenario Class (0)	
Case Type Subtype :	
Case Type Subtype (0)	
Correlation Rule :	
Correlation Rule (0)	

Figure 9. Security Attribute Administration

Using this UI an Administrator can map both Organizations and Users to different Security attributes.


User Attribute - Windows Internet Explorer
ORACLE Financial Services Enterprise Case Management
Welcome, CM Admin
Thursday, November 17, 2011
Home
Preferences
Administration
Administration >> User Administration >> Security Attribute Administration
Help
Expand All
Remove
Save
Cancel
Local intranet
100%

Choose User Type : User

Choose User : case_analyst1

User / Pool : USER

Line Organization : TestOrgA

Parent Organization : --

Own Case Flag : Yes

Own Alert Flag : Yes

Email Address : pavan@oracle.com

Business Organization :

Organization (0)

Jurisdiction :

Jurisdiction (0)

Business Domain : GEN,INST

Business Domain (2)

Business Domain Code	Business Domain Name	Business Domain Description
a	GEN	General
b	INST	Institutional Broker Dealer

Scenario Class :

Scenario Class (0)

Case Type Subtype :

Case Type Subtype (0)

Correlation Rule :

Correlation Rule (0)

Figure 10. Components of Security Attribute

Note: In order to update the user profiles before proceeding with mapping any security attributes, select the value **User** from the **Choose User Type** drop-down list. When chosen, all the updates made to user profiles through the User Maintenance UI are imported from the `CSSMS_USER_PROFILE` table of the OFSAAI configuration schema to the `KDD_REVIEW_OWNER` table of the mantas schema.

This action does not affect the security attributes that might be already mapped.

Once the user details are imported, the security attributes should be mapped/remapped.

The drop-down lists have options for both Organizations and Users. To map an organization, select the organization from the drop-down list and select the corresponding Organization in the **Choose User** drop-down list.

The **Choose User** drop-down list filters its values based on the value selected in the **Choose User Type** selection drop-down list. It shows only users, if the **User Type** is User; and it shows only organizations, if the **User Type** is Organization.

After selecting the desired user in **Choose User** drop-down list, the Administrator can map the following parameters to the selected user:

- Organization
- Jurisdiction
- Business Domain
- Scenario Class
- Case Type/Subtype
- Correlation Rule

Organization

A User or Organization's access to other Organization depends on the selection(s) made for this organization parameter. For Example, if a user is mapped Org1 and Org2, it implies that, user can access alert/case, which belongs to these two organizations, provided other security attributes are also matching.

Jurisdiction

Mapping of one or more jurisdictions to a user or organization, gives the privilege of accessing cases, alerts, account approval requests or pre-trade approval requests that belong to the mapped jurisdiction.

Business Domain

Mapping of one or more business domains to a user or organization gives privilege of accessing cases, alerts, account approval requests or pre-trade approval requests that belong to the mapped business domains.

Scenario Class

Mapping of one or more Scenario Classes to a user or organization gives the privilege of accessing alerts that belong to the mapped scenario class.

Case Type/Subtype

Mapping of one or more Case Types/Subtypes to a user or organization gives them the privilege of accessing cases that belong to the mapped Case Type/Subtype.

Correlation Rule

Mapping of one or more correlation rules gives the privilege of viewing the correlations generated based on the mapped correlation.

Additional Parameters

Other parameters, such as, Line Organization, Own Case Flag and Own Alert flag can be selected in the corresponding drop-down list mentioned in the screen and can be updated by clicking the **Save** button.

Note: The Own Alert and Case flag is required for taking ownership of the alerts and cases. If an alert user needs to perform a Promote To Case action, then the following pre-requisites should be fulfilled.

1. The user should be mapped to any one of the following user groups:
   - Case Supervisor
   - Case Analyst1
   - Case Analyst2

2. The user's 'Case Own' flag should be enabled by setting the value to 'Y'.
   Or
   The user should be mapped to the Case Initiator Role.

About Scenario Manager Login Accounts

Behavior Detection users gain access to the Scenario Manager application based on the following:

- User ID and password authentication enables access to the Scenario Manager.
- An associated functional role corresponds to particular user tasks and authorities.

Creating Scenario Manager Login Accounts

As administrator, the user setup process requires that you complete the following tasks:

1. Create a database login account and password (Refer to section *To Create the Database Login Account*, on page 45, for more information).
2. Set up an account and functional roles in the Scenario Manager. Before performing any tasks in the Scenario Manager, you must set up a user login account that establishes access and roles in the Scenario Manager. Perform these setups by adding records to the database (Refer to section *To Set Up an Account and Functional Roles*, on page 46, for more information).
3. Grant the database roles that the functional roles require. You can grant the role of Data Miner, or MNR to an *Oracle Financial Services Scenario Manager* user (Refer to section *To Grant a Database Role*, on page 46, for more information).

   Note: Oracle Financial Services suggests having only a few generic users in the database to use the Scenario Manager, as most organizations have an extremely small user population to execute these tools.

To Create the Database Login Account

The system instantiates the Behavior Detection database as a set of Oracle database tables. Therefore, each user whom the Oracle client authorizes to use the Scenario Manager must have login access to the Oracle database. As administrator, you must set up an Oracle login account for each user, and assign the `KDD_MNR` user role to this account.

Note: Behavior Detection does not support external logins (for example, OPS$accounts) in an Oracle environment. Users must provide an explicit password when logging on.

The assumption is that the client's system administrator has training and experience in performing such setups, and, therefore, does not require instructions here on how to perform this task. However, for information about setting up Oracle accounts, refer to the appropriate Oracle documentation.

Note: The Solaris and Oracle login user IDs do not have to be identical. However, the Oracle Financial Services Scenario Manager and Oracle login user IDs MUST be identical.

To Set Up an Account and Functional Roles

To create a Scenario Manager account and functional role, follow these steps:

1. Access the `KDD_USER` table.

   Table 16 defines the attributes for the `KDD_USER` table.

Table 16. `KDD_USER` Table Attributes

Column Name	Description
`USER_ID`	User's database login ID.
`USER_NM`	User's name.
`USER_ROLE_CD`	User's default database role.
`ACTV_FL`	Active user indication (Y or N).
`WRKLD_CD`	Not used by the Scenario Manager.

2. Enter the following information into the table using an SQL script:

   a. User database login ID in the `USER_ID` column. (The Scenario Manager and Oracle login user IDs must be identical.)

   b. User name in the `USER_NM` column.

   c. Default user role in the `USER_ROLE_CD` column.

      To use the Scenario Manager, the user needs the MNR (Data Miner) database role. The MNR database role is responsible for adjusting the pattern logic of existing scenarios and employs data mining techniques to create new patterns and scenarios.

   d. Flag of Y(es) or N(o) in the `ACTV_FL` column to specify whether the user is active.

   A sample SQL insert statement is:

```
INSERT INTO KDD_USER VALUES (‘KDD_MNR’, ‘KDD MINER’, ‘MNR’, ‘Y’,
‘FT’);
```

To Grant a Database Role

To grant a database role to the Scenario Manager `KDD_MNR` user, follow the steps:

1. Access the `KDD_USER_ROLE` table.

   Table 17 defines the attributes in the `KDD_USER_ROLE` table.

Table 17. `KDD_USER_ROLE` Table Attributes

Column Name	Description
`USER_ID`	User's login ID.
`USER_ROLE_CD`	User's database role.

2. Enter the following information into the table using an SQL script:

   - User login ID in the `USER_ID` column.
   - User role MNR in the `USER_ROLE_CD` column.

   A sample SQL insert statement is:

```
INSERT INTO KDD_USER_ROLE values (‘KDD_MNR’, ‘MNR’);
```

About Changing Passwords for System Accounts

Throughout the Behavior Detection application there are several system accounts that may require changing the password for security purposes.

Table 18 summarizes the different system account passwords used by the application, the subsystems that use those passwords, and instructions on how to change the passwords.

Table 18. System Account Passwords

System Account	Subsystem	Instructions
Data Ingest User (`INGEST_USER`)	Data Ingestion	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password. This password also needs to be changed on the Informatica pmserver: 1. Start the Informatica Workflow Manager. 2. Connect to the appropriate repository as the Administrator user. 3. Select the **Relational** option from the **Connections** drop-down menu. 4. Click the ingest_user connection that appears under Oracle. 5. Click **Edit**. 6. Type the desired password in the Password field. 7. Click **OK**. 8. Click **Close**. 9. Select the Save Repository item from the Repository drop-down menu.
Algorithm User (`KDD_ALG`)	Behavior Detection Services	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password.
Data Miner User (`KDD_MNR`)	Alert & Case Management Data Ingestion	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password.
Web Application User (`KDD_WEB`)	Alert & Case Management Services	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password.
Patch Database User	Patch Installer	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password.
Database Utility User	Database	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password.
Informatica Workflow Operator (Informatica Repository user ID that executes Behavior Detection workflows)	Data Ingestion	1. Use the Informatica pmpasswd utility to generate an encrypted version of the password. Do not use an encrypted version that has single quotes (') in it. 2. Find the `svrpwd='<password>'` property in the `<INSTALL_DIR>/ingestion_manager/scripts/env.sh` file and replace <password> with the desired encrypted password. Refer to the *Oracle Financial Services Behavior Detection Platform Installation Guide*, Release 6.1.3, for more information.

Table 18. System Account Passwords (Continued)

System Account	Subsystem	Instructions
Active Pages User (KDD_ALTIO)	Active Pages	**For the NetViz component**: Change the password for the properties `hibernate.connection.password` and `mantas.db.pwd` in the files `$ALTIO_HOME/WEB-INF/classes/hibernate.properties` and `$ALTIO_HOME/WEB-INF/classes/apps/NetVis/altio-app.xml`. **For all other Altio apps**: Change the password for the property `db.password` in the file `$ALTIO_HOME/WEB-INF/classes/apps/<Altio app directory>/altioapp.xml`.
Reports User (KDD_REPORT)	OBIEE Reports	Open the `$OracleBI_HOME/server/Repository` and expand the Physical Layer. Open the Connection Pool and change the Password parameter to set a new value of the `KDD_REPORT` schema password. **Note**: OBIEE is an optional application.

Note: When you change configuration files under the Alert & Case Management subsystem, you should always make your changes in the original Oracle Financial Services installation directories. If you have deployed Behavior Detection as a WAR, you must recreate the alert & case management and admin tools WARs using `create_am_war.sh` and `create_at_war.sh`. Then re-deploy the WAR using your Web Application server console. If you are running Behavior Detection directly out of the installation directories, you must restart the Behavior Detection web applications using your Web Application server console so that the system recognizes the new passwords. Refer to the section on application development in the *Oracle Financial Services Behavior Detection Platform Configuration Guide*, for more information.

About Configuring File Type Extensions

The list of file type extensions that are allowed to be attached while performing document attachment action should be configured as comma separated values in `CONFIGURATION` table of `OFSAAI` configuration schema in its `PARAMVALUE` column where `PARAMNAME` is `DOCUMENT_ALLOWED_EXTENSION`.

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:

- About Data Ingestion
- Process Flow
- Intra-Day Ingestion Processing
- Informatica Ingestion Parameters
- Alternatives to Standard Data Ingestion Practices
- Data Ingestion Directory Structure
- Startup and Shutdown
- Data Rejection During Ingestion
- Data Ingestion Archiving
- Miscellaneous Utilities
- Copying Thomson Reference Files
- Fuzzy Name Matcher Utility
- Refresh Temporary Table Commands
- Use of Control Data

About Data Ingestion

The Ingestion Manager receives, transforms, and loads Market, Business, and Reference data that alert detection processing requires. The Ingestion Manager typically receives Market data from a real-time Market data feed or file adapter interface, and both Business and Reference data through the file adapter interface. The Data Ingestion subsystem transforms Market, Business, and Reference data to create derived attributes that the detection algorithms require. The system extracts and transforms data and subsequently loads the data into the database. After loading the base tables, the Oracle client's job scheduling system invokes Informatica workflows to derive and aggregate data. The Informatica component also uses the Fuzzy Name Matcher Utility to compare names found in source data with names in the Watch List.

The Oracle client implements Ingestion Manager by setting up a batch process that conforms to the general flow that this chapter describes. Typically, the system uses a job scheduling tool such as Maestro or Unicenter AutoSys to control batch processing of Ingestion Manager.

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. Figure 11 shows how the Data Ingestion subsystem components participate in each workflow step to extract, transform, and load data. The workflow processes include the following:

- In **Extract or Preprocess Workflow**, preprocessor components receive the raw market data, business (firm) data, and reference data from their external interfaces. The components then perform data validation and prepare the data for further processing.
- In **Transform Workflow**, transformer components receive the preprocessed market and business data. The components then create derived attributes that support the downstream alert processing.
- In **Load and Transform Workflow**, loader components receive preprocessed Reference data and transformed market and business data. The components then load this data into the database. In Post Data Load, data transformations occur through Informatica: derivations and aggregations, risk assignment, and watch list processing (refer to Chapter 5, *Informatica Workflows* on page 133 for more information).

The following sections describe this process flow in more detail.

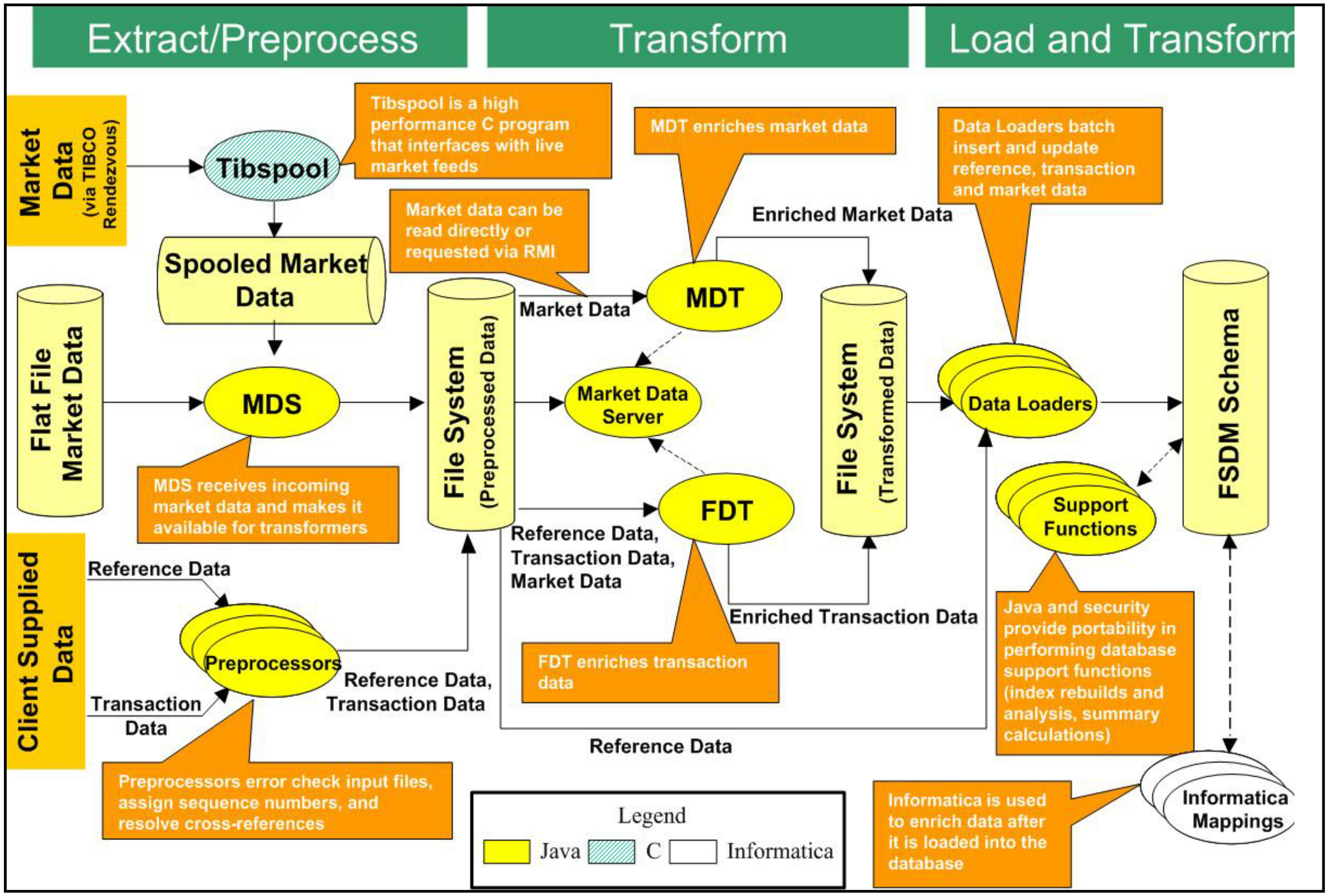

Extract/Preprocess
Transform
Load and Transform
Market Data (via TIBCO Rendezvous)
Tibspool
Tibspool is a high performance C program that interfaces with live market feeds
Spooled Market Data
Flat File Market Data
MDS
MDS receives incoming market data and makes it available for transformers
Market data can be read directly or requested via RMI
File System (Preprocessed Data)
Market Data
MDT enriches market data
MDT
Enriched Market Data
Market Data Server
FDT
Reference Data, Transaction Data, Market Data
Enriched Transaction Data
FDT enriches transaction data
File System (Transformed Data)
Data Loaders batch insert and update reference, transaction and market data
Data Loaders
FSDM Schema
Support Functions
Java and security provide portability in performing database support functions (index rebuilds and analysis, summary calculations)
Client Supplied Data
Reference Data
Transaction Data
Preprocessors
Reference Data, Transaction Data
Preprocessors error check input files, assign sequence numbers, and resolve cross-references
Reference Data
Legend
Java
C
Informatica
Informatica is used to enrich data after it is loaded into the database
Informatica Mappings

Figure 11. Data Ingestion Subsystem

Data Ingestion Process Summary

Figure 12 provides a high-level view of the Data Ingestion process for Oracle Financial Services Trading Compliance Solution (TC), Anti-Money Laundering (AML), Broker Compliance Solution (BC), Fraud (FR) and Insurance.

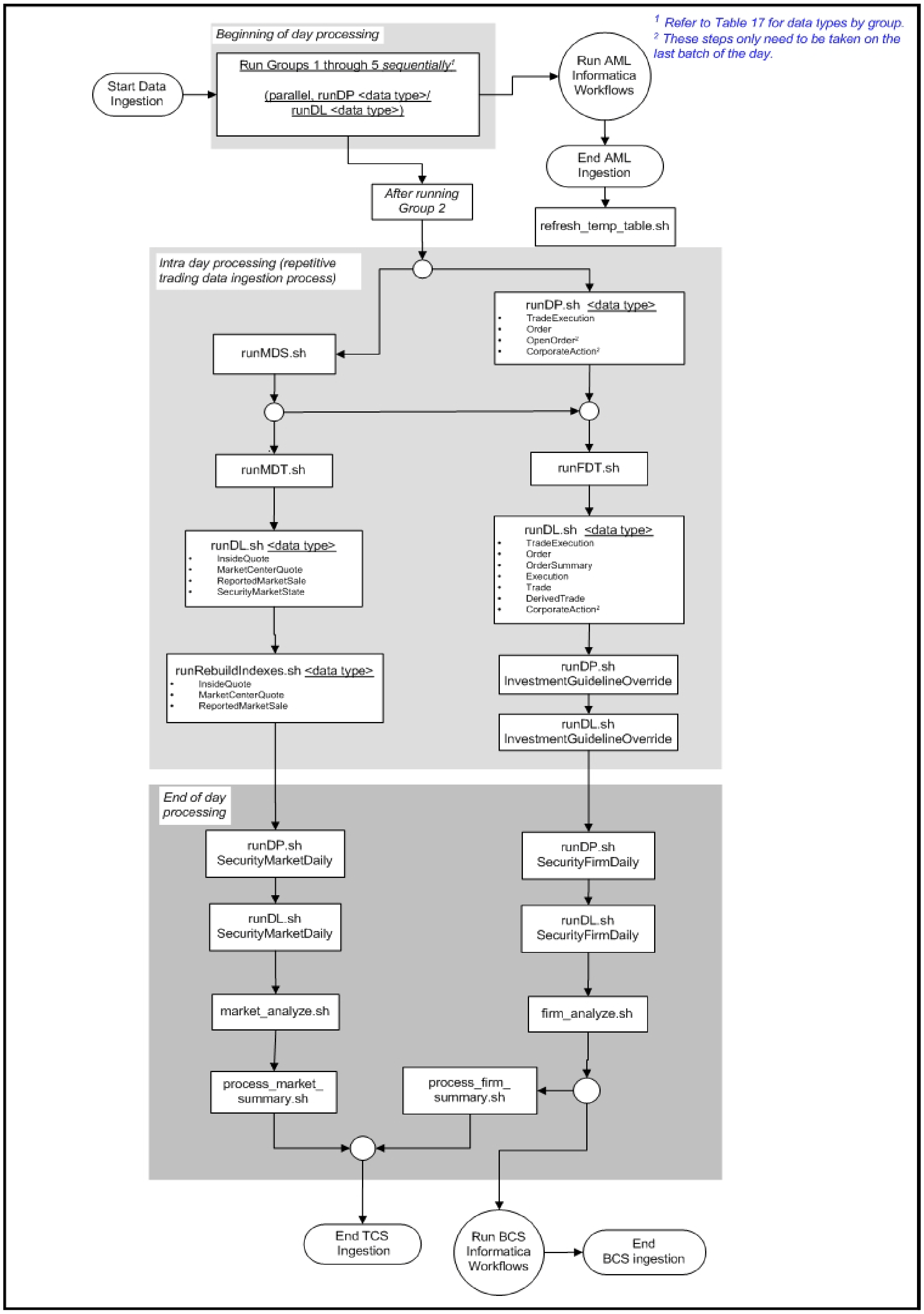

1 Refer to Table 17 for data types by group.
2 These steps only need to be taken on the last batch of the day.
Beginning of day processing
Start Data Ingestion
Run Groups 1 through 5 sequentially1
(parallel, runDP <data type>/ runDL <data type>)
Run AML Informatica Workflows
End AML Ingestion
refresh_temp_table.sh
After running Group 2
Intra day processing (repetitive trading data ingestion process)
runDP.sh <data type>
TradeExecution
Order
OpenOrder2
CorporateAction2
runMDS.sh
runMDT.sh
runFDT.sh
runDL.sh <data type>
InsideQuote
MarketCenterQuote
ReportedMarketSale
SecurityMarketState
runDL.sh <data type>
TradeExecution
Order
OrderSummary
Execution
Trade
DerivedTrade
CorporateAction2
runRebuildIndexes.sh <data type>
InsideQuote
MarketCenterQuote
ReportedMarketSale
runDP.sh InvestmentGuidelineOverride
runDL.sh InvestmentGuidelineOverride
End of day processing
runDP.sh SecurityMarketDaily
runDL.sh SecurityMarketDaily
runDP.sh SecurityFirmDaily
runDL.sh SecurityFirmDaily
market_analyze.sh
firm_analyze.sh
process_market_summary.sh
process_firm_summary.sh
End TCS Ingestion
Run BCS Informatica Workflows
End BCS ingestion

Figure 12. Data Ingestion Process

Alternate Process Flow for MiFID Clients

Derivations done by the FDT process for the MiFID scenarios, which use the Order Size Category, require the use of the Four-week Average Daily Share Quantity (4-wk ADTV) to define an order as small, medium, or large based on how it compares to a percentage of the 4-wk ADTV. The 4-wk ADTV is derived on a daily basis by the `process_market_summary.sh` script in the end-of day batch once the Daily Market Profile is collected for each security from the relevant market data source.

For firms using the MiFID scenarios and running a single end-of-day batch, the `process_market_summary.sh` script must be executed prior to running the `runFDT.sh` script such that the 4-wk ADTV for the Current Business Day incorporates the published Current Day Traded Volume.

Figure 13 depicts dependency between the `process_market_summary.sh` script and the `runFDT.sh` script.

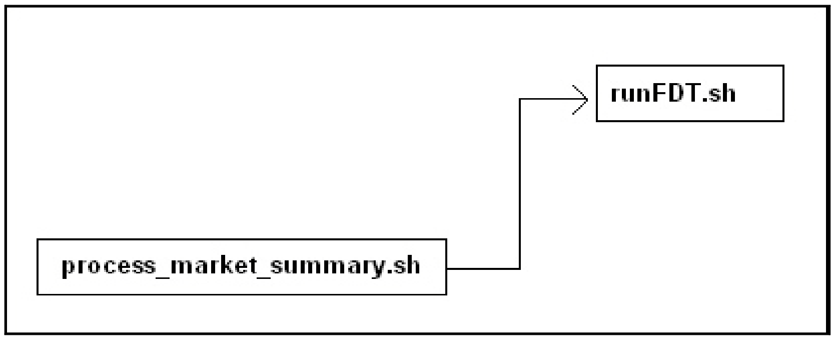


Figure 13. Dependency between *`process_market_summary.sh`* **and** *`runFDT.sh`*

For intra-day batch ingestion or intra-day execution of the MiFID scenarios, the process flow does not change from Figure 12. Since the current day's 4-wk ADTV is not available until the end of the day, the previous day's 4-wk ADTV is used to determine order size.

For additional information on configuring the percentage values used to define a MiFID-eligible order as Small, Medium, or Large, refer to the *Market Supplemental Guidance* section in the *Data Interface Specification*, Release 6.1.3.

Data Ingestion Flow Processes

The following sections take the high-level view of Figure 12 and divide the Data Ingestion flow into distinct processes:

- Beginning Preprocessing and Loading
- Preprocessing Trading Compliance Solution Data
- Processing Data through FDT and MDT
- Running Trading Compliance Solution Data Loaders
- Rebuilding and Analyzing Statistics
- Populating Market and Business Data Tables
- Processing Informatica Workflows and other Utilities

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 80, for more information.

Beginning Preprocessing and Loading

In Figure 12, section A, preprocessing begins. 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 client data in the `/inbox` directory.

Table 19 lists the data files by group.

Table 19. Data Files by Group

Group	Data Files	
1	AccountCustomerRole AccountPhone AccountEmailAddress AccountRealizedProfitAndLoss Country EmployeeToInsurancePolicy FrontOfficeTransaction InsuranceProduct InsurancePolicy InsurancePolicyBalance InsuranceSeller InsuranceSellerToLicense InsuranceTransaction Issuer InsurancePolicyToCustomer InsurancePolicyFeature CollateralValueCurrency SecuritiesLicense ExternalInvestmentAccountPosition	InvestmentGuideline InvestmentGuidelineToAccount LetterofIntent Loan LoanDailyActivity MarketCenter MarketIndex MarketIndexDaily Organization RegisteredRepresentativeComplaint ServiceTeamMember SystemLogonType WatchList OnlineAccount CollateralValueProduct EnergyAndCommodityInstrument
2	AccountGroup SecurityFirmDaily TrustedPair AccountAverageNetWorth FirmAccountPositionPair NaturalGasFlow PeerGroup	MatchedEntity SecurityMarketDaily Security AccountToPeerGroup MarketIndexMemberSecurity
3	Account Customer Employee FrontOfficeTransactionParty MarketTradingSession OrganizationRelationship AccountGroupAddress AccountGroupInvestmentObjective AccountGroupIOSMember AccountGroupMemberExperience BankerToOfficer GeneralUsageList LoanOriginationAction	AccountSupplementalAttribute CustomerSupplementalAttribute WatchListEntry RestrictionList AutomatedQuote LoanProduct MailHandlingInstructionActivity OrganizationToMortgageType ReferenceTableDetail ServiceVendor EnergyAndCommodityTrade LoanOriginationProduct

Table 19. Data Files by Group (Continued)

Group	Data Files	
4	AccountAddress AccountAssetAllocation AccountBalance AccountCollateral AccountFeature AccountGroupMember AccountInvestmentObjective AccountPosition AccountPositionPair AccountToCorrespondent AccountToCustomer AccountToOrganization AnticipatoryProfile AccountProfitAndLoss ControllingCustomer CustomerAddress CustomerBalance CustomerCountry CustomerEmailAddress CustomerPhone CustomerToCustomerRelationship AccountScheduledEvent LoanOrigination MailHandlingInstruction UncoveredOptionsDailyTradeSumary AccountPositionProfitAndLoss AccountIdentifierChangeHistory LoanOrigination AccountFees AccessEvents EmployeeFirmTransferHistory EmployeeToSecuritiesLicense	CustomerToProductsOffered CustomerToMarketsServed EmployeeAddress EmployeeEmailAddress EmployeePhone EmployeeToAccount EmployeeToOrganization EmployeeTradingRestriction FirmAccountPosition Fraud MarketNewsEvent MutualFundBreakpoint SecurityGroupMember SecurityInvestmentRating SecuritySelectListEntry SecurityTradingRestriction StructuredDeal SystemLogon OnlineAccountToAccount ManagedAccount AccountProfileStage EnergyAndCommodityFirmDaily EnergyAndCommodityMarketDaily EnergyAndCommodityMarketCenter FrontOfficeTransactionRemittanceDocument RelatedFrontOfficeTransactionInformation EnergyAndCommodityMarketTradingSession EnergyAndCommodityReportedMarketSale
5	AccountRestriction BackOfficeTransaction * ChangeLog InvestmentAdvisor SettlementInstruction SystemLogonToSystemLogonType Borrower	LoanOriginationCondition LoanOriginationConditionType LoanOriginationDocumentPrintLog LoanOriginationFeeDetail LoanOriginationNote LoanOriginationToService OptionsViolation RegisteredRepresentativeTradeCommission

* BackOfficeTransaction must be loaded after the AccountManagementStage utility has been executed (see Miscellaneous Utilities).

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 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:

- Prior to executing the `runMDS.sh` script, you should ingest the following reference data files:
  - Security
  - MarketCenter
- Prior to executing the `runDP.sh`, `TradeExecution`, and `runDL.sh` scripts, you should ingest the following reference data files:
  - Security
  - MarketCenter
  - CorporateAction
  - StructuredDeal
  - SettlementInstruction

Process Flow

The ingestion process flow is as follows:

1. Behavior Detection receives firm data in ASCII flat `.dat` files, which an Oracle 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 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 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 20 lists.

Table 20. 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.

Figure 14 summarizes preprocessing input and output directories.

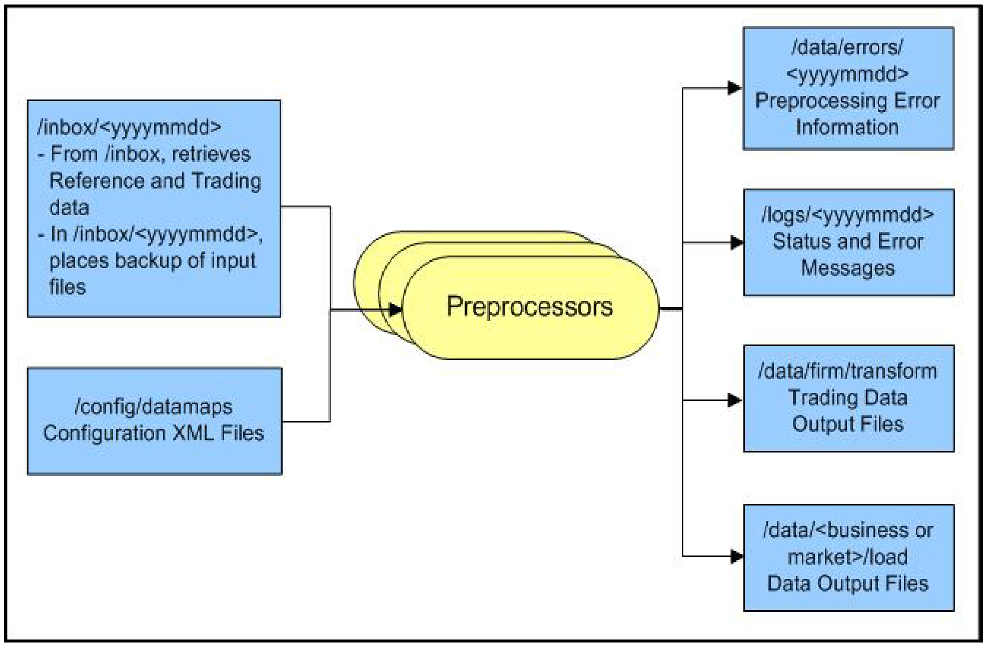

/inbox/<yyyymmdd>
- From /inbox, retrieves Reference and Trading data
- In /inbox/<yyyymmdd>, places backup of input files
/config/datamaps Configuration XML Files
Preprocessors
/data/errors/<yyyymmdd> Preprocessing Error Information
/logs/<yyyymmdd> Status and Error Messages
/data/firm/transform Trading Data Output Files
/data/<business or market>/load Data Output Files

Figure 14. Preprocessing Input and Output Directories

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
```

Figure 15 illustrates the Trading Compliance Solution data loading process.

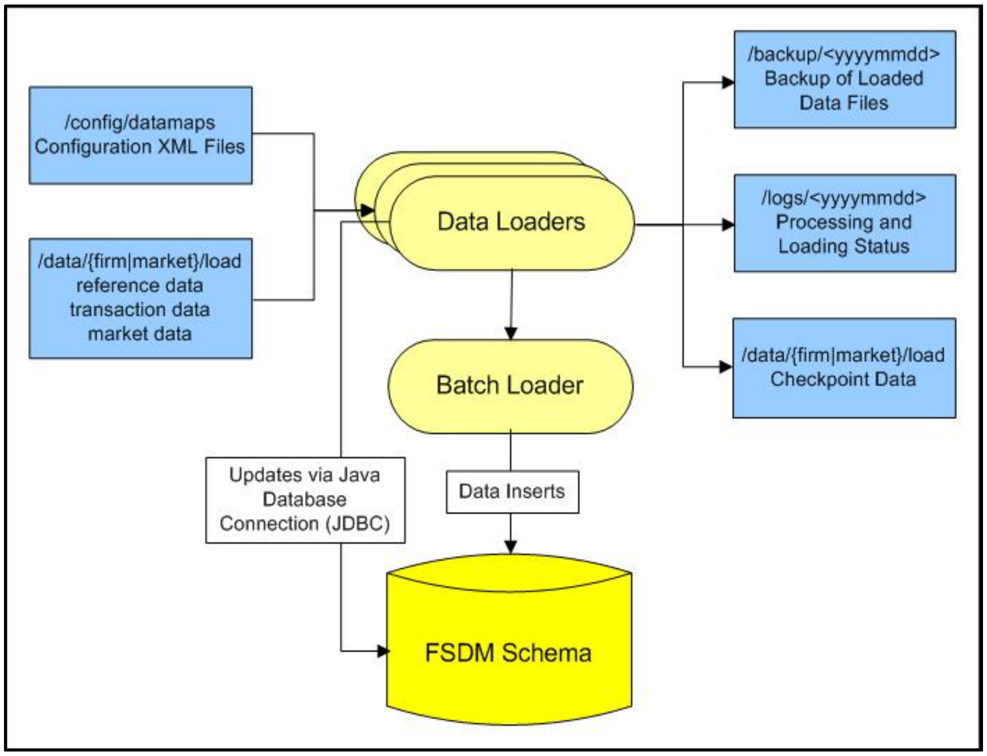


Figure 15. TC Data Loading Process

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.

Preprocessing Trading Compliance Solution Data

The Ingestion Manager preprocesses market and trading data as procedures in the following sections provide.

1. When Ingestion Manager satisfies dependencies from Group2 and preprocesses or loads the data in Group3, it executes the `runMDS.sh` script to process market data. This script invokes the Market Data server, which does the following:
   - Supports preprocessing of market data through the following mechanisms:
     - Preprocessing of queue-based equity market data from a market data I/O stream (for example, Reuters) through TIBCO.
     - Support of input of market data in flat files.
   - Assigns sequence numbers to market data records.
   - Stores market data so that Firm Data Transformer (FDT) and Market Data Transformer (MDT) can retrieve it efficiently.

   The Market Data server preprocesses market data files. The following provides a sample command:

   ```
   <INSTALL_DIR>/ingestion_manager/scripts/runMDS.sh
   ```

   This command initiates the Market Data server to process the ReportedMarketSale, InsideQuote, and MarketCenterQuote files, which the Oracle Financial Services client previously placed in the `/inbox` directory.

2. After Ingestion Manager preprocesses and loads the data in Group 2, it executes the `runDP.sh` script to process TC trading data. This script invokes:
   - Checking input trading data for errors
   - Assigning sequence IDs to records
   - Resolving cross-references to market reference data
   - Checking for missing fields or attributes

When Ingestion Manager executes `runMDS.sh`, it places output files in the directories in Table 21.

Table 21. ***runMDS.sh*** **and** ***runDP.sh*** **Output Directories**

Directory	Description
`/data/market/extract/<yyyymmdd>`	Market data intermediate files.
`/logs/<yyyymmdd>`	Preprocessing transformation and load status (in individual, date-stamped log files).
`/data/errors/<yyyymmdd>`	Records that failed validation.
`/inbox/<yyyymmdd>`	Backup of input files (for restart purposes, if necessary).

Figure 16 illustrates input and output directories for preprocessing market and trading data.

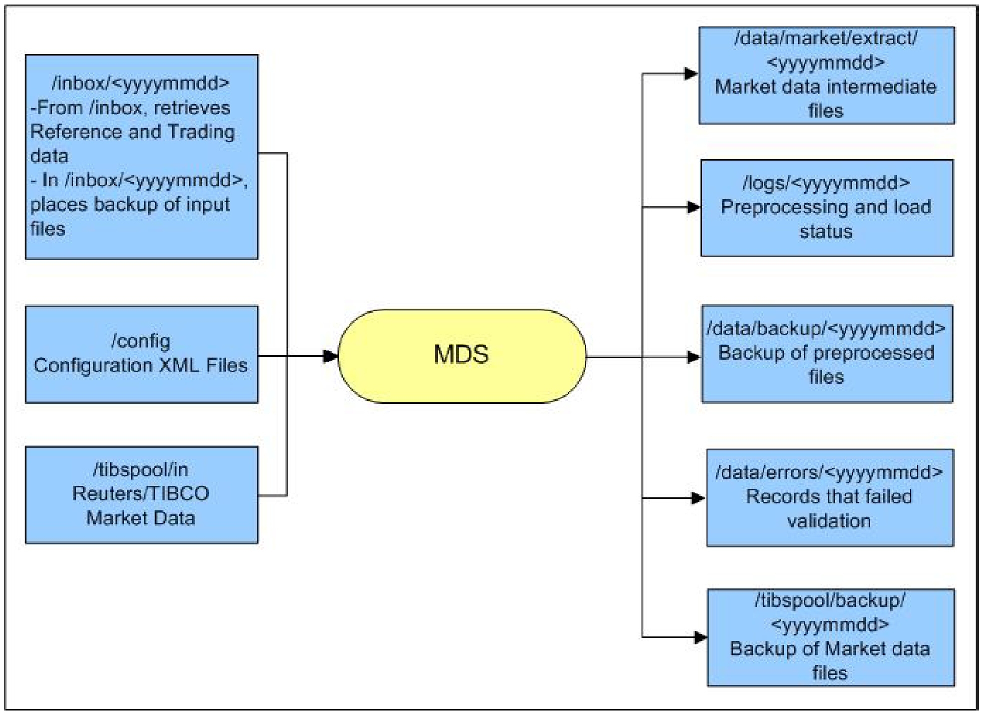

/inbox/<yyyymmdd>
-From /inbox, retrieves Reference and Trading data
- In /inbox/<yyyymmdd>, places backup of input files
/config
Configuration XML Files
/tibspool/in
Reuters/TIBCO Market Data
MDS
/data/market/extract/<yyyymmdd>
Market data intermediate files
/logs/<yyyymmdd>
Preprocessing and load status
/data/backup/<yyyymmdd>
Backup of preprocessed files
/data/errors/<yyyymmdd>
Records that failed validation
/tibspool/backup/<yyyymmdd>
Backup of Market data files

Figure 16. runMDS.sh Input and Output Directories

Preprocessing Alternative to the MDS

When ingesting market data in flat files, the Preprocessor can be used as an alternative to the MDS. The following commands can be run in parallel:

```
<INSTALL_DIR>/ingestion_manager/scripts/runDP.sh InsideQuote
<INSTALL_DIR>/ingestion_manager/scripts/runDP.sh MarketCenterQuote
<INSTALL_DIR>/ingestion_manager/scripts/runDP.sh ReportedMarketSale
```

The output of these commands will be the same as documented for the MDS. The benefit of this approach is that the Preprocessor has some performance enhancements which the MDS does not. If this alternative is used, everywhere that this document refers to the MDS, these three Preprocessors can be substituted.

Processing Data through FDT and MDT

When the Ingestion Manager completes preprocessing of TC trading data and market data, and prepared data output files, the Firm Data Transformer (FDT) and Market Data Transformer (MDT) can retrieve them.

Upon completion of data preprocessing through scripts `runDP.sh` and `runMDS.sh`, Ingestion Manager executes the `runFDT.sh` and `runMDT.sh` scripts. The `runMDT.sh` script can run as soon as `runMDS.sh` processing completes. However, `runMDS.sh` and `runDP.sh` must complete before `runFDT.sh` processing can begin.

FDT Processing

During execution of the `runFDT.sh` script, Ingestion Manager processes trade-related data, orders and executions, and trades through the Firm Data Transformer, or FDT (Figure 17). The FDT does the following:

- Enriches data.
- Produces summary records for orders and trades.
- Calculates derived values to support detection needs.
- Derives state chains (that is, order life cycle states, marketability states, and displayability states).
- Provides data for loading into FSDM schema.

The system executes the FDT with the `runFDT.sh` script; the following provides a sample command:

```
<INSTALL_DIR>/ingestion_manager/scripts/runFDT.sh
```

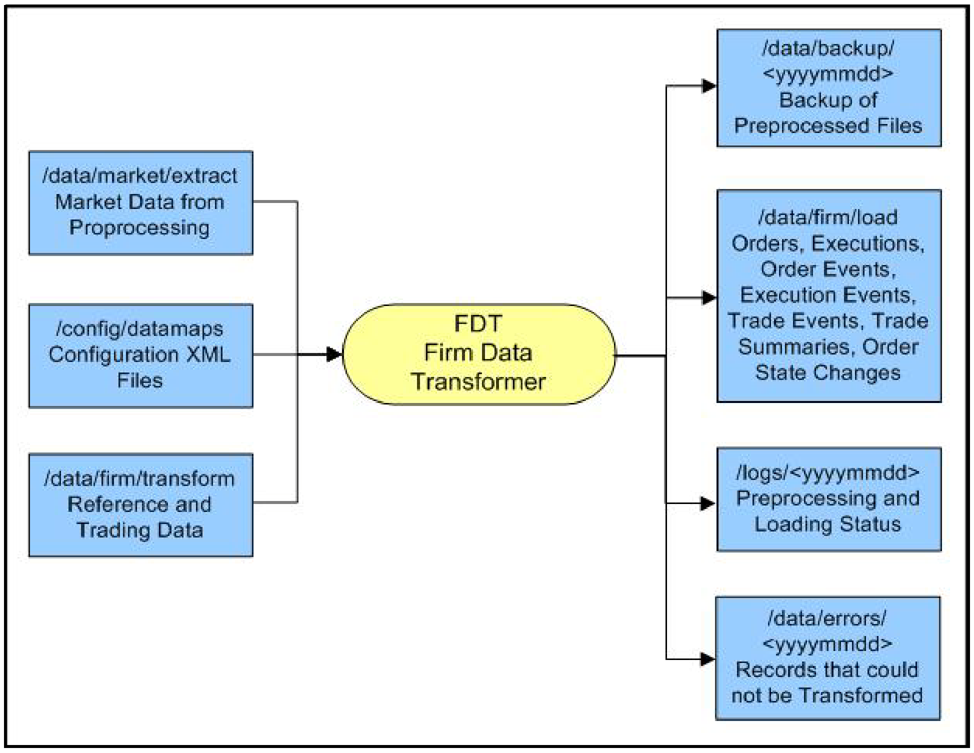

/data/market/extract Market Data from Proprocessing
/config/datamaps Configuration XML Files
/data/firm/transform Reference and Trading Data
FDT Firm Data Transformer
/data/backup/ <yyyymmdd> Backup of Preprocessed Files
/data/firm/load Orders, Executions, Order Events, Execution Events, Trade Events, Trade Summaries, Order State Changes
/logs/<yyyymmdd> Preprocessing and Loading Status
/data/errors/ <yyyymmdd> Records that could not be Transformed

Figure 17. Firm Data Transformer (FDT) Processing

The FDT:

- Processes all files that reside in the `/data/firm/transform` directory for the current date and batch.
- Terminates automatically after processing files that it found at startup.
- Ignores files that the system adds after processing begins; the system may process these files by starting FDT again, after exiting from the previous invocation.

When Ingestion Manager executes `runFDT.sh`, it places output files in the directories in Table 22.

Table 22. `runFDT.sh` Output Directories

Directory	Description
`/data/firm/transform`	Rollover data that processing saves for the next run of the FDT. Includes open and closed orders, old executions, old trades, old derived trades, lost order events, and lost trade execution events.
`/logs/<yyyymmdd>`	Status and error messages.
`/data/errors/<yyyymmdd>`	Records that the system was unable to transform.
`/data/backup/<yyyymmdd>`	Backup of preprocessed input files.
`/data/firm/load`	Transformed output files for loading into the database.

MDT Processing

During execution of the `runMDT.sh` script, Ingestion Manager processes market data (InsideQuote, MarketCenterQuote, and ReportedMarketSale) through the MDT. The Ingestion Manager also:

- Enriches data.
- Provides data for loading into FSDM schema.

The system executes the MDT with the `runMDT.sh` script; the following provides a sample command:

```
<INSTALL_DIR>/ingestion_manager/scripts/runMDT.sh
```

Figure 18 illustrates MDT data processing through `runMDT.sh`.

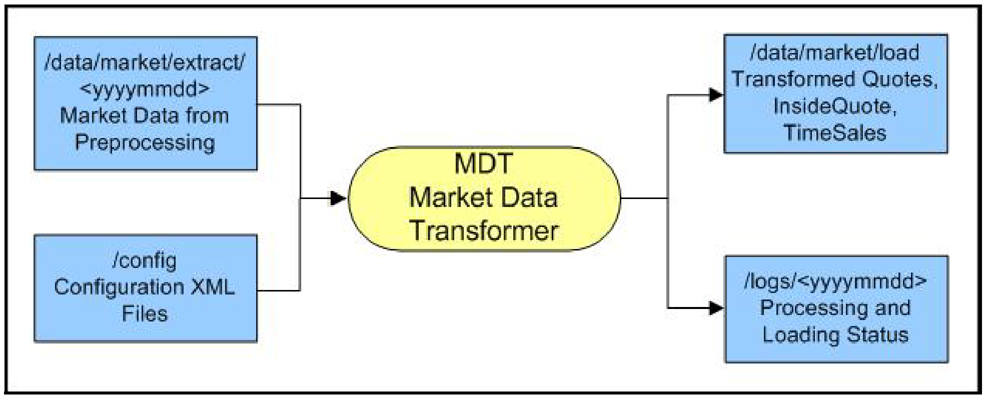

/data/market/extract/ <yyyymmdd> Market Data from Preprocessing
/config Configuration XML Files
MDT Market Data Transformer
/data/market/load Transformed Quotes, InsideQuote, TimeSales
/logs/<yyyymmdd> Processing and Loading Status

Figure 18. Market Data Transformer (MDT) Processing

When Ingestion Manager executes `runMDT.sh`, it places output files in the directories in Table 23.

Table 23. *runMDT.sh* Output Directories

Directory	Description
`/data/market/transform`	Checkpoint data kept from one run to the next.
`/logs/<yyyymmdd>`	Status and error messages.
`/data/market/load`	Transformed output files to be loaded into the database.

Running Trading Compliance Solution Data Loaders

When FDT and MDT processing complete, the system executes the `runDL.sh` script for TCS trading and market data load files. This activity loads data from the preprocessors and transformers into the FSDM schema.

The `FullRefresh` parameter in `DataIngest.xml` controls use of full refresh or delta mode. A value of <`true`> implies use of Full Refresh mode; a value of <`false`> implies use of Delta mode. Setting of the default can be to one or the other; overriding the default is possible for individual file types.

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

- **Full Refresh:** Truncates the entire table before loading the file. Use this mode when you plan to provide a complete set of records daily. You must set the `FullRefresh` parameter in `DataIngest.xml` to <`true`> to use the Full Refresh mode.
- **Delta Mode:** Updates existing data and inserts new data. Use this mode only when you plan to provide new or changed records daily. You must set the `FullRefresh` parameter in `DataIngest.xml` to <`false`> to use the Delta mode.

The system executes data loaders using the `runDL.sh` script; the following provides a sample command:

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

This command runs the data loaders for the order file that the FDT created previously.

The Ingestion Manager can execute the `runDL.sh` scripts for trading and market data simultaneously. For example, Ingestion Manager can load ReportedMarketSale, InsideQuote, MarketCenterQuote, and MarketState for market data simultaneously.

Figure 19 illustrates the Trading Compliance Solution data loading process.

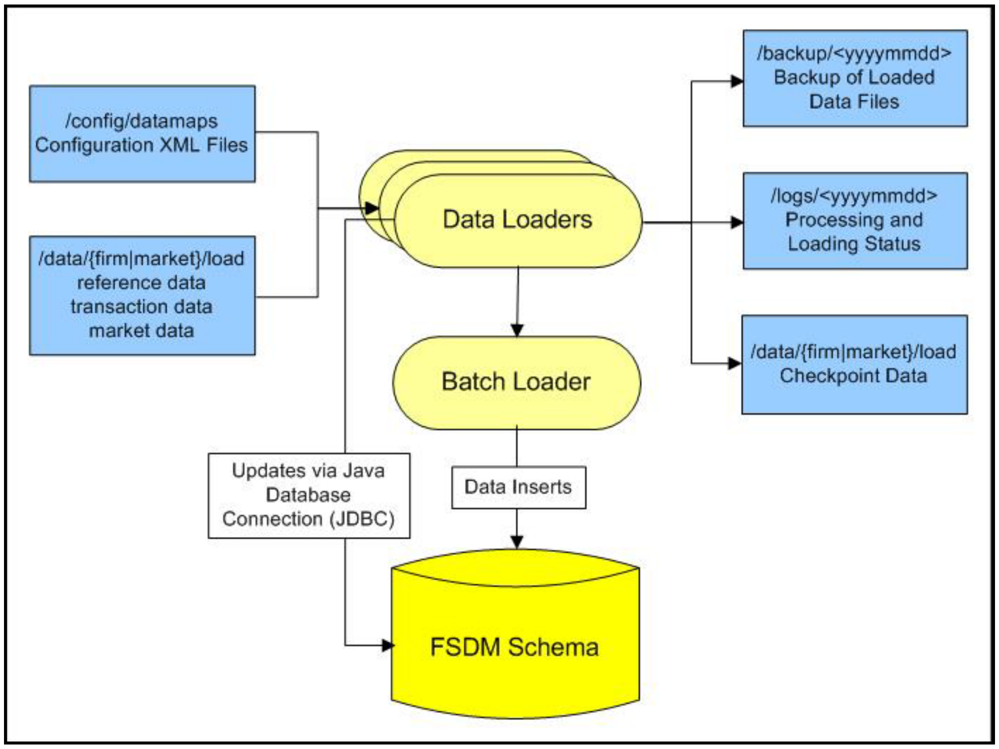

/config/datamaps Configuration XML Files
/data/{firm|market}/load reference data transaction data market data
Data Loaders
Batch Loader
Updates via Java Database Connection (JDBC)
Data Inserts
FSDM Schema
/backup/<yyyymmdd> Backup of Loaded Data Files
/logs/<yyyymmdd> Processing and Loading Status
/data/{firm|market}/load Checkpoint Data

Figure 19. TCS Data Loading Process

Rebuilding and Analyzing Statistics

When TCS market data loading is complete, Ingestion Manager does the following (Figure 12 on page 55):

1. Rebuilds database indexes by executing the `runRebuildIndexes.sh` script for the loaded market data files (refer to section *Rebuilding Indexes*, on page 69, for more information).
2. Analyzes data table characteristics (refer to section *Analyzing Statistics*, on page 69, for more information).

The following sections describe these procedures.

Rebuilding Indexes

During the data load process, Ingestion Manager drops database indexes on some tables so that use of Oracle direct-path loading can improve load performance for high-volume data. After loading is complete, Ingestion Manager rebuilds indexes using the `runRebuildIndexes.sh` script, which makes the table usable.

For example, the system executes the `runRebuildIndexes.sh` script on completion of the InsideQuote data loader:

```
<INSTALL_DIR>/ingestion_manager/scripts/runRebuildIndexes.sh
```
InsideQuote

The system then executes the `firm_analyze.sh` and `market_analyze.sh` scripts after rebuilding the indexes.

For example:

```
<INSTALL_DIR>/ingestion_manager/scripts/firm_analyze.sh
```

Analyzing Statistics

After rebuilding the indexes, Ingestion Manager uses either the `firm_analyze.sh` script (for trading data) or the `market_analyze.sh` script (for market data) to analyze data table characteristics. This activity improves index performance and creates internal database statistics about the loaded data.

Populating Market and Business Data Tables

To build and update trade and market summary data in the database, Ingestion Manager runs the `process_firm_summary.sh` and `process_market_summary.sh` scripts, as in Figure 12 on page 55.

The following examples illustrate execution of the scripts:

```
<INSTALL_DIR>/ingestion_manager/scripts/process_firm_summary.sh
<INSTALL_DIR>/ingestion_manager/scripts/process_market_summary.sh
```

After these two scripts complete processing, Data Ingestion for Trading Compliance Solution is finished.

Processing Informatica Workflows and other Utilities

When the Data Ingestion processes finish loading data into the FSDM, the Ingestion Manager performs the following tasks:

- Updates summaries of trading, transaction, and instruction activity.
- Assigns transaction and entity risk through watch list processing.
- Updates various Balances and Positions derived attributes.

Note: To successfully run Informatica workflows you must have installed Informatica. Refer to the *Oracle Financial Services Behavior Detection Platform Installation Guide*, for more information.

The system uses Informatica to perform these tasks. Figure 20 illustrates Informatica processing.

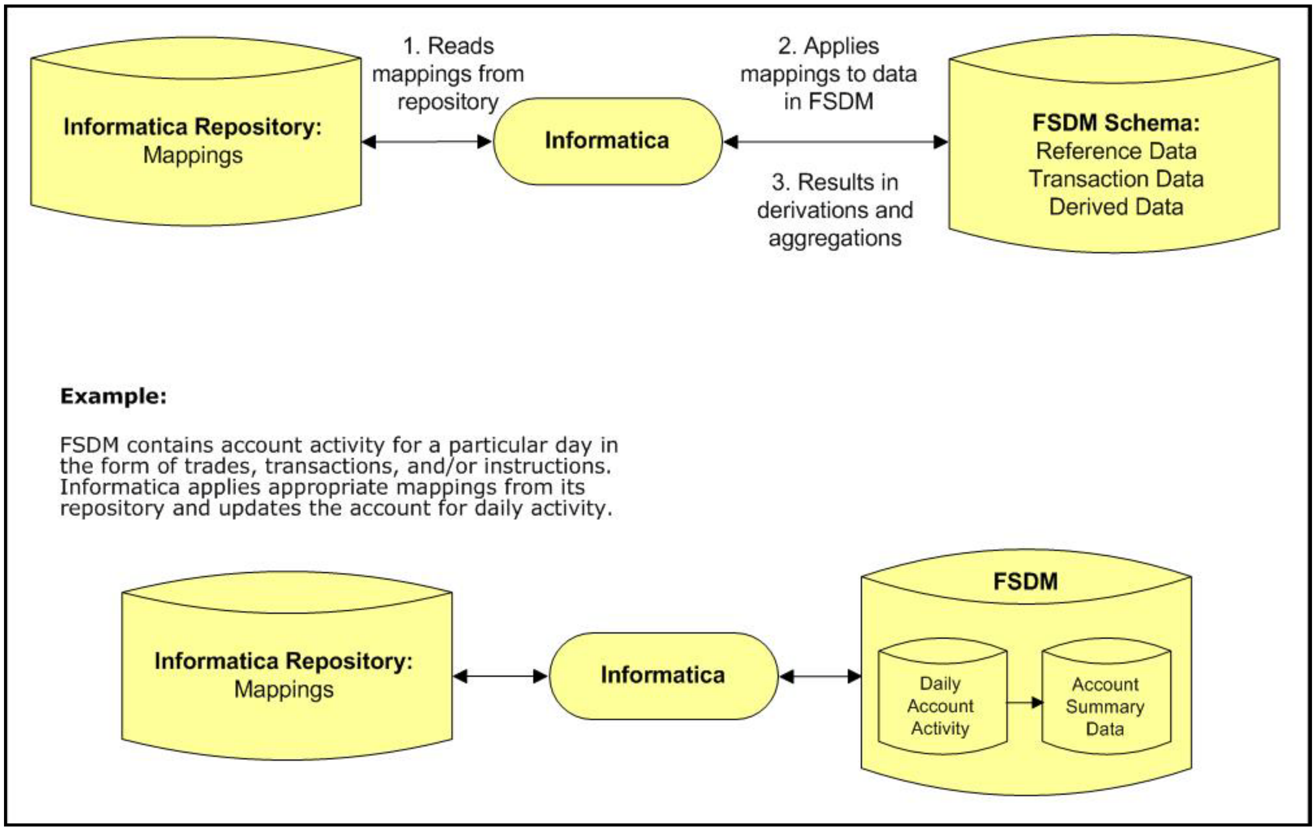

1. Reads mappings from repository
2. Applies mappings to data in FSDM
Informatica Repository: Mappings
Informatica
FSDM Schema: Reference Data Transaction Data Derived Data
3. Results in derivations and aggregations
Example:
FSDM contains account activity for a particular day in the form of trades, transactions, and/or instructions. Informatica applies appropriate mappings from its repository and updates the account for daily activity.
Informatica Repository: Mappings
Informatica
FSDM
Daily Account Activity
Account Summary Data

Figure 20. Informatica Workflow Processing

Informatica does the following:

- Reads mappings or workflows in its repository.
- Applies relevant workflows to FSDM: Reference, Transaction, and Derived data.
- Updates summary tables (Figure 21), Watch List content, and various Balance and Positions derived attributes (Figure 22).

.

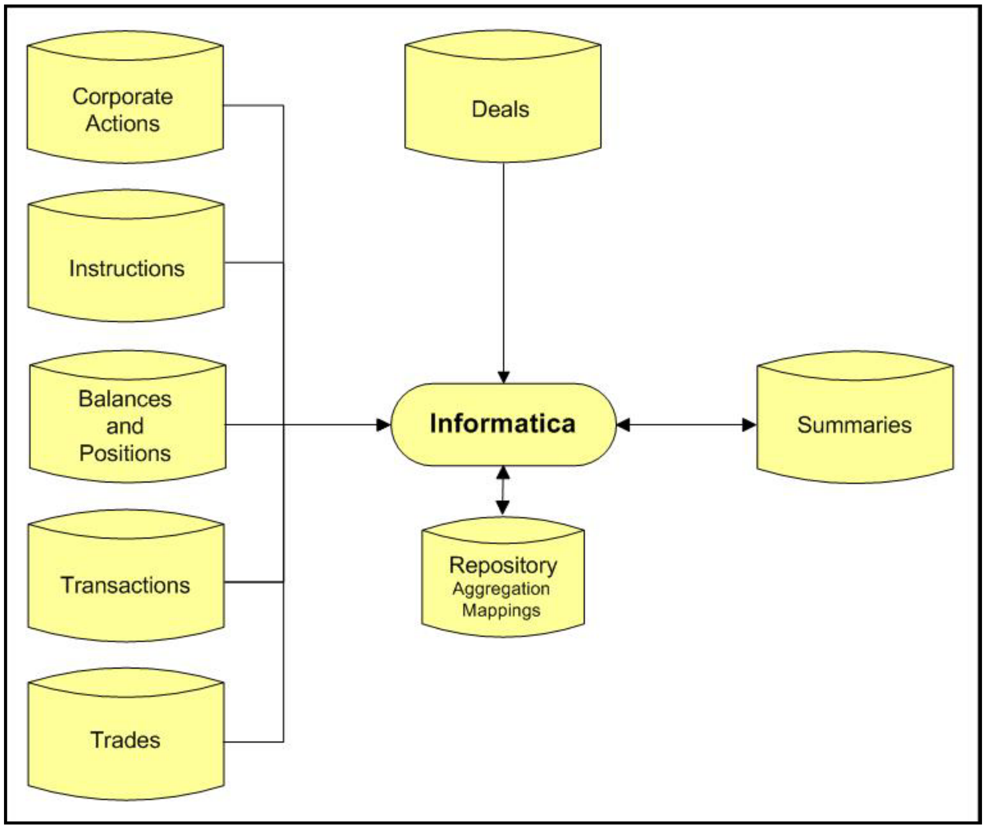


Figure 21. Informatica Summary Generation

Figure 22 illustrates Informatica Watch List processing and risk assignment.

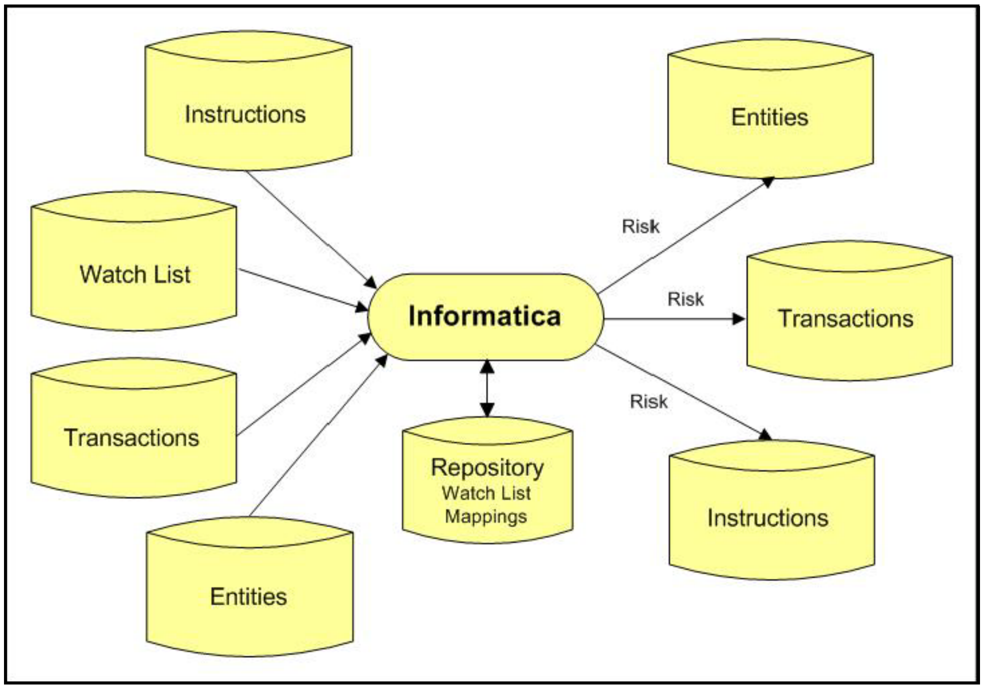


Figure 22. Informatica Watch List Processing

Refer to section *Alternatives to Standard Data Ingestion Practices*, on page 77, for more information about Watch List processing.

Intra-Day Ingestion Processing

Figure 23 provides a high-level flow of the intra-day ingestion process of extracting, transforming, and loading data.

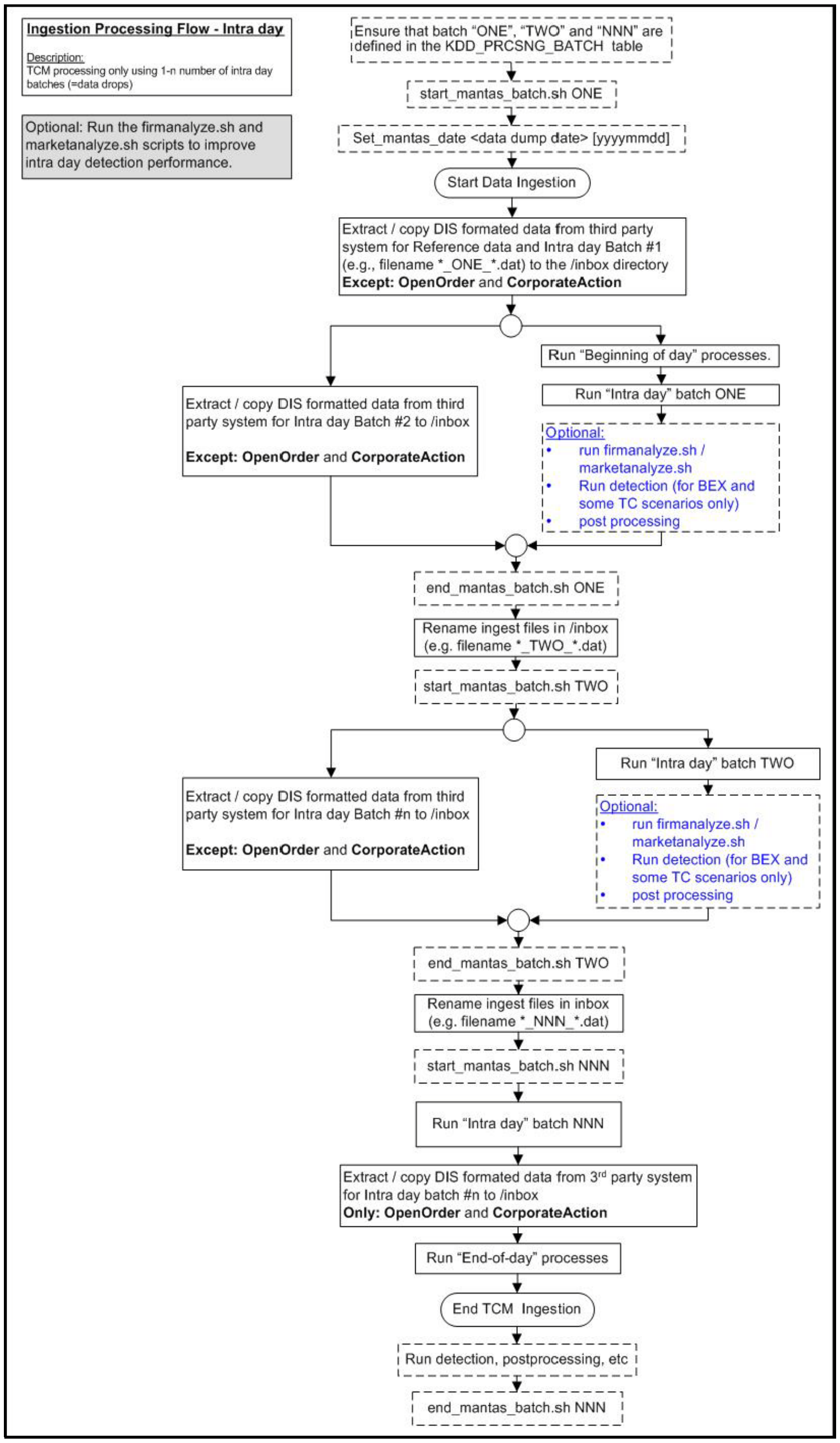

Ingestion Processing Flow - Intra day
Description:
TCM processing only using 1-n number of intra day batches (=data drops)
Optional: Run the firmanalyze.sh and marketanalyze.sh scripts to improve intra day detection performance.
Ensure that batch "ONE", "TWO" and "NNN" are defined in the KDD_PRCSNG_BATCH table
start_mantas_batch.sh ONE
Set_mantas_date <data dump date> [yyyymmdd]
Start Data Ingestion
Extract / copy DIS formated data from third party system for Reference data and Intra day Batch #1 (e.g., filename *_ONE_*.dat) to the /inbox directory Except: OpenOrder and CorporateAction
Run "Beginning of day" processes.
Run "Intra day" batch ONE
Extract / copy DIS formatted data from third party system for Intra day Batch #2 to /inbox Except: OpenOrder and CorporateAction
Optional: run firmanalyze.sh / marketanalyze.sh; Run detection (for BEX and some TC scenarios only); post processing
end_mantas_batch.sh ONE
Rename ingest files in /inbox (e.g. filename *_TWO_*.dat)
start_mantas_batch.sh TWO
Run "Intra day" batch TWO
Extract / copy DIS formatted data from third party system for Intra day Batch #n to /inbox Except: OpenOrder and CorporateAction
Optional: run firmanalyze.sh / marketanalyze.sh; Run detection (for BEX and some TC scenarios only); post processing
end_mantas_batch.sh TWO
Rename ingest files in inbox (e.g. filename *_NNN_*.dat)
start_mantas_batch.sh NNN
Run "Intra day" batch NNN
Extract / copy DIS formated data from 3rd party system for Intra day batch #n to /inbox Only: OpenOrder and CorporateAction
Run "End-of-day" processes
End TCM Ingestion
Run detection, postprocessing, etc
end_mantas_batch.sh NNN

Figure 23. Intra-Day Data Ingestion Processing

Intra-day processing references different processing groups as Figure 23 illustrates (for example, beginning-of-day processing and intra-day processing as Figure 12 illustrates). Multiple batches run throughout the day. As in Figure 23, you configure batch ONE, load and extract data, and then start processing. (Data for OpenOrder and CorporateAction is not included.) When batch ONE processing is complete, batch TWO processing begins. The same occurs for all other batches until all batch processing is complete.

You can run intra-day processing and add or omit detection runs at the end of (non end-of-day) ingestion batch runs. These cycles of detection should only run BEX and some TC scenarios. They detect only against that day's data and/or data for open batches, dependent on each scenario against which each batch is running. The last intra-day batch should be configured as the end-of-day batch.

You must run a final end-of-day batch that detects on all data loaded into the database for that day, not only looking at the batch that was last loaded. The system can display these alerts on the next day.

If you want to use either types of intra-day ingestion, you must set up intra-day batches and one end-of-day batch. If you do not, the FDT processes more market data than necessary and runs for a long period.

Table 24 provides an example of setting up the `KDD_PRCSNG_BATCH` table.

Table 24. Processing Batch Table Set-up

ONE	Intra-Day batch 1	1	NNN
TWO	Intra-Day batch 2	2	NNN
NNN	Intra-Day batch N+ end of day	3	NNN

Informatica Ingestion Parameters

Table 25 describes the default Informatica ingestion parameters that processing sets at installation. These parameters reside in the `$$PMrootDir/ParamFiles` file.

Table 25. Informatica Ingestion Parameters

File	Variable	Description
Multiple Files	`$$PARALLEL_NUM`	Controls how many sessions run in parallel (many workflows can *split* data into groups and run separate sessions in parallel). Processing uses this parameter only for performance tuning.
	`$$JURISDICTION`	Controls the jurisdiction placed on derived External Entity and Address records.
`prod_bsm.parm`	`$$PERCENTDIFF`	Helps determine how much a security must move by the end of the day to be considered a *win* or *loss*. If the security moves by less than a specified percentage, processing does not count it either way. If it moves by this percentage or more, it counts as a win or a loss, depending on whether the movement was beneficial to the account that made the trade.
`prod_mlm_brokerage.parm`	`$$InactivityMonths`	Specifies the number of months that processing aggregated to determine whether an account is inactive. If the sum of trades and transactions over this number of months is <= 3, the account is considered inactive. This setting can impact the *Escalation in Inactive Accounts* scenario. The default value is six months.
	`$$MonthsRetention`	Controls how long Behavior Detection retains non-standing instructions in the Instruction table.
`prod_orion.parm`	`$$LookBackDays`	Specifies the following: ● For sessions 720-727, set this parameter to 0. It controls how many days prior to today to examine for aggregating wire activity between pairs of entities. ● For sessions 701a and 701b, this parameter controls how many days of transactions to look across. Verify whether the new data contains reversals of prior transactions.

Table 25. Informatica Ingestion Parameters (Continued)

File	Variable	Description
`prod_orion.parm` (Continued)	`$$MIN_GEO_RISK`	Defines what is considered High Risk For the Account Profile attributes related to *High Risk Geography* (for example, Incoming High Risk Wire Count). Processing compares this parameter using a strict greater-than operation.
	`$$BASE_COUNTRY`	Sets the country for the derived institution when deriving institution records from transactions, if no address information is provided on the transaction.
	`$$PROCESS_PASS_THRU`	Identifies whether data ingestion determines if a transaction is pass-through or if the client is performing this determination. ● *Y* indicates that the Behavior Detection mapping runs. ● *N* indicates that the firm calculates this field and provides the value during ingestion.
	`$$PROCESS_SECONDARY_NAMES`	Tells data ingestion whether it should populate the secondary originator and secondary beneficiary name fields in the Front Office transaction table, or if it should leave it blank for the firm to populate. ● *Y* indicates that the Behavior Detection mapping runs. ● *N* indicates that the firm calculates this field and provides the value during ingestion.

	`$$RISK_ZONE_1_LOWER`
	`$$RISK_ZONE_1_UPPER`
	`$$RISK_ZONE_2_LOWER`
	`$$RISK_ZONE_2_UPPER`
	`$$RISK_ZONE_3_LOWER`
	`$$RISK_ZONE_3_UPPER`
	`$$RISK_ZONE_4_LOWER`
	`$$RISK_ZONE_4_UPPER`
		Defines the risk zones for Trusted Pairs functionality. Each risk zone has a lower and upper bound. The ranges of risk values within each zone are configurable but the number of risk zones shall remain at 4. If an implementation chooses not to use all Risk Zones then they can disable them by setting the risk ranges out of bounds. For example, Risk Zone 1 and Risk Zone 2 may have a lower and upper value of 0. Ingestion uses these risk zones to determine whether a party's effective risk has increased by enough points to move it to a higher risk zone and accordingly creates a new version of the trusted pair record with a status of Risk Esc Rec Cancel (RRC).

Note: On AIX and Solaris, the Informatica Ingestion process occasionally returns an invalid SQLException: java.sql.SQLException: ORA-01850: hour must be between 0 and 23, when running the `s_m_p0403_load_wl` session. The `s_m_p0403_load_wl` session runs successfully despite the reported error. This error should be ignored.

Alternatives to Standard Data Ingestion Practices

Table 26 describes alternatives to the following Data Ingestion processing options and provides advantages and disadvantages to each option:

- T+1 vs. intra-day
- Single source vs. multiple sources
- Live market data vs. file-based market data
- Truncating reference data vs. updating reference data
- Single instance vs. multiple instances

Table 26. Data Ingestion Options and Alternatives

Typical Process	Alternative Process
T+1 Process by which the current day's data becomes available on the following day or at the end of the current day. Advantages: ● Simplifies batch processing and scheduling. ● Supports truncating and reloading of reference data. ● Provides an ideal mechanism for executing AML and many TC scenarios. ● Provides support for multiple batch processing. ● Processing of data originates from one source in a single batch. Disadvantages: ● Delays availability of alerts until the following day. ● Limits the batch processing time window, especially for multi-regional data.	Intra-day Process by which Data Ingestion occurs on the day that the data becomes available. Intra-day processing focuses on collecting, transforming, and loading all Market and Firm data during market trading hours and throughout non-trading hours, if necessary. This allows viewing of detection results on the same day as generation of data. Advantages: ● Use of data in detection processing can occur on the same day (most useful for Best Execution scenarios). ● Processed data that originates from one source in multiple batches. ● Delivery of data by an Oracle client can occur in sets throughout the day, so that Behavior Detection processes the data as the system receives it. This spreads processing over a larger time period. Disadvantages: ● May take significantly longer to process a given amount of data when processing with multiple ingest cycles, as opposed to T+1 processing. ● Applies only to transactional trading and market data. ● Requires updating of reference tables, rather than truncating and reloading them. This impacts performance. ● Becomes difficult to implement and schedule in a multiple-batch environment.

Table 26. Data Ingestion Options and Alternatives (Continued)

Typical Process	Alternative Process
Single Source Process that considers all data to reside in one set of source systems for the same time. Self-containment of data sustains referential integrity across data types. Data processing occurs in a single ingestion cycle. Advantages: ● Simplifies configuration. ● Makes processing easier to monitor than that for multiple batch processing. Disadvantages: ● Eliminates support of timely delivery of data from multiple geographic regions (for example, Asia Pacific or Europe). ● Requires more processing power for a smaller batch window. This window shrinks in multi-regional deployment, as batch does not start until data from all regions is available.	Multiple Sources Multiple processes that follow the same rules for self-containment of data that single source processing follows. For example, transactions in one source are unable to reference an account in another source. Processing of each batch of data occurs in sequential batches. Behavior Detection does not allow overlap of batches. Also, you cannot combine processing of "*multiple sources*" with intra-day batch processing. Advantages: ● Spreads processing of multiple geographic regions across an entire day (which contributes to fewer hardware requirements). ● Makes alerts from a particular region available in a more timely manner. Disadvantages: ● Makes processing more complicated to configure and monitor. ● Makes completion of processing of later batches in a timely manner more difficult when a delay of an earlier batch occurs.
Live Market Data Mechanism for receiving market data from a live feed (for example, Reuters) through an application such as TIBCO. Advantages: ● Supports intra-day processing. ● Enables an Oracle client to leverage its existing live-feed infrastructure. Disadvantages: ● Requires an expensive, live-feed infrastructure if it does not already exist. ● Requires more skilled expertise to monitor and maintain.	File-based Market Data Mechanism for receiving data in flat files from an Oracle client that conform to formatting guidelines in the DIS. Advantages: ● Easier to maintain. ● Lowers total cost ownership (TCO). An Oracle client does not need a live-feed application. Disadvantages: ● Requires longer data preparation times, which delays the start of a batch cycle. ● Increases the costs of hardware and software for timely processing. ● Makes support of intra-day processing difficult due to latency in providing flat file market data.

Table 26. Data Ingestion Options and Alternatives (Continued)

Typical Process	Alternative Process
Truncate Reference Data Process for overwriting non-transactional data (for example, account information, customer lists, security symbols, etc.) on a daily basis. Truncating the data tables before loading them provides a faster mechanism for data loading. The Oracle client must provide a complete set of reference data if using this approach. Advantages: ● Supports an Oracle client that does not track updating of daily records well. ● Provides increased performance when a large percentage of records change on a daily basis. Disadvantages: ● Makes tables unavailable from the time the system truncates the data until it loads the data. ● Causes decreased performance if only a small percentage of records change on a daily basis. ● Makes multi-batch processing unsuitable, although the truncating process can be used for periodic or beginning-of-day table refresh.	Update Reference Data Mechanism that provides Behavior Detection with a set of records for updating in place in a table. A look-up of every record in the input determines whether Behavior Detection needs to update or insert data. Advantages: ● Requires a minimal amount of updates for better performance. ● Supports multiple batch processing and high data availability. Disadvantages: ● Causes decreased performance if a large percentage of records require updating. ● Requires an Oracle client to determine what records have changed from day to day.
Single Instance Mechanism by which a single instance of Data Ingestion software processes an input data stream. This configuration resides on one computer system. Advantages: ● Makes processing easier to configure and monitor. ● Simplifies the Oracle client data extraction process. Disadvantages: ● Limits performance as a result of the limitations of a single computer system, which results in a longer batch processing time.	Multiple Instances Mechanism by which configuration of multiple instances occurs on multiple computer systems. Each configuration processes a distinct data stream. Configuration spreads Data Ingestion instances across multiple computers. Advantages: ● Enables an Oracle client to scale for higher processing volume by using multiple computers. This increases bandwidth with the use of multiple I/O channels. ● Provides a lower cost structure with several smaller computers than with a single, large-capacity computer. Disadvantages: ● Makes configuration and monitoring of the ingestion process more difficult. ● Requires an Oracle client to split data prior to delivery for Data Ingestion.

Data Ingestion Directory Structure

The Data Ingestion subsystem components and data are organized in subdirectories below the `ingestion_manager` root level. Figure 24 illustrates the subdirectories that the `ingestion_manager` directory contains. Additionally, Table 27 provides details about each subdirectory.

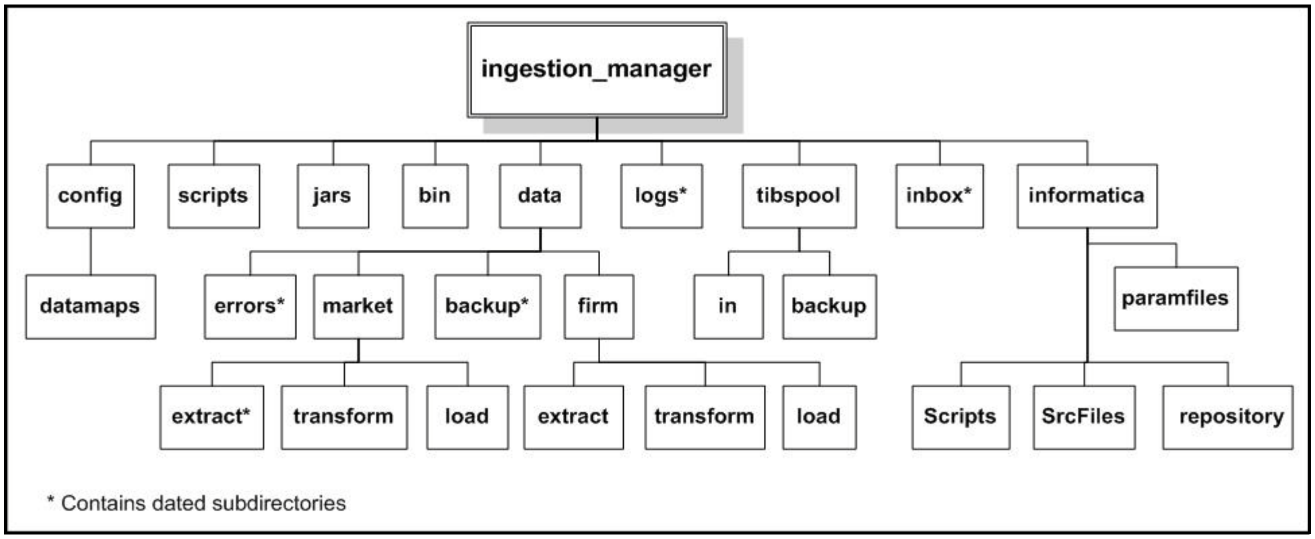


Figure 24. Data Ingestion Subsystem Directory Structure

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

Table 27. 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 82, for more information).
`config`	Contains files used to configure the Data Ingestion components (refer to *config Subdirectory*, on page 85, for more information).
`data/backup`	Contains backup files for the various Data Ingestion components (refer to *data/backup Subdirectory*, on page 105, for more information).
`data/errors`	Contains error files for various Data Ingestion components (refer to *data/errors Subdirectory*, on page 103, for more information).
`data/firm`	Contains Oracle client data files that Data Ingestion components write (refer to *data/firm Subdirectory*, on page 105, for more information).
`data/market`	Contains market data files that Data Ingestion components write (refer to *data/market Subdirectory*, on page 104, for more information).
`inbox`	Contains data files that the Oracle client provides (refer to *inbox Subdirectory*, on page 106, for more information).
`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. For more information about installation, refer to the *Oracle Financial Services Behavior Detection Platform Installation Guide*, Release 6.1.3.
`jars`	Contains the Java Archive (JAR) files to run Java Data Ingestion components implemented in Java (refer to *bin Subdirectory*, on page 82, for more information).
`logs`	Contains log files that Data Ingestion components write (refer to *logs Subdirectory*, on page 107, for more information).
`scripts`	Contains all the shell scripts for running Data Ingestion components (refer to *scripts Subdirectory*, on page 82, 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 106, 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*, on page 82, 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*, on page 82, 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 28 defines the run scripts for starting and stopping each component, and any special instructions.

Table 28. Run or Stop Scripts by Component

Script Names	Description or Special Instructions
`runTIBS.sh` `stopTIBS.sh`	Launches the TibSpool (TIBS) through the `runTIBS.sh` script or terminates it through the `stopTIBS.sh` script. Processing configures a symbol range (for example, A-C) for each instance of TIBS. This script uses the range to select the Market data that the TIBS captures. You can modify a section within the script that sets the level of detail logged from processing this script. The default is a level of *1* which is appropriate for normal processing. Use levels *2* and *3* for more detailed diagnostic logging. The command **`runTIBS.sh 1 2`** runs TibSpool to process market data range 1 (which DataIngest.xml defines) with a log level of 2. Market data ranges are set in `DataIngest.xml` (refer to *Data Ingest XML Configuration File*, on page 88, for more information).
`runTIBW.sh` `stopTIBW.sh`	Launches the Tibco Watcher (TIBW) through the `runTIBW.sh` script or terminates it through the `stopTIBW.sh` script. You can modify a section within the script that sets the level of detail logged from processing this script. The logging level specifies the level of logging detail. A level of *1* is the default. The command **`runTIBW.sh 1`** runs TibWatch with a log level of 1.
`runMDS.sh`	Launches an instance of MDS (`runMDS.sh`). The preprocessor terminates after it finishes preprocessing the data that is currently in its memory.

Table 28. Run or Stop Scripts by Component (Continued)

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. For valid component names (refer to Figure 12 on page 55). **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.
`runMDT.sh`	Launches the MDT (`runMDT.sh`). After receiving a soft-kill, the MDT terminates only when it has finished transforming all securities. You can stop the MDT immediately by using the UNIX **kill** command to stop the process ID for the Java process that is a child of the `runMDT.sh` script. **Note:** An MDT that you configure to run without polling (that is, batch mode) stops automatically when no more data remains for processing. However, running a `stopMDT.sh` script does not terminate batch processing.
`runFDT.sh`	Launches the FDT. No stop script is available for FDT; it stops after it processes all qualifying files that it finds in the `/data/firm/transform` directory at the time the process starts. The system processes an input file if the processing data and batch name are correct. You can stop the FDT immediately by using the UNIX **kill** command to stop the process ID for the Java process that is a child of the `runFDT.sh` process.
`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. For valid component names (refer to Figure 12 on page 55). **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.
`runDailySummary.sh`	Launches a process to collect daily summary information.
`runRebuildIndexes.sh <data type>`	Launches a process to rebuild the indexes of the given component. Processing requires this script only during use of a live market feed. A valid <component> value is one of InsideQuote, ReportedMarketSale, or MarketCenterQuote.
`process_firm_summary.sh`	Calls a database procedure to build summary statistics about the Oracle client (firm) data.
`process_market_summary.sh`	Calls a database procedure to build summary statistics about the Market data.
`market_analyze.sh`	Calls a database procedure to create internal database statistics for Market tables.
`firm_analyze.sh`	Calls a database procedure to create internal database statistics for Oracle client (firm) tables.

Table 28. Run or Stop Scripts by Component (Continued)

Script Names	Description or Special Instructions
`runIMC.sh`	Launches the Ingestion Manager Cleaner (IMC) utility. No stop script is available for IMC; the utility terminates after it finishes removing old data subdirectories and their contents.
`env.sh`	Contains common configuration settings required to run Data Ingestion subsystem components. The `run*.sh` and `stop*.sh` scripts use this script.
`truncate_table.sh <schema.tablename>`	Truncates a specified table in the database. Processing runs this script prior to loading reference data when an Oracle client wants to perform a full refresh of the data.
`runUtility.sh <datatype>`	Launches a Java based utility to derive the contents of a given database table. You need to run `runDL.sh <data type>` after this script has executed successfully. For example: runUtility.sh AccountProfile runDL.sh AccountProfile

The run scripts in Table 28 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 29 describes.

Table 29. 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.
`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.
`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.
`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 30 describes:

- DataIngestCustom.xml (refer to section *Data Ingest Custom XML Configuration File*, on page 86, for more information).
- DataIngest.properties (refer to section *Data Ingest Properties Configuration File*, on page 87, for more information).
- DataIngest.xml (refer to section *Data Ingest XML Configuration File*, on page 88, 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 30. 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 client regarding modification (refer to Table 31).
`DataIngest.xml`	XML configuration file that contains settings that normally remain as is (refer to Table 32).
`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 releases. Therefore, upon installation of a newer 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 version, the client can copy the older `DataIngestCustom.xml` file to `DataIngestCustom.xml` in the new installation.

Data Ingest Properties Configuration File

Table 31 describes the parameters for the `DataIngest.properties` configuration file.

Table 31. 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 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 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 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 31. DataIngest.properties File Configuration Parameters (Continued)

Property Name	Description	Example
`MDS.Adapter.RvSession.Service`	Service name for the TIBCO live market feed. Only Oracle 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 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 clients who opt to use the queue adapter to process live market data use this parameter.	tcp:7602

Data Ingest XML Configuration File

Table 32 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 32. 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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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.		
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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.		

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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 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`
`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 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.		

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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 client provides market data as flat files in accordance with the DIS.	`Reuters`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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`
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`FDT.TreatLostEventsAsErrors.value`	Identifies whether lost events found by the FDT (refer to section *Rejection During the Transformation Stage*, on page 110, 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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
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).	

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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.	

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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.	
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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`
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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`

Table 32. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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 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 33 identifies the error file signatures that each component can output to the `errors` subdirectory.

Table 33. 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 34 identifies the files that the MDT writes to subdirectories within the `load` subdirectory.

Table 34. 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 35 identifies the files that each data loader reads and loads into the database.

Table 35. 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 36 references the files that the system writes to the `backup` subdirectory, by component.

Table 36. 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 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 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 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 37 identifies sample output file names using the latter format.

Table 37. 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 38 identifies log files for each component, based on the file name's prefix.

Table 38. 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 a 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 109, for more information).

Backup Server Configuration

An Oracle 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 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 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 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 109, 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 110, 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 112, 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 database archiving process is explained in Figure 25.

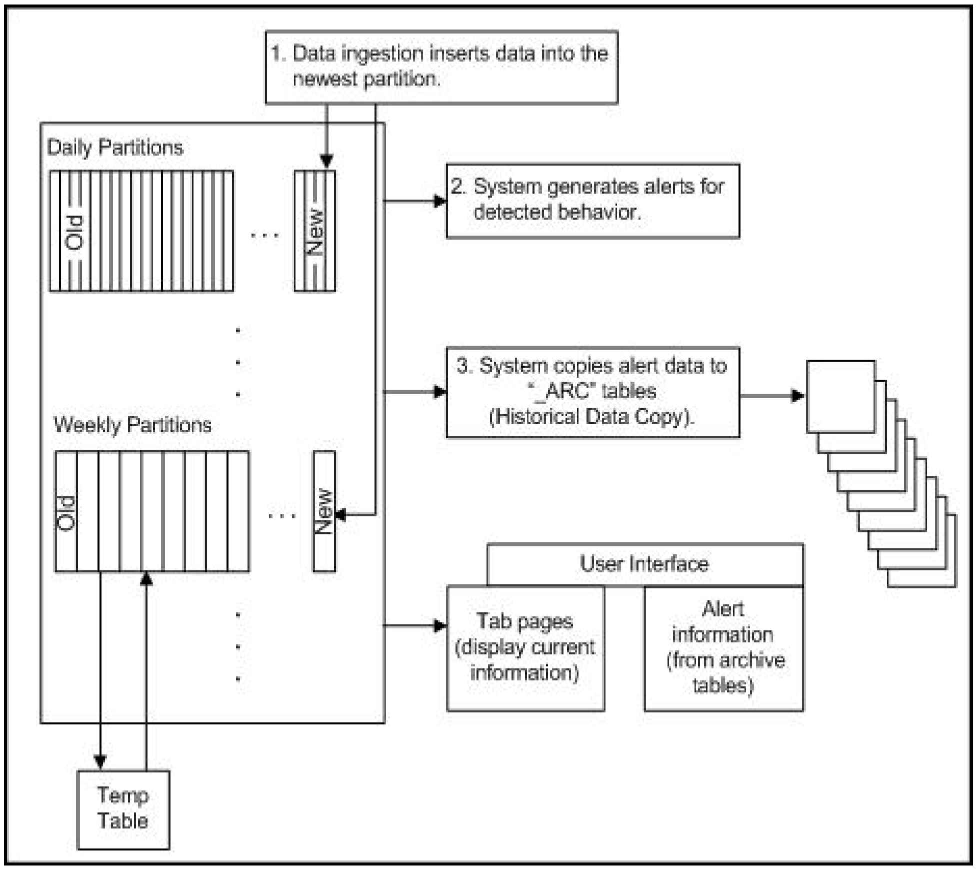

1. Data ingestion inserts data into the newest partition.
Daily Partitions
Old
New
2. System generates alerts for detected behavior.
3. System copies alert data to "_ARC" tables (Historical Data Copy).
Weekly Partitions
Old
New
User Interface
Tab pages (display current information)
Alert information (from archive tables)
Temp Table

Figure 25. Database Archiving Process

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.

For more information about the Platform user interface, refer to the *Oracle Financial Services Behavior Detection Platform User Guide.*

Miscellaneous Utilities

These utilities populate a single table in the data model. They should be executed after all the files in Table 19 have been loaded with the following exceptions:

- AccountManagementStage utility should be executed before loading BackOfficeTransaction.
- Some utilities should run in between Informatica workflows. Refer to Chapter 5, *Informatica Workflows* on page 133 for the sequence.

A utility should not be executed until all predecessors have executed successfully.

Commands to execute:

```
<INSTALL_DIR>/ingestion_manager/scripts/runUtility.sh <Utility Name>
<INSTALL_DIR>/ingestion_manager/scripts/runDL.sh <Utility Name>
```

These commands should be run serially. The utility has executed successfully only after both of these commands have successfully executed. A utility listed in the Predecessor column means that the given utility must be executed after the Predecessor utility. A utility listed in the Not In Parallel column means that the given utility can be executed before or after the Not In Parallel utility, but it cannot be executed in parallel (Table 39) .

Table 39. Miscellaneous Java based Utilities

Utility Name	Table Name	Predecessor	Not In Parallel
AccountGroupProductAllocation	ACCT_GRP_PRDCT_ALLOCATION		
AccountManagementStage	ACCT_MGMT_STAGE		
AccountPositionDerived	ACCT_POSN		AccountProfile
InterestedPartyToEmployee	INTERESTED_PARTY_EMP		
RegOToBorrower	REG_O_BORROWER		
UncoveredOptionExposureDaily	UNCVRD_OPTNS_EXPOSURE_DLY		AccountBalanceDerived
RegisteredRepresentativeProfile	RGSTD_REP_SMRY_MNTH		
AccountDailySecurityProfile	ACCT_SCRTY_SMRY_DAILY		
AccountDailyProfileTransaction	ACCT_TRXN_SMRY_DAILY		
AccountBalanceDerived	ACCT_BAL_POSN_SMRY	AccountPositionDerived	AccountProfile
AccountDailyProfileTrade	ACCT_TRADE_SMRY_DAILY	AccountDailySecurityProfile	
HouseholdBalance	HH_BAL_POSN_SMRY	AccountBalanceDerived	
InvestmentAdvisorProfile	NVSMT_MGR_SMRY_MNTH	AccountManagementStage	
AccountProfile	ACCT_SMRY_MNTH	AccountDailyProfileTrade, AccountDailyProfileTransaction	

Table 39. Miscellaneous Java based Utilities

Utility Name	Table Name	Predecessor	Not In Parallel
HouseholdProfileDerived	`HH_SMRY_MNTH`	AccountProfile	
ExternalInvestmentAccount ToAccount	`ACCT`		

Portfolio Manager Utility

The Portfolio Manager utility is used to populate the portfolio manager positions. This utility reads tables (*Account* and *Account Position*), populated while executing preprocessors (refer to *Beginning Preprocessing and Loading*, on page 57) and Informatica ingestion (refer to *Processing Informatica Workflows and other Utilities*, on page 69), and creates records to populate the `PORTFOLIO_MGR_POSN` table.

Commands to Execute:

```
runUtility.sh <Utility Name>
runDL.sh <Utility Name>
```

While executing these commands, replace `<Utility Name>` with `PortfolioManagerPosition`.

Example:

```
runUtility.sh PortfolioManagerPosition
runDL.sh PortfolioManagerPosition
```

Change Log Processing

There are two ways for the ingestion manager to create Change Log records:

- Client provides Change Log DIS files.
- Ingestion manager to generate Change Log records by comparing current day reference data records to previous day reference data records.
  There are two ways by which ingestion manager generate Change Log Records, they are:
  - Compare fields on a single reference data record that can be identified by a primary key.
    For example, an Account record can be identified by an Account Identifier. When an Account file is ingested, the Primary Customer Identifier on Account XYZ is compared to the Primary Customer Identifier currently in the database for Account XYZ. If they are different, then a Change Log record is created.
    This processing only accounts for updates to already existing records. Change Log record is not created for new reference data records or deleted reference data records.
  - Compare the set of values for a given field on several reference data records that map to a given key.
    For example, an Account Address record is identified with a combination of Account Identifier and Address Record Number.
    However, the information required is whether an Account Address record for a given Account has a field value that is different than any other Account Address record for that Account. For example, every Account Address record has a Country field. Lets say there are two Account Address records for Account XYZ in the database with values for Country of US and CN respectively. On the next day, an Account Address file is processed and there is an Account Address for Account XYZ with a value for Country of IR. A Change Log record is generated for the Country field of

this Account Address record. Futhermore, in the case of Account Address, it is not just the Account Identifier of an Account Address record that is of interest. The Address Purpose is also of interest. So when we look in the database for Account Address records that match a given Account Address record in a DIS file, we look to match both the Account Identifier field and the Address Purpose field.

This processing is controlled by configuration parameters in the ChangeLog section of `DataIngest.xml`. There is a Default subsection that defines parameters that are relevant to all data types. Each data type have their own subsection that defines relevant parameters as well as the fields that are checked for changes. Table 40 lists the Change Log parameters.

Table 40. Change Log Parameters

Parameter	Description
`class`	The Java class name to perform change log processing. For the first type of processing defined above, `ChangeLog` should be used. For the second type, `SetChangeLog` should be used.
`seqBatchSize`	Specifies the batch size when retrieving sequence ids for creating ChangeLog records.
`enabled`	Specifies whether ChangeLog processing should be done for the given data type.
`queryKey`	This is only relevant for the second type of processing (that is, when `class = SetChangeLog`). This defines the key that is used to query for reference data records matching the given one. In the Account Address example given above, the value would be AccountIdentifier, AddressPurpose. If this parameter is not present, then the primary key located in the given data type's data map file (for example `datamaps/AccountAddress.xml`) is used.
`outputKey`	This is only relevant for the second type of processing (that is, when `class = SetChangeLog`). This defines the set of fields that are mapped to Key1, Key2, Key3, and Key4 fields of a ChangeLog record. This can be different from the queryKey and primary key in order to match what is expected in ChangeLog DIS file records, and also to support Change Log Summary processing. If this parameter is not present, then the primary key located in the given data type's data map file (for example, `datamaps/AccountAddress.xml`) is used.

After these parameters are defined, each data type contains a parameter for every field that is checked for changes. If the value of the parameter is *true*, then changes are tracked for the given field. Otherwise, changes are not tracked.

ChangeLog records are produced when the preprocessor is run for the given data type (that is, `runDP.sh AccountAddress`). A ChangeLog file is produced with a name prefix containing the data type (that is, `ChangeLog_AccountAddress`). In order to load these ChangeLog files, the ChangeLog data loader is run (that is, `runDL.sh`

`ChangeLog`). This data loader is run after all preprocessors that produce ChangeLog records are complete and any ChangeLog DIS files are preprocessed (i.e. `runDP.sh ChangeLog`).

Change Log Summary Processing

There are three different parts of ChangeLog Summary scripts that must be run. They are:

- `AccountChangeLogSummary` - run the files in the following order:
  - `runUtility.sh AccountChangeLogSummary`
  - `runDL.sh AccountChangeLogSummary`
- `CustomerChangeLogSummary` - run the files in the following order:
  - `runUtility.sh CustomerChangeLogSummary`
  - `runDL.sh CustomerChangeLogSummary`
- `AccounttoCustomerChangeLogSummary` - run the files in the following order:
  - `runUtility.sh AccountToCustomerChangeLogSummary`
  - `runDL.sh AccountToCustomerChangeLogSummary`

All of these scripts must be run after the ChangeLog data loader is run (refer to *Change Log Processing*, on page 116, for more information). The `AccountChangeLogSummary` loads the `ACCT_CHG_LOG_SMRY` table, the `CustomerChangeLogSummary` loads the `CUST_CHG_LOG_SMRY` table, and the `AccounttoCustomerChangeLogSummary` loads the `CUST_ACCT_CHG_LOG_SMRY` table.

Network User Account Map

The Network User Account Map utility is used to populate the `NTWK_USER_ACCT_MAP` table. This table maps an employee to accounts in which the employee has a role. It depends on the Employee, Customer, Account To Customer, Account Customer Role, and Account Group Member DIS files being loaded with their respective `runDP/runDL` processes. The utility can be run by executing the command:

```
runNUAM.sh
```

Trade Blotter

Trade Blotter records are optionally created by the FDT and are loaded into the `KDD_TRADE_BLOTTER` and `KDD_TRADE_BLOTTER_ACTVY` tables. The FDT is configured by default to not create these records, so it needs to be configured to do so. The parameter `FDT.DeriveTradeBlotter.value` in the `DataIngestCustom.xml` file should be set to *true* to enable this functionality. These records can be loaded (after the FDT has been run) by executing the command:

```
runDL.sh TradeBlotter
runDL.sh TradeBlotterActivity
```

After all scenarios and post processing jobs have been run, an additional script needs to be run to score the trade blotter records based on the alerts that have been generated. This process updates the `KDD_TRADE_BLOTTER` table, and can be run by executing the command:

```
runScoreTradeBlotter.sh
```

Refer to *Score Trade Blotter*, on page 185, for more information.

Trusted Pair

The Trusted Pair DIS file is different from typical DIS files in that it is used to populate two separate tables, `KDD_TRUSTED_PAIR` and `KDD_TRUSTED_PAIR_MBR`. These tables can be populated by executing the commands:

```
runDP.sh TrustedPair
runDL.sh TrustedPair
runDL.sh TrustedPairMember
```

Note: Oracle Financial Services Behavior Detection supports only one method of managing trusted pairs per installation. Clients may elect to create and manage trusted pairs through the loading of trusted pairs via a DIS file or utilize the Behavior Detection user interface for creation and management of trusted pairs. However, both the methods should not be utilized concurrently.

Copying Thomson Reference Files

Behavior Detection uses the Thomson Financial Publishing US and Non-US institution lists. These lists are used to cross-reference bank identifiers of different types and different resolution (for example, FedWire, 8-digit BIC, and 11-digit BIC). The system matches banks that appear in transaction data to the Thomson data and generate the Institution Master table. Table 41 lists reference files and their sources.

If the reference files are not available, Behavior Detection derives only Institution Master records for accounts of the Firm that are part of a bank role. Thomson Financial Publishing updates these files monthly. Behavior Detection must obtain updates directly from Thomson Financial Publishing.

Table 41. Reference Files

Short Name	Description	Source	Processing Guidelines
US Banks	List of all financial institutions in the United States.	Thomson Financial Publishing Electronic Payments File (`rtsubase.txt`)	The default Informatica source file directory must contain the `rtsubase.txt` file. Behavior Detection processes any changes.
Non-US Banks	List of all financial institutions outside the United States.	Thomson Financial Publishing Global Bank Locations File (`tblabran.txt`)	The default Informatica source file directory must contain the `tblabran.txt` file. Behavior Detection processes any changes.

Copying Other Reference Files

To copy reference files other than those that Table 41 contains, use the following procedure.

To Copy Other Reference Files

To copy other reference files, follow these steps:

1. Change to the directory identified by `$PMSourceFileDir`, for example:
   `/software/informatica/pc7.1.4/MANTAS_REP/SrcFiles`
2. Copy the `file_country_update` file to this directory.
3. If you do not have the reference files from Thomson Financial, create empty files with the same names (that is, `rtsubase.txt` and `tblabran.txt`).

Fuzzy Name Matcher Utility

During Informatica workflow processing, the Fuzzy Name Matcher utility is used to match names of individuals and corporations (candidates) against a list of names (targets). The utility calculates a score that indicates how strongly the candidate name matches the target name. All matches are case-insensitive.

Although the system typically calls the utility from Informatica during Data Ingestion, you can also execute the utility through a UNIX shell script at the command line.

The Fuzzy Name Matcher Utility places its output in a delimited results file (one match per line); the utility overwrites a previous results file, if one exists. The output in the file includes:

- Match score (normalized to a number between 0 and 100)
- Delimiter
- Candidate name
- Delimiter
- Matched name on the target list

The results file contains all matches that are above a match threshold (that is, a single candidate name can match multiple names on the target list). Candidate names that do not match any target names do not appear in the file.

Logs

As the utility performs the name-matching process, it generates a log that it enters in the `<INSTALL_DIR>/ingestion_manager/fuzzy_match/log/Utilities.log` file (the logging process time-stamps all entries). The log file contains relevant error and warning information. (Refer to Appendix A, *Logging,* on page 299, for more information about these configurable files.)

You can modify the default logging configuration for this utility in the `/software/informatica/mantas/ingestion_manager/fuzzy_match/mantas_cfg/install.cfg` configuration file. Figure 26 on page 121 provides a sample file.

```
Log Configuration Items
#-------------------------------------------------------------
# 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 levels.
# Note, DIAGNOSTIC logging is used to log database statements and slows
# 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 full path and filename of the message library.
log.message.library=
# Specify where messages of a specific category should be logged.
# 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.
# The valid values are console, syslog, eventviewer, mantaslog, an e-mail
# address, or the full path to a file.
# If mantaslog is specified, the property log.mantaslog.location must be
# specified with the desired path to the logfile. If running the algorithms,
```

(Continued in next page)

Figure 26. Sample Fuzzy Name Matcher Utility `install.cfg` **File**

(Continued from previous page)

```
# the mantaslog filename has the format job<job #>-datetimestamp. For
# other subsystems, the format is mantaslog-datetimestamp.
# Note that category names cannot contain the following reserved words: fatal,
# warning, notice, diagnostic, trace, category, or location.
# Multiple locations can be listed for each property using a comma delimiter.
#log.category.EXAMPLE_CATEGORY.location=console,mantaslog

# Specify where messages of a specific severity and category should be logged
# to. The valid values are the same 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 will get logged to
# the location specified for the category by the above property, and if that
# does not exist, it will get logged to the default location configured below.
#log.EXAMPLE_CATEGORY.warning.location=syslog


# 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 displays on the console.
log.default.location=mantaslog

# Specify the location (directory path) of the mantaslog, if the mantaslog
# was chosen as the log output location anywhere above.
# Logging displays on the console if mantaslog was selected and this property is
# not given a value.
log.mantaslog.location=/tmp

# 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=
```

Using the Fuzzy Name Matcher Utility

The utility typically runs as part of automated processing that a job scheduling tool such as Maestro or Unicenter AutoSys manages. You can also execute the utility through a UNIX shell script, which the next section describes.

The following topics describe this process:

- Configuring the Fuzzy Name Matcher Utility.
- Executing the Fuzzy Name Matcher Utility.

Configuring the Fuzzy Name Matcher Utility

In addition to log configuration parameters, the `/software/ informatica/mantas/deployment/ingestion_manager/fuzzy_match/install.cfg` configuration file contains the configuration information that the Fuzzy Name Matcher Utility requires for processing. Figure 27 provides sample configuration parameters.

```
#--------------------------------------------------------------
#                Fuzzy Name Matcher Configuration Items
#--------------------------------------------------------------
fuzzy_name.match_multi=true
fuzzy_name.file.delimiter=~
fuzzy_name.default.prefix=P
fuzzy_name.max.threads=1
fuzzy_name.max.names.per.thread=1000
fuzzy_name.max.names.per.process=250000
fuzzy_name.B.stopword_file=/mantas/ingestion_manager/fuzzy_match/share/stopwords_b.dat
fuzzy_name.B.match_threshold=80
fuzzy_name.B.initial_match_score=75.0
fuzzy_name.B.initial_match_p1=2
fuzzy_name.B.initial_match_p2=1
fuzzy_name.B.extra_token_match_score=100.0
fuzzy_name.B.extra_token_min_match=2
fuzzy_name.B.extra_token_pct_decrease=50
fuzzy_name.B.first_first_match_score=1.0
fuzzy_name.P.stopword_file=/mantas/ingestion_manager/fuzzy_match/share/stopwords_p.dat
fuzzy_name.P.match_threshold=70
fuzzy_name.P.initial_match_score=75.0
fuzzy_name.P.initial_match_p1=2
fuzzy_name.P.initial_match_p2=1
fuzzy_name.P.extra_token_match_score=50.0
fuzzy_name.P.extra_token_min_match=2
fuzzy_name.P.extra_token_pct_decrease=50
fuzzy_name.P.first_first_match_score=0
```

Figure 27. Sample `install.cfg` Configuration Parameters

Table 42 describes the utility's configuration parameters as they appear in the `install.cfg` file. Note that all scores have percentage values.

Table 42. Fuzzy Name Matcher Utility Configuration Parameters

Parameter	Description
`fuzzy_name.stopword_file`	Identifies the file that stores the stop word list. The stop word file is either corporate or personal. The `<prefix>` token identifies corporate as *B* and personal as *P*. Certain words such as *Corp, Inc*, *Mr*, *Mrs*, or *the*, do not add value when comparing names.
`fuzzy_name.match_threshold`	Indicates the score above which two names are considered to match each other. The utility uses this parameter only when the `match_multi` property has a value of *`true`*. The allowable range is from 0 to 100.
`fuzzy_name.initial_match_score`	Specifies the score given for matching to an initial. The allowable range is 0 to 100; the recommended default is 75.
`fuzzy_name.initial_match_p1`	Specifies the number of token picks that must be made before awarding `initial_match_score`. The value is an integer >= 0. The default value is 2.
`fuzzy_name.initial_match_p2`	Specifies the number of token picks that must be made before awarding `initial_match_score` if only initials remain in one name. The value is an integer >= 0. The default value is 1.
`fuzzy_name.extra_token_match_score`	Indicates the score given to extra tokens. The allowable range is 0 to 100; the recommended default is 50.
`fuzzy_name.extra_token_min_match`	Specifies the minimum number of matches that occur before awarding `extra_token_match_score`. The range is any integer >= 0. The recommended setting for corporations is 1; for personal names is 2.
`fuzzy_name.extra_token_pct_decrease`	Determines the value of the `extra_token_match_score` parameter in regard to extra tokens. If multiple extra tokens are present, reduction of `extra_token_match_score` occurs for each additional extra token. The utility multiplies it by this number. For example, if `extra_token_match_score` = 50, and `extra_pct_decrease` is 50 (percent), the first extra token gets 50 percent, the second extra token gets 25 percent, the third token gets 12.5 percent, the fourth 6.25 percent, the fifth 3.125 percent, etc. The allowable range is 0 to 100. The recommended percentage for corporations is 100 (percent); for personal names, 50 (percent).
`fuzzy_name.first_first_match_score`	Allows the final score to be more heavily influenced by how well the first token of name #1 matches the first token of name #2. The allowable value is any real number >= 0. The recommended value for corporate names is 1.0; for personal names, 0.0.
`fuzzy_name.match_multi`	Determines how to handle multiple matches above the `match_threshold` value. If set to "true," the utility returns multiple matches. If set to "*`false`*," it returns only the match with the highest score.
`fuzzy_name.file.delimiter`	Specifies the delimiter character used to separate each columns in the result file and target name list file.

Table 42. Fuzzy Name Matcher Utility Configuration Parameters (Continued)

Parameter	Description
`fuzzy_name.min.intersection.first.letter.count`	Specifies the number of words per name whose first letters match. For example, if parameter value = 1 only the first letter of the first **or** last name would have to match to qualify. If the value = 2, the first letter of **both** the first and last name would have to match to qualify. **Warning:** By default, the value is set to 2. Oracle Financial Services recommends using the default value. You must not change the value to 1 or your system performance may slow down.
`fuzzy_name.default.prefix`	For entries that are not specified as business or personal name, default to this configuration set.
`fuzzy_name.max.names.per.process`	This property variable determines whether or not the fuzzy matcher algorithm will be run as a single process or as multiple sequential processes. If the total number of names between both the candidate name list and the target name list is less than the value of this property, then a single process will be run. If the number of names exceeds this property's value, then multiple processes will be run, based on how far the value is exceeded. For example, if the candidate name list contains 50 names, the target name list contains 50 names, and the fuzzy_name.max.names.per.process property is set to 200, then one process will be run (because the total number of names, 100, does not exceed 200). If the candidate list contains 400 names, the target name list contains 200 names, and the fuzzy_name.max.names.per.process property is set to 300, then four processes will be run (each with 100 candidate names and 200 target names so that the max number of names per process never exceeds 300). The ability to break apart one large fuzzy matcher process into multiple processes through this property can help to overcome per-process memory limitations imposed by certain Behavior Detection architectures.
`fuzzy_name.max.threads`	This parameter controls the number of threads to use when Fuzzy Name Matcher is being run. Oracle Financial Services recommends that this value is not set to a number higher than the number of processing cores on the system.
`fuzzy_name.max.names.per.thread`	The purpose of this parameter is to keep the processing threads balanced so that they perform work throughout the course of the fuzzy matcher job. That is, instead of splitting the number of names to process evenly across the threads, the value of this parameter can be set to a smaller batch-size of names so that threads that finish ahead of others can keep working.

Executing the Fuzzy Name Matcher Utility

To execute the Fuzzy Name Matcher Utility manually, use the following procedure.

To Execute the Fuzzy Name Matcher Utility

To execute the Fuzzy Name Matcher utility, type the following at the UNIX command line:

```
fuzzy_match.sh –t <target_name_list> -c <candidate_name_list> -r
<result_file>
```

where

- `target_name_list`: Name of the target name list file (names for which you are searching). This is a delimited file that contains the following tokens on each line:
  - Target name.
  - Delimiter, which is defined in the `install.cfg` file.
  - Business (B) or Personal (P) name.
- `candidate_name_list`: Name of the candidate name list file (names among which you are searching). This file contains one candidate name per line.
- `result_file`: Name of the results file.

  Note: The utility uses only the following characters in matching: a-z, hyphen, apostrophe, 0-9. It considers any other character as a separator between tokens. Also, the target names and candidate names must not contain delimiter characters.

If the utility completes successfully, it returns *0* at the command line; if it does not, it returns *–1*.

Using Informatica Identity Resolution API for Name Matching

The Fuzzy Name Matcher utility is used by two processes, Informatica workflows and Scan Watch List Web Service, within Behavior Detection. In Informatica workflows, the names on transactions are matched against names supplied on watch lists. In Scan Watch List Web Service, the names provided by a service caller are matched against names supplied on watch lists. This Fuzzy Name Matcher utility can be provided directly by Oracle Financial Services software (refer to section *Fuzzy Name Matcher Utility*, on page 120, for more information). It can also be provided through integration with the Informatica Identity Resolution API. Refer to Informatica Identity Resolution documentation, for information on installing and the name matching capabilities it provides.

Integration between Behavior Detection and Informatica Identity Resolution

The names being matched by both the Informatica workflows and the Scan Watch List Web Service are stored in the `WATCH_LIST` table. Informatica Identity Resolution keys are generated from these names and stored in the `SSA_PERSONAL_NAME` and `SSA_BUSINESS_NAME` tables. These keys are generated when the Scan Watch List Web Service is started (and while it is running through a polling agent) and/or an Informatica workflow (`w_ph_load_staging_fuzzy_matches`) is executed. The processes automatically generate these keys, you are not required to explicitly run these processes. Keys are only generated for names that are not stored in the SSA tables. The inactive names on watch list are not removed, however, these names are be filtered out and not matched. An excessive number of these inactive names can cause performance and storage issues. You can remove them by truncating `SSA_PERSONAL_NAME` and `SSA_BUSINESS_NAME` tables. You can use the following command to truncate the tables:

`truncate_table.sh SSA_PERSONAL_NAME` and `truncate_table.sh SSA_BUSINESS_NAME`.

These tables are refreshed from the `WATCH_LIST` table using either the Scan Watch List Web Service or the Informatica workflows.

Refer to the *Oracle Financial Services Behavior Detection Platform Installation Guide*, for information on configuring the location of the Informatica Identity Resolution software when installing Oracle Financial Services. In addition, the value of the `WatchListService/NameMatcher` parameter in `DataIngest.xml` must be changed to *`SSANameMatcher`*. This instructs the application to use the Informatica Identity Resolution integration when doing name matching in both the Informatica workflows and the Scan Watch List Web Service. `DataIngest.xml` also contains three parameters that affect how name matching is done with Informatica Identity Resolution. These are `Key Level`, `Search Level`, and `Match Level`. Collectively, they control the names returned when matching a given name.

Refresh Temporary Table Commands

Prior to running post-processing, you must execute database scripts after ingestion and prior to running AML scenarios. These scripts refresh the required temporary tables for selected AML scenario detection. Refer to section *Refreshing Temporary Tables*, on page 255, for more information.

Use of Control Data

After installing the Oracle Financial Services software, you can use control data that Oracle Financial Services Behavior Detection Platform provides to test end-to-end processing of data (that is, running data ingestion, executing scenarios, and viewing generated alerts in the Alert Management UI). This control data provides data that runs with the scenarios that you have purchased. Thus, you can verify that installation of the software is correct and works as designed.

To prepare the system for testing, follow these steps:

1. Complete the prerequisites for using control data (refer to section *Prerequisites for Using Control Data*, on page 128, for more information).
2. Prepare for ingestion of the control data (refer to section *Control Data Ingestion*, on page 129, for more information).
3. Install the control data (refer to section *Loading Control Data Thresholds*, on page 129, for more information).
4. Run Behavior Detection on control data to generate alerts (refer to section *Running Behavior Detection on Control Data*, on page 130, for more information).

Prerequisites for Using Control Data

Before you use control data to test your Behavior Detection installation, the following prerequisites must be fulfilled:

1. The maximum lookback that control data considers is of 13 months, which is for change in behavior scenarios. Hence, while creating control data ensure that it is spread over 22 different dates in 13 months.
2. The current day according to control data is 20091210.
3. Unless specified, set the current date as 20091210, to generate alerts on control data, before running Behavior Detection Platform.

Note: For more information about control data on your site, contact your Oracle Financial Services Administrator.

Control Data Ingestion

Control data uses a specific set of dates to ensure that all the scenarios are tested using this data. The maximum lookback that control data considers is of 13 months, which is for change in behavior scenarios. The control data is spread over 22 different dates in 13 months. The dates (YYYYMMDD format) being used by control data are:

Table 43. Dates used by Control Data

20081231	20091124
20090130	20091125
20090227	20091126
20090331	20091127
20090430	20091130
20090529	20091201
20090630	20091202
20090731	20091203
20090831	20091204
20090930	20091208
20091030	20091209
20091123	20091210

On all these dates, run the complete Behavior Detection batch and ingest the data for the respective date. Except for Behavior Detection and Post-Processing tasks, perform all other activities for the Control Data ingestion dates. Activities required during any Oracle Financial Services business day are - START BATCH > DRM > DATA INGESTION > BEHAVIOR DETECTION > POST PROCESSING > END BATCH.

Prior to running Behavior Detection on the control data, you must complete the following procedures.

1. Copy all control data from the golden data directory in the database subsystem (`/database/golden_data directory`) to the Ingestion Manager `/inbox` directory (refer to section *inbox Subdirectory*, on page 106, for more information).
2. Run ingestion for all the control data ingestion dates. Refer to section *Process Flow*, on page 52, for more information about the ingestion process.

Note: You need to adjust the partitions of the database tables as per the new dates, if you intend to process Control Data after the database upgrade to FSDM.

Loading Control Data Thresholds

To generate breaks on the control data, specific threshold sets and jobs are created. These threshold sets must be installed to the Behavior Detection system for use of control data and generation of test alerts.

1. Navigate to the directory
   `<INSTALL_DIR>/database/golden_data/threshold_sets`.
   This directory consists of test threshold sets of all the scenarios that are available with the OFSAAI system.
2. Execute shell script `load_tshld_set.sh`. This shell script installs the control data threshold sets for all the scenarios that are installed at your site. It also creates new jobs and template group ID's corresponding to all the scenarios installed. These template group ID's are same as the scenario ID's of installed scenarios.
3. Once the control data thresholds are installed, the system is ready for a test run, that is, generating test alerts.

Running Behavior Detection on Control Data

In order to generate alerts on the ingested control data, execute the new scenario jobs. These jobs consists of same template group ID as the scenario ID. (Refer to Chapter 2, *Behavior Detection Jobs*, on page 11, to get information regarding about running Behavior Detection Jobs.)

Important Notes

1. Run loaded scenarios with the system date as 20091210 with the following exceptions:
   a. For Portfolio Pumping scenario, the system date must be 20091204
   b. For Active Trading scenario, the system date must be 20091130
2. Check for system errors in the appropriate logs (refer to Appendix A, *Logging*, on page 299, for more information).
3. Run post-processing procedures.
4. Close the batch to enable display of alerts in the Behavior Detection UI.
5. Log in to the Behavior Detection UI with the correct user credentials.
6. Verify that you can view alerts in the UI.

The display of alerts signifies that installation of the system is correct and works as designed.

Note: The alerts that you can view depend on your user privileges.

Resetting the Environment

Once the testing is complete, you can reset the environment to its clean state by purging the test alerts (refer to section *Alert Purge Utility*, on page 215, for more on purging the alerts) and deleting the control data threshold sets. The `delete_thresholds.sh` shell script helps you to delete control data thresholds from the environment.

The `delete_thresholds.sh` script:

- is present in the `<INSTALL_DIR>/database/db_tools/bin` folder.
- takes `SCNRO_ID` (mandatory) and `TSHLD_SET_ID` (optional) as parameters.
- deletes the threshold sets that are NOT in the default Behavior Detection Product range of sequence IDs (113000000 to 117000000).

CHAPTER 5

Informatica Workflows

This chapter describes the derivation and aggregation of data through workflows in Informatica, after the Oracle Finanical Services Behavior Detection data ingestion process completes (refer to Chapter 4, *Data Ingestion,* on page 51, for more information).

This chapter focuses on the following topics:

- About Informatica Workflows
- AML Brokerage Workflows
- AML Banking Informatica Workflows
- Broker Compliance Informatica Workflows
- Fraud Detection Informatica Workflows
- Insurance Workflows

About Informatica Workflows

After the Oracle Finanical Services Behavior Detection Ingestion Manager loads the base tables, the process of deriving and aggregating data begins. The client's job scheduling system invokes Informatica workflows that perform this data manipulation. It is the client's responsibility, in consultation with Oracle Financial Services technical staff, to configure the job scheduling system for successful completion of this processing.
Processing uses the `<Informatica install directory>/Scripts/start_wkflw.sh` script to invoke individual workflows. The command includes the location and name for each workflow type that requires processing, in the following format:
<*Informatica install directory*>/Scripts/start_wkflw.sh <*folder*> <*workflow name*>
For example:
<*Informatica install directory*>/Scripts/start_wkflw.sh MLM_Brokerage_Common w_ph3041_create_addresses_from_inst

Informatica Workflow Types

The Oracle client's solution determines the required Informatica workflows, or a subset thereof:

- AML Brokerage (refer to *AML Brokerage Workflows,* on page 135, for more information).
- AML Banking (refer to *AML Banking Informatica Workflows,* on page 146, for more information).
- Broker Compliance (refer to *Broker Compliance Informatica Workflows,* on page 153, for more information).
- Fraud Detection (refer to *Fraud Detection Informatica Workflows,* on page 158, for more information).
- Insurance (refer to *Insurance Workflows,* on page 166, for more information).

Caution: If you are running multiple solutions, you must perform table comparisons to avoid running duplicate workflows.

Workflow Categories

Each workflow can include one or more of the following categories:

- Miscellaneous: Optional
- Miscellaneous: Pre-Watch List/Independent
- Watch List
- Miscellaneous: Post-Watch List
- Summary
- Balances and Positions

Note: The Informatica categories are not required for all solutions. Refer to each section in the above list for a complete list of required categories.

Workflow Processing

This chapter provides the required Informatica workflows for deriving and aggregating data based on the solution. Discussions of the workflows appear in the order that processing must execute them during data ingestion, and include figures and tables that describe each workflow. Sequence numbers that the accompanying figures and tables provide also reflect this order.

Where predecessors exist, processing of workflows cannot begin until completion of *predecessor* workflows. In the figures of this chapter, an arrow between workflows indicates a processing dependency. These dependencies may be internal to the workflow type (for example, Summary or Watch List), or external to the workflow type (for example, Summary workflow dependent on Miscellaneous workflow). Where applicable, the figures and tables describe each type of dependency.

In Figure 29, for example, processing can run workflow 2190 immediately after completion of workflow 2180 even if workflow 2070 has not completed. Execution of the workflows according to the sequence number resolves dependencies.

Workflow Tables

Table 44 provides a list of columns and a description for each column provided in the Informatica workflow tables that each section provides.

Table 44. Informatica Workflow Table Descriptions

Column	Description
Predecessor	Indicator that processing of workflows cannot begin until completion of *predecessor* workflows.
Sequence Number	Unique sequence number that indicates the order in which the workflows run. **Note:** Processing uniquely generates the sequence numbers that the figures represent. Therefore, they are subject to change in subsequent releases.
Folder	Folder location of each workflow.
Workflow Number	Unique, four-digit number that represents a particular workflow.
Workflow Name	Unique name of each workflow.

AML Brokerage Workflows

The following sections describe the Informatica workflows that are required for deriving and aggregating data for the AML Brokerage solution:

- Informatica Miscellaneous: Optional Workflows—AML Brokerage
- Informatica Miscellaneous: Pre-Watch List Workflows—AML Brokerage
- Informatica Watch List Workflows—AML Brokerage
- Informatica Miscellaneous: Post-Watch List Workflows—AML Brokerage
- Informatica Summary Workflows—AML Brokerage
- Informatica Balances and Positions Workflows—AML Brokerage

Each section provides a graphic that illustrates the workflow. An accompanying table describes the process by sequence, workflow number and name, and internal or external predecessors, if any.

Informatica Miscellaneous: Optional Workflows—AML Brokerage

Optional Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 28 illustrates Optional Miscellaneous Informatica workflows for AML Brokerage.

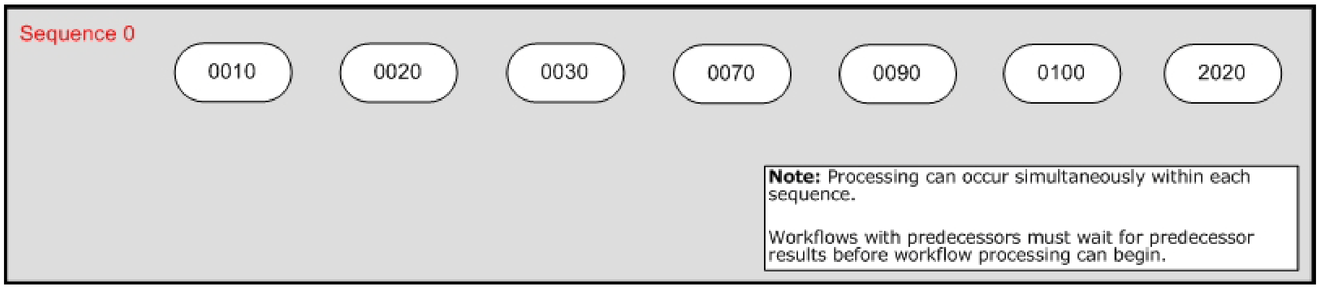

Sequence 0
0010
0020
0030
0070
0090
0100
2020
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 28. Optional Workflows—AML Brokerage

The application completes these workflows in one sequence; processing of the workflows does not require data from external predecessors.
Table 45 describes the Optional Miscellaneous workflows in Figure 28.

Table 45. Optional Workflows—AML Brokerage

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
0	ORION_Production	0010	w_p0010_dump_asm_to_file	
0	MLM_Brokerage_Production	0020	w_p0020_dump_csm_to_file	
0	MLM_Brokerage_Production	0030	w_p0030_dump_iasm_to_file	
0	ORION_Production	0070	w_p0070_reload_asm_from_dump_file	
0	MLM_Brokerage_Production	0090	w_p0090_reload_csm_from_dump_file	
0	MLM_Brokerage_Production	0100	w_p0100_reload_iasm_from_dump_file	
0	MLM_Brokerage_Production	2020	w_p2020_delete_nonstanding_instructions	

Informatica Miscellaneous: Pre-Watch List Workflows—AML Brokerage

Pre-Watch List Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 29 illustrates Pre-Watch List Miscellaneous Informatica workflows for AML Brokerage.

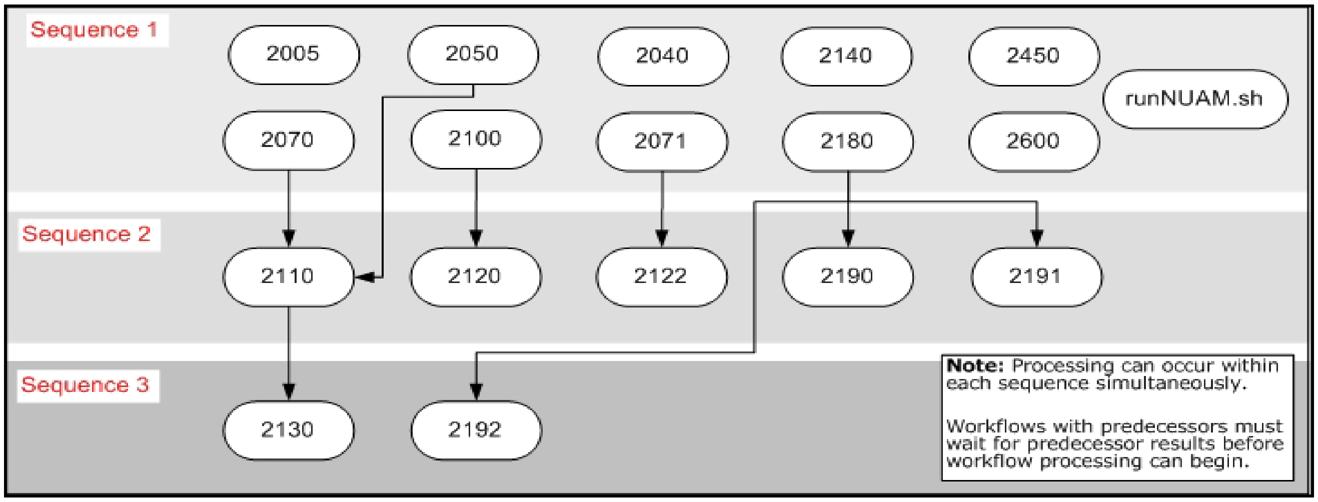

Sequence 1
2005
2050
2040
2140
2450
runNUAM.sh
2070
2100
2071
2180
2600
Sequence 2
2110
2120
2122
2190
2191
Sequence 3
2130
2192
Note: Processing can occur within each sequence simultaneously.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 29. Pre-Watch List Workflows—AML Brokerage

The application completes these workflows in three sequences; processing of the workflows does not require data from external predecessors.
Table 46 describes the Informatica Miscellaneous Pre-Watch List workflows in Figure 29.

Table 46. Pre-Watch List Workflows—AML Brokerage

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
1	ORION_Common	2005	w_p2005_set_cd_pd_dates	
1	MLM_Brokerage_Production	2040	w_p2040_update_cust_count_of_accounts	
1	ORION_Common	2050	w_ph2050_update_bot_reversals	
1	ORION_Common	2070	w_ph2070_truncate_cas	
1	MLM_Brokerage_Common	2071	w_ph2071_truncate_das	
1	ORION_Common	2140	w_ph2140_pass_thru_process	
1	ORION_Common	2180	w_ph2180_instn_identification	
1	ORION_Common	2450	w_ph2450_populate_ACCT_SRVC_TEAM	
1	ORION_Common	2600	w_ph2600_loan_smry_mnth	
2	MLM_Brokerage_Common	2122	w_ph2122_aggregate_tsv_offsetting_trades_to_tdtcs	MLM_Brokerage_Common:2071
2	ORION_Common	2190	w_ph2190_fotps_instn_processing	ORION_Common:2180
2	ORION_Common	2191	w_ph2191_anticipatory_profile_instn_processing	ORION_Common:2180
3	ORION_Common	2130	w_ph2130_update_fot_unrelated_party_code	
3	MLM_Brokerage_Common	2192	w_ph2192_update_inst_instn_seq_id	ORION_Common:2180

Informatica Watch List Workflows—AML Brokerage

Informatica Watch List workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. Figure 30 illustrates Informatica Watch List workflows for AML Brokerage.

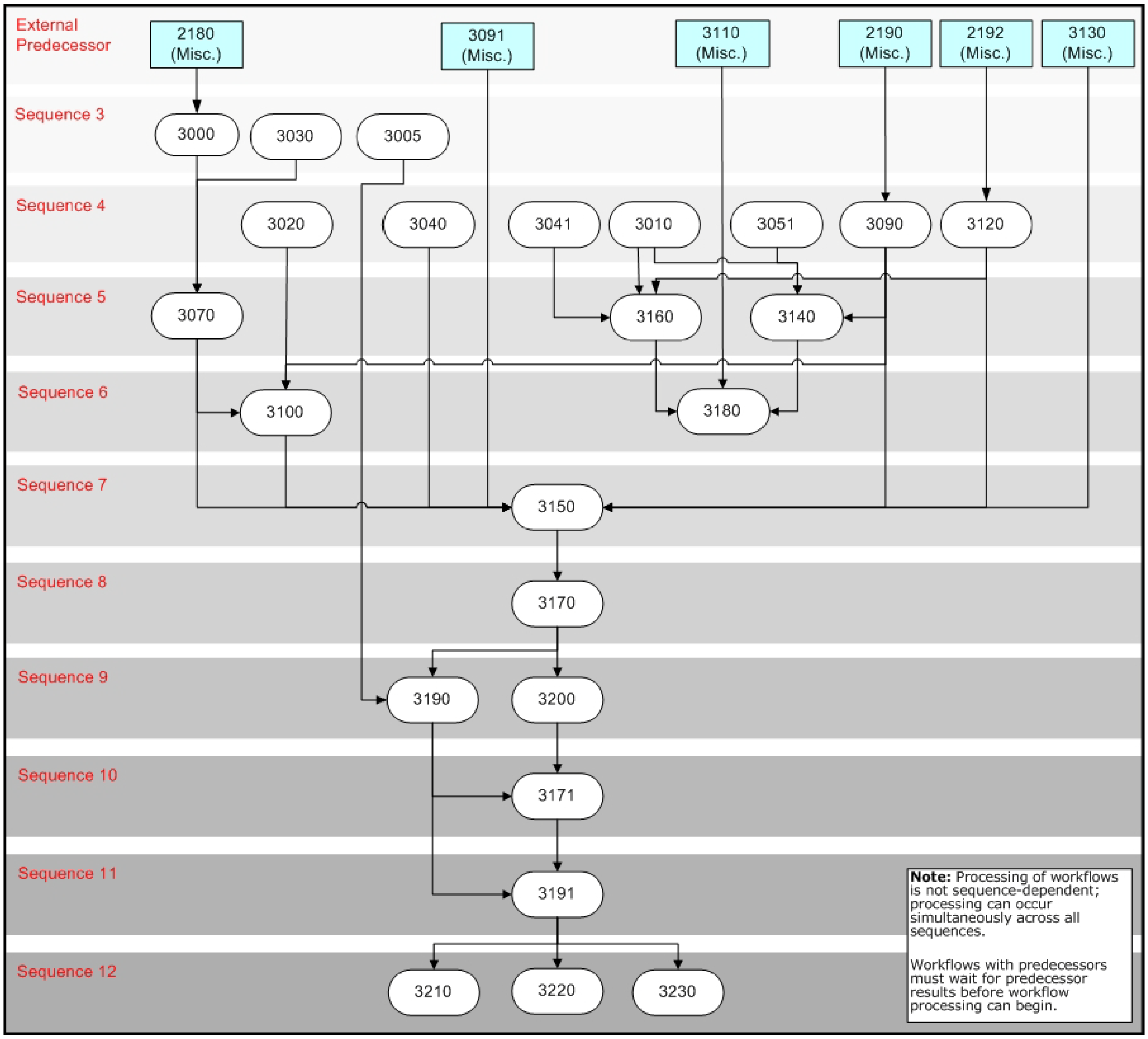


Figure 30. Watch List Workflows—AML Brokerage

The application completes these workflows in 12 sequences; processing of some workflows requires data from a number of Miscellaneous predecessor workflows. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 47 details the Watch List workflows in Figure 30.

Table 47. Watch List Workflows—AML Brokerage

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
3	ORION_Common	3000	w_ph3000_Adjust_WL_WLS	ORION_Common:2180
3	ORION_Common	3005	w_ph3005_apply_cust_KYC_risk	
3	ORION_Common	3030	w_ph3030_truncate_wls	
4	ORION_Common	3010	w_ph3010_truncate_ls	
4	ORION_Common	3020	w_ph3020_truncate_nms	
4	ORION_Common	3040	w_ph3040_truncate_wls2	
4	MLM_Brokerage_Common	3041	w_ph3041_create_addresses_from_inst	
4	ORION_Common	3051	w_ph3051_create_addresses_from_fotps	
4	ORION_Common	3090	w_ph3090_create_external_entities_from_fotps	ORION_Common:2190, ORION_Common:3091
4	MLM_Brokerage_Common	3120	w_ph3120_create_external_entities_from_inst	MLM_Brokerage_Common:2192
5	ORION_Common	3070	w_ph3070_load_watch_list_staging_table	ORION_Common:3000, ORION_Common:3030
5	ORION_Common	3140	w_ph3140_write_fotps_associations_to_ls	ORION_Common:3010, ORION_Common:3051, ORION_Common:3090
5	MLM_Brokerage_Common	3160	w_ph3160_write_inst_associations_to_ls	ORION_Common:3010, MLM_Brokerage_Common:3041, MLM_Brokerage_Common:3120
6	ORION_Common	3100	w_ph3100_load_staging_fuzzy_matches	ORION_Common:3020, ORION_Common:3070, ORION_Common:3090
6	ORION_Common	3180	w_ph3180_write_ls_to_link_tables	ORION_Common:3110, ORION_Common:3140, MLM_Brokerage_Common:3160
7	ORION_Common	3150	w_ph3150_load_staging_and_validate_watch_list	ORION_Common:3040, ORION_Common:3070, ORION_Common:3090, ORION_Common:3091, ORION_Common:3100, MLM_Brokerage_Common:3120, MLM_Banking_Common:3130

8	ORION_Common	3170	w_ph3170_update_staging_list_risk	ORION_Common:3150
9	ORION_Common	3190	w_ph3190_apply_risk_to_nonacct_entities	ORION_Common:3005, ORION_Common:3170
9	ORION_Common	3200	w_ph3200_apply_membership_to_entities	ORION_Common:3170
10	MLM_Brokerage_Common	3171	w_ph3171_update_account_customer_risk	ORION_Common:3190, ORION_Common:3200
11	ORION_Common	3191	w_ph3191_apply_risk_to_acct_entities	MLM_Brokerage_Common:3171, ORION_Common:3190
12	ORION_Common	3210	w_ph3210_update_fotps_activity_risk	ORION_Common:3191
12	MLM_Brokerage_Common	3220	w_ph3220_update_bot_activity_risk	ORION_Common:3191
12	MLM_Brokerage_Common	3230	w_ph3230_update_inst_activity_risk	ORION_Common:3191

Note: If you are running any of these combination you need to run workflows 3005 as well as 3171.

- Oracle Finanical Services Behavior Detection AML and KYC
- Oracle Finanical Services Behavior Detection Fraud and KYC
- Oracle Finanical Services Behavior Detection AML, Fraud, and KYC

Informatica Miscellaneous: Post-Watch List Workflows—AML Brokerage

Post-Watch List Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 31 illustrates Post-Watch List Miscellaneous Informatica workflows for AML Brokerage.

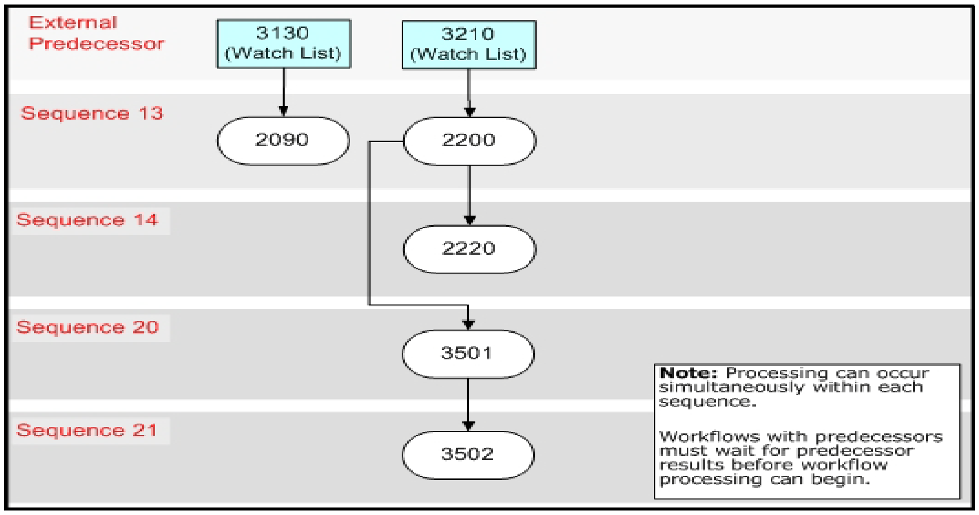

External Predecessor
3130 (Watch List)
3210 (Watch List)
Sequence 13
2090
2200
Sequence 14
2220
Sequence 20
3501
Sequence 21
3502
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 31. Post-Watch List Workflows—AML Brokerage

The application completes these workflows in four sequences; processing of workflows for one sequence requires data from an external predecessor.
Table 48 describes the Informatica Miscellaneous Post-Watch List workflows in Figure 31.

Note: For the workflow *w_ph3501_Exp_and_Risk_Review_TP*, customer can configure the *risk zones* and customize the *review reason text*.

- **Configuring Risk Zones**:

  The system compares a party's effective risk as it exists on the current version of the trusted pair member record with the party's current effective risk in the base table associated with the party ID type.

  If the party's effective risk has increased by enough points to move it to a higher *risk zone*, where risk zones are configurable ranges of risk values used to determine whether the increase in risk warrants a review of the pair, the system creates a new version of the trusted pair record with a status of Risk Esc Rec Cancel (RRC). However, if the party's risk has not increased by enough points to move it to a higher risk zone then no risk review action is initiated on the trusted pair. In any case, the party's risk will be updated on the applicable Trusted Pair member record.

  The threshold value by which an increase in effective risk will trigger a review of the trusted pair is configurable.

  The default risk zones are configured as:

  ```
  $$RISK_ZONE_1_LOWER=1
  $$RISK_ZONE_1_UPPER=3
  $$RISK_ZONE_2_LOWER=4
  $$RISK_ZONE_2_UPPER=5
  $$RISK_ZONE_3_LOWER=6
  $$RISK_ZONE_3_UPPER=7
  $$RISK_ZONE_4_LOWER=8
  $$RISK_ZONE_4_UPPER=10
  ```

  The ranges of risk values within each zone are configurable but the number of risk zones shall remain at 4. If an implementation chooses not to use all Risk Zones then they can *disable* them by setting the risk ranges out of bounds. For example, Risk Zone 1 and Risk Zone 2 may have a lower and upper value of 0.

- **Customizing Review Reason Text**:

  Where the party's effective risk has increased by enough points to move it to a higher *risk zone*, the system also records the reason for marking the record for review. This is done using the `TP_REVIEW_REASON_TX_PARTY1` and `TP_REVIEW_REASON_TX_PARTY2` parameters.

  Sample strings currently used for *review reason text* are as follows:

  `$$TP_REVIEW_REASON_TX_PARTY1=`Recommend Cancel - risk of <Party1> increased from <A> to <B>

  `$$TP_REVIEW_REASON_TX_PARTY2=` and risk of <Party2> increased from <C> to <D>

  The string for Review Reason Text parameters is translatable. You can change these strings except the values in angular brackets like <Party1>, <A>, <B>, <Party2>, <C>, and <D>.

If the system determines that the Trusted Pair record that has experienced a *threshold triggering risk increase* is still in a Risk Escalated Recommend Cancel (RRC) state (that is, a Supervisor has not reviewed the recommendation), the system appends the *new review reason text* to the *existing reason text* on the current Recommend Cancel version of the Trusted Pair record. A semi-colon (;) and a single space is used as the method of appending.

Note: While appending a *new review reason text* to the *existing text*, the system finds that appending text will result in the field exceeding 2500 characters. In this case, the system will overwrite the existing review reason text on the current Rec Cancel version of the Trusted Pair record with the current review reason text.

The above mentioned parameters for configuring *risk zones* and customizing *review reason text* are located in the `prod_orion.parm` file under the section *s_m_p1502_Risk_Review_Trusted_Pair*. Risk review only happens if `managing_tp_from_ui` is set to Y in the `installMantas.properties.sample` properties file.

Table 48. Post-Watch List Workflows—AML Brokerage

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
13	ORION_Common	2172	w_ph2172_update_jurisdiction_in_ag	
13	ORION_Common	2200	w_ph2200_build_trxn_tables_from_fotps	ORION_Common:3051, ORION_Common:3210
14	ORION_Common	2220	w_ph2220_update_fot_reversals	ORION_Common:2200
20	ORION_Common	3501	w_ph3501_Exp_and_Risk_Review_TP	ORION_Common:2200
21	ORION_Common	3502	w_ph3502_Flag_Trusted_Trxn	ORION_Common:3501

Informatica Summary Workflows—AML Brokerage

Informatica Summary workflows maintain monthly aggregations of customer activity. Figure 32 illustrates Informatica Summary workflows for AML Brokerage.

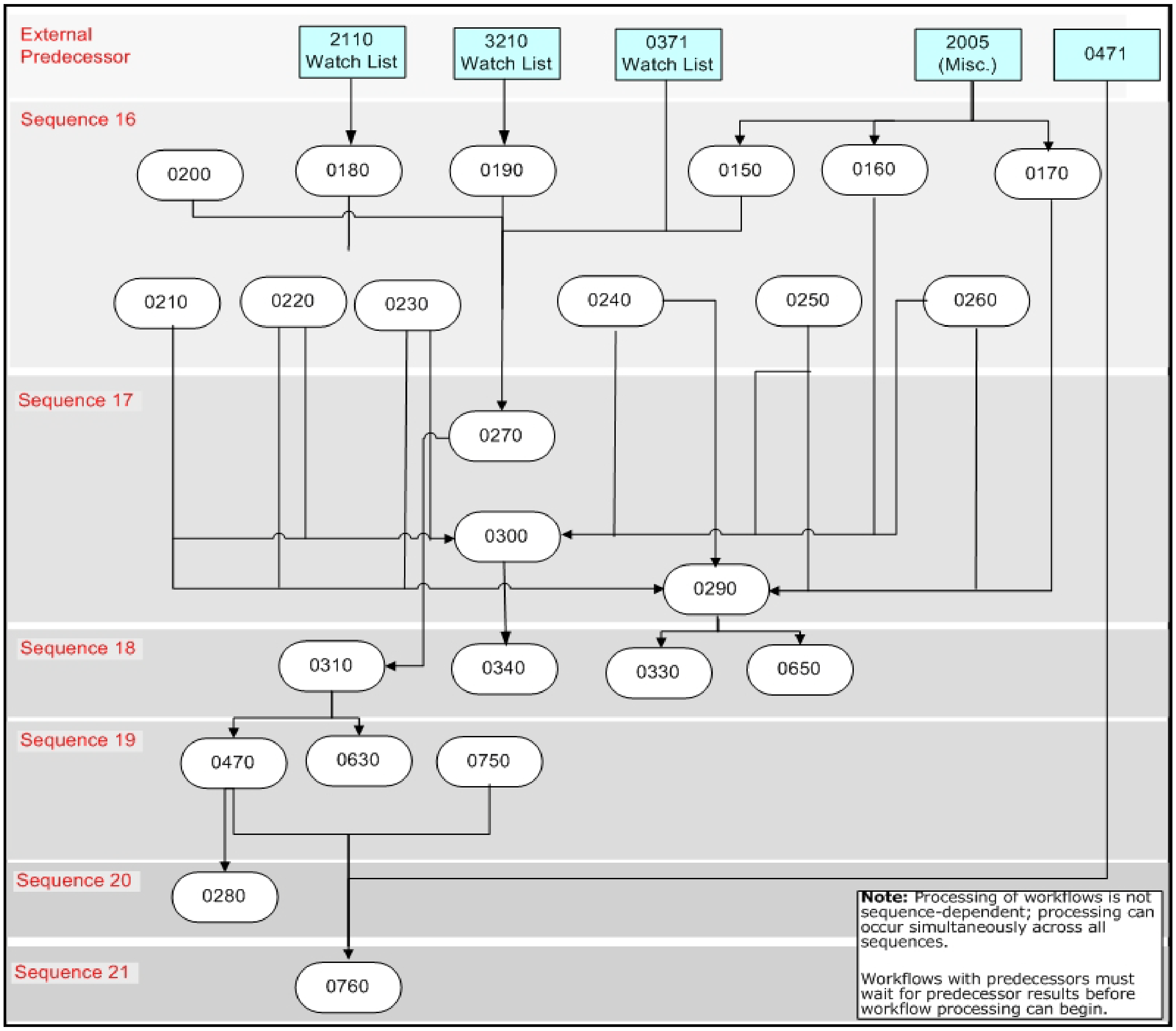


Figure 32. Summary Workflows—AML Brokerage

The application completes these workflows in six sequences; processing of workflows for some sequences requires data from external predecessors (refer to *Informatica Miscellaneous: Optional Workflows—AML Brokerage,* on page 136, and *Informatica Watch List Workflows—AML Brokerage,* on page 138, for more information).

These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.

Table 49 describes the Informatica Summary workflows in Figure 32.

Table 49. Summary Workflows—AML Brokerage

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
16	ORION_Production	0150	w_p0150_truncate_asms	ORION_Common:2005
16	MLM_Brokerage_P roduction	0160	w_p0160_truncate_csms	ORION_Common:2005
16	MLM_Brokerage_P roduction	0170	w_p0170_truncate_iasms	ORION_Common:2005
16	MLM_Brokerage_P roduction	0210	w_p0210_aggregate_bot_to_file	
16	MLM_Brokerage_P roduction	0220	w_p0220_aggregate_fotps_to_file	
16	MLM_Brokerage_P roduction	0230	w_p0230_aggregate_deals_to_file	
16	MLM_Brokerage_P roduction	0240	w_p0240_aggregate_instructions_to_file	
16	MLM_Brokerage_P roduction	0250	w_p0250_aggregate_trades_with_ca_to_file	
16	MLM_Brokerage_P roduction	0260	w_p0260_aggregate_trade_to_file	
16		9001	AccountDailyProfileTrade	
16		9002	AccountDailyProfileTransaction	
17		9005	AccountProfile	9001, 9002
17	ORION_Production	0190	w_p0190_aggregate_fotps_to_asms	ORION_Common:3210, ORION_Production:0150
17	MLM_Brokerage_P roduction	0290	w_p0290_join_instl_acct_activity_to_iasms	MLM_Brokerage_Production:0170, MLM_Br okerage_Production:0210, MLM_Brokerag e_Production:0220, MLM_Brokerage_Production:0230, MLM_Brokerage_Production:0240, MLM_Brokerage_Production:0250, MLM_Brokerage_Production:0260
17	MLM_Brokerage_P roduction	0300	w_p0300_join_instl_acct_activity_to_csms	MLM_Brokerage_Production:0160, MLM_Br okerage_Production:0210, MLM_Brokerag e_Production:0220, MLM_Brokerage_Production:0230, MLM_Brokerage_Production:0240, MLM_Brokerage_Production:0250, MLM_Brokerage_Production:0260
18	ORION_Production	0310	w_p0310_update_asm_for_daily_activity	9005, ORION_Production:0190
18	MLM_Brokerage_P roduction	0330	w_p0330_update_iasm_for_daily_activity	MLM_Brokerage_Production:0290
18	MLM_Brokerage_P roduction	0340	w_p0340_update_csm_for_daily_activity	MLM_Brokerage_Production:0300
18	ORION_Production	0650	w_p0650_build_IASD_from_ASMS	MLM_Brokerage_Production:0290
19	ORION_Production	0630	w_p0630_build_ATxSD_from_ASMS	ORION_Production:0310
19	ORION_Production	0750	w_ph0750_truncate_APTxSM	
19	MLM_Brokerage_P roduction	0280	w_p0280_create_clog_activity_records	ORION_Production:0310

20	ORION_Production	0760	w_ph0760_ASM_to_APTxSM	ORION_Production:0750

Informatica Balances and Positions Workflows—AML Brokerage

Informatica Balances and Positions workflows derive attributes that are useful in assessment of the financial status of an account, customer, or household. Figure 33 illustrates Informatica Balances and Positions workflows for AML Brokerage.

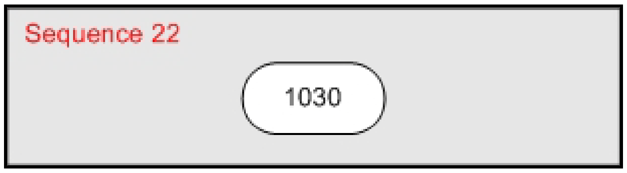


Figure 33. Balances and Positions Workflows—AML Brokerage

The application completes this workflow in one sequence; processing of the workflows does not require data from external predecessors.
Table 50 describes the Balances and Positions workflows in Figure 33.

Table 50. Balances and Positions Workflows—AML Brokerage

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
22	MLM_Brokerage_Production	1030	w_p1030_update_cbps_from_deal	

AML Banking Informatica Workflows

The following sections describe the required Informatica workflows for deriving and aggregating data for the AML Banking solution:

- Informatica Miscellaneous: Optional Workflows—AML Banking
- Informatica Miscellaneous: Pre-Watch List Workflows—AML Banking
- Informatica Watch List Workflows—AML Banking
- Informatica Miscellaneous: Post-Watch List Workflows—AML Banking
- Informatica Summary Workflows—AML Banking

Informatica Miscellaneous: Optional Workflows—AML Banking

Optional Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 34 illustrates Informatica Miscellaneous Optional workflows for AML Banking.

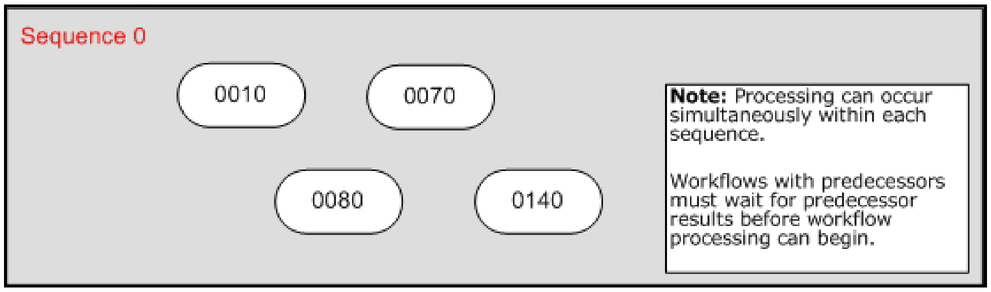

Sequence 0
0010
0070
0080
0140
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 34. Optional Workflows—AML Banking

The application completes these workflows in one sequence; processing of the workflows does not require data from external predecessors. Table 51 describes the Informatica Miscellaneous Optional workflows in Figure 34.

Table 51. Optional Workflows—AML Banking

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
0	ORION_Production	0010	w_p0010_dump_asm_to_file	
0	ORION_Production	0070	w_p0070_reload_asm_from_dump_file	
0	MLM_Banking_Production	0080	w_p0080_dump_cbsm_to_file	
0	MLM_Banking_Production	0140	w_p0140_reload_cbsm_from_dump_file	

Informatica Miscellaneous: Pre-Watch List Workflows—AML Banking

Pre-Watch List Informatica Miscellaneous workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. Figure 35 illustrates Informatica Pre-Watch List Miscellaneous workflows for AML Banking.

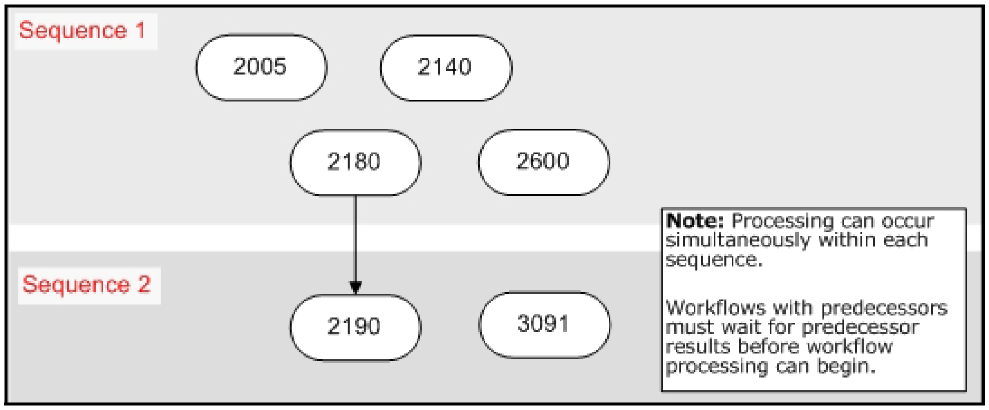

Sequence 1
2005
2140
2180
2600
Sequence 2
2190
3091
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 35. Independent Pre-Watch List Workflows—AML Banking

The system completes these workflows in two sequences; processing of the workflows does not require data from external predecessors. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 52 describes the Informatica Miscellaneous Independent Pre-Watch List workflows in Figure 35.

Table 52. Independent Pre-Watch List Workflows—AML Banking

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
1	ORION_Common	2005	w_p2005_set_cd_pd_dates	
1	ORION_Common	2140	w_ph2140_pass_thru_process	
1	ORION_Common	2180	w_ph2180_instn_identification	
1	ORION_Common	2600	w_ph2600_loan_smry_mnth	
2	ORION_Common	2190	w_ph2190_fotps_instn_processing	ORION_Common:2180
2	ORION_Common	3091	w_ph3091_create_PartyNmEE_from_fotps	

Informatica Watch List Workflows—AML Banking

Informatica Watch List workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. Figure 36 illustrates Informatica Watch List workflows for AML Banking.

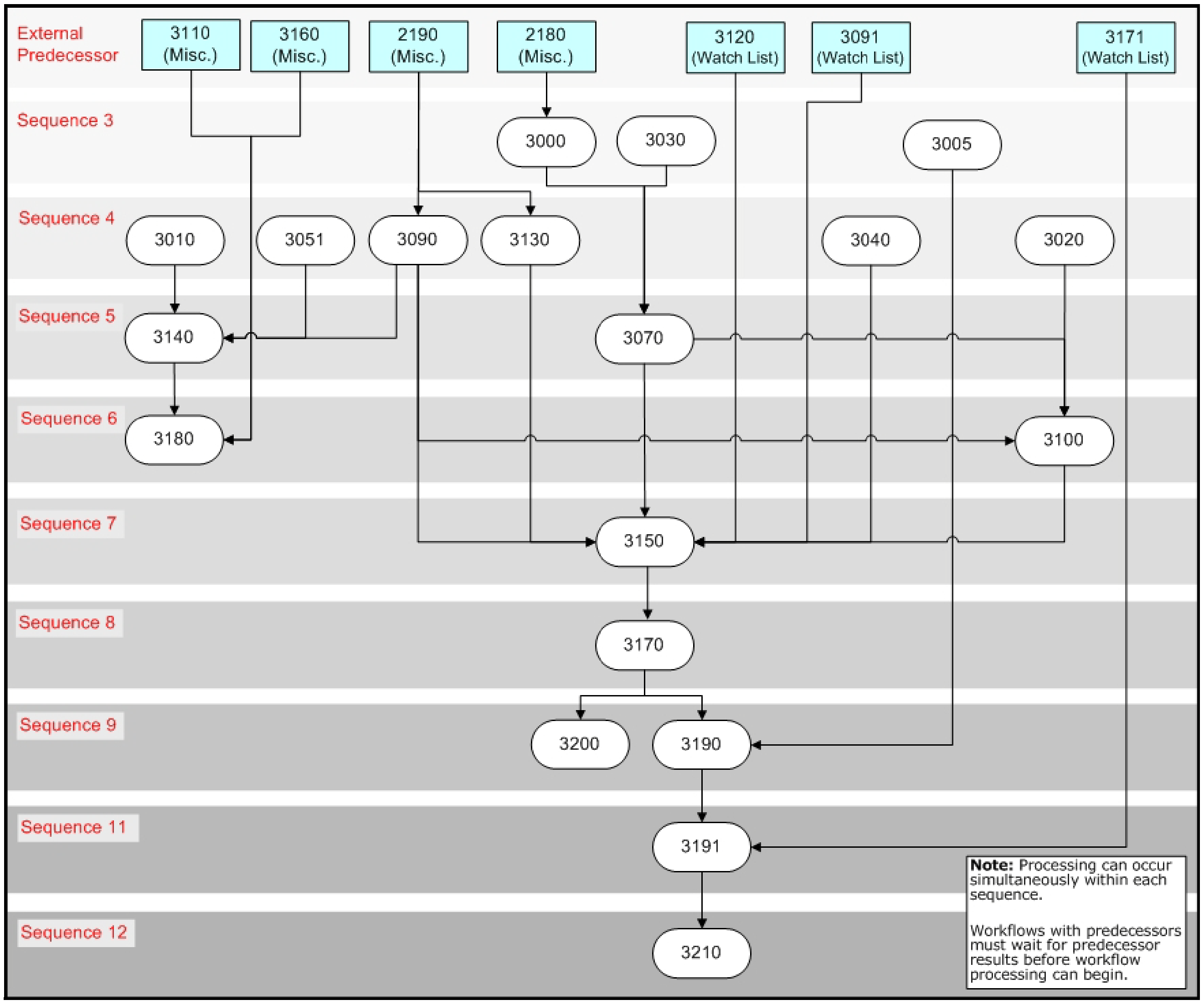


Figure 36. Watch List Workflows—AML Banking

The application completes these workflows in nine sequences. Processing of some workflows requires data from Miscellaneous predecessors (refer to *Informatica Miscellaneous: Optional Workflows—AML Banking,* on page 146, for more information). These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 53 describes the Informatica Watch List workflows in Figure 36.

Table 53. Watch List Workflows—AML Banking

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
3	ORION_Common	3000	w_ph3000_Adjust_WL_WLS	ORION_Common:2180
3	ORION_Common	3005	w_ph3005_apply_cust_KYC_risk	
3	ORION_Common	3030	w_ph3030_truncate_wls	
4	ORION_Common	3010	w_ph3010_truncate_ls	
4	ORION_Common	3020	w_ph3020_truncate_nms	
4	ORION_Common	3040	w_ph3040_truncate_wls2	
4	ORION_Common	3051	w_ph3051_create_addresses_from_fotps	
4	ORION_Common	3090	w_ph3090_create_external_entities_from_fotps	ORION_Common:2190, ORION_Common:3091
4	MLM_Banking_Com mon	3130	w_ph3130_create_client_banks_from_fotps	ORION_Common:2190
5	ORION_Common	3070	w_ph3070_load_watch_list_staging_table	ORION_Common:3000, ORION_Common:3030
5	ORION_Common	3140	w_ph3140_write_fotps_associations_to_ls	ORION_Common:3010, ORION_Common:3051, ORION_Common:3090
6	ORION_Common	3100	w_ph3100_load_staging_fuzzy_matches	ORION_Common:3020, ORION_Common:3070, ORION_Common:3090
6	ORION_Common	3180	w_ph3180_write_ls_to_link_tables	ORION_Common:3110, ORION_Common:3140, MLM_Brokerage_Common:3160
7	ORION_Common	3150	w_ph3150_load_staging_and_validate_watch_list	ORION_Common:3040, ORION_Common:3070, ORION_Common:3090, ORION_Common:3 091, ORION_Common:3100, MLM_Brokerage_Common:3120, MLM_Banking_Common:3130
8	ORION_Common	3170	w_ph3170_update_staging_list_risk	ORION_Common:3150
9	ORION_Common	3190	w_ph3190_apply_risk_to_nonacct_entities	ORION_Common:3005, ORION_Common:3170
9	ORION_Common	3200	w_ph3200_apply_membership_to_entities	ORION_Common:3170
11	ORION_Common	3191	w_ph3191_apply_risk_to_acct_entities	MLM_Brokerage_Common:3171, ORION_Comm on:3190
12	ORION_Common	3210	w_ph3210_update_fotps_activity_risk	ORION_Common:3191

Informatica Miscellaneous: Post-Watch List Workflows—AML Banking

Informatica Miscellaneous Post-Watch List workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. Figure 37 illustrates Informatica Miscellaneous Post-Watch List workflows for AML Banking.

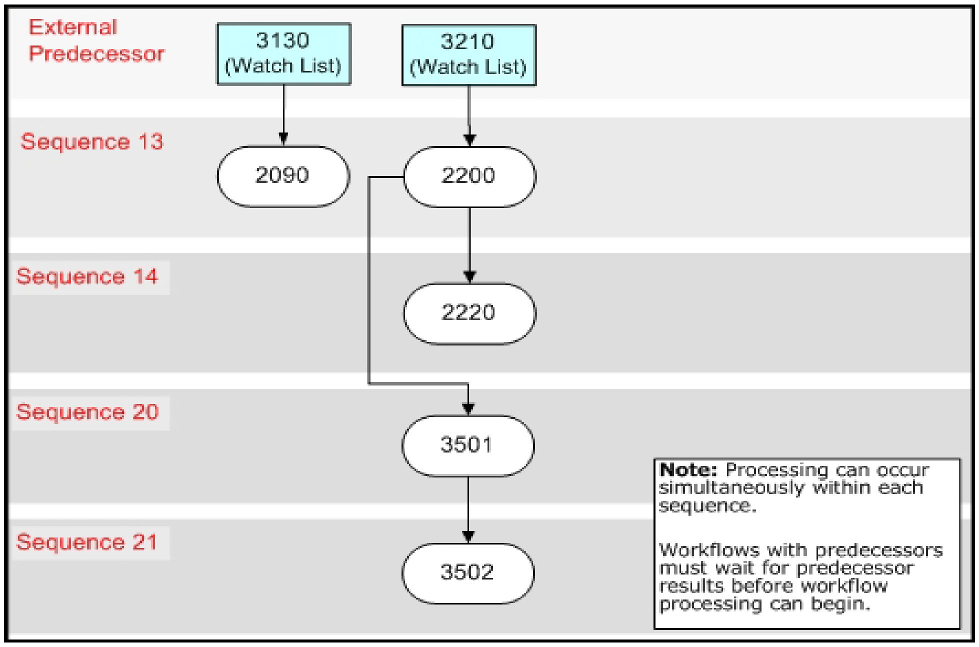

External Predecessor
3130 (Watch List)
3210 (Watch List)
Sequence 13
2090
2200
Sequence 14
2220
Sequence 20
3501
Sequence 21
3502
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 37. Post-Watch List Workflows—AML Banking

The application completes these workflows in four sequences. Processing of some workflows requires data from Miscellaneous predecessor workflows 3130 and 3210 (refer to *Informatica Miscellaneous: Optional Workflows—AML Banking,* on page 146, for more information). These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed. Refer to the note in the section *Informatica Miscellaneous: Post-Watch List Workflows—AML Brokerage,* on page 140, for more information.
Table 54 describes the Informatica Miscellaneous Post-Watch List workflows in Figure 37.

Table 54. Post-Watch List Workflows—AML Banking

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
13	MLM_Banking_Common	2090	w_ph2090_update_jurisdiction_in_cb	MLM_Banking_Common:3130
13	ORION_Common	2200	w_ph2200_build_trxn_tables_from_fotps	ORION_Common:3051, ORION_Common:3210
14	ORION_Common	2220	w_ph2220_update_fot_reversals	ORION_Common:2200
20	ORION_Common	3501	w_ph3501_Exp_and_Risk_Review_TP	ORION_Common:2200
21	ORION_Common	3502	w_ph3502_Flag_Trusted_Trxn	ORION_Common:3501

Informatica Summary Workflows—AML Banking

Informatica Summary workflows maintain monthly aggregations of customer activity. Figure 38 illustrates Informatica Summary workflows for AML Banking.

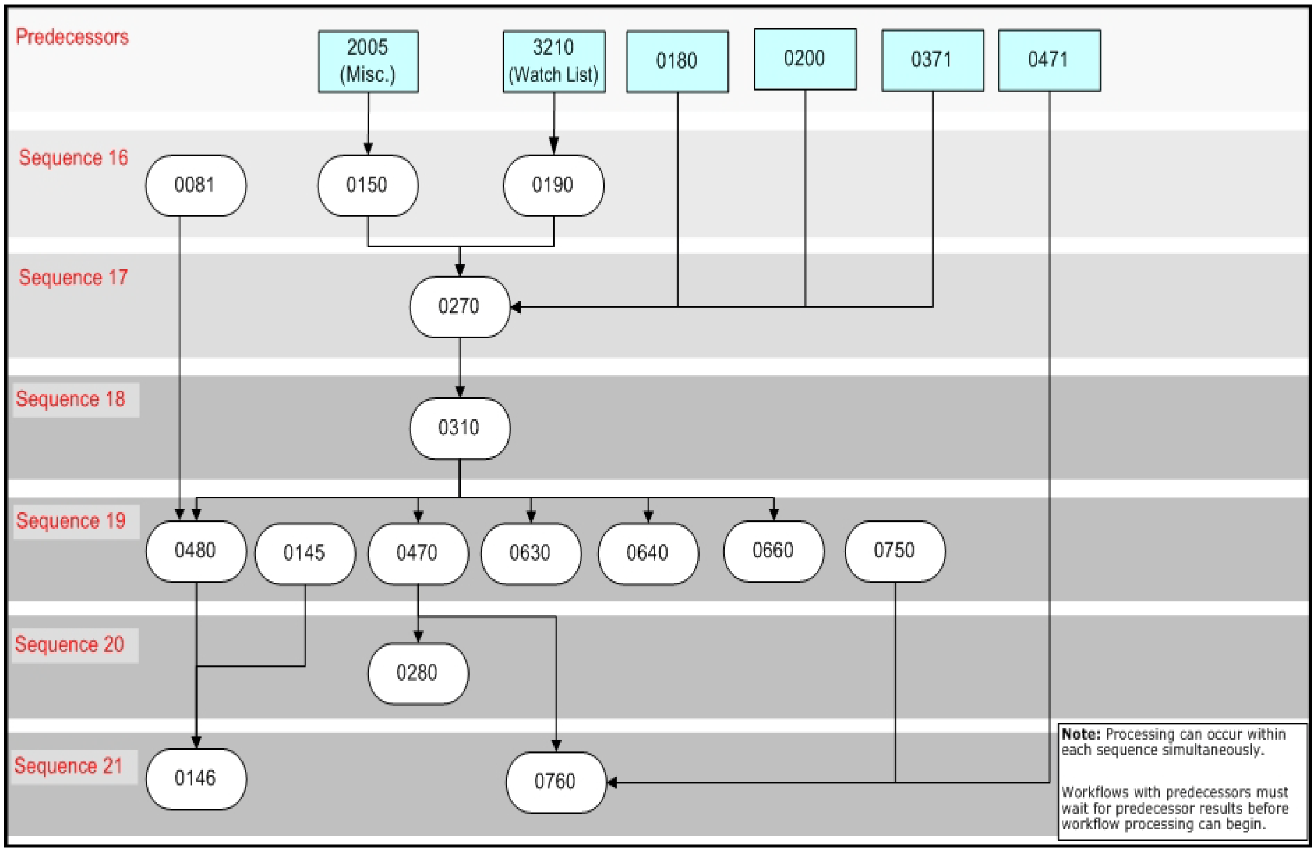


Figure 38. Summary Workflows—AML Banking

The application completes these workflows in five sequences; processing of some workflows requires data from external predecessors (refer to section *Informatica Miscellaneous: Optional Workflows—AML Banking,* on page 146 and *Informatica Watch List Workflows—AML Banking,* on page 148, for more information).

These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.

Table 55 describes the Informatica Summary workflows in Figure 38.

Table 55. Summary Workflows—AML Banking

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
16	MLM_Banking_Common	0081	w_ph0081_truncate_cbsm	
16	ORION_Production	0150	w_p0150_truncate_asms	ORION_Common:2005
16		9001	AccountDailyProfileTrade	
16		9002	AccountDailyProfileTransaction	
17		9005	AccountProfile	9001, 9002
17	ORION_Production	0190	w_p0190_aggregate_fotps_to_asms	ORION_Common:3210, ORION_Production:0150
18	ORION_Production	0310	w_p0310_update_asm_for_daily_activity	9005, ORION_Production:0190
19	MLM_Banking_Production	0145	w_ph0145_truncate_CBPTxSM	
19	MLM_Banking_Common	0480	w_ph0480_aggregate_asm_to_cbsm	MLM_Banking_Common:0081, ORION_Production:0310
19	ORION_Production	0630	w_p0630_build_ATxSD_from_ASMS	ORION_Production:0310
19	ORION_Production	0660	w_p0660_FOTPSR_to_AASD	ORION_Production:0310
19	ORION_Production	0750	w_ph0750_truncate_APTxSM	
19	MLM_Brokerage_Production	0280	w_p0280_create_clog_activity_records	ORION_Production:0310
20	ORION_Production	0760	w_ph0760_ASM_to_APTxSM	ORION_Production:0750
21	MLM_Banking_Production	0146	w_ph0146_CBSM_to_CBPTxSM	MLM_Banking_Production:0145, MLM_Banking_Common:0480

Broker Compliance Informatica Workflows

The following sections describe the Informatica workflows that are required for deriving and aggregating data for the Broker Compliance solution:

- Informatica Miscellaneous: Optional Workflows—Broker Compliance
- Informatica Miscellaneous: Pre-Watch List Workflows—Broker Compliance
- Informatica Summary Workflows—Broker Compliance
- Informatica Balances and Positions Workflows—Broker Compliance
- Informatica Miscellaneous: Post-Watch List Workflows—Broker Compliance

Informatica Miscellaneous: Optional Workflows—Broker Compliance

Optional Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 39 illustrates Optional Informatica Miscellaneous workflows for Broker Compliance.

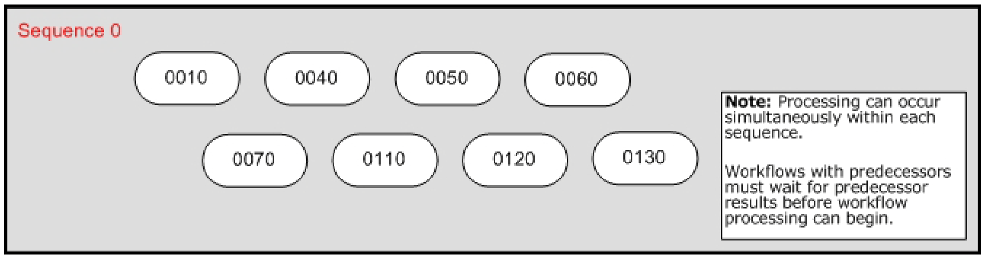

Sequence 0
0010
0040
0050
0060
0070
0110
0120
0130
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 39. Optional Workflows—Broker Compliance

The application completes these workflows in one sequence; processing of the workflows does not require data from external predecessors.
These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 56 describes the Informatica Miscellaneous Optional workflows in Figure 39.

Table 56. Optional Workflows—Broker Compliance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
0	ORION_Production	0010	w_p0010_dump_asm_to_file	
0	BSM_Production	0040	w_p0040_dump_masm_to_file	
0	BSM_Production	0050	w_p0050_dump_nmsm_to_file	
0	BSM_Production	0060	w_p0060_dump_rrsm_to_file	
0	ORION_Production	0070	w_p0070_reload_asm_from_dump_file	
0	BSM_Production	0110	w_p0110_reload_masm_from_dump_file	
0	BSM_Production	0120	w_p0120_reload_nmsm_from_dump_file	
0	BSM_Production	0130	w_p0130_reload_rrsm_from_dump_file	

Informatica Miscellaneous: Pre-Watch List Workflows—Broker Compliance

Pre-Watch List Informatica Miscellaneous workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions.

Figure 40 illustrates Informatica Miscellaneous Pre-Watch List workflows for Broker Compliance.

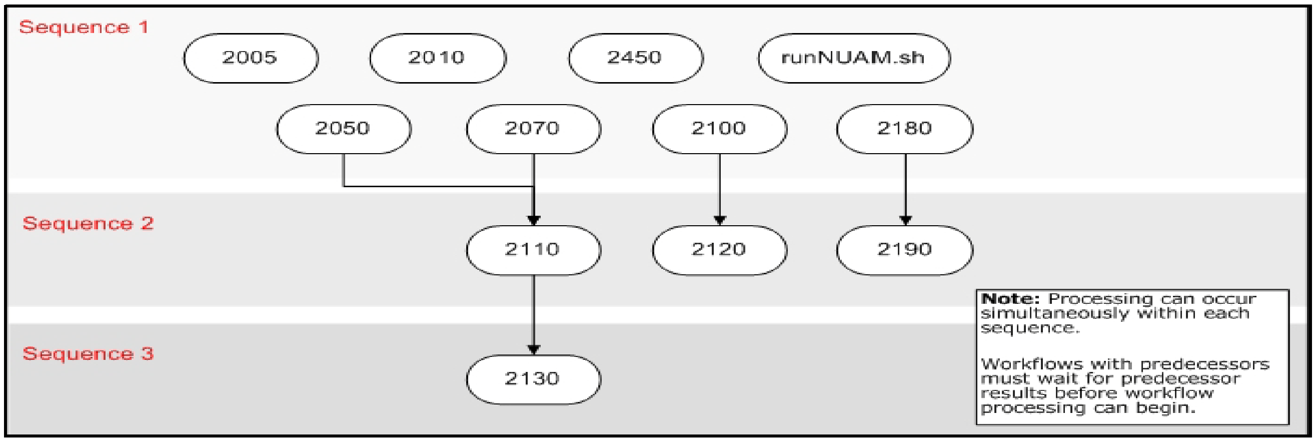

Sequence 1
2005
2010
2450
runNUAM.sh
2050
2070
2100
2180
Sequence 2
2110
2120
2190
Sequence 3
2130
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 40. Workflows—Broker Compliance

The application completes these workflows in three sequences; processing of the workflows does not require data from external predecessors. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.

Table 57 describes the Informatica Miscellaneous workflows in Figure 40.

Table 57. Workflows—Broker Compliance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
1	ORION_Common	2005	w_p2005_set_cd_pd_dates	
1	ORION_Common	2050	w_ph2050_update_bot_reversals	
1	ORION_Common	2070	w_ph2070_truncate_cas	
1	ORION_Common	2180	w_ph2180_instn_identification	
1	ORION_Common	2450	w_ph2450_populate_ACCT_SRVC_TEAM	
2	ORION_Common	2190	w_ph2190_fotps_instn_processing	ORION_Common:2180
2	ORION_Common	2130	w_ph2130_update_fot_unrelated_party_code	ORION_Common:2050 ORION_Common:2070

Informatica Summary Workflows—Broker Compliance

Informatica Summary workflows maintain monthly aggregations of customer activity. Figure 41 illustrates Informatica Summary workflows for BC.

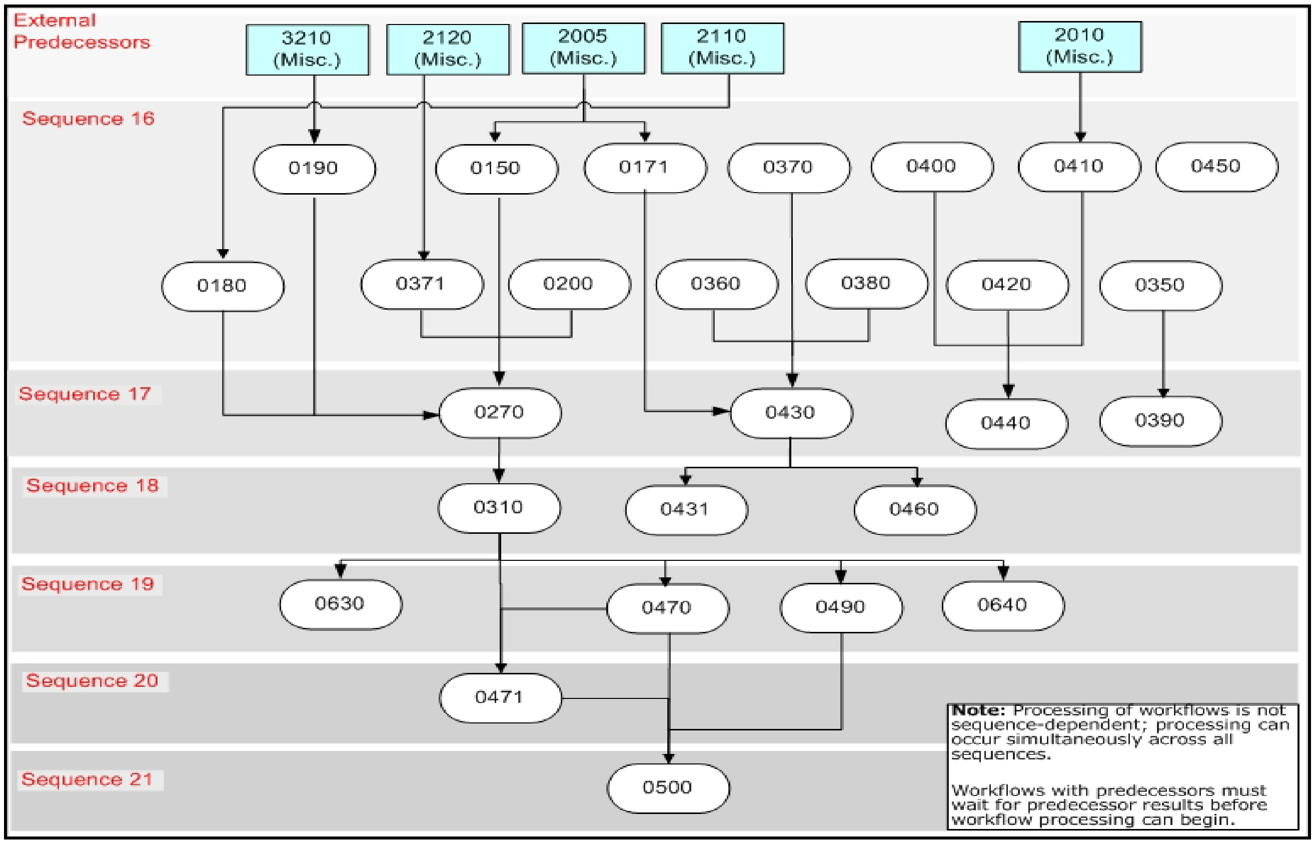


Figure 41. Summary Workflows—Broker Compliance

The application completes these workflows in five sequences; processing of some workflows requires data from external predecessors. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.

Table 58 describes the Summary workflows in Figure 41. The appearance of *N/A* in sequence number and predecessor columns implies that the workflows are *contingency* workflows. These workflows dump summaries to file and reload to the corresponding summary table.

Table 58. Summary Workflows—Broker Compliance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
16	ORION_Production	0150	w_p0150_truncate_asms	ORION_Common:2005
16	BSM_Production	0171	w_p0171_truncate_masms	ORION_Common:2005
16	BSM_Production	0360	w_p0360_aggregate_block_allcn_day_trades_to_file	
16	BSM_Production	0370	w_p0370_aggregate_block_allcn_trades_to_file	
16	BSM_Production	0380	w_p0380_aggregate_bot_inter_hh_jrnls_to_file	
16		9001	AccountDailyProfileTrade	
16		9002	AccountDailyProfileTransaction	
16		9003	InvestmentAdvisorProfile	
16		9004	RegisteredRepresentativeProfile	
17	ORION_Production	0190	w_p0190_aggregate_fotps_to_asms	ORION_Common:3210 ORION_Production:0150
17	BSM_Production	0430	w_p0430_join_daily_activity_to_masms	BSM_Production:0171, BSM_Production:0 360, BSM_Production:0370, BSM_Prod uction:0380
17		9005	AccountProfile	9001, 9002
18	ORION_Production	0310	w_p0310_update_asm_for_daily_activity	9005 ORION_Production:0190
18		9006	HouseholdProfileDerived	9005
18	BSM_Production	0431	w_p0431_build_MASD_from_MASMS	BSM_Production:0430
18	BSM_Production	0460	w_p0460_update_masm_for_daily_activity	BSM_Production:0430
19	ORION_Production	0630	w_p0630_build_ATxSD_from_ASMS	ORION_Production:0310

Informatica Balances and Positions Workflows—Broker Compliance

Informatica Balances and Positions workflows derive attributes that are useful in the assessment of the financial status of an account, customer, or household. Figure 42 illustrates Informatica Balances and Positions workflows for Broker Compliance Solution.

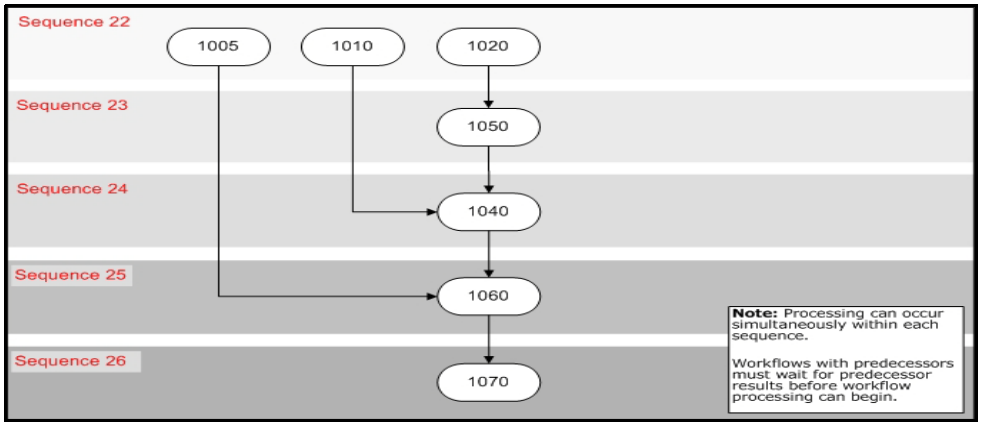

Sequence 22
1005
1010
1020
Sequence 23
1050
Sequence 24
1040
Sequence 25
1060
Sequence 26
1070
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 42. Balances and Positions Workflows—Broker Compliance

The application completes these workflows in five sequences; processing of the workflows does not require data from external predecessors. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 59 describes the Balances and Positions workflows in Figure 42.

Table 59. Balances and Positions Workflows—Broker Compliance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
22		9007	AccountPositionDerived	
23		9008	AccountBalanceDerived	9007
24		9009	HouseholdBalance	9008

Informatica Miscellaneous: Post-Watch List Workflows—Broker Compliance

Informatica Miscellaneous Post-Watch List workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. Figure 43 illustrates Informatica Miscellaneous Post-Watch List workflows for Broker Compliance.

Sequence 13
2172

Figure 43. Post-Watch List Workflows—Broker Compliance

The application completes this workflow in one sequence; processing of the workflows does not require data from external predecessors.
Table 60 describes the Post-Watch List workflows for Broker Compliance in Figure 43.

Table 60. Post-Watch List Workflows—Broker Compliance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
13	ORION_Common	2172	w_ph2172_update_jurisdiction_in_ag	

Fraud Detection Informatica Workflows

The following sections describe the Informatica workflows that are required for deriving and aggregating data for Fraud Detection:

- Informatica Miscellaneous: Optional Workflows—Fraud Detection
- Informatica Miscellaneous: Pre-Watch List Workflows—Fraud Detection
- Informatica Watch List Workflows—Fraud Detection
- Informatica Miscellaneous: Post-Watch List Workflows—Fraud Detection
- Informatica Summary Workflows—Fraud Detection

Informatica Miscellaneous: Optional Workflows—Fraud Detection

Optional Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 44 illustrates Optional Miscellaneous Informatica workflows for Fraud Detection.

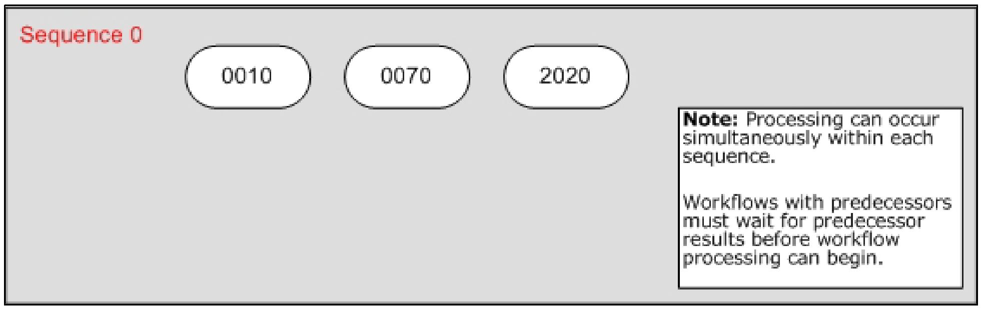

Sequence 0
0010
0070
2020
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 44. Optional Workflows—Fraud Detection

The application completes these workflows in one sequence; processing of the workflows does not require data from external predecessors.

Table 61 describes the Optional Miscellaneous workflows in Figure 44.

Table 61. Optional Workflows—Fraud Detection

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
0	ORION_Production	0010	w_p0010_dump_asm_to_file	
0	ORION_Production	0070	w_p0070_reload_asm_from_dump_file	
0	MLM_Brokerage_Production	2020	w_p2020_delete_nonstanding_instructions	

Informatica Miscellaneous: Pre-Watch List Workflows—Fraud Detection

Pre-Watch List Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 45 illustrates Pre-Watch List Miscellaneous Informatica workflows for Fraud Detection.

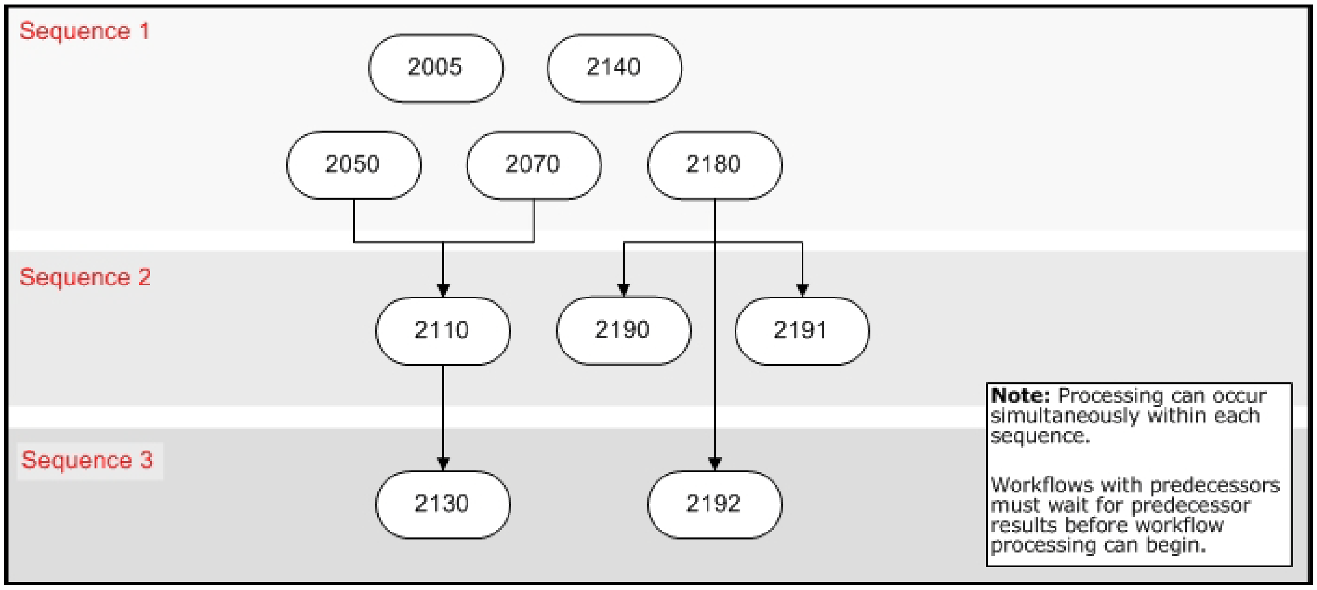

Sequence 1
2005
2140
2050
2070
2180
Sequence 2
2110
2190
2191
Sequence 3
2130
2192
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 45. Pre-Watch List Workflows—Fraud Detection

The application completes these workflows in three sequences; processing of the workflows does not require data from external predecessors.

Table 62 describes the Informatica Miscellaneous Pre-Watch List workflows in Figure 45.

Table 62. Pre-Watch List Workflows—Fraud Detection

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
1	ORION_Common	2005	w_p2005_set_cd_pd_dates	
1	ORION_Common	2050	w_ph2050_update_bot_reversals	
1	ORION_Common	2070	w_ph2070_truncate_cas	
1	ORION_Common	2140	w_ph2140_pass_thru_process	
1	ORION_Common	2180	w_ph2180_instn_identification	
2	ORION_Common	2190	w_ph2190_fotps_instn_processing	ORION_Common:2180
2	ORION_Common	2191	w_ph2191_anticipatory_profile_instn_processing	ORION_Common:2180
3	ORION_Common	2130	w_ph2130_update_fot_unrelated_party_code	
3	MLM_Brokerage_Common	2192	w_ph2192_update_inst_instn_seq_id	ORION_Common:2180

Informatica Watch List Workflows—Fraud Detection

Informatica Watch List workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions.

Figure 46 illustrates Informatica Watch List workflows for Fraud Detection.

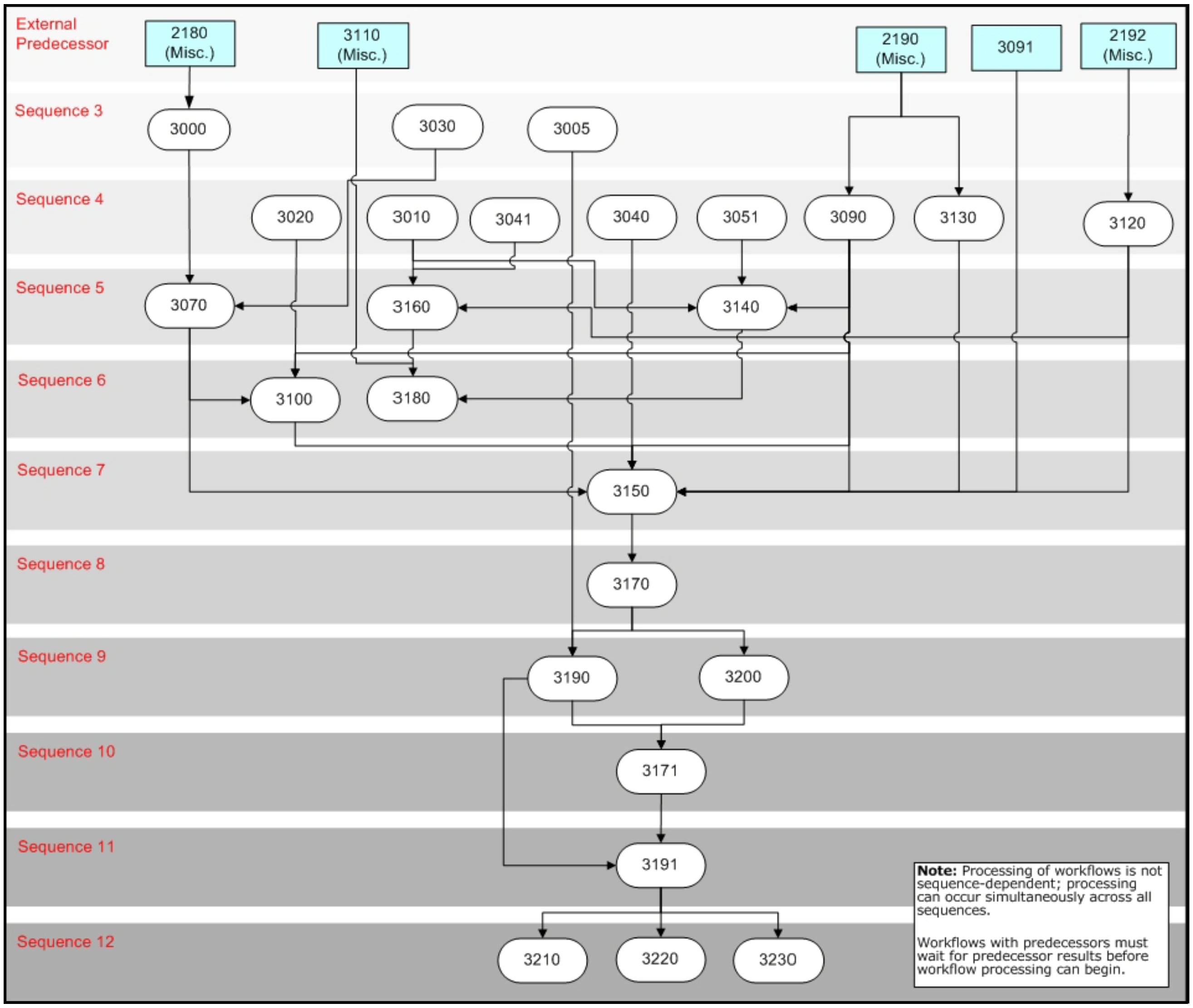


Figure 46. Watch List Workflows—Fraud Detection

The application completes these workflows in 10 sequences; processing of some workflows requires data from Miscellaneous predecessors. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.

Table 63 details the Informatica Watch List workflows in Figure 46.

Table 63. Watch List Workflows—Fraud Detection

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
3	ORION_Common	3000	w_ph3000_Adjust_WL_WLS	ORION_Common:2180
3	ORION_Common	3005	w_ph3005_apply_cust_KYC_risk	
3	ORION_Common	3030	w_ph3030_truncate_wls	
4	ORION_Common	3010	w_ph3010_truncate_ls	
4	ORION_Common	3020	w_ph3020_truncate_nms	
4	ORION_Common	3040	w_ph3040_truncate_wls2	
4	MLM_Brokerage_C ommon	3041	w_ph3041_create_addresses_from_inst	
4	ORION_Common	3051	w_ph3051_create_addresses_from_fotps	
4	ORION_Common	3090	w_ph3090_create_external_entities_from_fotps	ORION_Common:2190, ORION_Common:3091
4	MLM_Brokerage_C ommon	3120	w_ph3120_create_external_entities_from_inst	MLM_Brokerage_Common:2192
4	MLM_Banking_Com mon	3130	w_ph3130_create_client_banks_from_fotps	ORION_Common:2190
5	ORION_Common	3070	w_ph3070_load_watch_list_staging_table	ORION_Common:3000, ORION_Common:3030
5	ORION_Common	3140	w_ph3140_write_fotps_associations_to_ls	ORION_Common:3010, ORION_Common:3051, ORION_Common:3090
5	MLM_Brokerage_C ommon	3160	w_ph3160_write_inst_associations_to_ls	ORION_Common:3010, MLM_Brokerage_Com mon:3041, MLM_Brokerage_Common:3120
6	ORION_Common	3100	w_ph3100_load_staging_fuzzy_matches	ORION_Common:3020, ORION_Common:3070, ORION_Common:3090
6	ORION_Common	3180	w_ph3180_write_ls_to_link_tables	ORION_Common:3110, ORION_Common:3140, MLM_Brokerage_Common:3160
7	ORION_Common	3150	w_ph3150_load_staging_and_validate_watch_list	ORION_Common:3040, ORION_Common:3070, ORION_Common:3090, ORION_Common:3 091, ORION_Common:3100, MLM_Brokerage_Common:3120, MLM_Banking_Common:3130
8	ORION_Common	3170	w_ph3170_update_staging_list_risk	ORION_Common:3150
9	ORION_Common	3190	w_ph3190_apply_risk_to_nonacct_entities	ORION_Common:3005, ORION_Common:3170
9	ORION_Common	3200	w_ph3200_apply_membership_to_entities	ORION_Common:3170
10	MLM_Brokerage_C ommon	3171	w_ph3171_update_account_customer_risk	ORION_Common:3190, ORION_Common:3200

11	ORION_Common	3191	w_ph3191_apply_risk_to_acct_entities	MLM_Brokerage_Common:3171, ORION_Common:3190
12	ORION_Common	3210	w_ph3210_update_fotps_activity_risk	ORION_Common:3191
12	MLM_Brokerage_Common	3220	w_ph3220_update_bot_activity_risk	ORION_Common:3191
12	MLM_Brokerage_Common	3230	w_ph3230_update_inst_activity_risk	ORION_Common:3191

Informatica Miscellaneous: Post-Watch List Workflows—Fraud Detection

Post-Watch List Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 47 illustrates Post-Watch List Miscellaneous Informatica workflows for Fraud Detection.

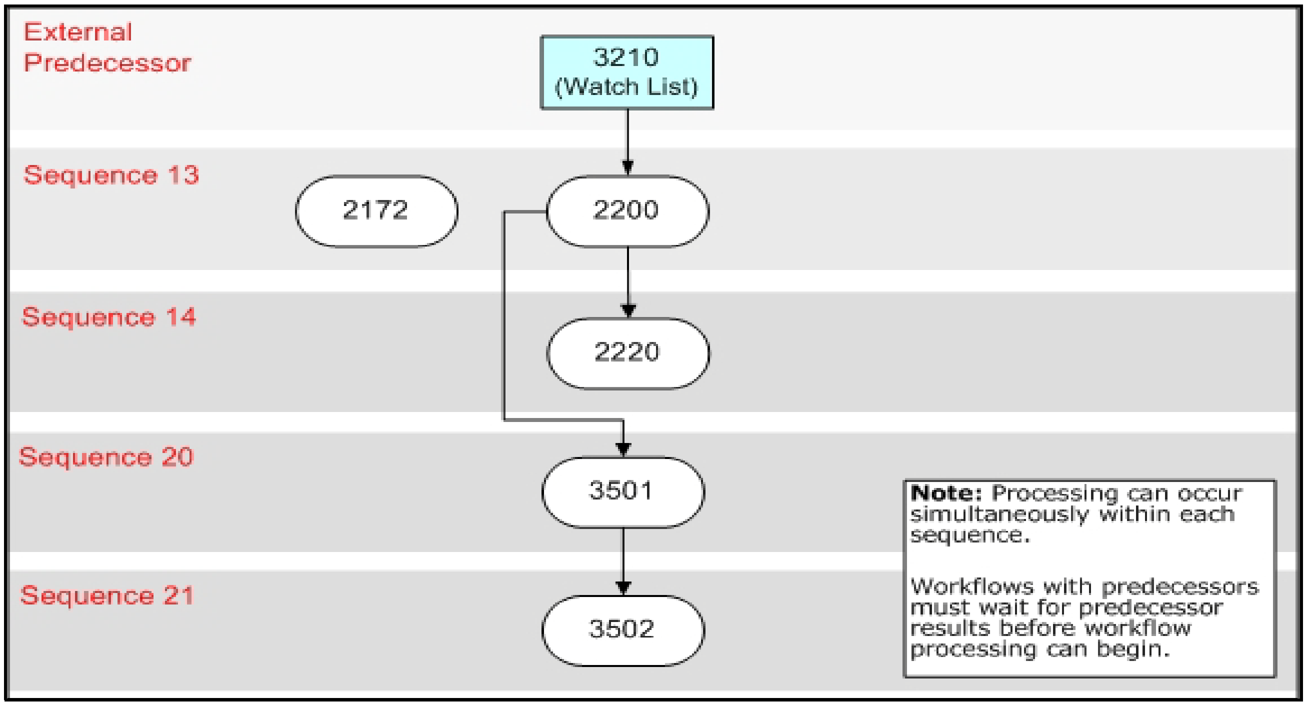

External Predecessor
3210 (Watch List)
Sequence 13
2172
2200
Sequence 14
2220
Sequence 20
3501
Sequence 21
3502
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 47. Post-Watch List Workflows—Fraud Detection

The application completes these workflows in four sequences; processing of a workflow requires data from a Watch List predecessor (refer to *Informatica Watch List Workflows—Fraud Detection,* on page 160, for more information). Refer to the note in the section *Informatica Miscellaneous: Post-Watch List Workflows—AML Brokerage,* on page 140, for more information.

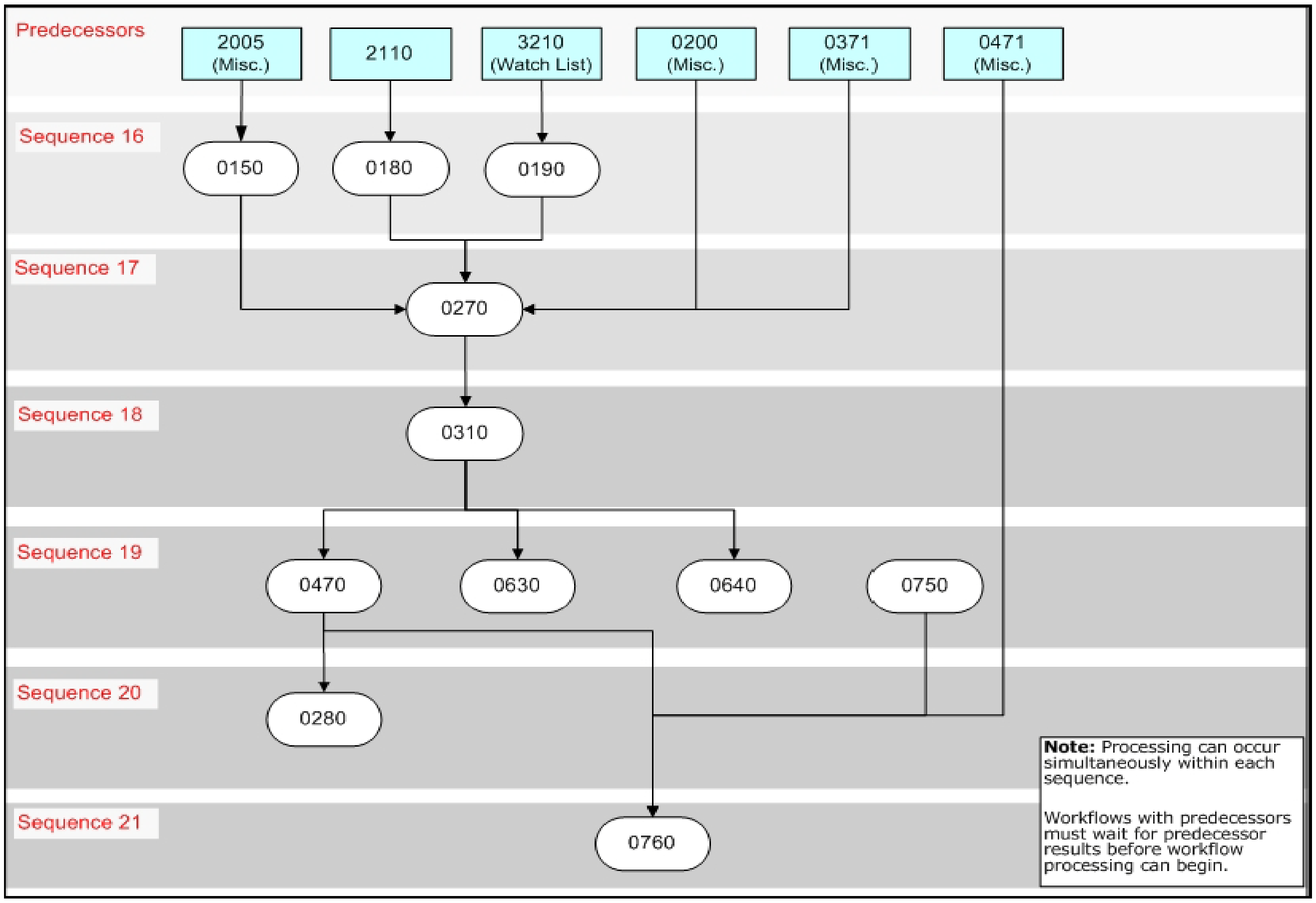
Table 64 describes the Informatica Miscellaneous Post-Watch List workflows in Figure 47.

Table 64. Post-Watch List Workflows—Fraud Detection

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
13	ORION_Common	2172	w_ph2172_update_jurisdiction_in_ag	
13	ORION_Common	2200	w_ph2200_build_trxn_tables_from_fotps	ORION_Common:3051, ORION_Common:3210
14	ORION_Common	2220	w_ph2220_update_fot_reversals	ORION_Common:2200
20	ORION_Common	3501	w_ph3501_Exp_and_Risk_Review_TP	ORION_Common:2200
21	ORION_Common	3502	w_ph3502_Flag_Trusted_Trxn	ORION_Common:3501

Informatica Summary Workflows—Fraud Detection

Informatica Summary workflows maintain monthly aggregations of customer activity. Figure 48 illustrates Informatica Summary workflows for Fraud Detection.



Note: Processing can occur simultaneously within each sequence.

Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 48. Summary Workflows—Fraud Detection

The application completes these workflows in five sequences; processing of some workflows requires data from Watch List and Miscellaneous predecessor workflows (refer to *Informatica Miscellaneous: Optional Workflows—Fraud Detection,* on page 158, and *Informatica Watch List Workflows—Fraud Detection,* on page 160, for more information).

These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.

Table 65 describes the Summary workflows in Figure 48.

Table 65. Summary Workflows—Fraud Detection

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
16	ORION_Production	0150	w_p0150_truncate_asms	ORION_Common:2005
16		9001	AccountDailyProfileTrade	
16		9002	AccountDailyProfileTransaction	
17	ORION_Production	0190	w_p0190_aggregate_fotps_to_asms	ORION_Common:3210 ORION_Production:0150
17		9005	AccountProfile	9001, 9002
18	ORION_Production	0310	w_p0310_update_asm_for_daily_activity	9005 ORION_Production:0190
19	ORION_Production	0630	w_p0630_build_ATxSD_from_ASMS	ORION_Production:0310
19	ORION_Production	0750	w_ph0750_truncate_APTxSM	
19	MLM_Brokerage_Production	0280	w_p0280_create_clog_activity_records	ORION_Production:0310
20	ORION_Production	0760	w_ph0760_ASM_to_APTxSM	ORION_Production:0750

Insurance Workflows

The following sections describe the Informatica workflows that are required for deriving and aggregating data for the Insurance Solution:

- Informatica Miscellaneous: Pre-Watch List Workflows—Insurance
- Informatica Watch List Workflows—Insurance
- Informatica Miscellaneous: Post-Watch List Workflows—Insurance
- Informatica Summary Workflows—Insurance

Each section provides a graphic that illustrates the workflow. An accompanying table describes the process by sequence, workflow number and name, and internal or external predecessors, if any.

Informatica Miscellaneous: Pre-Watch List Workflows—Insurance

Optional Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 49 illustrates an Informatica Miscellaneous Pre-Watch List workflow for Insurance.

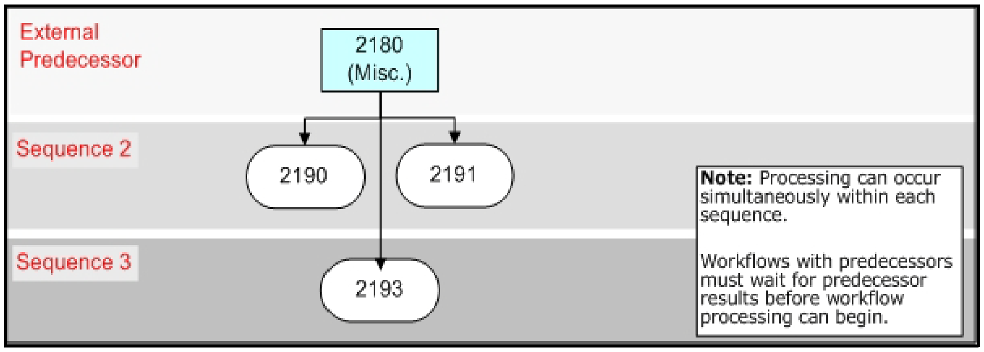

External Predecessor
2180 (Misc.)
Sequence 2
2190
2191
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.
Sequence 3
2193

Figure 49. Pre-Watch List Workflows—Insurance

The application completes these workflows in two sequences; processing of a workflow requires data from one predecessor. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 66 describes the Informatica Miscellaneous Pre-Watch List workflows for Insurance in Figure 49.

Table 66. Pre-Watch List workflows—Insurance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
2	ORION_Common	2190	w_ph2190_fotps_instn_processing	ORION_Common:2180
2	ORION_Common	2191	w_ph2191_anticipatory_profile_instn_processing	ORION_Common:2180
3	ORION_Common	2193	w_ph2193_INS_instn_processing	ORION_Common:2180

Informatica Watch List Workflows—Insurance

Informatica Watch List workflows facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions.
Figure 50 illustrates Informatica Watch List workflows for Insurance.

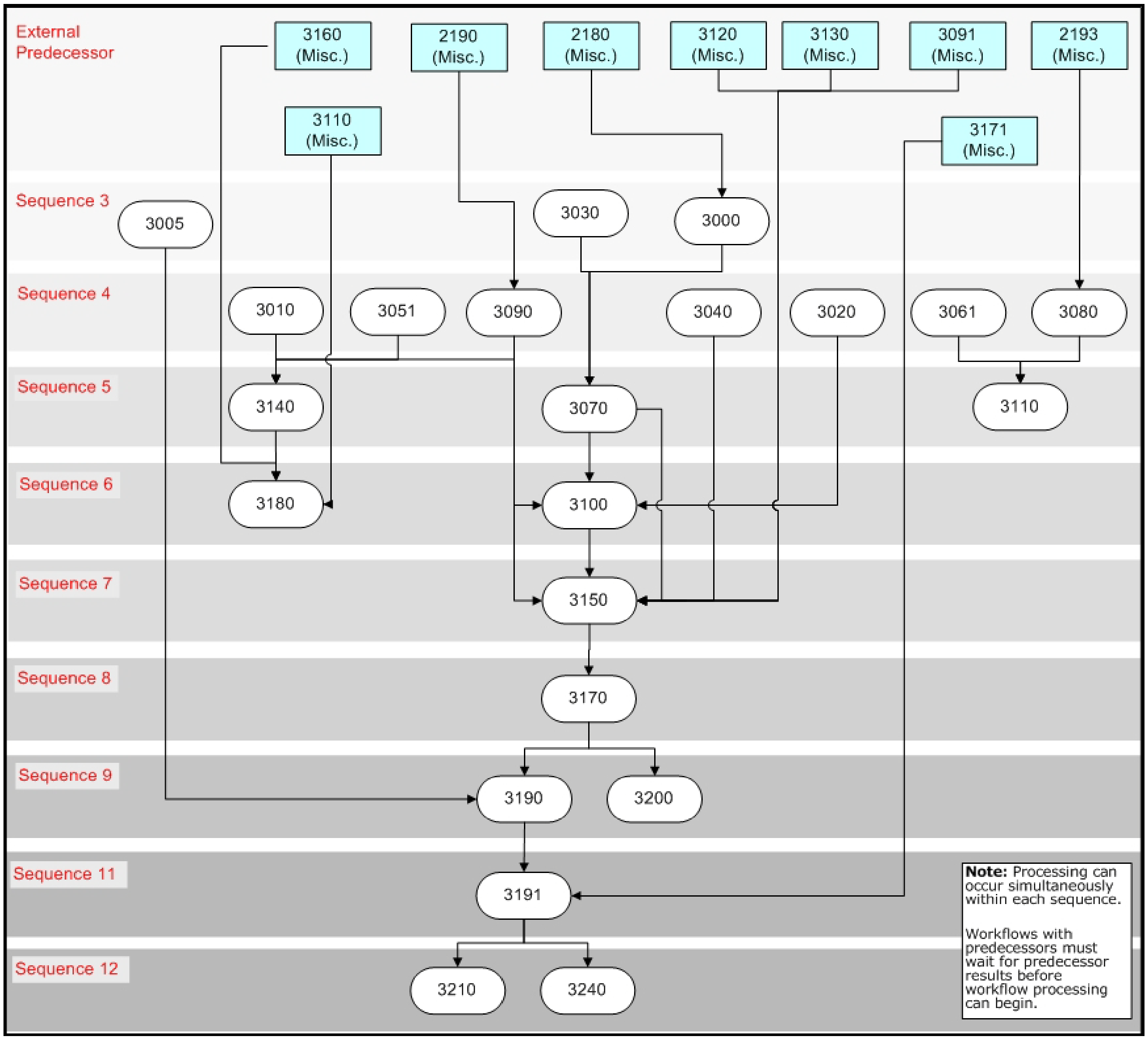


Figure 50. Informatica Watch List Workflows—Insurance

The application completes these workflows in nine sequences; processing of some workflows requires data from Miscellaneous predecessors. These workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 67 details the Informatica Watch List workflows for Insurance in Figure 50.

Table 67. Watch List Workflows—Insurance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
3	ORION_Common	3000	w_ph3000_Adjust_WL_WLS	ORION_Common:2180
3	ORION_Common	3005	w_ph3005_apply_cust_KYC_risk	
3	ORION_Common	3030	w_ph3030_truncate_wls	
4	ORION_Common	3010	w_ph3010_truncate_ls	
4	ORION_Common	3020	w_ph3020_truncate_nms	
4	ORION_Common	3040	w_ph3040_truncate_wls2	
4	ORION_Common	3051	w_ph3051_create_addresses_from_fotps	
4	ORION_Common	3061	w_ph3061_create_addresses_from_INS	
4	ORION_Common	3080	w_ph3080_create_external_entities_from_INS	ORION_Common:2193
4	ORION_Common	3090	w_ph3090_create_external_entities_from_fotps	ORION_Common:2190, ORION_Common:3091
5	ORION_Common	3070	w_ph3070_load_watch_list_staging_table	ORION_Common:3000, ORION_Common:3030
5	ORION_Common	3110	w_ph3110_write_INS_associations_to_ls	ORION_Common:3061, ORION_Common:3080
5	ORION_Common	3140	w_ph3140_write_fotps_associations_to_ls	ORION_Common:3010, ORION_Common:3051, ORION_Common:3090
6	ORION_Common	3100	w_ph3100_load_staging_fuzzy_matches	ORION_Common:3020, ORION_Common:3070, ORION_Common:3090
6	ORION_Common	3180	w_ph3180_write_ls_to_link_tables	ORION_Common:3110, ORION_Common:3140, MLM_Brokerage_Common:3160
7	ORION_Common	3150	w_ph3150_load_staging_and_validate_watch_list	ORION_Common:3040, ORION_Common:3070, ORION_Common:3090, ORION_Common:3 091, ORION_Common:3100, MLM_Brokerage_Common:3120, MLM_Banking_Common:3130
8	ORION_Common	3170	w_ph3170_update_staging_list_risk	ORION_Common:3150
9	ORION_Common	3190	w_ph3190_apply_risk_to_nonacct_entities	ORION_Common:3005, ORION_Common:3170
9	ORION_Common	3200	w_ph3200_apply_membership_to_entities	ORION_Common:3170
11	ORION_Common	3191	w_ph3191_apply_risk_to_acct_entities	MLM_Brokerage_Common:3171, ORION_Comm on:3190
12	ORION_Common	3210	w_ph3210_update_fotps_activity_risk	ORION_Common:3191
12	ORION_Common	3240	w_ph3240_update_INS_activity_risk	ORION_Common:3191

Informatica Miscellaneous: Post-Watch List Workflows—Insurance

Post-Watch List Miscellaneous workflows perform processing in support of workflows in multiple, functional areas. Figure 51 illustrates Post-Watch List Miscellaneous Informatica workflows for Insurance.

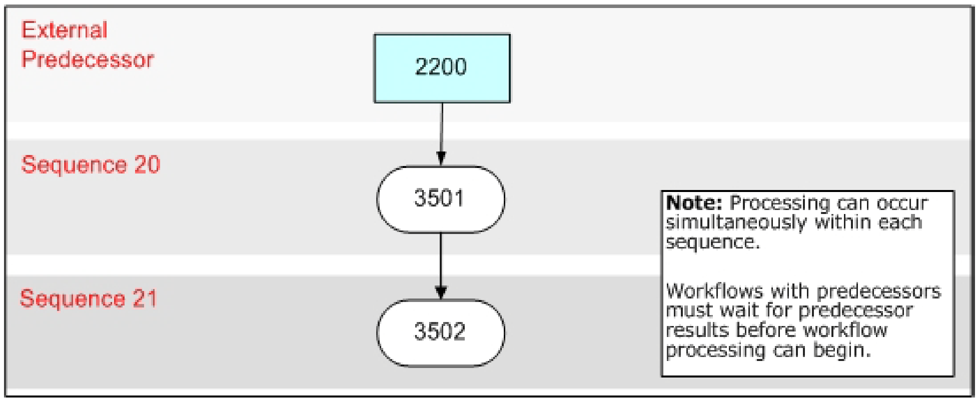

External Predecessor
2200
Sequence 20
3501
Sequence 21
3502
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 51. Post-Watch List Workflows—Insurance

The application completes these workflows in two sequences; processing of a workflow requires data from a predecessor. Refer to the note in the section *Informatica Miscellaneous: Post-Watch List Workflows—AML Brokerage,* on page 140, for more information.

Table 68 describes the Informatica Miscellaneous Post-Watch List workflows in Figure 51.

Table 68. Post-Watch List Workflows—Insurance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
20	ORION_Common	3501	w_ph3501_Exp_and_Risk_Review_TP	ORION_Common:2200
21	ORION_Common	3502	w_ph3502_Flag_Trusted_Trxn	ORION_Common:3501

Informatica Summary Workflows—Insurance

Informatica Summary workflows maintain monthly aggregations of customer activity. Figure 52 illustrates Informatica Summary workflows for Insurance.

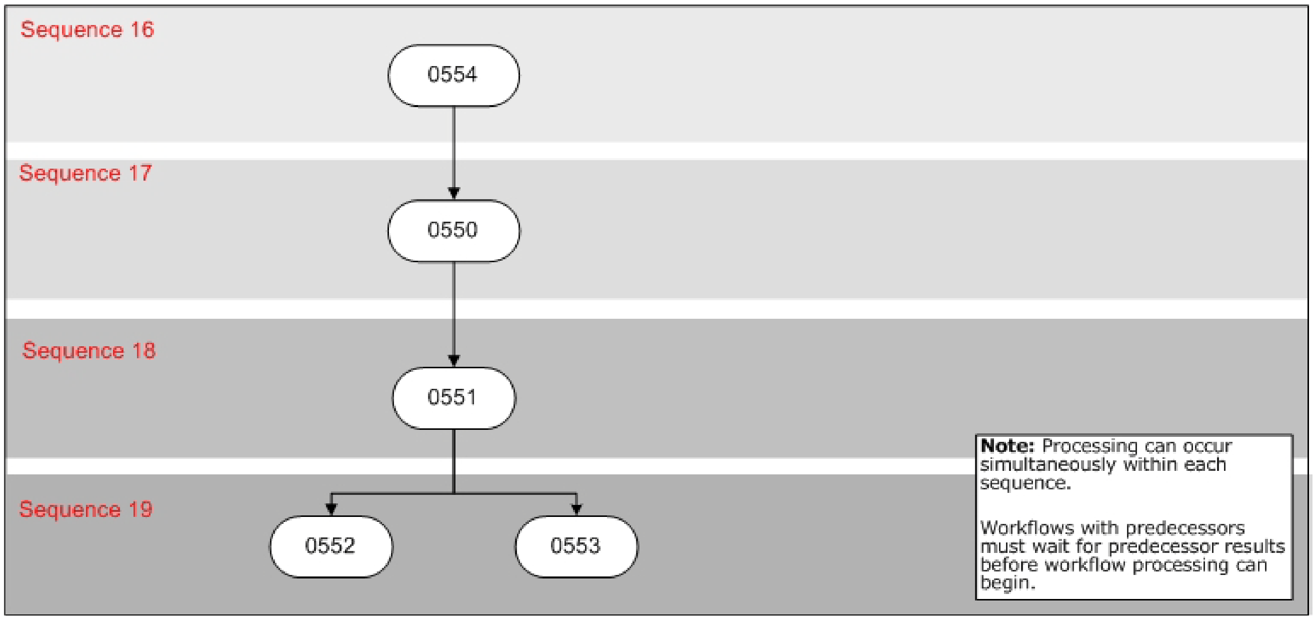

Sequence 16
0554
Sequence 17
0550
Sequence 18
0551
Sequence 19
0552
0553
Note: Processing can occur simultaneously within each sequence.
Workflows with predecessors must wait for predecessor results before workflow processing can begin.

Figure 52. Summary Workflows—Insurance

The application completes these workflows in four sequences; processing is not dependent on data from external predecessors. The workflows may proceed simultaneously in all sequences as soon as processing of any predecessors has completed.
Table 69 details the Watch List workflows for Insurance in Figure 52.

Table 69. Summary Workflows—Insurance

Sequence Number	Folder	Workflow Number	Workflow Name	Predecessor
16	ORION_Production	0554	w_ph0554_truncate_IPSS	
17	ORION_Production	0550	w_ph0550_aggregate_IPSS_from_IT	ORION_Production:0554
18	ORION_Production	0551	w_ph0551_IPSS_from_IPB	ORION_Production:0550
19	ORION_Production	0552	w_ph0552_build_IPSD_from_IPSS	ORION_Production:0551
19	ORION_Production	0553	w_ph0553_build_IPSM_from_IPSS	ORION_Production:0551

CHAPTER 6

Post-Processing Tasks

This chapter defines the following post-processing administrative tasks:

- About Post-Processing
- Match Scoring
- Alert Creation
- Update Alert Financial Data
- Alert Scoring
- Assignment
- Auto-Close
- Automatic Alert Suppression
- Highlight Generation
- Augment Trade Blotter
- Score Trade Blotter
- Historical Data Copy
- Alert Correlation
- Account Approval/Pre-Trade Approval Tasks

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.

- **Match Scoring:** Computes a ranking for scenario matches indicating a degree of risk associated with the detected event or behavior (Refer to *Match Scoring* on page 173, for more information).
- **Alert Creation:** Packages the scenario matches as units of work (that is, alerts), potentially grouping similar matches together, for disposition by end users (Refer to *Update Alert Financial Data* on page 176, for more information).
- **Update Alert Financial Data:** Records additional data for alerts such as the related Investment Advisor or Security involved in the alert.

- **Alert Scoring:** Ranks the alerts (including each match within the alerts) to indicate the degree of risk associated with the detected event or behavior (Refer to *Alert Scoring* on page 176, for more information).
- **Assignment:** Determines the user or group of users responsible for handling each alert or case (Refer to *Assignment* on page 177, for more information).
- **Auto-Close (optional):** Closes alerts that are of a lower priority to the business (Refer to *Auto-Close* on page 178, for more information).
- **Automatic Alert Suppression (optional):** Suppresses alerts that share specific scenario and focal entity attributes for a particular time frame (Refer to *Automatic Alert Suppression* on page 183, for more information).
- **Highlight Generation:** Generates highlights for alerts that appear in the alert list in the Alert Management subsystem and stores them in the database (Refer to *Highlight Generation* on page 184, for more information).
- **Augment Trade Blotter:** Provides the ability to differentiate between various types of trades using text-based codes. It also provides the ability to flag trades that require additional analysis before an analyst can mark trade as *Reviewed* or *Reviewed with Follow up*. (Refer to *Augment Trade Blotter* on page 184, for more information).
- **Score Trade Blotter:** Determines the maximum score of alerts generated in the same batch cycle associated with a trade; also determines the alert/trade mappings (Refer to *Score Trade Blotter* on page 185, for more information).
- **Historical Data Copy:** Identifies the records against which the current batch's scenario runs generated alerts and copies them to archive tables (Refer to *Historical Data Copy* on page 185, for more information).
- **Alert Correlation:** Uncovers relationships among alerts by correlating alerts to business entities and subsequently correlating alerts to each other based on these business entities (this latter step is optional). The relationships are discovered based on configurable rule sets (Refer to *Assignment* on page 177, for more information).
- **Account Approval/Pre-Trade Approval Tasks**: Two processes, Load Employee User Groups and Request Assignment, are required to successfully utilize the Account Approval (AA) and Pre-Trade Approval (PTA) application (Refer to *Account Approval/Pre-Trade Approval Tasks*, on page 192 for more information).

Note: You can re-run any failed post-processing job.

Order of Running Post-Processing Administrative Tasks

Run the post-processing administrative tasks in this order:

1. Match Scoring (501)
2. Alert Creation (502/503)
3. Update Alert Financial Data
4. Alert Scoring (504)
5. Assignment (505)
6. Auto-Close (506)
7. Automatic Alert Suppression (507)
8. Highlight Generation
9. Augment Trade Blotter
10. Score Trade Blotter
11. Historical Data Copy
12. Alert Correlation (508)

If the Oracle client has implemented the AA/PTA application, the Account Approval/Pre-Trade Approval Tasks may be run in parallel with any of the post-processing administrative tasks 1-12. Before running the AA/PTA application for the first time, the AA/PTA tasks must run in this order:

1. Load Employee User Groups
2. Request Assignment

Thereafter, the frequency of when the AA/PTA tasks are to be executed are determined by the Oracle client and may be executed independently of each other.

Match Scoring

Behavior Detection provides a mechanism to compute a score for matches to provide an initial prioritization. Match Scoring rules are created using the Scoring Editor from the Administration Tools. Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide,* Release 6.1.3*,* for more information.

Running the Match Scoring Job

The Match Scoring job is part of the Behavior Detection subsystem. Behavior Detection delivers job template group 501 to run the Match Scoring job.

To Run the Match Scoring Job

To run the Match Scoring job, follow the steps:

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

   ```
   start_mantas.sh 501
   ```

All new matches in the system are scored.

Alert Creation

Matches are converted into alerts with the Alert Creator processes. These processes are part of the Behavior Detection subsystem.

The system uses two types of Alert Creator jobs:

- Multi-match Alert Creator generates alerts for matches that share a common focus, are from scenarios in the same scenario class, and possibly share other common attributes. Each focus type has a separate job template.
- Single-match Alert Creator generates 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. These jobs can be modified using Administration Tools. Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*, for more information.

The following sections describe running each type of Alert Creator.

To Run Multi-match Alert Creator

To run the multi-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 502
   ```

   where `502` is the job template that Behavior Detection provides to run the Alert Creator algorithm.

To Run Single Match Alert Creator

To run the single 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.

Understanding Advanced Alert Creator Configuration

The Alert Creator algorithm can support grouping strategies that the Administration Tools do not support. To use these advanced strategies, you must enter Alert Creator rules directly into the database. The following section discusses these advanced rules.

Advanced Rules

The executable retrieves new, unowned single matches generated from specified types of scenarios. It then groups them based on one of four implemented algorithms and a specified list of bindings for grouping. It requires parameter settings to designate:

- Choice of grouping algorithm to use.
- Scenario types associated with the set of matches to consider for grouping.
- Bindings on which to base break group compatibility.

Grouping Algorithms

When grouping algorithms, choose from the following:

- **BIND_MATCH:** The Alert Creation module creates alerts based on matches with matching bindings/values based on a provided list of bindings to use when determining *groupability*.
- **BIND_BEHAVIOR_SCENARIO_CLASS:** The Alert Creation module creates alerts based on matches with matching scenario class code and with matching bindings/values based on a provided list of bindings to use when determining *groupability*.
- B**IND_BEHAVIOR_SCENARIO:** The Alert Creation module creates alerts based on matches with matching scenario ID and with matching bindings/values based on a provided list of bindings to use when determining *groupability*.
- **BIND_BEHAVIOR_PATTERN:** The Alert Creation module creates alerts based on matches with matching pattern ID and with matching bindings/values based on a provided list of bindings to use when determining *groupability*.
- **SINGLE_ALERT_MATCH:** The Alert Creation module creates alerts for all remaining matches. A alert is created for each of the remaining matches, as long as they bind one of the centricity names in the bindings string. This is the *catch all* algorithm that ensures that all matches that have a bound centricity value and a corresponding alert is created.

For a `BIND_MATCH` grouping rule, the system compares bindings (`KDD_BREAK_BINDING`) values for matches to determine whether it can group matches together into an alert.

For example, the grouping algorithm interprets `!TRADER ?ASSOC_SCRTY` to create an alert; each break set to be grouped must have a `TRADER` binding in which the values for that binding must match and each must either have an `ASSOC_SCRTY` binding in which the values match OR each must be missing the `ASSOC_SCRTY` binding. Alerts that mentioned `ASSOC_SCRTY` could only be grouped with other alerts that mentioned `ASSOC_SCRTY`. Similarly, alerts that did not mention `ASSOC_SCRTY` could only be grouped with other alerts that did not mention `ASSOC_SCRTY`.

This list is order-dependent and at least one binding should be marked as required using an exclamation point (!) to prevent grouping of all miscellaneous matches into one big break. The order helps determine the centricity in the first binding name in the binding string. The centricity name is used to determine the alert's centricity ID.

Update Alert Financial Data

Oracle Financial Services provides some enhanced data on alerts to support searching by alerts based on business data. For example, Trader-focused alerts may be searched based on the security involved in the activity. Update Alert Financial Data is the process that populates this information.

To update alert financial data, run the following command from the `<INSTALL_DIR>/database/db_tools/bin` directory:

```
upd_kdd_review_fin.sh <batch_id> <YYYYMMDD>
```

If `<batch_id>` and the batch date `<YYYYMMDD>` are not provided, the system derives this data for matches created in the current batch. The log for this process is under the `<INSTALL_DIR>/database/db_tools/logs` directory. The name of the file is `run_stored_procedure.log`.

Alert Scoring

Oracle Financial Services provides a mechanism to compute a score for alerts to provide an initial prioritization. The score is an integer and will be bounded by a configurable minimum and maximum value.

This module has two different strategies for computing the alert's score. All strategies are based on the score of the alert's matches. The strategies are:

- **Max Match Score:** The score of the alert equals the alert's highest scoring match.
- **Average Match Score:** The score of the alert equals the average of its matches score.

Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*, Release 6.1.3, for more information.

Running the Alert Scoring Job

To run an Alert Scoring Job, follow the steps:

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

   ```
   start_mantas.sh 504
   ```

   where, `504` is the job template that Oracle Financial Services provides to run the Alert Scoring algorithm.

Assignment

Oracle Financial Services provides a mechanism to assign alerts and cases to a predefined owner (either an individual user or a pool of users). When performing alert or case assignment, the module fetches new, unowned alerts or cases for a given product and assigns them to an owner using a rule-based strategy.

You can configure assignment rules by using the Administration Tools. Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*, for more information.

Running the Assignment Job

The Assignment Job is part of the Behavior Detection subsystem. Oracle Financial Services provides default job templates and job template groups for running Assignment Job. You can modify these jobs using the Administration Tools. Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*, for more information.

To run an Assignment job, follow the steps:

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

   ```
   start_mantas.sh 505
   ```

   where, `505` is the job template that Oracle Financial Services provides to run the Assignment algorithm.

Auto-Close

Oracle Financial Services provides a mechanism to close alerts automatically that do not warrant investigation. The system can close alerts based on their age, status, score, focus type, generating scenario, or any combination of these attributes. The system regularly evaluates all candidate alerts and closes each alert that satisfies the criteria. The system maintains closed alerts for audit purposes and they are still available for display
(for example, from the Entity History page in the Oracle Financial Services UI) and processing (for example, by reopening an alert).

Defining the Auto-Close Alert Algorithm

The KDD_AUTO_CLOSE_ALERT table provides all operation sets, and their respective operations, that the system uses to determine whether it should close an alert. The table includes the following:

- Operations are logical expressions that can be used to close alerts (for example, alert score > 50, age > 30). A set of operations based on the same attribute (for example, score) form an operation set.
- The OPRTN_SET_ID column is a grouping of mutually exclusive operations. Each operation specifies the next step that is applied to alerts that satisfy the operation. This next step is either to close the alert or execute the Next operation Set (NEXT_OPRTN_SET_ID column), or branch to further evaluate the alerts.
- The XPRSN_ORDER_ID column sets up an order of precedence by which the system attempts to satisfy the operations.

  Note: Enter NULL into the XPRSN_ORDER_ID column if that entry is linked from another entry that has a value in the XPRSN_ORDER_ID column.
- The ALERT_ATTR_ID column identifies the attribute of the alert for evaluation.
- The OPRTR_CD column specifies the type of operation to be performed. Allowed values are =, !=, >, <, >=, <=, contains, or IN.

  Note: While using IN operator, the right-hand side variables should be separated by| (for example, NW|OP).
- The value in the VALUE_TX column provides the right-hand side of the operation being evaluated.
- If the current operation is satisfied, and it is not the final operation in the operation set (indicated by a NULL value in the NEXT_OPRTN_SET_ID column), the process jumps to the NEXT_OPRTN_SET_ID. If the NEXT_OPRTN_SET_ID is NULL, and the operation is true, the system closes the alert.
- The DMN_CD column is the Oracle Financial Services application product code.
- The CLS_ACTIVITY_TYPE_CD column specifies the activity type code of the closing action to associate with an alert that is closed by this rule.

  Note: The CLS_ACTIVITY_TYPE_CD column is optional. If the column is NULL, the system uses the default auto-close activity type code.

- The CMMNT_TX column specifies an optional text comment to associate with an alert that is closed by this rule.

The Auto-Close Alert algorithm does not close a locked alert. The system locks an alert when an analyst investigates it, and then unlocks it when the analyst releases it. All locked alerts are skipped until the next time the Auto-Close Alert algorithm is run. The Oracle Financial Services administrator must fill in rows in the KDD_AUTO_CLOSE_ALERT table with the criteria for auto-closing the alerts.

The system uses the KDD_REVIEW table to provide available attributes for use in the Auto-Close algorithm.

Refer to the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide*, Volume 2, Release 6.1.3, for more information about the KDD_REVIEW table.

To Set Up Auto-Close Rules

To set up auto-close rules, follow the steps:

1. Formulate the criteria for auto-closing alerts using the attributes in the Alert Closing Attributes (KDD_AUTO_CLOSE_ALERT) table. The Alert Identifier (ALERT_ATTR_ID) column is needed later in this set of instructions.

   Table 70 describes commonly used Alert Closing Attributes.

Table 70. Commonly Used Alert Closing Attributes

Alert Attribute	Alert Identifier (ALERT_ATTR_ID)
Alert Age	113000057
Due Date	113000024
Focus Type	113000010
Last Action	113000038
Owner’s Organization	113000056
Previous Match Count All	113000054
Previous Match Count Same Scenario	113000053
Scenario	113000013
Score	113000022
Status	113000008
Status Name	113000055
Processing Batch Name	113000068
Jurisdiction	113000067
Previous Match Count Same Scenario Class	113000064
Scenario Class	113000014

To View All Alert Closing Attributes

To view a full set of Alert Closing Attributes, run the following query:

1.
```
Select A.ATTR_ID, A.ATTR_NM
From KDD_ATTR A, KDD_DATASET_ATTR B
where A.ATTR_ID=B.ATTR_ID and B.DATASET_ID=113000002
```
   Note: If the alert attribute that corresponds with a particular alert identifier contains a NULL value, the Auto-Close algorithm does not interpret these values and returns a fatal Behavior Detection error.

2. Formulate operations for the auto-closing criteria.

   Operations contain only one mathematical operator (for example, >, <, or =). Operation sets include one or more operations chained together by the `NEXT_OPRTN_SET` column.

3. Determine an order of precedence for the operations (that is, what to test first, second, and so forth).

   Each operation's precedence must be unique within the `KDD_AUTO_CLOSE_ALERT` table.

   Note: An error occurs if two operations have the same precedence. All operations must have a precedence or the system does not test them.

4. Assign an operation ID to each operation. This ID must be unique within `KDD_AUTO_CLOSE_ALERT`.

5. Assign an operation ID to each operation within each operation set.

   Use IDs close together for operations within the same operation set. The system uses this ID to link together operations within the same operation set by placing the next ID for testing in the Next Operation ID (`NEXT_OPRTN_SET_ID`) column.

6. Determine the rows to insert into the `KDD_AUTO_CLOSE_ALERT` table from the following columns:

   - `OPRTN_SET_ID` is the operation set ID.
   - `XPRSN_ORDER_ID`, the operation ID, the precedence must be unique for each operation across the table. This column can contain a NULL value.

     Note: When an operation set is reached by linking from another operation set, you can leave the `XPRSN_ORDER_ID` at `NULL`. For operations sets that are not reached through another operation set, the `XPRSN_ORDER_ID` is required.

   - `ALERT_ATTR_ID` (Refer to Step 1).
   - `OPRTR_CD` is the mathematical operator for the operation.
   - `VALUE_TX` is the right-hand side of the operation.
   - `NEXT_OPRTN_SET_ID` is the ID that identifies the next operation in the operation set, or NULL if no operations exist.

     Note: Inserting an ID into the `NEXT_OPRTN_SET` column previously called creates a loop and results in an error.

- DMN_CD is the Oracle Financial Services application product code.
- The CLS_ACTIVITY_TYPE_CD column specifies the activity type code of the closing action.
- The CMMNT_TX column specifies an optional text comment.

  Note: The activity type code that CLS_ACTCVY_TYPE_CD specifies must exist in the KDD_ACTIVITY_TYPE_CD table and the KDD_ACTIVITY_TYPE_CD. Verify that the AUTO_CLOSE_FL is set to 'Y' for this code to be valid.

7. Insert the needed rows into the KDD_AUTO_CLOSE_ALERT table.

Sample Auto-Closing Alert Rule

You may want to close an alert when the match score is less than 75 and the status code is equal to *NW* (New), or the review is more than 30 days old. If so, follow the steps:

1. Determine the ATTR_ID for the columns to reference in the KDD_REVIEW table.

   ```
   SCORE has ATTR_ID 113000022.
   STATUS has ATTR_ID 113000008.
   AGE has ATTR_ID 113000057.
   ```

2. Formulate the operations:

   The match score is less than 75 and the status code is equal to
   `NW = (SCORE < 75) AND (STATUS = NW)`

   Reviews more than thirty days old = `(AGE > 30)`

3. Determine an order of precedence for the criteria.

   For example, to determine whether reviews are more than thirty days old, assign `(AGE > 30)` a precedence of 1, and `(SCORE < 75) AND (STATUS = NW)` a precedence of 2.

4. Assign an operation ID to each operation within the operation set.

   The operation ID must be unique within the database. The numbers may be any number not already in the table.

   ```
   OPRTN_SET_ID 100 -> (SCORE < 75) AND (STATUS = NW)
   OPRTN_SET_ID 200 -> (AGE > 30)
   ```

5. Assign an ID to each operation within the already divided operations:

   ```
   OPRTN_SET_ID 100 -> (SCORE < 75)
   OPRTN_SET_ID 101 -> (STATUS = NW)
   OPRTN_SET_ID 200 -> (AGE > 30)
   ```

6. Assign the next operation set to chain the operations together.

   Optionally: assign or close an activity type code and/or comment to the operation.

7. Insert the rows into the KDD_AUTO_CLOSE_ALERT table.

Table 71 resembles the entries into the `KDD_AUTO_CLOSE_ALERT` table for the (AGE > 30) auto-close alert.

Table 71. `KDD_AUTO_CLOSE_ALERT` (`AGE > 30`)

OPRTN_SET_ID	XPRSN_ORDER_ID	ALERT_ATTR_ID	OPRTR_CD	VALUE_TX	NEXT_OPRTN_SET_ID	DMN_CD	CLS_ACTIVITY_TYPE_CD	CMMNT_TX
200	1	1130000057	>	30	NULL	MTS	MTS 203	Close if age greater than 30

Note: The `NEXT_OPRTN_SET_ID` is `NULL` because this operation set contains only one operation. Table 72 shows how to set it to the next operation's ID within the operation set.

Table 72 resembles entries into the `KDD_AUTO_CLOSE_ALERT` table for the `(SCORE < 75)` and `(STATUS = NW)` auto-close alert.

Table 72. `KDD_AUTO_CLOSE_ALERT` (`SCORE < 75`) and (`STATUS = "NW"`)

OPRTN_SET_ID	XPRSN_ORDER_ID	ALERT_ATTR_ID	OPRTR_CD	VALUE_TX	NEXT_OPRTN_SET_ID	DMN_CD	CLS_ACTIVITY_CD	CMMNT_TX
100	2	113000022	<	75	101	MTS	NULL	NULL
101	NULL	113000008	=	NW	NULL	MTS	NULL	NULL

Running the Auto-Close Alert

Auto-Close Alert is part of the Behavior Detection subsystem. Oracle Financial Services provides default job templates and job template groups for running Auto-Close Alert. You can modify these jobs using the Administration Tools. Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*, Release 6.1.3, for more information.

To Run Auto-Close Alert

To run Auto-Close Alert, follow the steps:

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

   ```
   start_mantas.sh 506
   ```

   where, `506` is the job template that Oracle Financial Services provides to run the Auto-Close algorithm.

Automatic Alert Suppression

The Alert Management subsystem provides actions that enable an analyst to specify that the system close a particular entity's alerts on a specific scenario automatically. This is called *Alert Suppression*. The system runs the Alert Suppression algorithm to close newly-generated alerts that match an active suppression rule.

The system can suppress alerts with the status of NEW based on their creation date, generating scenario, and focal entity. The algorithm evaluates all candidate alerts and suppresses each alert that satisfies the criteria. The suppressed alerts, to which the system assigns a status of Closed, remain for audit purposes and are still available for display (for example, through the Entity History page) and processing (for example, reopening an alert).

Defining the Suppress Alert Algorithm

The Suppress Alert algorithm does not suppress locked alerts. The system locks an alerts while an analyst takes an action on it, and then unlocks the alert when the analyst releases it. The system skips all locked alerts until the next time it runs the Suppress Alert component. When a user takes an action on an existing alert to suppress future alerts, the suppression rule populates the `KDD_AUTO_SUPPR_ALERT` table with the criteria for automatically suppressing and canceling suppression of the alerts.

- Refer to the *Oracle Financial Services Behavior Detection Platform User Guide*, Release 6.1.3, for detailed information about initiating and canceling Alert Suppression.
- Refer to the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide*, Volume 2, Release 6.1.3, for information about the `KDD_AUTO_SUPPR_ALERT` table.

Running the Suppression Job

The suppression job is part of the Behavior Detection subsystem. Oracle Financial Services provides default job templates and job template groups for running Auto-Close Alert. You can modify these jobs using the Administration Tools. Refer to the *Oracle Financial Services Behavior Detection Platform Administration Tools User Guide*, Release 6.1.3, for more information.

To Run the Suppression Job

To run the suppression job, follow the steps:

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

   ```
   start_mantas.sh 507
   ```

   where, `507` is the job template that Oracle Financial Services provides to run the suppression job algorithm.

Highlight Generation

The Alert Management subsystem displays alert and match highlights in the Alert List and Alert Context sections of the Oracle Financial Services UI. The system calculates and stores these highlights in the database as part of the batch cycle using the following shell script:

The system generates highlights for alerts that appear in the alert list in the Alert Management subsystem and stores them to the database using the following shell script:

```
run_highlights.ksh
```

This script is part of the Database Tools that resides in the `<INSTALL_DIR>/database/db_tools/bin` directory. This script attaches to the database using the user that the `utils.database.username` property identifies in the `<INSTALL_DIR>/database/db_tools/ mantas_cfg/install.cfg` file. You run highlight generation after the creation of alerts and before the system ends the batch with the `end_mantas_batch.sh` script.

By default, Behavior Detection writes log messages for this script in the `<INSTALL_DIR>/database/db_tools/logs/highlights.log` file.

Augment Trade Blotter

Oracle Financial Services Behavior Detection provides the ability to differentiate between various types of trades (for example, Client is Above 64 and Cancelled Trade) using text-based codes. It also provides the ability to flag trades that require additional analysis before an analyst can mark trade as *Reviewed* or *Reviewed with Follow up*. For this purpose, the `run_augment_trade_blotter.sh` script calls the `P_AUGMENT_TRADE_BLOTTER` procedure, which takes batch date as an optional input parameter. If batch date is not specified, the procedure operates on the current business date. This procedure iterates through each trade, and calls the `P_INSERT_TRADE_ATTRIBUTE` and `P_UPDATE_REQ_ANALYSIS_FL` procedures.

The database procedure `P_INSERT_TRADE_ATTRIBUTE` contains the logic to assign characteristic codes to a trade. It inserts data in the `KDD_TRADE_ATTRIBUTE` table. The `KDD_TRADE_ATTRIBUTE` table contains the association between the trade (`TRADE_SEQ_ID`) and its characteristic text code (`ATTR_TYPE_CD`).

The database procedure `P_UPDATE_REQ_ANALYSIS_FL` contains the logic to identify trades, which require additional analysis. This procedure updates the `REQ_ANALYSIS_FL` column of the `KDD_TRADE_BLOTTER` table, setting it to *Y* for trades requiring additional analysis.

To augment trade blotter data, run the following command:

`run_augment_trade_blotter.sh <yyyymmdd>`, where `<yyyymmdd>` is an optional input parameter. If batch date `<yyyymmdd>` is not provided, the system takes the current batch date from the `DATA_DUMP_DT` column of the `KDD_PRCSNG_BATCH_CONTROL` table.

The log for this script is written in the `run_stored_procedure.log` file under the `<INSTALL_DIR>/database/db_tools/logs` directory.

This script is a part of the database tools and resides in the `<INSTALL_DIR>/database/db_tools/bin` directory.

Note: This utility can be run anytime after data ingestion of Trade Blotter has been successfully completed.

Score Trade Blotter

There is certain information that must be processed in order for the Alert Management system to be able to display the Trade Blotter data. This includes the score of the trades and the mapping between alerts and trades. The system can determine the maximum score of alerts generated in the same batch cycle associated with a trade as well as determine the alert/trade mappings by the execution of the following shell script:

```
runScoreTradeBlotter.sh
```

Note: This script is part of the Ingestion Manager subsystem and resides in the `<INSTALL_DIR>/ingestion_manager/scripts` directory.

Historical Data Copy

Behavior Detection maintains records that are directly involved with detected behaviors in a set of archive, or ARC, tables. The Historical Data Copy (HDC) process identifies the records against which the current batch's scenario runs generated alerts and copies them to the ARC tables.

The `run_hdc.ksh` and `upd_kdd_review_fin.sh` must run upon completion of all detection and other alert post-processing (for example, scoring and assignment), but before the system ends the batch with the following shell script:

```
end_mantas_batch.sh
```

Note: This script is part of the Database Tools that reside in the `<INSTALL_DIR>/database/db_tools/bin` directory.

The `run_hdc.ksh` shell script manages the HDC process. This process connects to the database as the user that the `truncate.database.username` property identifies in the `<INSTALL_DIR>/database/db_tools/ mantas_cfg/install.cfg` file. This property should identify the *ingest user*, a user in the database with write access to tables in both the Market and Business schemas.

To improve performance, you can adjust two configurable parameters in the `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` file, which Table 73 describes.

Table 73. HDC Configurable Parameters

Parameter	Recommended Value	Descriptions
`hdc.batchsize`	10000	Number of break match key IDs are included in each batch thread for data retrieval.
`hdc.maxthreads`	2x (Number of CPUs)	Maximum number of concurrent threads that HDC uses for retrieving data to tune performance.

By default, Behavior Detection writes log messages for this script in the `<INSTALL_DIR>/database/db_tools/logs/hdc.log` file.

Alert Correlation

Oracle Financial Services Behavior Detection provides a mechanism to correlate alerts to business entities and optionally to each other based on configurable rule sets. This functionality is performed by the Alert Correlation process. Details on configuring the data paths to correlate alerts to business entities as well as information on constructing the rules to correlate alerts to each other is provided in the following sub-sections.

Running the Alert Correlation Job

Alert Correlation is a part of the Behavior Detection subsystem. The system delivers job template group 508 to run the Alert Correlation job (for information on how to run this process though a web service, Refer to the *Oracle Financial Services Services Guide*, Release 6.1.3).

To run an Alert Correlation job, follow the steps:

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

   ```
   start_mantas.sh 508
   ```

   where, `508` is the job template that Oracle Financial Services provides to run the Alert Correlation algorithm.

Understanding Alert Correlation Configuration

As mentioned above, Alert Correlation performs two major tasks—correlating alerts to business entities and correlating alerts to alerts. The second step is optional, and is governed by the `correlate_alerts_to_alerts` job parameter delivered with the template job associated to group 508. If this parameter's value is set to *true* then this step will be performed, and if this value is set to *false* then it will not be performed. The only exception to this is if the *correlate alerts to alerts* feature is not licensed. A license for this feature can be obtained from your engagement representative, and details on enabling the license file can be found in the *Oracle Financial Services Behavior Detection Platform Installation Guide*, Release 6.1.3.

The other job parameter associated with Alert Correlation is *correlation_actions*. This parameter has a value of a comma-delimited list that defines what optional actions to take against a correlation that is found by the *correlate alerts to alerts* task. The currently-supported actions are *prioritize*, which will assign a score to the correlation, and *promote_to_case*, which will promote a correlation to a case. Both actions have associated parameters that are defined and dictated by the rule that generated the correlation (these rule sets are discussed below). Note that the *promote_to_case* action is also a licensable feature (dependent on Integrated Case Management license). The same information as above applies in terms of obtaining and configuring a license file.

Both parameters above can be configured by changing their associated `VALUE_TX` values in the `KDD_PARAM_BINDING` table.

In addition to the job parameters, there is a certain metadata that needs to be in place in order to successfully run Alert Correlation. These include the definitions of the paths used to correlate alerts to business entities and the correlation rules that define the criteria for correlating alerts to alerts, and the parameters associated to any subsequent actions performed (if this step in the process is chosen to be run). Details on this metadata is provided in the following sub-sections.

Business Entity Paths

The business entity paths are currently managed through manual interaction with the `KDD_BUS_NTITY_PATH` and `KDD_BUS_NTITY_PATH_CFG` tables in the FSDM. These tables are populated with a comprehensive set of sample data paths. However, the following information will assist in modifying these paths or adding to them. The structure of the tables is as follows (Table 74):

Table 74. `KDD_BUS_NTITY_PATH` (Metadata Table)

Column Name	Primary Key	Foreign Key	Column Type	Nullable (Y/N)	Default
`PATH_ID`	*		NUMBER(10)	No	
`PATH_NM`			VARCHAR2(50)	No	
`QUERY_DEF_NM`			VARCHAR2(50)	Yes	
`ALERT_FOCUS_ID`		`KDD_CENTRICITY.CNTRY_ID`	NUMBER(10)	Yes	
`MTCHD_TABLE_NM`		`KDD_EJB_NAME.EJB_NM`	VARCHAR2(50)	Yes	
`BUS_NTITY_ID`		`KDD_CENTRICITY.CNTRY_ID`	NUMBER(10)	Yes	

The purpose of this table is to define paths that can be used by the Alert Correlation algorithm to perform the first step in its process, correlating alerts to business entities. The way such a path is established is by first defining whether the origin of the path should be an alert's focus or one of its matched records. This is established by either populating the ALERT_FOCUS_ID column (indicating that the origin should be the focus of the alert), or the MTCHD_TABLE_NM column (indicating that the origin should be a matched record of the alert). The destination of the path (the business entity we are trying to correlate to by executing this path) is defined by the BUS_NTITY_ID column. The actual SQL we would need to execute to establish the relationship between the alert's focus or matched record and this business entity is defined by an element in a query definition file. The QUERY_DEF_NM column corresponds to the element name in the query definition file. The query definition file itself can be found here:

```
{INSTALL_DIR}/behavior_detection/algorithms/share/xml/
querydefs/QBD_BusEntityPaths.xml
```

The PATH_ID and PATH_NM in the table above are simply used to establish unique identifiers for this path.

The above paths may not necessarily apply to all types of alerts, and they may have different levels of importance depending on what types of alerts they are applied to. This variance is defined by a path configuration, which is stored in the KDD_BUS_NTITY_PATH_CFG table. Its structure is as follows:

Table 75. KDD_BUS_NTITY_PATH_CFG (Metadata Table)

Column Name	Primary Key	Foreign Key	Column Type	Nullable (Y/N)	Default
PATH_CFG_ID	*		NUMBER(10)	No	
PATH_ID		KDD_BUS_NTITY_PATH.PATH_ID	NUMBER(10)	No	
SCNRO_ID		KDD_SCNRO.SCNRO_ID	NUMBER(10)	Yes	
SCNRO_CLASS_CD		KDD_SCNRO_CLASS.SCNRO_CLASS_CD	VARCHAR2(3)	Yes	
PRSDNC_NB			NUMBER(10)	Yes	

We can choose to apply the path identified by the PATH_ID in this table to only alerts of a certain scenario or scenario class. This is established by populating either the SCNRO_ID or the SCNRO_CLASS_CD column, respectively. If neither of these columns are populated, this path configuration is considered for an alert of any scenario or scenario class. The "importance" or "strength" of a correlation determined by this path may vary depending on the scenario or scenario class of the alert. This is defined by the PRSDNC_NB (the lower the number, the higher the precedence). A NULL PRSDNC_NB indicates not to apply this PATH_ID to any alerts of this SCNRO_ID or SCNRO_CLASS_CD.

Correlation Rules

Once alerts are correlated to business entities, the alert-to-business entity relationships can be used to correlate alerts to each other. Alerts will be grouped into a correlation if they share common business entities, and if they meet the criteria defined in the Alert Correlation Rules. These rules are managed through the Alert Correlation Rule Migration Utility (see *Alert Correlation Rule Migration Utility*, on page 287). The logic of an Alert Correlation Rule is defined in XML, and the Alert Correlation Rule Migration Utility is responsible for reading this XML from a file, validating it, and inserting it into the `KDD_CORR_RULE` table. The following is an example of the rule logic defined in an Alert Correlation Rule XML file, followed by detailed descriptions of the elements contained in the XML file:

```
<CorrelationRule id="123" name="Possible Identity Theft">
  <MinAlertCount>2</MinAlertCount>
  <PrecedenceThreshold>5</PrecedenceThreshold>
  <AlertAttrOperations>
        <![CDATA[ (BOTH.JRSDCN_CD IN ("AMEA","IND")) OR
        (FROM.SCORE_CT = TO.SCORE_CT) ]]>
  </AlertAttrOperations>
  <Lookback number="1" unit="D"/>
  <Scenarios>
     <Scenario id="234"/>
     <Scenario id="345"/>
  </Scenarios>
  <ExistingCorrelationParams>
     <ExtendFlag>TRUE</ExtendFlag>
     <NonExtendableCaseStatuses>
        <CaseStatus>CCL</CaseStatus>
        <CaseStatus>NVST</CaseStatus>
     </NonExtendableCaseStatuses>
  </ExistingCorrelationParams>
  <Actions>
     <Scoring strategy="MAX" incStrategy="ALERT_COUNT"/>
     <CasePromotion>
       <FocusTypePref>CU,AC</FocusTypePref>
       <AlertCorrAttrOperations>
         <![CDATA[(CORR.BUS_NTITY_ID = 5) AND
         (CORR.PRECEDENCE_NB <= 6)]]>
       </AlertCorrAttrOperations>
       <ExistingCasePromoteLossRcvryData>TRUE
       </ExistingCasePromoteLossRcvryData>
       <Case type="AML" subtype="SURV" subClassTagLevel1="CHK_FRD"
       subClassTagLevel2="ALTD_INST"/>
     </CasePromotion>
  </Actions>
</CorrelationRule>
```

- **MinAlertCount** (*required*): The minimum number of alerts involved in a correlation for it to be considered a valid correlation. The minimum acceptable value is 2.
- **PrecedenceThreshold** (*required*): Number indicating the maximum precedence value that a business entity shared between alerts must have in order to be considered a correlation by this rule. The lower the precedence number the stronger the relationship. Alerts will not be considered for the correlation unless the precedence number associated with the business entity-to-alert is less than or equal to (<=) the value defined.
- **AlertAttrOperations** (*optional*): Defines operations used to further constrain the alerts to be used for correlation. An operation consists of an alert attribute (identified by `ATTR_NM`) compared to a string literal (for example, a *from alert* and *to alert* can be correlated if they both have `JRSDCN_CD = "AMEA"`, represented by `BOTH.JRSDCN IN ("AMEA", "IND"))` above, or an alert attribute compared to the same attribute (for example, a *from alert* and *to alert*

can be correlated if `FROM.SCORE_CT = TO.SCORE_CT`). The set of supported comparison operators are: `=`, `!=`, `<`, `>`, `<=`, `>=`, `IN`, and `NOT IN`. Note that because the `SCNRO_ID` attribute of both alerts and correlations can potentially have multiple values, only the `IN` and `NOT IN` operators should be used in expressions involving `SCNRO_ID`. The rest of the operators can only support a single value operands. Also, there should be no space in the scenario id list specified. For example, `BOTH.SCNRO_ID IN (115600002,114690101)` is correct and `BOTH.SCNRO_ID IN (115600002, 114690101)` is incorrect.

Multiple operations can be strung together by logical `AND` and `OR` operators and operation precedence can be defined with parentheses. Note that the text of an *AlertAttrOperation* must be wrapped in a `CDATA` tag as above to account for any special XML characters contained in the expression (for example, > or <).

- **Lookback** (*optional*): The *number* attribute indicates the number of seconds/minutes/hours/days to look back from the current date/time to create a time window in which to consider alerts for correlation. This is a create timestamp of the alert. The *unit* attribute identifies the unit of the lookback number. Possible values are S, M, H, D, and CM for seconds, minutes, hours, days, and current month, respectively. All of these require a valid number value except for CM, which essentially just makes the lookback the 1st of the current month (for example, if the current date is October 14, we will lookback to October 1 if the CM unit is selected). The create timestamp of the alert is used to determine whether or not an alert falls within the lookback period.

  Note: Do not use a unit less granular than a day in rules intended for batch alerts (S, M, and H are intended for posted alerts). For batch processing, use D or CM as a unit.

- **Scenarios** (*optional*): Identifies the Scenario(s) an alert should have been generated from in order to be considered for a correlation by this rule. If not specified, system will consider all the scenarios.
- **ExistingCorrelationParams** (*required*): Defines the conditions for extending existing correlations. When a new correlation is discovered, it is possible that it is a superset (with only the triggering alert not already included in the existing correlation) of a correlation that has previously been identified. `ExtendFlag` defines whether this correlation rule can result in extending an existing correlation. If this is set to FALSE (do not extend) then a new correlation is created when this condition is identified. If it is set to TRUE then the existing correlation is added to unless it has already been promoted to a case that has a status identified in the `CaseStatus` tags of `NonExtendableCaseStatuses`.
- **Actions** (*optional*): Once correlations are discovered, multiple types of actions can be taken on the correlation. These actions and their associated parameters are defined in between the Actions tags. The current set of possible actions include scoring the correlation and promoting the correlation to a case.
- **Scoring** (*optional*): The *strategy* attribute defines whether the correlation score should be derived from the max of the associated alert scores (`MAX_SCORE`) or the average of the associated alert scores (`AVERAGE_SCORE`). The *incStrategy* attribute provides the option of defining a fixed score to be added to the

correlation score. The possible values can be `ALERT_COUNT` (each additional alert above *MinAlertCount* adds to the score), `SCENARIO_COUNT` (each distinct scenario (starting with the second scenario) involved in the correlation adds to the score), or `NONE` (the score is not incremented above what has already been calculated).

Note: The calculated correlation score is bounded by the values of the *min_correlation_score* and *max_correlation_score* parameters found in the following configuration files:

`<INSTALL_DIR>/behavior_detection/algorithms/mantas_cfg/install.cfg` (for the Alert Correlation batch algorithm)

`<INSTALL_DIR>/services/install.cfg` (for the Alert Correlation step of the PostAlert operation of the Alert Management Service)

- **CasePromotion** (*optional*): Defines the parameters used to determine whether a newly discovered correlation should be promoted to a case. Correlations that are already part of a case (for example, when a correlation is extended) are not considered by this type of rule, except the `ExistingCasePromoteLossRcvryData` element, which determines whether or not to augment the existing case's fraud loss and recovery data with the related data from the new alerts added to the case. Logical operations based on attributes of the correlation (including scenarios involved in the correlation) defined under *AlertCorrAttrOperations* can be used to determine whether or not the correlation should be promoted to a case. The syntax, supported operators, and others are same as that of the *AlertAttrOperations* defined above (including the `CDATA` requirement). If the conditions result in the promotion of a correlation to a case, the resulting type, subtype, subclass tag level 1, and subclass tag level 2 of the case are determined by the *type*, *subtype*, *subClassTagLevel1*, and *subClassTagLevel2* attributes of the Case element. The focus of the case is determined by using the ordered list of preferred business entity types defined in the `FocusTypePref` element. In the example above, if the alerts involved in the associated correlation are correlated to the same `CUSTOMER` then `CUSTOMER` would become the focus of the case. If not, and if they are correlated to the same `ACCOUNT`, `ACCOUNT` would become the focus of the case. If not, the correlation will not be promoted to a case.

Activating or Deactivating Correlation Rules

Running the Alert Correlation job will execute only those correlation rules that are designated as Active. Rules that are designated as Inactive will be ignored and not executed. To deactivate an active correlation rule the correlation rule metadata must be modified to change `KDD_CORR_RULE.STATUS_CD` from a value of `ACT` to `NULL`. To activate an inactive rule, modify `KDD_CORR_RULE.STATUS_CD` from a value of `NULL` to `ACT`. Changes made to the metadata are effective immediately and will be utilized the next time alert correlation is run.

Sample Alert Correlation Rules

Oracle Financial Services Behavior Detection delivers two sample alert correlation rules:

- **Correlated Alerts by Business Entity**: Groups alerts created in the past month based on a common correlated business entity. For example, this rule would correlate all alerts with a business entity-to-alert correlation on customer CU12345 that were created in the past month.
- **Potential Identity Theft**: Groups alerts created in the past seven days that are generated on one or more specified scenarios where the alerts share a common correlated business entity. Specified scenarios are those scenarios which identify behaviors that, in isolation or when considered as a whole, may be indicative of identity theft. For example, this rule would correlate all alerts generated on one or more of the specified scenarios with a business entity-to-alert correlation to CU12345 that were created in the past seven days.

The application installs these sample alert correlation rules in the `<INSTALL_DIR>/database/db_tools/data` directory.

Account Approval/Pre-Trade Approval Tasks

If the Oracle client has implemented the Account Approval (AA)/Pre-Trade Approval (PTA) application, the following processes must be executed to successfully utilize both AA and PTA:

- **Load Employee User Groups**: Captures the employees of the Oracle client along with their system logons.It maps each employee to the Employee User Group, which gives each employee access to the AA and PTA application to submit new account approval and pre-trade approval requests, respectively.
- **Request Assignment**: Determines the group of users responsible for handling account approval requests and pre-trade approval requests.

For this purpose, the following scripts, which is part of the Database Tools that resides in the `<INSTALL_DIR>/database/db_tools/bin directory`, call specific database procedures:

Table 76. AA/PTA Scripts

Task	Script	Database Procedure/Description
Load Employee User Groups	`run_apprvl_load_emp_ug.sh`	`P_PTA_EMP_UG_POPULATION` Maps employees within the Oracle client to the AA/PTA pre-packaged Employee User Group (CREMPLOYEEUG). All employee users provided in the System Logon file with the Source System set to 'MTS' are mapped to CREMPLOYEEUG, which gets captured in the `CSSMS_USR_GROUP_MAP` table. This gives an employee access to the AA/PTA application. To identify all the employees that must have access to AA/PTA, the Employee and System Logon files must be provided. The system logon IDs for all employees within the Oracle client captured in the System Logon file must have the Source System (`SRC_SYS_CD`) field value set to 'MTS'. Refer to the *Oracle Financial Service Behavior Detection Platform Data Interface Specification* for details on how to populate the Employee and System Logon files.
Request Assignment	`run_apprvl_assign_request.sh`	`P_PTA_BA_OWNER_ASGNMT` Auto-assigns newly submitted Account Approval and Pre-Trade Approval requests to a pool of users (POOL). When assigning the requests to a POOL, the module fetches new, unassigned requests and assigns each request to a pool based on the auto-assignment rules defined in CR Default Assignee parameter via the Manage Installation Parameter screen. Refer to the *Oracle Financial Services Behavior Detection Platform Configuration Guide* for details on how to configure the CR Default Assignee parameter.
Auto Approve/Reject PTA Requests	`run_apprvl_auto_apprv_rejct.sh`	`P_PTA_AUTO_APPRV_REJECT` This procedure is used to Auto-Approve and/or Auto-Reject newly submitted Pre-Trade Approval requests. To enable/disable the auto-approve functionality, refer to the CR Pre-Trade Auto Approve parameter via the Manage Installation Parameter screen. To enable/disable the auto-reject functionality, refer to the CR Pre-Trade Auto Reject parameter via the Manage Installation Parameter screen. Refer to the *Oracle Financial Services Behavior Detection Platform Configuration Guide* for details on how to configure the CR Default Assignee parameter.

Before running the AA/PTA application for the first time, the AA/PTA tasks must run in this order:

1. Load Employee User Groups
2. Request Assignment

After both scripts have been successfully executed upon initialization of the AA/PTA application, both scripts can be run independently of each other and how frequent each script is run is determined by the Oracle client.

CHAPTER 7

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
- Batch Export Utility
- Calendar Manager Utility
- Data Retention Manager
- Database Statistics Management
- Flag Duplicate Alerts Utility
- Notification
- Refreshing Temporary Tables
- Truncate Manager
- ETL Process for Threshold Analyzer Utility
- Process to Deactivate Expired Alert Suppression Rules

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 215, 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 223, for more information).
- **Calendar Manager**: Updates calendars in the Oracle Financial Services FSDM system based on predefined business days, holidays, and *days off*, or non-business days (Refer to *Calendar Manager Utility,* on page 236, 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 241, for more information).

- **Database Statistics Management**: Manages statistics in the database (Refer to *Database Statistics Management,* on page 251).
- **Flag Duplicate Alerts**: Enables you to run a script daily after the generation of alerts to identify pairs of alerts that are possible duplicates and adds a system comment to each alert (Refer to *Flag Duplicate Alerts Utility,* on page 253, for more information).
- **Notification**: Enables you to configure users of Alert Management and Case Management to receive UI notifications based upon actions taken on alerts or cases, to which, they are associated or when the alert or case is nearing a due date. (Refer to *Notification,* on page 254, for more information).
- **Refreshing Temporary Tables**: Refreshes temporary tables that the behavior detection process uses and estimates statistics for the newly populated tables (Refer to *Refreshing Temporary Tables,* on page 255, for more information).
- **Truncate Manager**: Truncates tables that require complete replacement of their data (Refer to *Truncate Manager,* on page 258, for more information).

Figure 53 illustrates the frequency with which you use these batch-related database utilities when managing activities: daily, weekly, monthly, annually, or as needed.

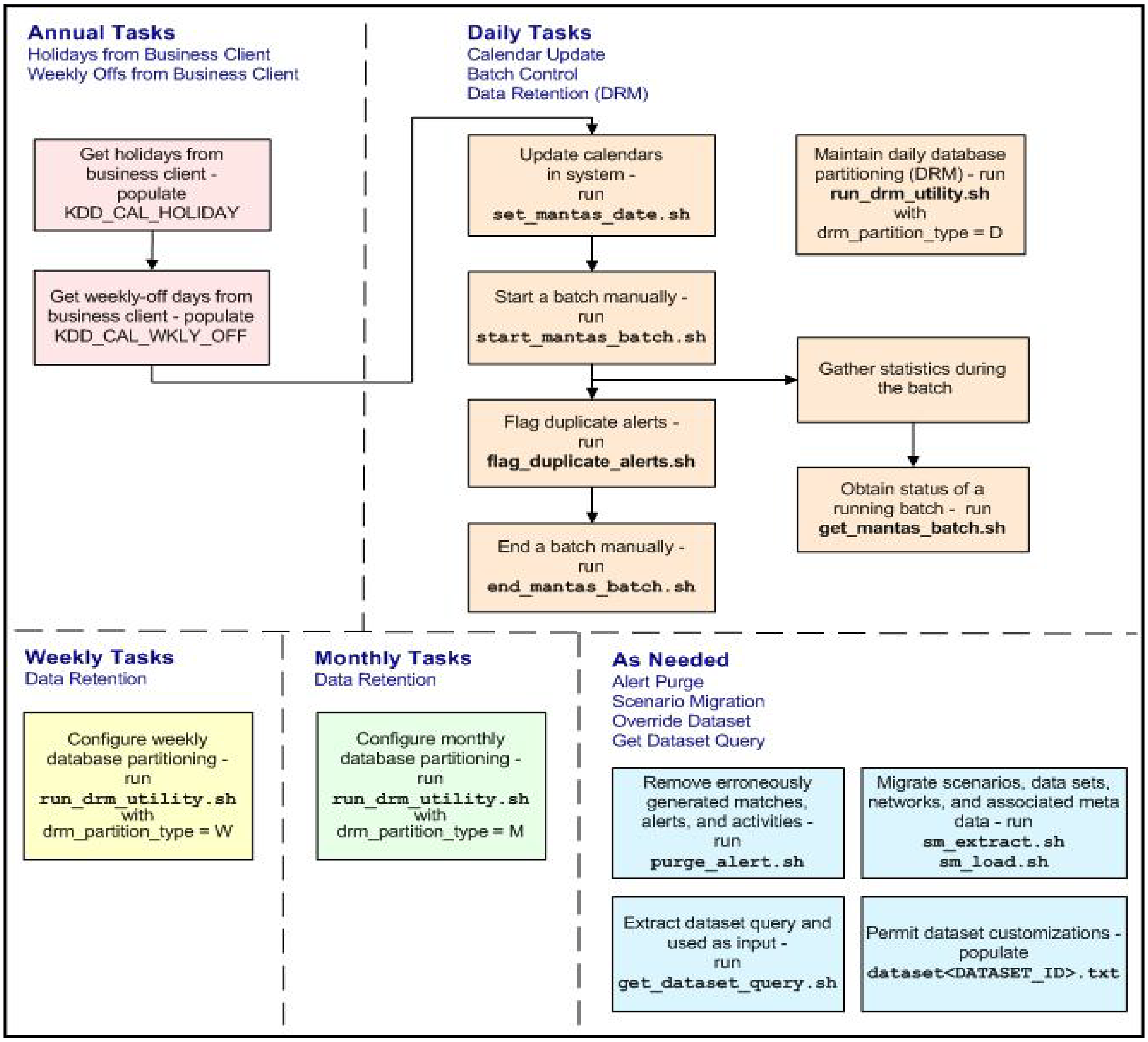

Annual Tasks
Holidays from Business Client
Weekly Offs from Business Client
Get holidays from business client - populate KDD_CAL_HOLIDAY
Get weekly-off days from business client - populate KDD_CAL_WKLY_OFF
Daily Tasks
Calendar Update
Batch Control
Data Retention (DRM)
Update calendars in system - run 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 53. Managing Database Activities with Utilities

As Figure 53 illustrates, daily tasks are initially dependent on the annual tasks that you perform, such as obtaining holiday and weekly off-days from an Oracle Financial Services client. Daily tasks can include updating Behavior Detection calendars and managing batch processes. You may need to configure data partitioning on a daily, weekly, or monthly basis.

Tasks that you perform when needed can include deleting extraneous or invalid matches and alerts, or migrating scenarios and other information from one environment to another (for example, from test to production).

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 (Refer to the following section, *install.cfg File,* on page 198, for more information).
- `categories.cfg` file (Refer to *categories.cfg File,* on page 208, for more information).

`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 54 (which appears on the next several pages) 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=6.1.3

#### 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 54. 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 55 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/mantas6.1.3/database/db_tools/logs/
alert_purge.log
log.category.BATCH_CONTROL.location=/users/orion/mantas6.1.3/database/db_tools/logs/
batch_control.log
log.category.CALENDAR_MANAGER.location=/users/orion/mantas6.1.3/database/db_tools/logs/
calendar_manager.log
log.category.DATA_RETENTION_MANAGER.location=/users/orion/mantas6.1.3/database/db_tools/
logs/DRM_Utility.log
log.category.TRUNCATE_MANAGER.location=/users/orion/mantas6.1.3/database/db_tools/logs/
truncate_manager.log
log.category.COMMON_UTILITIES.location=/users/orion/mantas6.1.3/database/db_tools/logs/
common_utilities.log
log.category.EXTRACT.location=/users/orion/mantas6.1.3/database/db_tools/logs/extract.log
log.category.LOAD.location=/users/orion/mantas6.1.3/database/db_tools/logs/load.log
```

(Continued on next page)

(Continued from previous page)

```
log.category.REFRESH_TEMP_TABLE.location=/users/orion/mantas6.1.3/database/db_tools/logs/
refresh_temp_table.log
log.category.RUN_STORED_PROCEDURE.location=/users/orion/mantas6.1.3/database/db_tools/logs
/
run_stored_procedure.log
log.category.GET_DATASET_QUERY.location=/users/orion/mantas6.1.3/database/db_tools/logs/
get_dataset_query.log
log.category.PUSH_EMAIL.location=/users/orion/mantas6.1.3/database/db_tools/logs/
push_email.log
log.category.HIGHLIGHT_GENERATOR.location=/users/orion/mantas6.1.3/database/db_tools/logs/
highlight_generator.log
log.category.REPORT.location=/users/orion/mantas6.1.3/database/db_tools/logs/report.log
log.category.DATA_ANALYSIS_TOOL.location=/users/orion/mantas6.1.3/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 55. Sample Logging Information in the `categories.cfg` **File**

Configuring Console Output

Figure 56 displays a section of the sample `categories.cfg` file from Figure 55. Note the log routing information in bold text.

```
log.category.ALERT_PURGE.location=console,/users/orion/mantas6.1.3/database/db_tools/logs/
  alert_purge.log
log.category.BATCH_CONTROL.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  batch_control.log
log.category.CALENDAR_MANAGER.location=console,/users/orion/mantas6.1.3/database/db_tools/
  logs/calendar_manager.log
log.category.DATA_RETENTION_MANAGER.location=/users/orion/mantas6.1.3/database/db_tools/
  logs/DRM_Utility.log
log.category.TRUNCATE_MANAGER.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  truncate_manager.log
log.category.COMMON_UTILITIES.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  common_utilities.log
log.category.EXTRACT.location=/users/orion/mantas6.1.3/database/db_tools/logs/extract.log
log.category.LOAD.location=/users/orion/mantas6.1.3/database/db_tools/logs/load.log
log.category.REFRESH_TEMP_TABLE.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  refresh_temp_table.log
log.category.RUN_STORED_PROCEDURE.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  run_stored_procedure.log
log.category.GET_DATASET_QUERY.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  get_dataset_query.log
log.category.PUSH_EMAIL.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  push_email.log
log.category.HIGHLIGHT_GENERATOR.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  highlight_generator.log
log.category.REPORT.location=/users/orion/mantas6.1.3/database/db_tools/logs/report.log
log.category.DATA_ANALYSIS_TOOL.location=/users/orion/mantas6.1.3/database/db_tools/logs/
  data_analysis_tool.log
```

Figure 56. 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 56, 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 the application has the necessary information for populating its own business calendars.

This section covers the following topics:

- Loading Holidays (Refer to *Loading Holidays,* on page 211, for more information).
- Loading Non-business Days (Refer to *Loading Non-business Days,* on page 213, for more information).

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 57 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 57. Sample `KDD_CAL_HOLIDAY` Table Loading Script

Table 77 provides the contents of the `KDD_CAL_HOLIDAY` table.

Table 77. `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 58 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 58. 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 78 provides the contents of the KDD_CAL_WKLY_OFF table.

Table 78. 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 (Refer to *Directory Structure,* on page 215, for more information).
- Logs (Refer to *Logs,* on page 216, for more information).
- Precautions (Refer to *Precautions,* on page 216, for more information).
- Using the Alert Purge Utility (Refer to *Using the Alert Purge Utility,* on page 216, for more information).
- Sample Alert Purge Processes (Refer to *Sample Alert Purge Processes,* on page 221, for more information).

Directory Structure

Table 79 provides the directory structure for the Alert Purge Utility.

Table 79. 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 198, and Appendix A, *Logging*, on page 299, for more information.

Precautions

You use the utility to rid the system of falsely-generated matches and alerts. Other than recorded information in the `<INSTALL_DIR>/database/db_tools/logs/Alert_Purge.log` file, the system does not capture audit information for this process. The utility does not update other alerts' prior counts as a result of purging alerts.

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

Run the Alert Purge Utility:

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

Using the Alert Purge Utility

The Alert Purge Utility is not part of an automated batch process that an application such as Maestro or Unicenter AutoSys controls. You run this manual process only when necessary (Figure 53). 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 (Figure 54). 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 59. 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 80 describes required and optional parameters for this utility.

Table 80. 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 s`cenario, 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.

Table 80. Alert Purge Utility Parameters (Continued)

Parameter	Description
`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.
`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:

- A 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 81 defines the matches that relate to these alerts and actions:

Table 81. 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 81 (Refer to section *Sample Purge Alerts Utility Run: Situation One*, on page 222, for more information).
- Sample Purge Alerts Utility Run: Situation Two shows how to purge all matches in the first job in Table 81, regardless of their alert affiliation (Refer to section *Sample Purge Alerts Utility Run: Situation Two*, on page 222, 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 222, for more information).

Sample Purge Alerts Utility Run: Situation One

To purge only those alerts that contain the first job in Table 81, 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 81, 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 198, 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 or with multiple batches. You can use multiple batches to permit intra-day processing to generate results several times per day, or to separate processing based on servicing multiple time zones.

Behavior Detection provides two types of batches:

- **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).

- **Intra-day**: Used when loading data between end-of-day batches to obtain more frequent detection results. For example, running a batch of trading-compliance scenarios at 10:00 A.M. can identify behaviors relevant to the opening of the market without waiting for the end of the day to be able to act.

Directory Structure

Table 82 provides the directory structure for the Batch Control Utility, in `<INSTALL_DIR>/database/db_tools/`:

Table 82. 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 198, and Appendix A, *Logging*, on page 299, 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 54 on page 207). 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 60. 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 93 on page 239, 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

The Oracle Financial Services FSDM 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 83 provides the contents of the KDD_PRCSNG_BATCH table. (Refer to the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide*, Volume 2, for more information about this table.)

Table 83. 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 84.

Table 84. 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 85 describes.

Table 85. 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 236, 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 241, 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 86 describes.

Table 86. `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 210, 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 86).

2. The utility determines whether a batch is running by querying the `KDD_PRCSNG_BATCH_CONTROL` table (Table 87).

Table 87. `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 88 describes the `KDD_PRCSNG_BATCH_HIST` table.

Table 88. `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 61. 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 61. 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 83 on page 226).

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 87 on page 228).

2. Records information about the batch in the `KDD_PRCSNG_BATCH_` HIST table (Refer to Table 88 on page 228). This table contains a history of all batches that appear by start date and end date. Figure 62 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 62. `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 210, 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 87 on page 228).
2. The utility returns the batch name to standard output.

Batch Export Utility

The Batch Export utility allows Oracle Financial Services clients to export batches of alerts and cases for archival purposes. Clients can now export groups of alerts and cases to a configured location. The exported data includes XML metadata for easier searching of the archived documents, and attachments associated with the investigation record.

Exporting alerts and cases are accomplished in a batch process (by running a script). Filters can be configured to define the alerts and cases that are exported.

The batch export process is initiated by running the script:
`run_pdf_archival_utility.sh`

Note: Prior to using the export functionality, you must have at least one alert or case, your export directory location must be configured, and the Alert Management UI must be running.

Overview

1. The `run_pdf_archival_utility.sh` script initiates a call to the Investigation Export Web Service.
2. Using standard logging procedures, the results of the Batch Export are noted to a log file (`pdf_archival.log`).
3. If export filter criteria has not been defined for specific alert or case IDs, then the system updates the last run date with a date and timestamp at the completion of the process.
4. The Web service stores the exported data in the configured directory.

   Exported data includes multiple file types: XML metadata, PDF alert/case information, and other files (such as, document or spreadsheet attachments).

Note: The export Web service stores the exported data in a configured location within your system. For security purposes, clients should restrict user access to the configured directory.

Directory Structure

Each PDF is stored in a configured directory location. Table 89 provides the directory structure for the exported data in `<INSTALL_DIR>/database/db_tools/`

.

Table 89. Directory Components

Directory Components	Description
`alert_mgmt/WEB-INF/classes/conf/mantas_cfg/install.cfg`	Property to define the directory system where the exported data is stored in the absolute path: `investigation.pdf.export.dir`
Child Directory	Each PDF is exported to a child directory labeled with that alert or case identifier as well as the date and time. (Date and time of the batch action.) This is under the directory defined by: `investigation.pdf.export.dir`
	The directory name format: ● `Alert_[AlertID]_Date_Time` ● `Case_[Case ID]_Date_Time`
	Date format: `YYYYMMDD` Time format: `HH_MM_SS`

Table 89. Directory Components (Continued)

Directory Components	Description
Child Directory Contents	The PDF file name format: ● `Alert_[AlertID].pdf` ● `Case_[CaseID].pdf`

	`Alert_[ALERTID]_info.xml`
	`Case_[CASEID]_info.xml`

	Document or spreadsheet attachments

Configuration Parameters

`<INSTALL_DIR>/alert_management/alert_mgmt/WEB-INF/classes/conf/mantas_cfg/install.cfg:`

Exported data is stored in the absolute path: `investigation.pdf.export.dir`

`<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg`

- Set the maximum number of pdf export threads:
  `pdf.archival.maxthreads=3`
- Number of alerts/cases per export web service call:
  `pdf.archival.service.batchsize=5`
- URL of the Alert Management service:

```
alertmanagement.service.url=http://<alert-management-server>:
<server-port>/<mantas-context>/services/AlertManagementService
```

For example, if the alert management subsystem is running on a server prod1.mantas.com and server port is 1001, then alertmanagement.service.url should be set to
`http://prod1.mantas.com:1001/mantas/services/AlertManagementService`

Using Batch Export

To start a Batch Export, follow these steps:

1. Verify that the Alert Management UI is running.
2. Verify that the
   `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg`

   configuration file contains the correct source database connection information and correct configuration information for this utility.
3. Execute the script `run_pdf_archival_utility.sh` located in the `db_tools/bin` directory.
4. Upon completion of a successful export you will receive a confirmation message. If the data was not successfully exported, then you will receive an error message with details of the error.

Script Details

The `run_pdf_archival_utility.sh script` (default) calls the Investigation Export Web service, which runs under the Alert Management UI to export alert and case information.

This script exports all alerts and cases that were closed after the last execution of the utility (without any parameters set).

`run_pdf_archival_utility.sh` [`ALERT_ID`=<comma-separate alert ids>] [`CASE_ID`=<comma-separate case ids>]

Example Usage

The `ALERT_ID` and `CASE_ID` are optional. If none of these are passed to the script (`run_pdf_archival_utility.sh`), then the alerts and cases that were closed after the last execution of the script are archived, as in the following example:

```
run_pdf_archival_utility.sh
```

When either an `ALERT_ID` or `CASE_ID` or both are passed, then the corresponding alerts and cases are archived, as in the following examples:

1. `run_pdf_archival_utility.sh ALERT_ID=1234,3435,4354,3454`
2. `run_pdf_archival_utility.sh CASE_ID=8999,8984,9800,8498`
3. `run_pdf_archival_utility.sh ALERT_ID=1234,3435,4354,3454 CASE_ID=8999,8984,9800,8498`

Note: Running the script with `ALERT_ID` or `CASE_ID` or both, does not update the last execution date. The `ALERT_ID` and `CASE_ID` should be numbers only. Use `KDD_REVIEW.REVIEW_ID` values as alert or case ID.

Attachments

Attachments associated with the alerts and cases being exported are copied to the child directory corresponding to the alert or case to which they are associated.

XML Metadata

The exported data includes XML metadata for easier searching of the archived documents.

The XML file name uses the following file formats:

- `Alert_[AlertID]_info.xml`
- `Case_[CaseID]_info.xml`

Note: Alert metadata xsd:
`<alert-mgmt>/WEB-INF/classes\conf/ui_config/dtd/mantasAlertExportData.xsd`

Table 90. Alert Metadata

Alert Metadata	
System Name	Hard-coded to Oracle Financial Services Behavior Detection
PDF	File name
Element – ExportDate	System Date

Table 90. Alert Metadata (Continued)

Alert Metadata	
Element – AlertID	KDD_REVIEW.REVIEW_ID
Element – CreationDate	KDD_REVIEW.CREAT_TS
Element – CurrentStatus	KDD_REVIEW_STATUS.STATUS_CD
Element – FocalEntityDisplayID	KDD_REVIEW.FOCAL_NTITY_DSPLY_ID
Element – FocalEntityName	KDD_REVIEW.FOCAL_NTITY_DSPLY_NM
Element – FocusTypeCode	KDD_CENTRICITY.CNTRY_TYPE_CD
Element – ScenarioDisplayName	KDD_REVIEW.SCNRO_DISPL_NM
Element – ScenarioClassCode	KDD_REVIEW.SCNRO_CLASS_CD
Element – OwnerUserName	KDD_REVIEW_OWNER.OWNER_ID
Element – ClosingAction	KDD_REVIEW.CLS_ACTVY_TYPE_CD
Element – CloserUserName	KDD_REVIEW_OWNER.OWNER_ID
Element – DateEnteredStatus	KDD_REVIEW.STATUS_DT
Element – AssociatedAccountServiceTeamID	KDD_REVIEW_FINANCIAL.SRVC_TEAM_INTRL_ID
Element – AssociatedAccountRepresentativeID	KDD_REVIEW_FINANCIAL.RGSTD_REP_ID
Element – AssociatedAccountRepresentativeName	KDD_REVIEW_FINANCIAL.RGSTD_REP_NM
Element – AssociatedAccountRepresentativeLineOrganizationID	KDD_REVIEW_FINANCIAL.BRNCH_INTRL_ID
Element – AssociatedAccountRepresentativeLineOrganizationName	KDD_REVIEW_FINANCIAL.BRNCH_NM
Element – AssociatedAccountRepresentativeSupervisoryOrganizationID	KDD_REVIEW_FINANCIAL.SPRVSRY_ORG_INTRL_ID
Element – AssociatedAccountRepresentativeSupervisoryOrganizationName	KDD_REVIEW_FINANCIAL.SPRVSRY_ORG_NM
Element – AssociatedSecurityID	KDD_REVIEW_FINANCIAL.SCRTY_INTRL_ID
Element – AssociatedSecurityName	KDD_REVIEW_FINANCIAL.SCRTY_DSPLY_NM
Element – AssociatedTraderID	KDD_REVIEW_FINANCIAL.TRDR_INTRL_ID
Element – AssociatedTraderName	KDD_REVIEW_FINANCIAL.TRDR_DSPLY_NM
Element – AssociatedInvestmentManagerID	KDD_REVIEW_FINANCIAL.NVSMT_MGR_INTRL_ID
Element – AssociatedInvestmentManagerName	KDD_REVIEW_FINANCIAL.NVSMT_MGR_DSPLY_NM
***Element – Actions:**	
ActionName	KDD_ACTIVITY_TYPE_CD.ACTVY_TYPE_SHORT_NM
DateActionTaken	KDD_ACTIVITY.START_DT
UserNameTakingAction	KDD_REVIEW_OWNER.OWNER_ID
OwnerUserName	KDD_REVIEW_OWNER.OWNER_ID
AlertStatus	KDD_REVIEW_STATUS.STATUS_CD
Element – Scenarios:	
ScenarioName	KDD_SCNRO.SCNRO_SHORT_NM
ScenarioCatalogID	KDD_SCNRO.SCNRO_CTLG_ID

*Covers all actions taken on alert, not just the closing action.

Note: Case metadata xsd:
`<alert-mgmt>/WEB-INF/classes/conf/ui_config/dtd/mantasCaseExportData.xsd`

Table 91. Case Metadata

Case Metadata	
System Name	Hard-coded to Oracle Financial Services Behavior Detection
PDF	File name
Element – ExportDate	System Date
Element – CaseID	`KDD_REVIEW.REVIEW_ID`
Element – CreatorUserName	`KDD_REVIEW_OWNER.OWNER_ID`
Element – CreationDate	`KDD_REVIEW.CREAT_TS`
Element – CurrentStatus	`KDD_REVIEW_STATUS.STATUS_CD`
Element – FocalEntityDisplayID	`KDD_REVIEW.FOCAL_NTITY_DSPLY_ID`
Element – FocalEntityName	`KDD_REVIEW.FOCAL_NTITY_DSPLY_NM`
Element – FocalTypeCode	`KDD_CENTRICITY.CNTRY_TYPE_CD`
Element – CaseType	`KDD_CASE_TYPE.CASE_TYPE_NM`
Element – CaseSubType	`KDD_CASE_TYPE.CASE_SUB_TYPE_NM`
Element – CaseSubClass1	`KDD_CASE_SUBCLASS_TAG.CASE_SUB_CLASS_LVL1_CD`
Element – CaseSubClass2	`KDD_CASE_SUBCLASS_TAG.CASE_SUB_CLASS_LVL2_CD`
Element – CaseTitle	`KDD_REVIEW_CASE.CASE_TITLE`
Element – CaseDescription	`KDD_REVIEW_CASE.CASE_DESCRIPTION`
Element – OwnerUserName	`KDD_REVIEW_OWNER.OWNER_ID`
Element – ClosingAction	`KDD_REVIEW.CLS_ACTVY_TYPE_CD`
Element – CloserUserName	`KDD_REVIEW_OWNER.OWNER_ID`
Element – DateEnteredStatus	`KDD_REVIEW.STATUS_DT`
Element - CreatorUserName	`KDD_REVIEW_OWNER.OWNER_ID`
Element - OwnerUserName	`KDD_REVIEW_OWNER.OWNER_DSPLY_NM`
***Element – Actions:**	
ActionName	`KDD_ACTIVITY_TYPE_CD.ACTVY_TYPE_SHORT_NM`
DateActionTaken	`KDD_ACTIVITY.START_DT`
UserNameTakingAction	`KDD_REVIEW_OWNER.OWNER_ID`
OwnerUserName	`KDD_REVIEW_OWNER.OWNER_ID`
CaseStatus	`KDD_REVIEW_STATUS.STATUS_CD`
Element – Linked Alerts:	
Element – AlertID	`KDD_REVIEW_LINK.END_REVIEW_ID`

*Covers all actions taken on case, not just the closing action.

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 system calendars. 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 198, for more information).

This section contains the following topics:

- Directory Structure (Refer to *Directory Structure,* on page 236, for more information).
- Logs (Refer to *Logs,* on page 236, for more information).
- Calendar Information (Refer to *Calendar Information,* on page 237, for more information).
- Using the Calendar Manager Utility (Refer to *Using the Calendar Manager Utility,* on page 237, for more information).

Directory Structure

Table 92 provides the directory structure for the Calendar Manager Utility, in `<INSTALL_DIR>/database/db_tools/`.

Table 92. 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 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 198, and Appendix A, *Logging*, on page 299, for more information.

Calendar Information

The Calendar Manager Utility obtains all holidays and weekly off-days for loading into the calendars by retrieving information from the `KDD_CAL_HOLIDAY` and `KDD_CAL_WKLY_OFF` tables (Refer to Table 77 and Table 78). 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 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 54). 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
```

- `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 and case age in Business Days, the calendar date of the current system date and the calendar date of the alert or case 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 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. 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 210, for more information.

Updating the `KDD_CAL` Table

As previously discussed, the Calendar Manager Utility retrieves information that it needs for updating business calendars from the `KDD_CAL_HOLIDAY` and `KDD_CAL_WKLY_OFF` database tables. It then populates the `KDD_CAL` table accordingly (Refer to the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide*, Volume 2, for more information). That is, for each calendar name found in the `KDD_CAL_WKLY_OFF` and `KDD_CAL_HOLIDAY` tables, the utility creates entries in `KDD_CAL`.

Table 93 provides the contents of the KDD_CAL table.

Table 93. 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 93. `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 53 on page 197). 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 94 provides the directory structure for the Data Retention Manager.

Table 94. 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

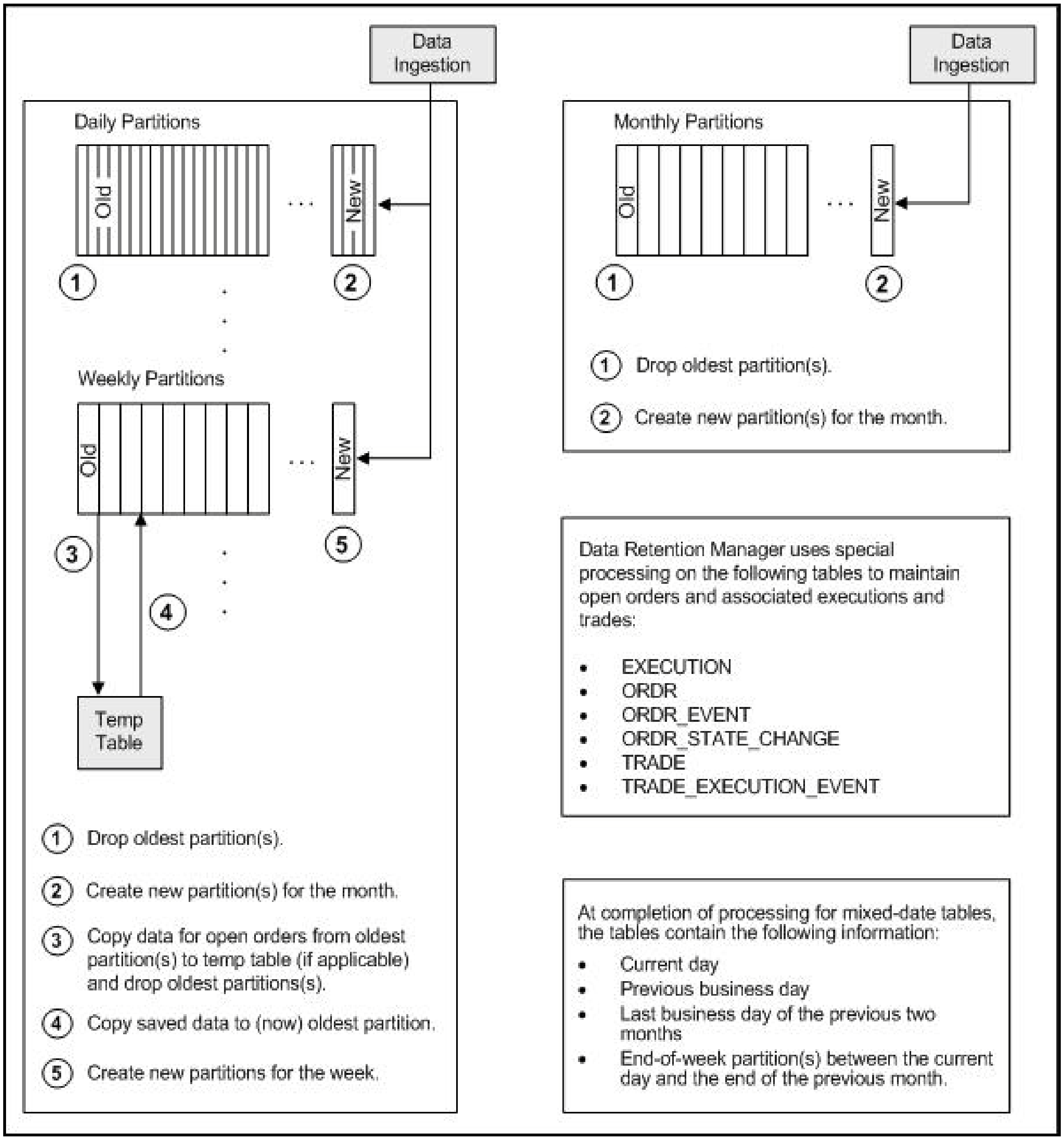
Oracle stored procedures implement Data Retention Manager and conducts some logging on the database server. A configuration parameter in the `install.cfg` file controls the path to which you store the logs on the database server.

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 198, and Appendix A, *Logging*, on page 299, for more information.

Processing Flow

Figure 63 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
Daily Partitions
Old
New
1
2
Weekly Partitions
Old
New
3
4
5
Temp Table
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.
Data Ingestion
Monthly Partitions
Old
New
1
2
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 63. 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 54 on page 207 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.
```

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

Table 95 describes these parameters.

Table 95. 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 to *Calendar Manager Utility,* on page 236, for more information).

To Run the Data Retention Manager

To run Data Retention Manager manually, follow these steps:

1. Access the directory where the shell script resides:

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

2. Start the batch shell script with the parameters in Table 95:

   ```
   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 96.

Table 96. 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 95 on page 245).

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 97 on page 249, 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 236, 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 Table 95 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 Table 95 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 97 describes the table's contents.

Table 97. `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 97. `BUSINESS.KDD_DR_MAINT_OPRTN` Table Contents (Continued)

Column Name	Description
`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 98 provides the table's contents.

Table 98. `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 99 describes the table's contents.

Table 99. `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 93 on page 239 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.

Note: For Case Management Schema, there is no separate script for managing Oracle database statistics. But for improved query performance, we have to manage the Oracle database statistics periodically. Following are the sample commands.

To analyze table wise use, use the following commands:

```
ANALYZE table <Table name> compute statistics;
```

Example: `ANALYZE table KDD_CASES compute statistics;`

We can also perform whole schema analyze periodically.

Flag Duplicate Alerts Utility

The Flag Duplicate Alerts Utility enables you to run a script daily after the generation of alerts. This script identifies the pairs of alerts that are possible duplicates. It then adds a system comment to each alert and identifies the paired alert in the comment as a *Possible Duplicate*.

External Entity-focused scenarios in Behavior Detection can generate alerts either on external identifiers (for example, external account ID) or on names of parties outside the bank. The logic of the scenarios only generates the name-focused alerts when the name has been found with multiple (or no) external identifiers. This check is made across all transactions, not just the transactions involved in a particular alert. As a result, a single run of an External Entity-focused scenario can generate alerts involving the exact same transactions, one alert focused on the external Party ID, and one alert focused on the external Party Name.

Using the Flag Duplicate Alerts Utility

The Flag Duplicate Alerts Utility looks at alerts that meet the following criteria:

- Entity focus (EN)
- Status of New (NW)
- Generated in the current running batch on the current date

The utility selects and compares alerts that meet the listed criteria above. It then determines whether generation of the alert is based on the same set of transactions for the same scenario, with different focuses (for example, one alert is an ID and the other is a Name). The utility flags these alerts as possible duplicates and adds a system comment in the Action History section of the Alert Details page (each alert cross-references the other). For example:

`Possible duplicate of alert` **`xxxxx`**.

Executing the Flag Duplicate Alerts Utility

Use the following procedure to execute the Flag Duplicate Alerts Utility.

To Execute the Flag Duplicate Alerts Utility

To execute the Flag Duplicate Alerts Utility, run the following script after the Alert Creator, Assigner, and Auto-Close processes (jobs) have completed:

```
<INSTALL_DIR>/database/db_tools/bin/flag_duplicate_alerts.sh
```

The system writes log information for this process to the following location:

```
<INSTALL_DIR>/database/db_tools/logs/run_stored_procedure.log
```

Notification

Notifications appear on the UI on the Home page and help alert users to items requiring their attention.

Notifications can be classified into two categories (depending on the method of generation):

- Event Based
- Batch Based

Event Based

These notifications are always associated with an event. Following are the event based notifications:

- **New Case Creation notification**: Whenever a user manually creates a new case, a notification is generated to the owner of the case and if owner is a pool then notification is generated to all the users who fall under that pool. If the user who created the case is also assigned as the owner, no notification is generated.
- **Re-assigned case notification**: Notification is generated to new owner of the case upon reassignment of the case. If the user who reassigned the case is also the new owner, no notification is generated. If the new owner is a pool then notification is generated to all users who are members of the organization represented by that pool.
- **Re-assigned alerts notification**: Notification is generated to the new owner of the Alert upon reassignment of the alert. If the user who reassigned the alert is also the new owner, no notification is generated. If the new owner is a pool then notification is generated to all users who are members of the organization represented by that pool.
- **Alert Data Transfer Unsuccessful**: In Asynchronous alert data transfer mode if there is Unsuccessful data transfer during promotion of an alert to a case or linking of an alert to a case, then notification is generated to the User who is taking the action, the owner of the alert, the owner of the case, and the assigned to user of the case.

Batch Based

These notification are result of processing of `end_mantas_batch.sh`.. Following are the batch based notifications:

- **Cases Near Due Date notification**: Notification is generated to the owner of the cases if the due date of the case falls within the configurable parameter set in the Installation parameter table.
- **Alerts Near Due Date notifications**: Notification is generated to the owner of the alerts if the due date of the alert falls within the configurable parameter set in Installation parameter table.

The above notifications are generated after the complete execution of Batch (provide the batch name) and can be seen in the Notification Grid in landing page. Each user sees the notification which is relevant to him.

Note: User can set the parameter of near due date and display of notification in `KDD_INSTALL_PARAMS` table (Refer to the *Configuration guide,* for more information).

Refreshing Temporary Tables

Some behavior detection patterns use the temporary tables as part of the detection process.

Logs

The `log.category.REFRESH_TEMP_TABLE.location` property in the `<INSTALL_DIR>/database/db_tools/mantas_cfg/categories.cfg` file controls logging for this process. The system writes log information for this process to the following location:

```
<INSTALL_DIR>/database/db_tools/logs/refresh_temp_table.log
```

Using Refreshing Temporary Tables

The MINER schema defines these tables; the tables have corresponding views that are used to populate them. Prior to running these patterns, run the `refresh_temp_table.sh` script. The script has the following calling signature:

```
refresh_temp_table.sh <table_name> <view_name>
```

where:

- `table_name` identifies the name of the table to populate.
- `view_name` identifies the name of the view to run to populate the table.

This procedure deletes all records in the target table prior to running the view to populate it. It then estimates statistics for the newly populated table. This procedure logs into the database with the user that the `utils.miner.user` property identifies in the `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` file.

Populate Temporary Tables for Scenarios

Scenarios typically depend on Data Ingestion to complete processing, however the IML-HiddenRelationships-dINST, ML-NetworkOfAcEn-fAC, and CST-LOSSES scenarios depend on population of Temp Tables to populate data. The Link Analysis scenario also depends on the network job creation before the sequence matcher part of the scenario runs.

IML-HiddenRelationships-dINST

To populate the temporary tables for IML-HiddenRelationships-dINST scenario, follow the steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel):

```
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_JRNL
TMP_HIDREL_NT_JRNL_VW
```

```
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_WIRE
TMP_HIDREL_NT_WIRE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_ACTAXID
TMP_HIDREL_NT_ACTAXID_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_ACADDR
TMP_HIDREL_NT_ACADDR_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_ACPHONE
TMP_HIDREL_NT_ACPHONE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_ACEMAIL
TMP_HIDREL_NT_ACEMAIL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_ACPSWRD
TMP_HIDREL_NT_ACPSWRD_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_INST
TMP_HIDREL_NT_INST_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_WIREACBENE
TMP_HIDREL_NT_WIREACBENE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_WIREACORIG
TMP_HIDREL_NT_WIREACORIG_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_CUACTAXID
TMP_HIDREL_NT_CUACTAXID_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_CUACADDR
TMP_HIDREL_NT_CUACADDR_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_CUACPHONE
TMP_HIDREL_NT_CUACPHONE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_HIDREL_NT_CUACEMAIL
TMP_HIDREL_NT_CUACEMAIL_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698616.
3. Execute the scenario job. The product job template ID is 116200024.

ML-NetworkOfAcEn-fAC

To populate the temporary tables for ML-NetworkOfAcEn-fAC scenario, follow the steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel):

```
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTADDR
TMP_NETACENCU_NT_ACCTADDR_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTEMAIL
TMP_NETACENCU_NT_ACCTEMAIL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTPHONE
TMP_NETACENCU_NT_ACCTPHONE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTPSWRD
TMP_NETACENCU_NT_ACCTPSWRD_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTTAXID
TMP_NETACENCU_NT_ACCTTAXID_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACADDR
TMP_NETACENCU_NT_CUACADDR_VW
```

```
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACEMAIL
TMP_NETACENCU_NT_CUACEMAIL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACPHONE
TMP_NETACENCU_NT_CUACPHONE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACTAXID
TMP_NETACENCU_NT_CUACTAXID_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_JRNL
TMP_NETACENCU_NT_JRNL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_WIREACBENE
TMP_NETACENCU_NT_WIREACBENE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_WIREACORIG
TMP_NETACENCU_NT_WIREACORIG_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_WIRETRXN
TMP_NETACENCU_NT_WIRETRXN_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698120.
3. Execute the scenario job. The product job template ID is 114698631.

FR-NetworkOfAcEn-fAC

To populate the temporary tables for FR-NetworkOfAcEn-fAC scenario, follow these steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel.):

```
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTADDR
TMP_NETACENCU_NT_ACCTADDR_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTEMAIL
TMP_NETACENCU_NT_ACCTEMAIL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTPHONE
TMP_NETACENCU_NT_ACCTPHONE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTPSWRD
TMP_NETACENCU_NT_ACCTPSWRD_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_ACCTTAXID
TMP_NETACENCU_NT_ACCTTAXID_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACADDR
TMP_NETACENCU_NT_CUACADDR_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACEMAIL
TMP_NETACENCU_NT_CUACEMAIL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACPHONE
TMP_NETACENCU_NT_CUACPHONE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_CUACTAXID
TMP_NETACENCU_NT_CUACTAXID_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_JRNL
TMP_NETACENCU_NT_JRNL_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_WIREACBENE
TMP_NETACENCU_NT_WIREACBENE_VW
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_WIREACORIG
TMP_NETACENCU_NT_WIREACORIG_VW
```

```
$DB_TOOLS/refresh_temp_table.sh TMP_NETACENCU_NT_WIRETRXN
TMP_NETACENCU_NT_WIRETRXN_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698120.
3. Execute the scenario job. The product job template ID is 117350084.

CST-Losses

To populate the temporary tables for CST-LOSSES scenario, follow the steps:

1. Execute this refresh temporary table process:

   ```
   $DB_TOOLS/refresh_temp_table.sh VWCST_lOSSES_AC_ASM_TMP
   VWCST_lOSSES_AC_ASM
   ```

2. Execute the link analysis/network generation job.
3. Execute the scenario job.

Truncate Manager

The Data Ingestion subsystem calls the `run_truncate_manager.sh` script to truncate tables that require complete replacement of their data.

Logs

The `log.category.TRUNCATE_MANAGER.location` property in the `<INSTALL_DIR>/database/db_tools/mantas_cfg/categories.cfg` file controls logging for this utility. The system writes log information for this process to the following location:

```
<INSTALL_DIR>/database/db_tools/logs/truncate_manager.log
```

Using the Truncate Manager

The `run_truncate_manager.sh` script takes the table name as an argument; the table must exist in the BUSINESS schema (which the `schema.business.owner` property identifies). The script logs into the database using the user that the `truncate.database.username` property specifies in the `<INSTALL_DIR>/database/db_tools/ mantas_cfg/install.cfg` file.

The script has the following calling signature:

```
run_truncate_manager.sh <table_name>
```

Note: This process is not intended to be called independently; only the Ingestion Manager subsystem should use it.

ETL Process for Threshold Analyzer Utility

For inserting and updating records into the `KDD_TA_ML_DATA`, `KDD_TA_BC_DATA`, and `KDD_TA_TC_DATA` tables, there are two shell scripts that are used to call the database procedures. These are:

- `run_insert_ta_utility.sh` – This script calls the `P_TA_ML_INSERT_BREAKS`, `P_TA_BC_INSERT_BREAKS`, and `P_TA_TC_INSERT_BREAKS` procedures, which insert data into the `KDD_TA_ML_DATA`, `KDD_TA_BC_DATA`, and `KDD_TA_TC_DATA` tables, respectively, based on the `CREAT_TS` of the alerts in relation to the `LAST_RUN_DT` from `KDD_TA_LAST_RUN` (values for `RUN_TYPE_CD` are `ML_I`, `BC_I`, and `TC_I`). There is one optional parameter (`DEBUG_FL`) for this shell script (defaults value to `FALSE`). If you provide a value of TRUE as an argument then information (insert commands) is also loaded into the `KDD_TA_INS_DEBUG` table. It also updates the `LAST_RUN_DT` column in the `KDD_TA_LAST_RUN` table (values for `RUN_TYPE_CD` are `ML_I`, `BC_I`, and `TC_I`) with the date (sysdate) the procedure was last run.
- `run_update_ta_utility.sh` – This script calls the `P_TA_ML_UPDATE`, `P_TA_BC_UPDATE`, and `P_TA_TC_UPDATE` procedures, which update `QLTY_RTNG_CD` in the `KDD_TA_ML_DATA, KDD_TA_BC_DATA`, and `KDD_TA_TC_DATA` tables, respectively, for any *Review* closed since the last run based on `LAST_RUN_DT` from `KDD_TA_LAST_RUN` (values for `RUN_TYPE_CD` are `ML_U`, `BC_U`, and `TC_U`). The `CLS_CLASS_CD` value from `KDD_REVIEW` is used as the new `QLTY_RTNG_CD`. There are no parameters needed for this shell script. It also updates the `LAST_RUN_DT` column in the `KDD_TA_LAST_RUN` table (values for `RUN_TYPE_CD` are `ML_U`, `BC_U`, and `TC_U`) with the date (sysdate) the procedure was last run.

The log for these scripts is written in the `run_stored_procedure.log` file under the `<INSTALL_DIR>/database/db_tools/logs` directory.

Note: The `LAST_RUN_DT` column in the `KDD_TA_LAST_RUN` table is only updated for *inserts* and *updates* if at least one or more records were inserted or updated. The `LAST_RUN_DT` column is not updated for significant errors that resulted in no records being updated. These scripts are a part of the database tools and reside in the `<INSTALL_DIR>/database/db_tools/bin` directory.

You can run this utility anytime. (In other words, it is not necessary to run this utility during specific processing activities.)

Process to Deactivate Expired Alert Suppression Rules

The following shell script should be executed in order to deactivate Alert Suppression Rules that have expired based on the current system date:

```
-- run_upd_suppression_recs.sh
```

This script should be run as the last step in batch processing just prior to ending the batch. It is important that this script is run after post-processing has been completed (that is, not before the Alert Suppression job is executed).

CHAPTER 8 Administrative Utilities

Oracle Financial Services Behavior Detection Platform 6.1.3 provides utilities that enable you to set up or modify a selection of database processes. This chapter focuses on the following topics:

- About Administrative Utilities
- Data Analysis Tool
- Get Dataset Query with Thresholds Utility
- Scenario Migration Utility
- Alert Correlation Rule Migration Utility
- Watch List Service
- Watch List Service
- Alert Processing Web Services
- Password Manager Utility

About Administrative Utilities

Several Behavior Detection database utilities that configure and perform system pre-processing and post-processing activities are not tied to the batch process cycle:

- **Data Analysis Tool:** Assists a Data Miner or Data Analyst in determining how well a customer has populated the Production Data Model (Refer to *Data Analysis Tool,* on page 262, for more information).
- **Get Dataset Query with Thresholds:** Enables you to extract dataset SQL complete with substituted thresholds for analysis of the SQL outside of the Behavior Detection application (Refer to *Get Dataset Query with Thresholds Utility,* on page 274, for more information).
- **Scenario Migration:** Extracts scenarios, datasets, networks, and associated metadata from a database to flat files and loads them into another environment (Refer to *Scenario Migration Utility,* on page 275, for more information).

Figure 53 on page 197, illustrates the frequency with which you use these database 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. For information about these resources, and examples, Refer to Chapter 7, *Common Resources for Administrative Utilities*, on page 198, for more information.

Data Analysis Tool

The Data Analysis Tool enables you to determine how well a customer has populated the Production Data Model. By reviewing the quality of data in each of the tables that the schema identifies, the Data Analysis Tool indicates how well the supplied data can support scenarios. The tool does not make "judgments" about data quality. Rather, it provides a repeatable way to run a set of analytical queries across the data. You can then use the results to direct further analysis.

The Data Analysis Tool:

- Counts all table rows in the schema.
- Identifies unique values and their distribution against the table.
- Determines the number of null occurrences for a specified column.
- Determines the number of padded spaces that occur for a specified column.
- Checks referential integrity between tables.

The following sections provide instructions for using the tool:

- Configuring the Data Analysis Tool
- Using the Data Analysis Tool
- Logs
- Troubleshooting the Data Analysis Tool

The tool provides its results in either a text or Hypertext Markup Language (HTML) file. You can then use these results to direct an investigation for data quality.

Note: To use the Data Analysis Tool effectively, you must have basic knowledge of Structured Query Language (SQL) and Extensible Markup Language (XML).

Configuring the Data Analysis Tool

The Data Analysis Tool uses the `install.cfg` and `analysis.xml` (or similar) configuration files. You edit either file in a text editor such as vi. To produce well-formed XML files, however, you should edit the XML file in a validating XML editor.

To Configure General Tool Properties

Behavior Detection deploys the Data Analysis Tool as one of the utilities under the database tools. Basic configuration for these tools is through the `install.cfg` file that resides in `<INSTALL_DIR>/database/db_tools/mantas_cfg`.

Table 100 provides the configuration instructions for the properties that the Data Analysis Tool uses in the `install.cfg` file.

Table 100. Configuration Instructions for the `install.cfg` File

Property	Description	Example
`database.driverName`	Database connection driver that the utility is to use.	`database.driverName =oracle.jdbc.driver.OracleDriver`
`utils.database.urlName`	Database connection string that the Data Analysis Tool is to use.	`utils.database.urlName =jdbc:oracle:oci:@PROD_DB`
`schema.business.owner`	Database user for the BUSINESS schema.	`schema.business.owner=BUSINESS`
`schema.market.owner`	Database user for the MARKET schema.	`schema.market.owner=MARKET`
`dat.database.username`	User name for the database. The Data Analysis Tool connects to the database as the INGEST_USER for the appropriate privileges.	`dat.database.username= INGEST_USER`
`dat.database.password`	Password for the database. This is set by the Password Manager Utility.	
`dat.analysis.input`	Path and name for the XML input file. By default, this is the `analysis.xml` file under the `<install_dir>/database/db_tools/mantas_cfg` directory. You can override this at the command line.	`dat.analysis.input=/opt/mantas/database/db_tools/mantas_cfg/analysis.xml`
`dat.analysis.output`	Path and file name of output file for the analysis report. You can override this at the command line.	`dat.analysis.output=/opt/mantas/database/db_tools/data/ analysis.html`
`dat.output.format`	Output format for the report. Acceptable output formats are HTML or TEXT.	`dat.output.format=HTML`
`dat.output.delimiter`	Not currently used. The delimiter for the format TEXT is always a comma (",").	

For additional information about the `install.cfg` file, Refer to *install.cfg File*, on page 198, for more information.

To Configure the Analysis XML File

The `analysis.xml` configuration file specifies the queries that you can use to analyze the data that the database schema provides. You can perform the following types of queries:

- Distinct Values for Fields of Interest Analysis (Refer to *Distinct Values for Fields of Interest Analysis,* on page 265, for more information).
- Null and Padded Space Count Analysis (Refer to *Null and Padded Space Count Analysis,* on page 266, for more information).
- Join Counts Analysis for referential integrity between two or more tables (Refer to *Join Counts Analysis,* on page 267, for more information).
- Other Queries as configured (Refer to *Other Queries,* on page 270, for more information).

Analysis Constraints

For both distinct value counts and null counts, you can specify optional constraints. The XML format for two of the files is identical. For a join analysis, the XML format uses a filter element that is similar to a constraint. However, you must specify the table name.

To specify a constraint, use the `<CONSTRAINT>` element. The `<CONSTRAINT>` element requires three attributes:

- **Field:** Database field name to which the constraint applies.
- **Value:** Value being compared.
- **Operator:** Operator used in the comparison.

  Table 101 lists valid code operators.

Table 101. XML Code Operators

XML Code Operator	Comparison Operator
GT	>
LT	<
EQ	=
LTE	<=
GTE	>=
NEQ	<>
EMPTY	Blank Character

The following code sample illustrates the use of the `<CONSTRAINT>` element:

```
<CONSTRAINT field="DATA_DUMP_DT" operator="EQ"
value="15-NOV-2006" />
```

To include a constraint that filters out null columns, use the EMPTY operator and set the value to `is not null`. The following example illustrates the use of the EMPTY operator:

```
<CONSTRAINT field="DATA_DUMP_DT" operator="EMPTY" value="is not
null" />
```

You can also use the EMPTY operator to perform more complex comparisons than those that other operators support that Table 101 lists. When using the EMPTY operator, the generated SQL statement includes the field name, a space, and the text within the value string. As such, representation of more complex operations is possible.

An AND operator joins any existing, multiple `<CONSTRAINT>` elements.

When adding date constraints as in the first example above, you must specify the date in the same format as the database's NLS Date Format (Oracle Financial Services recommends `DD-MON-YYYY` as the default format).

Distinct Values for Fields of Interest Analysis

Identifying the table and one or more column combinations of interest provides a combination of distinct values and number of occurrences in the table. The following code illustrates the required structure of this analysis within the following elements:

```
<ANALYSIS>
  <TABLES>
    <analysis for distinct values occurs here>
  </TABLES>
</ANALYSIS>
```

The name attribute of the `<TABLE>` element identifies the table against which this analysis is applied. The `<VALUES>` element identifies targeted columns. The field attribute of the `<COLUMN>` element sets each database column.

Application of filters to an analysis is possible if the `<CONSTRAINT>` element identifies the filter. The following code illustrates the structure for using a filter:

```
<TABLE name="table name">
<!-- get distinct value for one column -->
  <VALUES>
    <COLUMN field="column name"/>
       <!-- Constraint feature is optional.
            May contain one or more constraints. -->
        <CONSTRAINT field="column name" operator="operator"
                        value="filter value" />
  </VALUES>
<!-- get distinct value for many columns -->
  <VALUES>
    <COLUMN field="column name"/>
    <COLUMN field="column name"/>
       <!-- Constraint feature is optional.
            May contain one or more constraints. -->
       <CONSTRAINT field="column name"
        operator="operator"value="filter value" />
  </VALUES>
</TABLE>
```

`The following XML code illustrates use of a filter:`

```
<ANALYSIS>
  <TABLES>
    <TABLE name="ACCT">
      <VALUES>
        <COLUMN field="ACCT_TYPE1_CD"/>
        <COLUMN field="ACCT_TYPE2_CD"/>
      </VALUES>
    </TABLE>
    <TABLE name="CUST">
       <VALUES>
         <COLUMN field="CUST_TYPE_CD"/>
```

```
            <CONSTRAINT field="DATA_DUMP_DT" operator="EQ"
             value="15-NOV-2006" />
          </VALUES>
        </TABLE>
    </TABLES>
<ANALYSIS>
```

This XML code executes the following queries:

```
select ACCT_TYPE1_CD, ACCT_TYPE2_CD, count(1)
from ACCT
group by ACCT_TYPE1_CD, ACCT_TYPE2_CD

select CUST_TYPE_CD, count(1)
from CUST
where DATA_DUMP_DT='15-NOV-2006'
group by CUST_TYPE_CD
```

Null and Padded Space Count Analysis

Null and padded space count analysis provides the number of occurrences for null values and padded spaces for a particular field in a table. You perform this analysis by identifying the table and one or more columns of interest. The null analysis feature has the following limitations:

- The feature is optional.
- The field identified for the specified table can be analyzed only once within the <NULLS> element per table.
- The filtering feature for the null analysis is optional and can have multiple constraints.

The structure to perform this analysis is:

```
<ANALYSIS>
   <TABLES>
     <!-- analysis for null counts occurs here -->
   </TABLES>
</ANALYSIS>
```

Within the <TABLE> element, the name attribute identifies the table to be analyzed. The targeted columns are identified within the <NULLS> element. The field attribute in the <NULL> element sets each column name. Apply filters to the analysis within the <CONSTRAINT> element. The following code illustrates the structure for the a null and padded space count analysis:

```
<TABLE name="table name">
<!-- May contain one or more columns -->
   <NULLS><!-- With no constraints -->
     <NULL field="column name"/><!-- With constraints -->
     <NULL field="column name">
       <!-- Constraint feature is optional.
            May contain one or more constraints. -->
        <CONSTRAINT field="column name" operator="operator"
                         value="filter value" />
     </NULL>
   </NULLS>
</TABLE>
```

The following XML code sample is an example of the correct structure:

```
<TABLE name="ACCT">
   <NULLS>
     <NULL field="ACCT_TYPE1_CD"/>
     <NULL field="RGSTN_TYPE_CD">
       <CONSTRAINT field="DATA_DUMP_DT" operator="EQ"
                          value="15-NOV-2006" />
     </NULL>
   </NULLS>
<TABLE name="ACCT">
```

This code executes the following queries:

```
SELECT sum(case when ACCT_TYPE1_CD is null then 1 else 0 end)as
NULL_CT0,
sum(case when ACCT_TYPE1_CD <> ltrim(rtrim(ACCT_TYPE1_CD))
then 1 else 0 end) as SPACE_CT0,
sum(case when RGSTN_TYPE_CD is null
and DATA_DUMP_DT='15-NOV-2006' then 1 else 0 end) as NULL_CT1,
sum(case when RGSTN_TYPE_CD <> ltrim(rtrim(RGSTN_TYPE_CD))
and DATA_DUMP_DT='15-NOV-2006' then 1 else 0 end) as SPACE_CT1
FROM ACCT a
```

Join Counts Analysis

A join identifies the relationship between two tables by common fields. Checking for join counts determines the referential integrity between two or more tables. Determine join counts as follows:

- Simple join between two or more tables (Refer to *Simple Join,* on page 267, for more information).
- Simple join between two or more tables with filter restriction (Refer to *Simple Join with Filter Restriction,* on page 268, for more information).
- Join count of distinct values for specific column (Refer to *Join Counts Analysis,* on page 267, for more information).

The join count analysis is structured within the following elements:

```
<ANALYSIS>
     <JOINS>
         <!-- analysis for referential integrity here -->
     </JOINS>
</ANALYSIS>
```

Simple Join

A join is set within the <JOIN> element. To retrieve the join count between two or more tables, the joins are identified within the <MULTIJOIN> element. Within this <MULTIJOIN> element, multiple <JOIN> elements can be set.

Because a join retrieves the join count between two or more tables, <LEFT> and <RIGHT> elements are used to indicate the tables. The <LEFT> element identifies the first table and its field using the table and column attributes. The table and column attributes for the <RIGHT> element identify the second table and field. The structure for a simple join count analysis is:

```
<MULTIJOIN>
<!-- May contain more than one JOIN element -->
  <JOIN>
```

```
        <LEFT table="table name" column="column" />
        <RIGHT table="table name" column="column" />
      </JOIN>
    </MULTIJOIN>
```

The following XML code provides an example:

```
<ANALYSIS>
  <JOINS>
   <MULTIJOIN>
     <JOIN>
       <LEFT table="ACCT" column="ACCT_INTRL_ID" />
       <RIGHT table="CUST_ACCT" column="ACCT_INTRL_ID" />
     </JOIN>
   </MULTIJOIN>
   <MULTIJOIN>
     <JOIN>
       <LEFT table="ACCT" column="ACCT_INTRL_ID" />
       <RIGHT table="CUST_ACCT" column="ACCT_INTRL_ID" />
     </JOIN>
     <JOIN>
       <LEFT table="CUST" column="CUST_INTRL_ID" />
       <RIGHT table="CUST_ACCT" column="CUST_INTRL_ID" />
     </JOIN>
   </MULTIJOIN>
  </JOINS>
</ANALYSIS>
```

This XML code executes the following queries:

```
select count(1)
from ACCT a, CUST_ACCT b
where a.ACCT_INTRL_ID=b.ACCT_INTRL_ID

select count(1)
from ACCT a, CUST_ACCT b, CUST c
where a.ACCT_INTRL_ID=b.ACCT_INTRL_ID
and c.CUST_INTRL_ID=b.CUST_INTRL_ID
```

Simple Join with Filter Restriction

Adding a filter to the joins determines the join count between tables with a restriction. A filter uses the table, field, operator, and value attributes to set the restriction. The operator is limited to the XML code operators in Table 101 on page 264, for more information.

The structure is organized in the same manner as a Simple Join with an added <FILTER> element. The following code illustrates the structure:

```
<MULTIJOIN>
   <JOIN>
     <LEFT  table="table name" column="column" />
     <RIGHT table="table name" column="column" />
   </JOIN>
   <!-- Optional. May contain one or more filters. -->
   <FILTER table="table name" column="column" operator=
                 "operator" value="filter value" />
</MULTIJOIN>
```

The <FILTER> element is optional in the join analysis. Multiple filters can be applied to a join. The AND operator is appended to each filter condition upon creation of the query. The following XML code illustrates the use of a filter with a simple join analysis:

```
<ANALYSIS>
  <JOINS>
    <MULTIJOIN>
      <JOIN>
        <LEFT table="ACCT" column="ACCT_INTRL_ID" />
        <RIGHT table="CUST_ACCT" column="ACCT_INTRL_ID" />
      </JOIN>
      <FILTER table="ACCT" column="DATA_DUMP_DT"
                     operator="GTE" value="01-NOV-2006" />
      <FILTER table="ACCT" column="DATA_DUMP_DT"
                     operator="LTE" value="05-NOV-2006" />
    </MULTIJOIN>
  </JOINS>
</ANALYSIS>
```

This code executes the following query:

```
select count(1) from ACCT a, CUST_ACCT b
where a.ACCT_INTRL_ID=b.ACCT_INTRL_ID
and a.DATA_DUMP_DT>='01-NOV-2006' and
a.DATA_DUMP_DT<='05-NOV-2006'
```

To filter for values that are null or not null, set the operator to EMPTY and the value to IS NULL or IS NOT NULL, respectively.

Join Count by Distinct Column

To determine a join count of the number of distinct values for a specified column within the joined tables, include the <DISTINCT_COUNT> element as content to the <MULTIJOIN> element. The targeted table and its column are set to the table and column attributes, respectively. The following sample demonstrates integration of the <DISTINCT_COUNT> element in the analysis:

```
<MULTIJOIN>
  <JOIN>
    <LEFT table="table name" column="column" />
    <RIGHT table="table name" column="column" />
  </JOIN>
  <!-- Optional. Can only have one DISTINCT_COUNT within
       the MULTIJOIN element. -->
  <DISTINCT_COUNT table="table name" column="column" />
</MULTIJOIN>
```

The <DISTINCT_COUNT> element is optional in the join analysis.

The following XML sample code illustrates use of the <DISTINCT_COUNT> element:

```
<ANALYSIS>
  <JOINS>
    <MULTIJOIN>
      <JOIN>
       <LEFT table="ACCT" column="ACCT_INTRL_ID" />
       <RIGHT table="CUST_ACCT" column="ACCT_INTRL_ID" />
```

```
          </JOIN>
          <FILTER table="ACCT" column="DATA_DUMP_DT" operator=
                       "EQ" value="02-NOV-2006" />
         <DISTINCT_COUNT table="ACCT" column="ACCT_TYPE_CD" />
        </MULTIJOIN>
      </JOINS>
    </ANALYSIS>
```

This sample code executes the following query:

```
select count(DISTINCT a.ACCT_TYPE_CD)
from ACCT a, CUST_ACCT b
where a.ACCT_INTRL_ID=b.ACCT_INTRL_ID and
a.DATA_DUMP_DT='02-NOV-2006'
```

Other Queries

The Data Analysis Tool also supports providing SQL queries directly in the analysis XML file. A query has two components: the query title and the query itself. As queries often contain characters that are "reserved" in XML, you should follow the example below for "escaping" the SQL to ensure that it does not become corrupted.

```
<QUERIES>
     <SQLQUERY title="title">
      select col1, col2 from some_table
      where some_condition
     </SQLQUERY>
</QUERIES>
```

The following XML sample code illustrates use of the <QUERIES> element:

```
 <ANALYSIS>
   <QUERIES>
     <SQLQUERY title="FO Transaction Roles"><![CDATA[ select
        FOT.mantas_PRODUCT_TYPE_CD,
        FOTPS.PARTY_ROLE_CD, count(1) as RoleCt
        from FO_TRXN_STAGE FOT, FO_TRXN_PARTY_STAGE FOTPS
        where FOT.TRXN_INTRL_ID = FOTPS.TRXN_INTRL_ID
        group by FOT.mantas_PRODUCT_TYPE_CD, FOTPS.PARTY_ROLE_CD
        order by 1, 2]]></SQLQUERY>
  </QUERIES>
 </ANALYSIS>
```

This code runs the query in the `<SQLQUERY>` element and writes the results to the output file. For SQL queries, the results are always in HTML. Your code can contain any number of `<SQLQUERY>` elements. The system runs each query in sequence after the other components of analysis are complete.

SQLQUERY Element Rules

Several cautions and notes are specific to the <`SQLQUERY`> element:

- If your query contains characters that XML standards reserve (for example, > or <), you must place your query within a CDATA block.
- Verify that no white space exists between the SQL query opening tag and the CDATA tags (for example, `<![CDATA[ ...`) and the closing tag (for example, `...]]>`).

- Processing extracts column headers in the output from the SQL query itself. When performing calculations in return columns, it is best to alias the return columns for output.
- Line breaks and comments in the SQL are acceptable, but you should use /* */ style comments in lieu of single-line comments for safety.
- The tool does not perform any schema-name substitution. Therefore, verify that any schema names match the database contents. The database user (for example, INGEST_USER) has aliases for most tables you may need to analyze. Thus, running the tool as INGEST_USER should prevent you from needing schema names in queries.

Using the Data Analysis Tool

After editing the configuration files, you can run the Data Analysis Tool as a foreground or background process.

Table 102 lists the XML input files delivered for use with the Data Analysis Tool.

Table 102. Data Analysis Tool XML Input Files

File	Description
`analysis_aml.xml`	Analysis configuration specific for data required by Anti-Money Laundering scenarios and Ingestion Manager operations to support them.
`analysis_aml_ui.xml`	Analysis configuration specific for data displayed in support of Anti-Money Laundering scenarios.
`analysis_iaml.xml`	Analysis configuration specific for data required by Institutional Anti-Money Laundering scenarios and Ingestion Manager operations to support them.
`analysis_iaml_ui.xml`	Analysis configuration specific for data displayed in support of Institutional Anti-Money Laundering scenarios.
`analysis_bc.xml`	Analysis configuration specific for data required by Broker Compliance scenarios and Ingestion Manager operations to support them.
`analysis_bc_ui.xml`	Analysis configuration specific for data displayed in support of Broker Compliance scenarios.

You can also create your own files using the provided files as a template. Place files that you create in the `mantas_cfg` directory that the DTD can locate. If you place your files in a different directory, you must modify the DTD reference in the XML files to qualify the path to the DTD.

To Run the Data Analysis Tool

Go to the `<install_dir>/database/db_tools/bin` directory and execute the following command:

```
run_data_analysis_tool.sh [bg] [-i input_file.xml] [-o outputfile]
```

Table 103 describes the command line arguments that the Data Analysis Tool uses.

Table 103. Command Line Arguments

Argument	Explanation
`bg`	If provided, runs the tool in the background. You can then disconnect your Unix or Linux session without interrupting the tool's operation. The system directs any output from the screen to the `nohup.out` file in the directory from which you ran the tool.
`-i input_file`	Uses an input analysis file (Table 102) other than the one that `install.cfg specifies`. Omission of this argument causes the Data Analysis Tool to use the default file in `install.cfg`.
`-o output_file`	Writes the output to a file other than the one that `install.cfg specifies`. Omission of this argument causes the Data Analysis Tool to use the default file in `install.cfg`.

Logs

The Data Analysis Tool writes status and error messages to the configured log file. The default location for this log file is:

```
<install_dir>/database/db_tools/logs/data_analysis_tool.log
```

The system writes any system-type errors that prevent the tool from connecting to or operating this log file. It also writes data errors to the log and includes them in the data analysis report output (Refer to *Understanding the Data Analysis Report*, on page 272, for more information).

Understanding the Data Analysis Report

The tool generates a data analysis report, which resides in the location you specified in the `install.cfg` file or with the command line "`-o`" argument.

Note: Oracle Financial Services recommends that you view the output report using Microsoft Excel because this HTML file has specific HTML formatting for Excel.

Table 104 describes sections of the output report.

Table 104. Data Analysis Report Output

Section	Description
Table Count Summary	Contains the row count of each table in the configured database excluding the KDD, archive, and temp tables.
Field Distribution Summary Table	Groups by table the unique values for the identified fields and number of times each value occurs in the table. This summary table appears only in the report if the analysis for Distinct Values for Fields of Interest and Its Count was configured in the XML file. In addition, quotes enclose any values with padded spaces to identify spaces in the value.
Null Summary Count Table	Groups by table the number of nulls present and values with padded spaces for the identified fields in each table. This summary table only appears in the report if the analysis for Null and Padded Space Count has been configured in the XML file.
Referential Integrity Table Summary	Displays the join analysis, the number of rows returned between the joined tables, and the table count for each table being joined. This summary only appears in the report if the analysis for Join Counts has been configured in the XML file.
Query Results	Displays the results of queries specified in the QUERIES section of the analysis file.
SQL Report	Lists all of the SQL run to produce the other sections of the report.
Error Report	Displays any errors that occurred when any of the queries were performed.

Troubleshooting the Data Analysis Tool

Table 105 lists common Data Analysis Tool errors and their solutions.

Table 105. Troubleshooting Data Analysis Tool Errors

Error Message	Cause	Solution
java.io. FileNotFoundException <path & filename>	The system cannot find the file specified.	Verify the `install.cfg` file indicates the correct path.
java.lang. RuntimeException: Tables `<table 1>` and `<table 2>`	Tables <`table 1`> and <`table 2`> are already joined in this fashion.	In the `analysis.xml` file, remove duplicate join contents in the <`JOIN`> element.

Get Dataset Query with Thresholds Utility

Processing uses the Get Dataset Query with Thresholds Utility to store a dataset query in the Behavior Detection database with the threshold names and not with the threshold values. When the Behavior Detection engine executes a scenario, it substitutes the correct threshold values in the SQL query before submitting it to the database. Tracking of the query that executes in the database occurs only through the Behavior Detection engine log file when it runs in trace mode.

Using the Get Dataset Query With Thresholds Utility

Processing extracts the dataset query and uses it as input for tuning and execution plan generation.

Note: This utility does not recursively substitute thresholds in child datasets. Therefore, if a dataset being extracted has a reference to another dataset, manual extraction of that dataset must also occur.

Table 106 describes the parameters to provide with the `get_dataset_query.sh` script:

Table 106. Get Dataset Query Variables

Parameter	Description
`Dataset ID`	Unique identifier of the dataset for retrieval.
`Threshold Set ID`	Unique identifier of the threshold set for retrieval.

Executing the Get Dataset Query with Thresholds Utility

The following section provides instructions to execute the Get Dataset Query with Thresholds Utility.

To Execute the Get Dataset Query with Thresholds

To execute the Get Dataset Query with Thresholds Utility, follow the steps:

1. After the Alert Creator process completes, execute the `get_dataset_query.sh` script as follows:

   ```
   <INSTALL_DIR>/database/db_tools/bin/get_dataset_query.sh <Data-
   set ID> <Threshold Set ID>
   ```

   The dataset query automatically prints to standard output, which you can copy and paste into any other application.

   When the dataset query does not find a dataset, output is:

   ```
   Error: Dataset not found.
   ```

   When the dataset query does not find a threshold set, output is:

   ```
   Error: Threshold Set not found.
   ```

   Optional: Redirect the output into a text file as follows:

   ```
   <INSTALL_DIR>/database/db_tools/bin/get_dataset_query.sh <Data-
   set ID> <Threshold Set ID> query.sql
   ```

Scenario Migration Utility

You use the Scenario Migration Utility to migrate scenarios, datasets, networks, and associated metadata from the development environment to the production environment.

To provide a list of scenarios, datasets, or networks, you edit the `scnros.cfg`, `dataset.cfg`, or the `network.cfg` files prior to scenario extraction or loading.

The Scenario Migration Utility creates and migrates the following metadata files:

- **Scenarios:** The `<scenario catalog identifier>.<scenario id>.xml` file contains scenario metadata for core Behavior Detection tables. It also may contain scenario metadata for optional tables.
- **Datasets:** The `<dataset idDS>.xml` file contains dataset metadata for core Behavior Detection tables.
- **Networks:** The `<network>NW.xml` file contains network metadata for core Behavior Detection tables.

  Note: When the Scenario Migration Utility extracts these files, you can version-control them or store them in the Oracle Financial Services client's archival system.

To help avoid accidental loading of a scenario into the incorrect environment, the Scenario Migration utility enables you to *name* your source and target environments. On extract, you can specify the environment name to which you plan to load the scenario. If you attempt to load it to a different environment, the system displays a warning prompt.

Logs

The Scenario Migration Utility produces two log files (Figure 64 on page 277): `load.log` and `extract.log`. These files reside in the following location:

```
<INSTALL_DIR>/database/db_tools/logs
```

Using the Scenario Migration Utility

This section covers the following topics, which describe configuring and executing the Scenario Migration Utility, including extracting and loading metadata:

- Configuring the Scenario Migration Utility
- Extracting Scenario Metadata
- Loading Scenario Metadata

Configuring the Scenario Migration Utility

The `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Scenario Migration and other utilities require for processing. Figure 64 provides sample information from the `install.cfg` file that is specific to this utility.

```
############### 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/oriont/mantas6.1.3/database/db_tools/mantas_cfg/dep.cfg
sm.release=6.1.3
#### 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=${database.driverName}
extract.conn.url=jdbc:oracle:oci:@T2O9S8
#Source System Id
extract.system.id=TEST_ENVIORNMENT
# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
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/oriont/mantas6.1.3/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/oriont/mantas6.1.3/database/db_tools/data/temp
```

(Continued on next page)

(Continued from previous page)

```
# 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 restricted, 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
load.system.id=PROD_ENVIRONMENT
# 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
# 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}
load.dirname
# Specify whether threshold can be updated
load.threshold.update=Y
verify.target.system
```

Figure 64. Sample `install.cfg` File for Scenario Migration

Note: In the `install.cfg` file, entries are in the form `Property1=${Property2}`. That is, the value for Property1 is the value that processing assigns to Property2. As such, if you change Property2's value, Property1's value also changes.

Configuring the Environment

To configure the environment for scenario migration, modify the parameters that the sample `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` shows (Refer to Table 107 on page 278). The tables in the following sections describe the parameters specific to the Scenario Migration Utility.

Configuring General Scenario Migration

Table 107 describes general scenario migration parameters.

Table 107. General Scenario Migration Parameters

Parameter	Description
`score.include`	Flag that indicates whether scenario migration includes scenario scoring metadata; value is "Y" or "N" (the default).
`wrapper.include`	Flag that indicates whether scenario migration includes wrapper metadata; value is "Y" or "N" (the default).
`sm.depfile`	Location of the scenario migration dependencies file, `<INSTALL_DIR>/data-base/db_tools/mantas_cfg/dep.cfg`.
`sm.release`	Version of the Scenario Migration Utility.

Caution: Oracle Financial Services strongly recommends that you maintain scores and threshold values in a single environment. Maintaining these attributes in multiple environments and migrating the scenarios between the environments can cause the loss of threshold set-specific scoring rules.

Configuring Scenario Extraction

Table 108 describes scenario extraction parameters.

Table 108. Scenario Extraction Parameters

Parameter	Description
`extract.product.range.only`	Flag that indicates the components of the scenario that are shipped with the product; value is "Y" or "N" (the default).
`extract.database.username`	User to use to connect to the database when extracting scenarios (DB_UTIL_USER).
`extract.database.password`	Password for the above user.
`extract.conn.driver`	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
`extract.conn.url`	Database connection string that the Scenario Migration Utility is to use.
`extract.system.id`	System from which the scenario was extracted.
`extract.schema.mantas`	MANTAS schema owner in the database into which extraction of the scenarios occurs (MANTAS).
`extract.schema.business`	Business schema owner in the database into which extraction of the scenarios occurs (BUSINESS).
`extract.schema.market`	Market schema owner in the database into which extraction of the scenarios occurs (MARKET).
`extract.user.miner`	DATA MINER schema owner in the database into which extraction of the scenarios occurs (`KDD_MNR`).
`extract.miner.password`	Password for the above user.
`extract.dirname`	Full path to the target directory where the utility writes extracted metadata (`<INSTALL_DIR>/database/db_tools/data)`.
`extract.backup.dir`	Full path to the target directory where the utility writes backups of the extracted metadata (`<INSTALL_DIR>/database/db_tools/data/temp`).
`extract.product.range.only`	Indicator (Y or N) of whether to extract custom patterns, jobs, thresholds, threshold sets, and scoring rules when extracting a scenario. Set to Y to prevent extraction of these entities.
`extract.product.range.check`	(For internal use only.) Indicator (Y or N) of whether to fail the extraction of a scenario if any metadata has sequence IDs outside the product range. Set to Y to fail the extraction.

Configuring Scenario Load

Table 109 describes scenario load parameters.

Table 109. Scenario Load Parameters

Parameter	Description
`load.database.user-name`	User to use to connect to the database when extracting scenarios (DB_UTIL_USER).
`load.database.pass-word`	Password for the above user.
`load.conn.driver`	Database connection driver that the utility is to use (ora-cle.jdbc.driver.OracleDriver).
`load.conn.url`	Database connection string that the Scenario Migration Utility is to use.
`load.ignore.cus-tom.patterns=N`	When set to N, custom patterns will not be ignored. This mode should be used when migrating scenarios between environments within the client's environment. If a custom pattern is not in the loaded XML file, then it will be deacti-vated. When set to Y, any custom patterns will be ignored by the load process, and should continue to operate.
`load.schema.mantas`	MANTAS schema owner in the database in which loading of the scenario occurs (MANTAS).
`load.schema.business`	BUSINESS schema owner in the database in which load-ing of the scenario occurs (BUSINESS).
`load.schema.market`	MARKET schema owner in the database in which loading of the scenario occurs (MARKET).
`load.user.miner`	DATA MINER schema owner in the database in which load-ing of the scenario occurs (`KDD_MNR`).
`load.miner.password`	Password for the above user.
`load.threshold.update`	Threshold values from the incoming scenario. ● Selecting N retains the threshold values from the tar-get environment. ● Selecting Y updates thresholds in the target environ-ment to values from the incoming file.

Table 109. Scenario Load Parameters (Continued)

Parameter	Description
`load.system.id`	Name that is assigned to the system into which this instance of Scenario Migration loads metadata. The system compares the value for this setting to the target system in the metadata file.
`load.dirname`	Directory from which the system loads scenario, network, and dataset XML files.
`verify.target.system`	Check target name upon loading metadata files. ● Setting to N prevents Scenario Migration from checking the load.system.id against the target system specified when the scenario, network or dataset was extracted. ● Setting to Y enables this check. If the target in the XML file does not match the setting for load.system.id or the target is present in XML file but the load.system.id is blank then the system prompts you for an appropriate action. You can then continue with load or abandon the load, and you can apply the same answer to all other files in the session of Scenario Migration or allow the utility to continue prompting on each XML file that has a mismatch.

Extracting Scenario Metadata

Scenario metadata includes XML files that contain the table data for scenario, dataset, and network logic. The `sm_extract.sh` script invokes a Java tool, which creates these files. You start this script as follows:

```
sm_extract.sh <mode> [-notarget | -target <name>
```

where:

- `mode` (mandatory) is the scenario, network, or dataset.
- `-notarget`, if included, implies that the system does not save the target environment to the generated XML files.
- `-target <name>` identifies the same target (in `<name>`) for all extracted XML files.

If you do not specify `-notarget` or `-target <name>` on the command line, the system prompts you to supply a target environment on each extracted file.

To Extract Scenario Metadata

To extract scenario, dataset, and network metadata, follow these steps:

1. Navigate to the following directory:

   `cd <INSTALL_DIR>/db_tools`

2. Edit the metadata configuration files with identifying information for the scenarios, datasets, or networks for extraction:

   - `<scnro_ctlg_id>` in the `scnros.cfg` file

     and/or

   - `<scnro_ctlg_id>.<scnro_id>` in the `scnros.cfg` file

     Note: Providing both `<scnro_ctlg_id>` and `<scnro_id>` in the `scnros.cfg` file allows finer granularity when extracting scenarios. If you provide both a scenario catalog ID and a scenario ID on a line, you must separate them with a period.

   - `<data_set_id>` in the `dataset.cfg` file
   - `<network_id>` in the `network.cfg` file

3. Execute the `sm_extract.sh` script in this order:

   a. Enter `sm_extract.sh dataset` to extract dataset metadata.

   b. Enter `sm_extract.sh scenario` to extract scenario metadata.

   c. Enter `sm_extract.sh network` to extract network metadata.

Loading Scenario Metadata

The `sm_load.sh` script loads translated XML table data files into the target database.

To Load Scenario Metadata

To load scenario, dataset, and network metadata, use the following procedure.

Note: To avoid corrupting the Behavior Detection process, never load scenarios while the process is running.

1. Navigate to the following directory:

   `cd <INSTALL_DIR>/db_tools`

2. *Optional:* Edit the metadata configuration files (that is, `scnros.cfg`, `dataset.cfg`, and `network.cfg`) with identifying information for the scenarios, datasets, or networks that you want to load:

   - `<scnro_ctlg_id>` in the `scnros.cfg` file

     and/or

   - `<scnro_ctlg_id>` in the `scnros.cfg` file

     Note: Providing both `<scnro_ctlg_id>` and `<scnro_id>` in the `scnros.cfg` file allows finer granularity when loading scenarios. You must separate values with a period per line.

   - `<data_set_id>` in the `dataset.cfg` file
   - `<network_id>` in the `network.cfg` file

3. Copy the XML files you plan to load into the directory that the load.dirname specifies in the `install.cfg` file (Figure 54 on page 207).
4. Execute the `sm_load.sh` script:
   a. Enter `sm_load.sh dataset` to load dataset metadata.
   b. Enter `sm_load.sh scenario` to load scenario metadata.
   c. Enter `sm_load.sh network` to load network metadata.

Scenario Migration Best Practices

Migrating scenarios from one environment to another requires a unified process in order to prevent conflicts and errors. This section describes the recommended best practices for scenario migration for any existing Oracle Financial Services Behavior Detection system.

Caution: Not following the recommended best practices while loading scenarios to the targeted system may cause one or more sequence ID conflicts to occur, and your scenario will not be loaded. Once a conflict occurs, the metadata in the target environment needs to be corrected before the scenario can be successfully loaded.

To execute the recommended best practices, you should have an intermediate level knowledge of the scenario metadata, and be familiar with scenario patterns, thresholds, threshold sets, and so on. Basic SQL are required, as well as access privileges to the MANTAS schema. You must also be able to update records through SQLPLUS or a similar DB utility.

Process Overview

Scenario metadata is stored in many tables, with each table using a unique sequence ID for each of its records. If scenarios, thresholds, and scoring rules are modified in multiple environments using the same sequence ID range, then conflicts may occur when you migrate scenarios to these environments. To prevent conflict, you must set different sequence ID ranges in each of the environments.

The recommended best practices contain two basic points:

- Make changes in only one environment
- Separate the sequence ID ranges

Best Practices

Prepare to implement the recommended best practices before installing the application. Once the application is installed you should execute these steps to avoid scenario migration problems.

Make changes in only one environment

1. Only make changes to scenarios, thresholds, threshold sets, and scoring rules in the source environment.
2. Test and confirm your changes in the source environment.
3. Extract scenarios from the source environment and migrate them to all of your target environments.

Conflicting sequence IDs are often the cause errors when you migrate a scenario, so it is important to separate the sequence ID range.

Separate Sequence ID ranges

1. Review the `MANTAS.KDD_COUNTER` table, which contains all sequence ID ranges and current values. (For information about the `MANTAS.KDD_COUNTER` table, Refer to the *Oracle Financial Services Financial Services Data Model, Ref Guide Vol. 1*)
2. Start your sequence ID ranger at 10,000,000 and separate each environment by 10,000,000. The Oracle Financial Services Behavior Detection product sequence ID range is >100,000,000.

Sequences to Modify

You should set these sequences before doing any work on scenarios, thresholds, or scoring rules.

Table 110, Table 111, and Table 112 list sequences involved and sample values.

Table 110. Environment 1 (Development)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
KDD_ATTR	ATTR_ID_SEQUENCE	10000000	10000000	19999999
KDD_AUGMENTATION	AGMNT_INSTN_ID_SEQ	10000000	10000000	19999999
KDD_DATASET	DATASET_ID_SEQUENCE	10000000	10000000	19999999
KDD_JOB	JOB_ID_SEQ	10000000	10000000	19999999
KDD_LINK	LINK_ID_SEQ	10000000	10000000	19999999
KDD_LINK_ANLYS_NTWRK_DEFN	NTWRK_DEFN_ID_SEQ	10000000	10000000	19999999
KDD_LINK_ANLYS_TYPE_CD	TYPE_ID_SEQ	10000000	10000000	19999999
KDD_LINK_SUMMARY	LINK_SUMMARY_ID_SEQ	10000000	10000000	19999999
KDD_LINK_TYPE_SUMMARY	LINK_TYPE_SUMMARY_ID_SEQ	10000000	10000000	19999999
KDD_NODE	NODE_ID_SEQ	10000000	10000000	19999999
KDD_NTWRK	NTWRK_ID_SEQ	10000000	10000000	19999999
KDD_PARAM_SET	PARAM_SET_ID_SEQ	10000000	10000000	19999999
KDD_PTTRN	PTTRN_ID_SEQ	10000000	10000000	19999999
KDD_RULE	RULE_ID_SEQ	10000000	10000000	19999999
KDD_SCNRO	SCNRO_ID_SEQ	10000000	10000000	19999999
KDD_SCORE	SCORE_ID_SEQ	10000000	10000000	19999999
KDD_SCORE_HIST	SCORE_HIST_SEQ_ID_SEQ	10000000	10000000	19999999
KDD_TSHLD	TSHLD_ID_SEQ	10000000	10000000	19999999
KDD_TSHLD_HIST	HIST_SEQ_ID_SEQ	10000000	10000000	19999999
KDD_TSHLD_SET	TSHLD_SET_ID_SEQ	10000000	10000000	19999999

Table 111. Environment 2 (Test/UAT)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
KDD_ATTR	ATTR_ID_SEQUENCE	20000000	20000000	29999999
KDD_AUGMENTATION	AGMNT_INSTN_ID_SEQ	20000000	20000000	29999999
KDD_DATASET	DATASET_ID_SEQUENCE	20000000	20000000	29999999
KDD_JOB	JOB_ID_SEQ	20000000	20000000	29999999
KDD_LINK	LINK_ID_SEQ	20000000	20000000	29999999
KDD_LINK_ANLYS_NTWRK_DEFN	NTWRK_DEFN_ID_SEQ	20000000	20000000	29999999
KDD_LINK_ANLYS_TYPE_CD	TYPE_ID_SEQ	20000000	20000000	29999999
KDD_LINK_SUMMARY	LINK_SUMMARY_ID_SEQ	20000000	20000000	29999999
KDD_LINK_TYPE_SUMMARY	LINK_TYPE_SUMMARY_ID_SEQ	20000000	20000000	29999999
KDD_NODE	NODE_ID_SEQ	20000000	20000000	29999999
KDD_NTWRK	NTWRK_ID_SEQ	20000000	20000000	29999999
KDD_PARAM_SET	PARAM_SET_ID_SEQ	20000000	20000000	29999999
KDD_PTTRN	PTTRN_ID_SEQ	20000000	20000000	29999999
KDD_RULE	RULE_ID_SEQ	20000000	20000000	29999999
KDD_SCNRO	SCNRO_ID_SEQ	20000000	20000000	29999999
KDD_SCORE	SCORE_ID_SEQ	20000000	20000000	29999999
KDD_SCORE_HIST	SCORE_HIST_SEQ_ID_SEQ	20000000	20000000	29999999
KDD_TSHLD	TSHLD_ID_SEQ	20000000	20000000	29999999
KDD_TSHLD_HIST	HIST_SEQ_ID_SEQ	20000000	20000000	29999999
KDD_TSHLD_SET	TSHLD_SET_ID_SEQ	20000000	20000000	29999999

Table 112. Environment 3 (PROD)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
KDD_ATTR	ATTR_ID_SEQUENCE	30000000	30000000	39999999
KDD_AUGMENTATION	AGMNT_INSTN_ID_SEQ	30000000	30000000	39999999
KDD_DATASET	DATASET_ID_SEQUENCE	30000000	30000000	39999999
KDD_JOB	JOB_ID_SEQ	30000000	30000000	39999999
KDD_LINK	LINK_ID_SEQ	30000000	30000000	39999999
KDD_LINK_ANLYS_NTWRK_DEFN	NTWRK_DEFN_ID_SEQ	30000000	30000000	39999999
KDD_LINK_ANLYS_TYPE_CD	TYPE_ID_SEQ	30000000	30000000	39999999
KDD_LINK_SUMMARY	LINK_SUMMARY_ID_SEQ	30000000	30000000	39999999
KDD_LINK_TYPE_SUMMARY	LINK_TYPE_SUMMARY_ID_SEQ	30000000	30000000	39999999
KDD_NODE	NODE_ID_SEQ	30000000	30000000	39999999
KDD_NTWRK	NTWRK_ID_SEQ	30000000	30000000	39999999
KDD_PARAM_SET	PARAM_SET_ID_SEQ	30000000	30000000	39999999

Table 112. Environment 3 (PROD) (Continued)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
KDD_PTTRN	PTTRN_ID_SEQ	30000000	30000000	39999999
KDD_RULE	RULE_ID_SEQ	30000000	30000000	39999999
KDD_SCNRO	SCNRO_ID_SEQ	30000000	30000000	39999999
KDD_SCORE	SCORE_ID_SEQ	30000000	30000000	39999999
KDD_SCORE_HIST	SCORE_HIST_SEQ_ID_SEQ	30000000	30000000	39999999
KDD_TSHLD	TSHLD_ID_SEQ	30000000	30000000	39999999
KDD_TSHLD_HIST	HIST_SEQ_ID_SEQ	30000000	30000000	39999999
KDD_TSHLD_SET	TSHLD_SET_ID_SEQ	30000000	30000000	39999999

In order to update your database tables with recommended values, use SQLPLUS or a similar tool.
A sample SQL statement to update a set of sequence is:

```
UPDATE KDD_COUNTER
set min_value = 10000000,
    max_value = 19999999,
    current_value = 10000000
where sequence_name in
('DATASET_ID_SEQUENCE',
 'ATTR_ID_SEQUENCE',
 'PARAM_SET_ID_SEQ',
 'PTTRN_ID_SEQ',
 'RULE_ID_SEQ',
 'SCNRO_ID_SEQ',
 'JOB_ID_SEQ',
 'TSHLD_ID_SEQ',
 'NTWRK_ID_SEQ',
 'LINK_ID_SEQ',
 'NODE_ID_SEQ',
 'LINK_SUMMARY_ID_SEQ',
 'NTWRK_DEFN_ID_SEQ',
 'TYPE_ID_SEQ',
 'TAB_ID_SEQ',
 'LINK_TYPE_SUMMARY_ID_SEQ',
 'TSHLD_SET_ID_SEQ',
 'HIST_SEQ_ID_SEQ',
 'AGMNT_INSTN_ID_SEQ',
 'SCORE_ID_SEQ',
 'SCORE_HIST_SEQ_ID_SEQ');
Commit;
```

Repeat for each environment, remembering to change the values for min, max, and current.

Alert Correlation Rule Migration Utility

Use the Alert Correlation Rule Migration Utility to migrate correlation rules and associated audit trails between development environment and the production environment.

To provide a list of correlation rules, you create a file listing the correlation rule names prior to correlation rules extraction or loading. The Alert Correlation Rule Migration Utility creates and migrates the following metadata file:

`<CorrelationRuleName>.xml`

This file contains correlation rule metadata, and additionally, an audit trail of the correlation rule for core Oracle Financial Services Behavior Detection tables. To avoid accidental loading of correlation rules into the incorrect environment, the Alert Correlation Rule Migration Utility enables you to *name* your source and target environments. On extract, you can specify the environment name to which you plan to load the correlation rule. If you attempt to load it to a different environment, the system displays a warning prompt.

Logs

The Alert Correlation Rule Migration Utility produces two log files (Figure 65 on page 289): `load.log` and `extract.log`. These files reside in the following location:

`<INSTALL_DIR>/database/db_tools/logs`

Using the Alert Correlation Rule Migration Utility

This section covers the following topics, which describe configuring and executing the Alert Correlation Rules Migration Utility, including extracting and loading metadata:

- Configuring the Alert Correlation Rules Migration Utility
- Extracting Alert Correlation Rule
- Loading Alert Correlation Rule

Configuring the Alert Correlation Rules Migration Utility

The `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Alert Correlation Rule Migration and other utilities require for processing. Figure 65 provides sample information from the `install.cfg` file that is specific to this utility.

```
################ CORRELATION RULE MIGRATION CONFIGURATION #######################

#### GENERAL CORRELATION RULE MIGRATION SETTINGS
# Specify the name of the configuration file containing the names of correlation rules to be
migrated. This property is specific to the Correlation Rule Migration Utility
corrRuleMig.CorrRuleFileNm=/users/mantas/database/db_tools/mantas_cfg/corrRule.cfg

#### EXTRACT (These properties are shared by Correlation Rule Migration Utility with the
Scenario Migration Utility)

# 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:@T2O9S8

#Source System Id
extract.system.id= ENVIORNMENT

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

# File Paths for Extract

#Specify the full path in which to place extracted Correlation Rules
extract.dirname=/users/mantas/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/mantas/database/db_tools/data/temp
```

(Continued on next page)

(Continued from previous page)

```
#### LOAD (These properties are shared by Correlation Rule Migration Utility with the
Scenario Migration Utility)

# 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= PROD_ENVIRONMENT
# Specify the schema names for Load
load.schema.mantas=${schema.mantas.owner}

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

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

# Specify whether the Audit Trail (History Records) are to be loaded or not. This property
is specific to the Correlation Rule Migration Utility
corrRuleMig.loadHistory=Y


# Specify the URL to be used for refreshing the Correlation Rules. This property is specific
to the Correlation Rule Migration Utility
aps.service.url=http:/./10.155.114.70:8080/mantas/services/AlertProcessingService
```

Figure 65. Sample `install.cfg` File for Alert Correlation Rule Migration

Note: In the `install.cfg` file, entries are in the form `Property1=${Property2}`. That is, the value for Property1 is the value that processing assigns to Property2. As such, if you change Property2's value, Property1's value also changes.

Configuring the Environment

To configure the environment for alert correlation rule migration, modify the parameters that the sample `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` shows (Refer to Table 113 on page 290). The tables in the following sections describe the parameters specific to the Alert Correlation Rule Migration Utility.

Configuring General Alert Correlation Rule Migration

Table 113 describes general alert correlation rule migration parameters.

Table 113. General Alert Correlation Rule Migration Parameters

Parameter	Description
`corrRuleMig.CorrRule-FileNm`	Location of the file containing the list of Alert Correlation Rule names to be migrated. `<INSTALL_DIR>/database/db_tools/mantas_cfg/<FileName>.cfg`

Note: If the file name containing the list of Alert Correlation Rule Names is not provided, the utility displays a warning message and extracts/loads the default alert correlation rules specified in this file:
`<INSTALL_DIR>/database/db_tools/mantas_cfg/corrRule.cfg`

Configuring Alert Correlation Rule Extraction

Table 114 describes alert correlation rule extraction parameters.

Table 114. Alert Correlation Rule Extraction Parameters

Parameter	Description
`extract.database.username`	User to use to connect to the database when extracting alert correlation rules (DB_UTIL_USER).
`extract.database.password`	Password for the above user.
`extract.conn.driver`	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
`extract.conn.url`	Database connection string that the Alert Correlation Rule Migration Utility is to use.
`extract.system.id`	System from which the alert correlation rule was extracted.
`extract.schema.mantas`	MANTAS schema owner in the database into which extraction of the alert correlation rule occurs (MANTAS).
`extract.dirname`	Full path to the target directory where the utility writes extracted metadata (`<INSTALL_DIR>/database/db_tools/data`).
`extract.backup.dir`	Full path to the target directory where the utility writes backups of the extracted metadata (`<INSTALL_DIR>/database/db_tools/data/temp`).

Configuring Alert Correlation Rule Load

Table 115 describes alert correlation rule load parameters.

Table 115. Alert Correlation Rule Load Parameters

Parameter	Description
`load.database.username`	User to use to connect to the database when loading alert correlation rules (DB_UTIL_USER).
`load.database.password`	Password for the above user. This is set by the Password Manager Utility.
`load.conn.driver`	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
`load.conn.url`	Database connection string that the Alert Correlation Rule Migration Utility is to use.
`load.schema.mantas`	MANTAS schema owner in the database in which loading of the alert correlation rule occurs (MANTAS).
`load.system.id`	Name that is assigned to the system into which this instance of Alert Correlation Rule Migration loads metadata. The system compares the value for this setting to the target system in the metadata file.
`load.dirname`	Directory from which the system loads alert correlation rule(s) XML files.
`verify.target.system`	Check target name upon loading metadata files. ● Setting to N prevents Alert Correlation Rule Migration from checking the load.system.id against the target system specified when the alert correlation rule was extracted. ● Setting to Y enables this check. If the target environment in the XML file does not match the setting for load.system.id or the target environment is present in XML file but the load.system.id is blank then the system prompts you for an appropriate action. You can then continue with load or abandon the load, and you can apply the same answer to all other files in the session of Alert Correlation Rule Migration or allow the utility to continue prompting on each XML file that has a mismatch.
`corrRuleMig.loadHistory`	Load audit trail records on load. ● Setting to N prevents Alert Correlation Rule Migration Utility from loading the audit trail records from the XML file into the database. ● Setting to Y enables the system to load the audit trail records from the XML file into the database.
`aps.service.url`	Web service URL of the AlertProcessing service to be used for refreshing the correlation rules.

Note: Irrespective of whether the user specifies N or Y for the `corrRuleMig.loadHistory` parameter, a default audit trail record indicating the current load event is inserted into the database.

Extracting Alert Correlation Rule

Alert correlation rule metadata includes XML files that contain the table data for the alert correlation rule along with their audit trails, if any. The `sm_extract.sh` script invokes a Java tool, which creates these files.

To extract alert correlation rule metadata, follow these steps:

1. Create a metadata configuration file (`<someFileName>.cfg`) with identifying information for the alert correlation rules to be extracted.

    - `<corr_rule_name>` in the `<someFileName>.cfg` file

      and/or

    - `<corr_rule_name>=<corr_rule_file_name>`

      Note: Providing both `<corr_rule_name>` and `<corr_rule_file_name>` in the `<someFileName>.cfg` file allows the user a flexibility to specify the actual filename that contains the metadata information for the respective alert correlation rule. If you provide both an alert correlation rule name and an alert correlation rule file name on a line, you must separate them with an equals (=) sign. It is recommended that the alert correlation rule file name be specified without an extension.

2. Navigate to the following directory:
   ```
   cd <INSTALL_DIR>/database/db_tools/mantas_cfg
   ```
3. Edit the `install.cfg` file to include the path for the above created file against the tag `corrRuleMig.CorrRuleFileNm`.
4. Navigate to the following directory:
   ```
   cd <INSTALL_DIR>/database/db_tools/bin
   ```
5. Execute the `sm_extract.sh` script as follows:
   ```
   sm_extract.sh correlation
   ```

Note: The utility performs the following validations upon extraction of an alert correlation rule:

- The attribute value for `type` attribute in the XML tag `<Case/>` should exist in the `CASE_TYPE_CD` column of the `CASE` schema table `KDD_CASE_TYPE_SUBTYPE`.
- The attribute value for `subtype` attribute in the XML tag `<Case/>` should exist in the `CASE_SUB_TYPE_CD` column of the `CASE` schema table `KDD_CASE_TYPE_SUBTYPE`.
- The attribute value for `<subClassTagLevel1/>` should exist in the `CASE_SUB_CLASS_LVL1_CD` column of the `CASE` schema table `KDD_SUBCLASS1`.
- The attribute value for `<subClassTagLevel2/>` should exist in the `CASE_SUB_CLASS_LVL2_CD` column of the `CASE` schema table `KDD_SUBCLASS2`.
- The `CDATA` section of the XML tag `<AlertAttrOperations/>` is validated as follows:

- The valid operations should be one of BOTH, TO and FROM.
- The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.
- The TO and FROM alert operations can be used only to compare alert attribute values to each other, and not to a literal.
- The FROM alert operation should always precede the TO alert operation.
- The BOTH alert operator must be used to compare alert attribute values to a literal.
- The expression can be nested to any arbitrary length provided it confirms to a general syntax:
  Operand Operator Operand [Logical_Operator Operand Operator Operand]

  For example,
  a) `BOTH.SCORE_CT >= 0`
  b) `BOTH.SCORE_CT >= 0 AND FROM.SCORE_CT = TO.SCORE_CT`

  Note: A space character is expected between each Operand and Operator combination.
- The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
- The alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_REVIEW` and `KDD_REVIEW_FINANCIAL`.

● The CDATA section of the XML tag `<AlertCorrAttrOperations>` is validated as follows:

- The Correlation Alert operation should be CORR.
- The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.

  Note: The `SCNRO_ALERT_CT` attribute works fine when used with the IN or NOT IN operators. The alert correlation job gives an error when the `SCNRO_ALERT_CT` attribute is used with operators like >,>=,<,<=,= and!=. This attribute is unlikely to be used in a correlation expressions but if it is used then it is recommended to use only with the IN or NOT IN operators.
- The expression can be nested to any arbitrary length provided it confirms to a general syntax:
  Operand Operator Operand [Logical_Operator Operand Operator Operand]

  For Example:
  a) `CORR.SCNRO_ID >= 0`
  b) `CORR.SCNRO_ID >= 0 AND CORR.SCNRO_ID = CORR.SCNRO_ID`

  Note: A space character is expected between each Operand and Operator combination.

- The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
- The Correlation Alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_ALERT_CORR` and `KDD_ALERT_CORR_SCNRO`.

Loading Alert Correlation Rule

The `sm_load.sh` script loads translated XML table data files into the target database.

To load alert correlation rule metadata, follow these steps:

1. Create a metadata configuration file (`<someFileName>.cfg`) with the rule names of the alert correlation rules to be loaded.
   - `<corr_rule_name>` in the `<someFileName>.cfg` file

     and/or
   - `<corr_rule_name>=<corr_rule_file_name>`

     Note: Providing both `<corr_rule_name>` and `<corr_rule_file_name>` in the `<someFileName>.cfg` file allows the user a flexibility to specify the actual filename that contains the metadata information for the respective alert correlation rule. If you provide both an alert correlation rule name and an alert correlation rule file name on a line, you must separate them with an equals (=) sign. It is recommended that the alert correlation rule file name be specified without an extension.
2. Navigate to the following directory:
   ```
   cd <INSTALL_DIR>/database/db_tools/mantas_cfg
   ```
3. Edit the `install.cfg` file to include the path for the above created file against the tag `corrRuleMig.CorrRuleFileNm`.
4. Copy the XML files you plan to load into the directory that the `load.dirname` specifies in the `install.cfg` file.
5. Navigate to the following directory:
   ```
   cd <INSTALL_DIR>/database/db_tools/bin
   ```
6. Execute the `sm_load.sh` script as follows:
   ```
   sm_load.sh correlation
   ```

Note: The utility performs the following validations upon loading of an alert correlation rule:

- The attribute value for *type* attribute in the XML tag `<Case/>` should exist in the `CASE_TYPE_CD` column of the `CASE` schema table `KDD_CASE_TYPE_SUBTYPE`.
- The attribute value for *subtype* attribute in the XML tag `<Case/>` should exist in the `CASE_SUB_TYPE_CD` column of the `CASE` schema table `KDD_CASE_TYPE_SUBTYPE`.

- The attribute value for `<subClassTagLevel1/>` should exist in the `CASE_SUB_CLASS_LVL1_CD` column of the `CASE` schema table `KDD_SUBCLASS1`.
- The attribute value for `<subClassTagLevel2/>` should exist in the `CASE_SUB_CLASS_LVL2_CD` column of the `CASE` schema table `KDD_SUBCLASS2`.
- The CDATA section of the XML tag `<AlertAttrOperations/>` is validated as follows:
  - The valid operations should be one of BOTH, TO and FROM.
  - The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.
  - The TO and FROM alert operations can be used only to compare alert attribute values to each other, and not to a literal.
  - The FROM alert operation should always precede the TO alert operation.
  - The BOTH alert operator must be used to compare alert attribute values to a literal.
  - The expression can be nested to any arbitrary length provided that it confirms to a general syntax:
    Operand Operator Operand [Logical_Operator Operand Operator Operand]

    For example,
    a) `BOTH.SCORE_CT >= 0`
    b) `BOTH.SCORE_CT >= 0 AND FROM.SCORE_CT = TO.SCORE_CT`

    Note: A space character is expected between each Operand and Operator combination.
  - The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
  - The alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_REVIEW` and `KDD_REVIEW_FINANCIAL`.
- The CDATA section of the XML tag `<AlertCorrAttrOperations>` is validated as follows:
  - The Correlation Alert operation should be CORR.
  - The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.
  - The expression can be nested to any arbitrary length provided that it confirms to a general syntax:
    Operand Operator Operand [Logical_Operator Operand Operator Operand]

For Example:

a) `CORR. SCNRO_ID >= 0`

b) `CORR.SCNRO_ID >= 0 AND CORR.SCNRO_ID = CORR.SCNRO_ID`

Note: A space character is expected between each Operand and Operator combination.

- The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
- The Correlation Alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_ALERT_CORR` and `KDD_ALERT_CORR_SCNRO`.

Watch List Service

Watch list web service enables you to query the Behavior Detection Watch List tables to determine if a given name (or a name closely matching the given name) is on a watch list. Refer to the *Oracle Financial Services Services Guide*, for more details on how the service can be called and the results that are returned.

Watch List Service uses three scripts to start, stop, and re-read the Watch List tables in case of changes. These scripts are placed in `ingestion_manager/scripts` folder. The scripts are:

- `startWatchList.sh` script: to start the web service.
- `shutdownWatchList.sh` script: To stop the web service.
- `initWatchList.sh` script: this script causes the Watch List Service to re-read the Watch List tables in case there have been any changes to it while the service has been running.

There is a polling agent that runs as part of the web service that will query the Watch List tables on a configurable interval (default of 10 seconds). This is to done to keep the service in sync with the Watch List Management Utility

Alert Processing Web Services

The Alert Processing Web service provides the ability to execute additional processing steps during a call to the existing PostAlert service operation, currently delivered with Investigation Management. Details on this service can be found in the *Oracle Financial Services Services Guide.*

Instructions on Administering the Alert Processing Services

Alert Processing Service provides two scripts, one to start the service and one to stop the service. These scripts can be found in the {`INSTALL_DIR`}/services/scripts directory. Details of the scripts are as follows:

- `startWebServices.sh`: Run this script to start the web service.
- `shutdownWebServices.sh`: Run this script to stop the web service.

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 307, 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 116 outlines the logging levels that the Common Logging component supports.

Table 116. 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.

Scenario Manager

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

```
{INSTALLED DIRECTORY}/behavior_detection/toolkit/mantas_cfg/etc/
mantas_toolkit_message_lib_en.dat
```

All messages numbers that this section contains must be within the range of 130,000 - 169,999.

Services

The following file is the message library for the Services:

```
<INSTALL_DIR>/services/server/webapps/mantas/WEB-INF/classes/conf/
mantas_cfg/etc/mantas_alert_management_message_lib_en.dat
```

All messages numbers that this section contains must be within the range of 210,000 - 249,999.

Alert Management/Case Management

The following logs contain the message library for both Alert and Case Management applications:

Web server Logs

The following file is the message library for the Web server logs:

```
<mantas reveleus web context>/logs/UMMService.log
```

Application server logs

The following file is the message library for the Application Server logs:

```
<FIC_APP_HOME>/common/ficserver/logs/revappserver.log
```

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:

- Administration Tools:
  `<FIC_HOME>/AM/admin_tools/WEB-INF/classes/conf/mantas_cfg/install.cfg`
- Database:
  `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg`
- Scenario Manager:
  `{INSTALLED DIRECTORY}/behavior_detection/toolkit/mantas_cfg/install.cfg`
- Behavior Detection:
  `<INSTALL_DIR>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg`
- Alert Management/Case Management:
  - `Web Server logs:`

    `Logging levels can be configured in the below mentioned file:`

    `<mantas reveleus web home>/conf/RevLog4jConfig.xml`

    `In below mentioned tag.`

    `<root>`

    <priority value ="debug" />

    <appender-ref ref="ConsoleAppender1"/>

    </root>

    - Below mentioned logger levels are available:

      DEBUG

      INFO

      WARN

      SEVERE

      FATAL
  - `Application Server logs:`

    `Logging levels can be configured in the below mentioned file:`

    `<$FIC_HOME>/conf/RevLog4jConfig.xml`

    `<root>`

    <priority value ="debug" />

    <appender-ref ref="ConsoleAppender1"/>

    </root>

    - Below mentioned logger levels are available:

DEBUG

INFO

WARN

SEVERE

FATAL

- Services:
  `<INSTALL_DIR>/services/server/webapps/mantas/WEB-INF/classes/conf/mantas_cfg/install.cfg`

  `<INSTALL_DIR>/services/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.
```

Figure 66. 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 117 identifies the valid values for this property.

Table 117. 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 Oracle Financial Services 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 118 identifies the configurable properties for logging in an Oracle Financial Services client's environment.

Table 118. 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://logging.apache.org/log4j/docs/ manual.html) for more details about the log message format.
`log.message.library`	To be specified at installation.	Identifies the full path and filename of the message library.
`log.max.size`	2000000000	Determines the maximum size (in bytes) of a log file before the system creates a new log file. For more information (Refer to *Log File Sizes,* on page 307, 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 307, 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 307, 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 118. 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 117 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 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.

APPENDIX B

Oracle Financial Services Software Updates

This appendix describes the application of Behavior Detection Platform software updates in *Oracle Financial Services Behavior Detection Platform*:

- Oracle Financial Services Software Updates - Hotfix
- Hotfix Effect on Customization

Behavior Detection Platform Software Updates - Hotfix

A hotfix is a package that includes one or more files that are used to address a defect or a change request in Oracle Financial Services Behavior Detection. Typically, hotfixes are small patches designed to address specific issues reported by the clients.

Hotfixes can affect the following areas in Behavior Detection:

- The User Interface (UI)
- Scenarios (patterns and datasets)
- Post-Processing jobs
- Performance
- Informatica/Ingestion

Each hotfix includes a `readme.txt` file, which describes the step-by-step process to install the hotfix.

Hotfixes are delivered to clients in the following ways:

- E-mail
- Secure FTP

Hotfix Effect on Customization

When a hotfix is installed it can affect your customizations on the *User Interface* and *Scenarios.*

User Interface

If your UI customizations are correctly isolated to the `custom` directory, then the impact should be minimal. It is possible, however, that the hotfix changes information in the base product that you have customized. In that case, you can not see the effect of the hotfix. To minimize this, be sure to avoid copying more than necessary to the `custom` directory. For example, you should not copy the entire `BF_Business.xml` file to override a few fields, you should create a new file in the `custom` directory that only contains the fields you are overriding.

The hot fixes delivered will include installation and deployment instructions in the fix documentation.

Scenarios

If you have customized scenarios (changed dataset logic or changed scenario logic), then applying a hotfix to that scenario will remove those customizations. If you customized datasets by creating a dataset override file (Refer to *Applying a Dataset Override,* on page 13 for more information), then your custom dataset continues to be used after applying the hotfix. It is possible that your custom dataset prevents the scenario fix from being evident (if the dataset you customized was one of the items changed by the hotfix). It is also possible that the hotfix changes the fields it expects from the dataset you customized, causing the scenario to fail. For scenarios you have customized, you should always test the scenario hotfix without your customizations in place, then re-apply them to the scenario, if necessary.

APPENDIX C

Report Management

This appendix contains the configuration and integration details of the Report Framework with Oracle Financial Services user interface. The term client app is consistently used to Refer to the application that must be attached to the Report Framework. It provides online approval and archival capability.

Report Management

The Report Management is an Active Pages interface that allows you to retrieve a list of reports based on the report criteria that are approved or awaiting approval. You can select from the list of report to display either the live version of the scheduled reports awaiting approval or archived versions of reports that have been approved. In addition, you can view comments associated to a report.

Report Management enables to access a report based on the user's viewable organization, jurisdiction, and role. Each report is mapped to one or more user roles set up during deployment.

Report Management is accessible as an application on the Active Pages Desktop.

Prerequisites for Integrating Report Framework

Before you use Report Framework, the following prerequisites must be fulfilled:

1. Altio's `pdfservlet.war` file must be present on Altio Presentation Server.
2. Report Framework must be installed on the same Altio Presentation Server as the client application.
3. A PDF template with form fields to act as placeholders must be present in the templates directory. The PDF template can be created using any PDF Writer.
4. The client App should contain a function called generatePDF that invokes an HTTP Service. This service makes a call to the PDF Generation servlet. This Altio function determines the data it requires to archive as a PDF.
5. Configure the following parameters in the client app to enable you to connect to the Report Framework
   - `pdfservlet.war` location
   - PDF Template location

6. Set the parameters in the client app to be consider as default – in the `KDD_REP_XXX` tables. The set parameters are disabled when the client app is launched from the Report Management Console. (Refer to Table 119 for more information on Metadata).

Table 119, Refer to the Metadata creation for Report Framework.

Table 119. Metadata Creation for Report Framework

Logical Name	Table Name	Table Description	Comments
Report Definition	`KDD_REPORT_DEFN`	This table is the definition of a report. It identifies the name, frequency and any links to Active Pages applications.	The Report Type Code should be *AP* for reports to be Oracle Financil Services6.1.3 outside of the Behavior Detection UI. The Active Pages Application Name should point to the Altio project name and the view should point to the path of the default view file to be loaded. The Access Control Name should detail the acl required to access reports of this type.
Report Definition Parameters	`KDD_REP_DEFN_PARAM`	This table contains a list of parameters that are valid for a particular type of report.	
Report Template	`KDD_REPORT_TEMPLATE`	This table stores a template for a report. This template is used to generate report tracking records for different time periods.	Please note that the Calendar names that you specify here must exists in the `KDD_CAL` table. Also, a report definition can have multiple templates associated with it for different organizations or calendars.
Report Template Jurisdiction	`KDD_REPORT_TEMPLATE_JRSDCN`	This table stores jurisdictions to be used with a particular report template.	
Report Template Parameters	`KDD_REPORT_TEMPLATE_PARAM`	This table stores the values of parameters specific to the report template.	The parameters and their corresponding values would be set as default report parameters when the report is loaded through the Report Framework's Report Management console.
Report	`KDD_REPORT_TRACK`	This table stores an instance of a report to be signed off.	This table is populated by the `run_report_client.ksh` utility.

Table 119. Metadata Creation for Report Framework (Continued)

Logical Name	Table Name	Table Description	Comments
Report Jurisdiction	`KDD_REPORT_TRACK_JRSDCN`	This table stores the Jurisdictions associated to an instance of the report. This table would be populated by the `run_report_client.ksh` utility.	This table is populated by the `run_report_client.ksh` utility.
Report Parameters	`KDD_REPORT_TRACK_PARAM`	This table stores the parameter values used to filter the instance of the report.	This table is populated by the `run_report_client.ksh` utility.
Report Comments	`KDD_REPORT_TRACK_CMMNT`	This table stores the comments that the user make against a report.	This table stores workflow or signoff comments added by the user

Refer to the *Oracle Financial Services Behavior Detection Platform FSDM Reference Guide,* Vol.1, *Business Data*, for more information.

Report Framework Limitations

The report framework has the following limitations:

- Report Framework cannot import graphs and charts.
- Only the data for which placeholders are created in the PDF Template can be exported.

Caution: Click **Close** button to close the report. Closing the report using the browser close button will lock the report. This would then prevent other users from editing the report for a short duration.

APPENDIX D Informatica Workflow Details

This appendix lists the Informatica workflow details used in the application and the use of that workflow.

Informatica Workflows

Table 120 provides the list of workflows and a description for each Informatica workflow. The workflows are listed by folders.

Table 120. Informatica Workflow Description

Workflow Number	Workflow Name	Description
BSM_Common Folder		
0350	`w_ph0350_truncate_rrst`	This workflow truncates the registered representative security table.
0490	`w_ph0490_truncate_hhsm`	This workflow truncate household summary month.
0500	`w_ph0500_aggregate_asm_to_hhsm`	This workflow aggregates household summary month out of the account summary month stage table.
1005	`w_ph1005_truncate_hhbps`	This workflow truncates household balance position table.
1010	`w_ph1010_update_abps_asgmt_risk_from_app`	This workflow calculates multiple fields in the account balances and position summary table. The algorithm is a simple mathematical difference between two sums from two different queries, plus the risk of straddled pairs. Updates are made to assignment risk attributes.
1020	`w_ph1020_update_ap_uncovered_options_from_app`	This workflow processes account option position pair data and updates the corresponding account position records. Updates are made to the attributes relating to uncovered option contracts.
1040	`w_ph1040_update_abps_option_mktval_long_from_app`	This workflow processes account option position pair data and updates the corresponding account balance records. Updates are made to option market value long attributes.
1050	`w_ph1050_aggregate_ap_to_abps`	This workflow aggregates current-day security positions by product category and account for update of the account balance record.
1060	`w_ph1060_aggregate_abps_to_hhbps`	This workflow aggregates account balance data into household balances.
1070	`w_ph1070_aggregate_ap_to_hhbps`	This workflow aggregates account positions data by household and updates the corresponding household balances record.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
BSM_Production Folder		
0040	`w_p0040_dump_masm_to_file`	This workflow is used for the nightly dump of Managed Account Summary month table for the current month to a flat file.
0050	`w_p0050_dump_nmsm_to_file`	This workflow is used for the nightly dump of Investment Manager Summary Month table for the current month to a flat file.
0060	`w_p0060_dump_rrsm_to_file`	This workflow is used for the nightly dump of Registered Representative Summary Month table for the current month to a flat file.
0110	`w_p0110_reload_masm_from_dump_file`	This workflow mapping rebuilds Managed Account Summary Month from Dump File.
0120	`w_p0120_reload_nmsm_from_dump_file`	This workflow can be used to rebuild Investment Manager Summary Month from dump file.
0130	`w_p0130_reload_rrsm_from_dump_file`	This workflow can be used to rebuild Registered Representative Summary Month Table from dump file.
0171	`w_p0171_truncate_masms`	This workflow truncates managed account summary daily.
0360	`w_p0360_aggregate_block_allcn_day_trades_to_file`	This workflow is used for the daily aggregation of the block allocation day trades data. This populates the managed account summary file.
0370	`w_p0370_aggregate_block_allcn_trades_to_file`	This workflow is used for the daily aggregation of the block allocation trades data. This populates the managed account summary file.
0371	`w_p0371_aggregate_day_trades_to_file`	This workflow is used for the daily aggregation of the day trades data. This populates the account summary file.
0380	`w_p0380_aggregate_bot_inter_hh_jrnls_to_file`	This workflow is used for the daily aggregation of the back office transaction data. This populates the managed account summary file.
0390	`w_p0390_aggregate_trades_to_rrst`	This workflow aggregates trades into registered representative security.
0400	`w_p0400_aggregate_nm_inter_hh_jrnls_to_file`	This workflow is used for the daily aggregation of the back office transaction data. It calculates Investment Manager Summary Journal Data.
0410	`w_p0410_aggregate_nm_net_worth_to_file`	This workflow is used for the calculation of the Daily change in Delinked Sub-Accounts and Net worth. It loads Investment Manager Summary Journal Data.
0420	`w_p0420_aggregate_nm_third_party_trxns_to_file`	This workflow is used for the daily aggregation of the front office transaction data. It calculates the Investment Manager Summary Third Party Data elements.
0430	`w_p0430_join_daily_activity_to_masms`	This workflow joins three flat files that contain the data for Inter HH Journal and block allocation trades together. The result is inserted into the `MANGD_ACCT_SMRY_TEMP` table.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
0431	w_p0431_build_MASD_from_MASMS	This workflow updates the Managed Account Summary Daily Table from its corresponding staging table.
0440	w_p0440_update_nmsm_for_daily_acti vity	The workflow updates Investment Manager Summary Month with Daily Data.
0450	w_p0450_update_rrsm_for_daily_acti vity	This workflow updates Registered Representative Sum-mary Month table from Registered Representative Summary Temp.
0460	w_p0460_update_masm_for_daily_acti vity	This workflow updates the Managed Account Summary Month Table from its corresponding staging table.
0471	w_p0471_update_asm_for_profit_and_ loss	This workflow update the profit and loss related fields in the account summary month table.
2010	w_p2010_update_nm_from_ma	This workflow aggregates managed account net worth and count of sub-accounts, grouping by Investment Manager. It updates the Investment Manager table.
MLM_Banking_Common Folder		
0081	w_ph0081_truncate_cbsm	This workflow truncates Client Bank Summary.
0480	w_ph0480_aggregate_asm_to_cbsm	This workflow performs daily re-aggregation of the Cor-respondent Bank Summary Month table out of the account summary month stage.
2090	w_ph2090_update_jurisdiction_in_cb	This workflow updates the jurisdiction for CLIENT_BANK (Correspondent Bank).
3130	w_ph3130_create_client_banks_from_ fotps	This workflow maintains the Correspondent Bank table. It derives the records from the `FOTPS` table.
MLM_Banking_Production Folder		
0080	w_p0080_dump_cbsm_to_file	This workflow can be used for the nightly dump of Cor-respondent Bank Summary Month Table to File (current month).
0140	w_p0140_reload_cbsm_from_dump_file	This workflow can be used to rebuild Correspondent Bank Summary Month table from dump file.
0145	w_ph0145_truncate_CBPTxSM	This workflow truncates the `CB_PEER_TRXN_SMRY_MNTH` table.
0146	w_ph0146_CBSM_to_CBPTxSM	This workflow populates the `CB_PEER_TRXN_SMRY_MNTH` table from `CLIENT_BANK_SMRY_MNTH` table.
MLM_Brokerage_Common Folder		
2071	w_ph2071_truncate_das	This workflow is used to Truncate the `DAILY_AGG_STAGE` table.
2122	w_ph2122_aggregate_tsv_offsetting_ trades_to_tdtcs	This workflow populates `DAILY_AGG_STAGE` table with aggregated TRADE Data. `DAILY_AGG_STAGE` table in turn is used to populate `OFFSETING_ACCT_PAIRS` and `TRADE_DAILY_TOT_CT_STAGE` tables.
2192	w_ph2192_update_inst_instn_seq_id	This workflow marks all institutions with an Oracle Financial Services generated INTSN_SEQ_ID in `INSTRUCTION`.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
3041	w_ph3041_create_addresses_from_inst	This workflow maintains the addresses in the Derived Address table. It derives the addresses from the INSTRUCTION table.
3120	w_ph3120_create_external_entities_from_inst	This workflow maintains the External Entity table. It derives the entities from the INSTRUCTION table.
3160	w_ph3160_write_inst_associations_to_ls	Load the Link Stage with any entity associations from instruction.
3171	w_ph3171_update_account_customer_risk	This workflow updates the risk on the ACCT based on KYC, Primary customer, as well as other external risks. It does not update the risk based on the EL process.
3220	w_ph3220_update_bot_activity_risk	This workflow updates the risk related values to all parties in BOT.
3230	w_ph3230_update_inst_activity_risk	This workflow updates the risk related values to all parties in INSTRUCTION.
MLM_Brokerage_Production Folder		
0020	w_p0020_dump_csm_to_file	This workflow can be used for the nightly dump of Customer Summary Month Table to File (current month).
0030	w_p0030_dump_iasm_to_file	This workflow can be used for the nightly dump of Institutional Account Summary Month Table to File. (Current Month).
0090	w_p0090_reload_csm_from_dump_file	This workflow can be used to rebuild Customer Summary Month Table from dump file.
0100	w_p0100_reload_iasm_from_dump_file	This workflow can be used to rebuild Institutional Account Summary Month Table from dump file.
0160	w_p0160_truncate_csms	This workflow truncates Customer Summary Month Stage.
0170	w_p0170_truncate_iasms	This workflow truncates institutional account summary month stage.
0210	w_p0210_aggregate_bot_to_file	This workflow aggregates (sums and counts) various fields from BOT for the account summary table. It puts the results in a flat file for later processing.
0220	w_p0220_aggregate_fotps_to_file	This workflow aggregates (sums & counts) various fields from the FOTPS for the institution account summary and customer summary tables. It puts the results in a flat file for later processing.
0230	w_p0230_aggregate_deals_to_file	This workflow is used for Daily Aggregation of Deals. This populates the Institutional account deal activity file and customer deal activity file.
0240	w_p0240_aggregate_instructions_to_file	This workflow performs Daily Aggregation of Instructions. This populates the Institutional account deal activity file and customer deal activity file.
0250	w_p0250_aggregate_trades_with_ca_to_file	This workflow performs Daily Aggregation of Trades with Corporate Actions. This populates the Institutional account deal activity file and customer deal activity file.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
0260	`w_p0260_aggregate_trade_to_file`	This workflow performs Daily Aggregation of Trades. This populates the Institutional account deal activity file and customer deal activity file.
0280	`w_p0280_create_clog_activity_records`	This workflow creates Change Log records that indicate a change in an accounts activity level as measured by the sum of deposits, withdrawals, and trades over a configurable time period (months).
0290	`w_p0290_join_instl_acct_activity_to_iasms`	This workflow joins institutional account and customer summary files and load them into institutional account summary month stage table.
0300	`w_p0300_join_instl_acct_activity_to_csms`	This workflow joins institutional account and customer summary files and load them into customer summary month stage table.
0330	`w_p0330_update_iasm_for_daily_activity`	This workflow performs Update of Institutional Account Summary Month Table from its corresponding staging table.
0340	`w_p0340_update_csm_for_daily_activity`	This workflow updates the Customer Summary Month Table from its corresponding staging table.
1030	`w_p1030_update_cbps_from_deal`	This workflow counts the records in the Deal table which has an end date greater than or equal to the current date by customer. It updates Customer Balance position table.
2020	`w_p2020_delete_nonstanding_instructions`	This workflow can be used to delete old Instructions.
2040	`w_p2040_update_cust_count_of_accounts`	This workflow calculates the total number of accounts for an institutional customer.
ORION_Common Folder		
2005	`w_p2005_set_cd_pd_dates`	This workflow sets the CD_Date and PD_Date values to the CDPDdate file. The post_session_success_command updates the Parameter file with CD_Date and PD_Date entries.
2050	`w_ph2050_update_bot_reversals`	This Workflow handles reverserals for Back Office Transactions.
2070	`w_ph2070_truncate_cas`	The workflow truncates `CUST_ACCT_STAGE`.
2100	`w_ph2100_truncate_atps`	This workflow truncates `ACCT_TRD_POSN_STAGE`.
2110	`w_ph2110_update_bot_unrelated_party_code`	This workflow populates the unrelated party code for BOT.
2120	`w_ph2120_load_ATPS_from_trade`	This workflow populates trades into a staging table for use later in calculating open/close position and day trades. All non-canceled trades for a day are inserted.
2130	`w_ph2130_update_fot_unrelated_party_code`	This workflow populates the unrelated party code for FOT.
2140	`w_ph2140_pass_thru_process`	This workflow kicks off the Pass Thru process. It generates second originator and beneficiary records for Front Office Transaction. It also sets the pass thru flag based on the a set of expressions.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
2172	w_ph2172_update_jurisdiction_in_ag	This workflow updates the jurisdiction for ACCT_GROUP.
2180	w_ph2180_instn_identification	This workflow creates unique identifiers for banks based on the third party vendors.
2190	w_ph2190_fotps_instn_processing	This workflow marks all institutions with an Oracle Financial Services generated INTSN_SEQ_ID in FOTPS.
2191	w_ph2191_anticipatory_profile_instn_processing	This workflow marks all institutions with an Oracle Financial Services generated INTSN_SEQ_ID in the Anticipatory Profile tables.
2193	w_ph2193_INS_instn_processing	This workflow marks all institutions with an Oracle Financial Services generated INTSN_SEQ_ID in the insurance table.
2200	w_ph2200_build_trxn_tables_from_fotps	This workflow builds the records for all front office transaction tables.
2220	w_ph2220_update_fot_reversals	This workflow adjusts the reversals for front office transaction tables.
2450	w_ph2450_populate_ACCT_SRVC_TEAM	This workflow maintains ACCT_SRVC_TEAM out of ACCT_SRVC_TEAM_MEMBER.
2600	w_ph2600_loan_smry_mnth	This workflow maintains LOAN_SMRY_MNTH table.
3000	w_ph3000_Adjust_WL_WLS	This workflow takes into account the modifications of the WL based on the new user interface WL utility.
3005	w_ph3005_apply_cust_KYC_risk	This workflow finds CUST records that have effective risks that are lower than the KYC risk for the same customer and updates the CUST record's effective risk with the high KYC risk.
3010	w_ph3010_truncate_ls	The workflow truncates Link Stage.
3020	w_ph3020_truncate_nms	The workflow truncates Name Match Stage.
3030	w_ph3030_truncate_wls	The workflow truncates Watch List Stage.
3040	w_ph3040_truncate_wls2	The workflow truncates Watch List Stage2.
3051	w_ph3051_create_addresses_from_fotps	This workflow maintains the addresses in the Derived Address table. It derives the addresses from the FOTPS table.
3061	w_ph3061_create_addresses_from_INS	This workflow maintains the addresses in the Derived Address table. It derives the addresses from the INSURANCE table.
3070	w_ph3070_load_watch_list_staging_table	This workflow determines changes in the Watch List table Each entry is classified as Add, No Change, or Retire based on the comparison of the current-day watch list data to the previous-day watch list data. The output is written to a temporary table.
3080	w_ph3080_create_external_entities_from_INS	This workflow maintains the External Entity table. It derives the entities from the INSURANCE table.
3090	w_ph3090_create_external_entities_from_fotps	This workflow maintains the External Entity table. It derives the entities from the Front Office transaction table.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
3091	w_ph3091_create_PartyNmEE_from_fotps	This workflow maintains the names in External Entities from FOTPS.
3100	w_ph3100_load_staging_fuzzy_matches	This workflow is a wrapper for the fuzzy matching mappings and scripts.
3110	w_ph3110_write_INS_associations_to_ls	This workflow loads the Link Stage with any entity associations from INSURANCE.
3140	w_ph3140_write_fotps_associations_to_ls	This workflow loads the Link Stage with any entity associations from FOTPS.
3150	w_ph3150_load_staging_and_validate_watch_list	This workflow validates entities and their membership lists. It starts with WLS and creates records in WLS2.
3170	w_ph3170_update_staging_list_risk	This workflow defines the highest risk for each entity based on various sources.
3180	w_ph3180_write_ls_to_link_tables	This workflow writes link-stage associations to various link tables.
3190	w_ph3190_apply_risk_to_nonacct_entities	This workflow applies all risk related values to all entity table except account.
3191	w_ph3191_apply_risk_to_acct_entities	This workflow applies all risk related values to ACCT table.
3200	w_ph3200_apply_membership_to_entities	This workflow maintains the membership tables based on the current WL processing results.
3210	w_ph3210_update_fotps_activity_risk	This workflow updates the risk related values to all parties in FOTPS.
3240	w_ph3240_update_INS_activity_risk	This workflow updates the risk related values to all parties in INSURANCE.
3501	w_ph3501_Exp_and_Risk_Review_TP	This mapping sets the status of a Trusted Pair to expire based on its Expiry Date. Also, if $$TP_RISK_REVIEW_FLAG is set to 'Y' then this mapping reviews/updates the risks for IA and EE parties associated with trusted pairs to reflect the latest risk as in the base tables. If they have increased by substantial amount to move them to a next risk zone it is recommending risk cancellation (RRC).
3502	w_ph3502_Flag_Trusted_Trxn	This mapping flags the transactions as trusted or not trusted based upon entry in the kdd_trusted_pair and kdd_trusted_pair_mbr tables. It only looks at today's transactions.
ORION_Production Folder		
0010	w_p0010_dump_asm_to_file	This workflow is used for the nightly dump of Account Summary Month Table to file for the current month.
0070	w_p0070_reload_asm_from_dump_file	This workflow can be used to rebuild ACCT Summary Month table from the dump file This workflow truncates Account Summary Month Stage.

Table 120. Informatica Workflow Description (Continued)

Workflow Number	Workflow Name	Description
0150	`w_p0150_truncate_asms`	This workflow truncates Account Summary Month Stage.
0180	`w_p0180_aggregate_bot_to_file`	This workflow is used for the daily aggregation of the back office transaction data. It populates the account summary file.
0190	`w_p0190_aggregate_fotps_to_file`	This workflow aggregates (sums and counts) various fields from the FOTPS for the account summary table. It puts the results in a flat file for later processing.
0200	`w_p0200_aggregate_trade_to_file`	This workflow aggregates (sums and counts) various fields from daily trades for the account summary table. It puts the results in a flat file for later processing
0270	`w_p0270_join_retail_acct_activity_to_asms`	This workflow joins retail account daily activity from files and loads the data to account summary stage table.
0310	`w_p0310_update_asm_for_daily_activity`	This workflow updates the Account Summary Month Table from its corresponding staging table.
0470	`w_p0470_update_asm_for_net_worth`	This workflow updates Account Summary Month, current month data, with daily net worth balance data.
0550	`w_ph0550_aggregate_IPSS_from_IT`	This workflow aggregates the Insurance transaction records into Insurance Transaction stage.
0551	`w_ph0551_IPSS_from_IPB`	This workflow aggregates the Insurance policy balance into Insurance Transaction stage.
0552	`w_ph0552_build_IPSD_from_IPSS`	This workflow performs updates of Insurance Policy Summary Daily Table from the Insurance Policy Summary Month Stage table.
0553	`w_ph0553_build_IPSM_from_IPSS`	This workflow performs updates of Insurance Policy Summary Month Table from its corresponding staging table.
0554	`w_ph0554_truncate_IPSS`	This workflow truncates the Insurance Policy Summary Stage table.
0630	`w_p0630_build_ATxSD_from_ASMS`	This workflow maintains Account transaction summary daily from account summary stage table.
0640	`w_p0640_build_ATdSD_from_ASMS`	This workflow maintains Account trade summary daily from account summary stage table.
0650	`w_p0650_build_IASD_from_ASMS`	This workflow, unlike its stated name, builds institutional account summary daily from the corresponding staging table.
0660	`w_p0660_FOTPSR_to_AASD`	This workflow populates `ACCT_ATM_SMRY` table from `FO_TRXN_PARTY_STAGE_RISK` table based on `TRXN_EXCTN_DT`.
0750	`w_ph0750_truncate_APTxSM`	This workflow truncates the `ACCT_PEER_TRXN_SMRY_MNTH` table.
0760	`w_ph0760_ASM_to_APTxSM`	This workflow populates the `ACCT_PEER_TRXN_SMRY_MNTH` table from `ACCT_SMRY_MNTH` table.

APPENDIX E

Moving Oracle Financial Services System Environment

This appendix describes the steps to move Oracle Financial Services Behavior Detection system from one environment to another.

Moving PROD Environment to UAT

After taking backup of Oracle database, follow the steps to move Oracle Financial Services Production environment to the UAT:

1. Move database to UAT using the backup.

   Configure Informatica (Refer to *Chapter 3, Installing the Application Server,* in *Installation Guide*, for more information on Informatica).

2. Update the Informatica counter in `KDD_COUNTER` table (Refer to *Installation Guide*, for more information on updating the `KDD_COUNTER` table).
3. Initialize all Case Sequence present in the Case Schema.
4. Setup the Oracle Financial Services Behavior Detection Platform User Interface (UI), (Refer to *Chapter 4, Installing Oracle Financial Services User Interface*, in *Installation Guide*).
5. Set up Oracle Financial Services Alert Management and Case Management (Refer to *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration Release 7.3* and *Oracle Financial Services Enterprise Alert and Case Management Stage 3 Installation Guide,* for more information). Refer to *Scenario Migration Best Practices,* on page 283, for more information.

Index