Administration Guide

Oracle Financial Services:

Anti-Money Laundering |Fraud| Trading Compliance |Broker Compliance |Energy and Commodity Trading Compliance| Personal Trading Approval

Release 8.0.5.0.0
October 2017

Administration Guide

Oracle Financial Services:

Anti-Money Laundering|Fraud| Trading Compliance|Broker Compliance|Energy and Commodity Trading Compliance| Personal Trading Approval

Release 8.0.5.0.0
October 2017

Part Number: E85394-01

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

Part Number: E60570_01
First Edition (October 2017)

Trademarks

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

Revision History

The following table describes the revision history of the Administration Guide.

Date	Edition	Description
October 2017	First edition of 8.0.5.0.0	In Chapter 3, *Managing Data,* updated the following sections: ● Processing Data Using FDT and MDT ● Generating Change Logs with Hive ● Setting Up Batches Created Chapter 6, *Managing Personal Trading Approval*, to provide OFS Personal Trading Approval application-specific information. In Appendix E, *BD Datamap Details,* added the following sections: ● Firm Data Transfer Datamaps

Date	Edition	Description
March 2017	First edition of 8.0.4.0.0	Global changes made: ● Changed `OFSBDF Installed` to `OFSAAI Installed` in all references to code found in this document ● Changed references to FCCM to BD in this document Chapter 1, *About Oracle Financial Services Behavior Detection (OFSBD),* updated the following sections: ● Deployment View image ● Tiers and subsystems-related content Chapter 2, *Managing User Administration and Security Configuration,* moved the following sections to *Appendix C, User Administration:* ● Managing User Group and User Roles ● Managing User Groups ● Defining User Access Properties and Relationships ● Accessing objects under Metadata Browser In the same chapter, moved *Managing Additional Configurations*, *Configuring Alert and Case Management*, and *Assign Employee Users to Personal Trading Approval Attestation Questionnaire* sections to the BD Configuration guide as these sections are related to configuration. Introduced a new chapter, Chapter 3, *Managing Data,*. This chapter has been written using content from chapters *Data Ingestion-Flat File Interface, Ingestion Data from Staging Area, and BDF Datamaps* of the previous version. Introduced two new sections in the chapter: ● Trade Finance Datamaps: Includes datamaps related to trade finance. ● Trusted Pair: Includes information about the trusted pair DIS file. In Chapter 3, *Managing Data,* provided content related to hive and sections for change logs in BDF and T2T In Chapter 7, *Managing Batch Processing Utilities,* ● Updated the *Managing ETL Process for Threshold Analyzer Utility* section In *Appendix D, Managing Data:* ● Added *Group 1* and *Country* data file. ● Added the *Add CustomerCreditRating* data file in *Group 2*. ● Moved the *Customer* data file from *Group 4* to *Group 3*. ● Moved the *CustomerIdentificationDocument* data file from *Group 5* to *Group 2*. ● Added the DQ group names and corresponding T2T and H2T names
July 2016	Second edition of 8.0.2.0.0	In Chapter 3, *Managing Data,* updated the following tables: ● DataIngest.properties File Configuration Parameters ● BDF Datamap Configuration Parameters In Chapter 7, *Managing Batch Processing Utilities,* updated the following figure and tables: ● Sample install.cfg File ● Scenario Extraction Parameters ● Scenario Load Parameters

Date	Edition	Description
Feb 2016	First edition of 8.0.2.0.0	In Chapter 4, Ingestion Data from Staging Area, added the following table: ● *Data Trade Finance related Datamaps*
July 2015		The following general changes have been made: ● Moved from Multiple Schemas to Unified Schemas ● Ingestion and Detection supported from OFSAA batch.

Contents

APPENDIX E Processing Derived Tables and Fields 327

APPENDIX F BD Datamap Details 341

List of Figures

List of Tables

About this Guide

This guide explains the concepts behind the Oracle Financial Services Behavior Detection (OFSBD), 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

This *Administration Guide* is designed for use by the Installers and System Administrators. Their roles and responsibilities, as they operate within OFSBD, include the following:

- **Installer:** Installs and configures OFSBD at a specific deployment site. The Installer also installs and upgrades any additional Oracle Financial Services solution sets and requires access to deployment-specific configuration information, such as machine names and port numbers).
- **System Administrator:** Configures, maintains, and adjusts the system, and is usually an employee of a specific Oracle customer. The System Administrator maintains user accounts and roles, monitors data management and alert management, archives data, loads data feeds, and performs post-processing tasks. In addition, the System Administrator can reload cache.

Note: Administrators who have access to any of the Financial Crime and Compliance Management (FCCM) modules such as Enterprise Case Management, Anti-Money Laundering, Fraud, and so on, will get unrestricted access to the administration utilities that are required to administer the module.

Prerequisites for an Administrator User

User must have knowledge of UNIX and LINUX.

Scope of this Guide

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

OFSBD is powered by advanced data mining algorithms and sophisticated pattern recognition technologies. It provides an open and scalable infrastructure that supports rich, end-to-end functionality across all Oracle Financial Services solution sets. OFSBD's extensible, modular architecture enables a customer to deploy new solution sets readily as the need arises.

This guide provides information about how to administer the following products:

- Anti-Money Laundering (AML)
- Fraud
- Energy and Commodity Trade Compliance (ECTC)
- Broker Compliance (BC)
- Trader Compliance (TC)
- Personal Trading Approval (PTA)

Note: Your implementation may not include all of these products.

How this Guide is Organized

The *Administration Guide*, includes the following chapters:

- *Chapter 1, About Oracle Financial Services Behavior Detection (OFSBD),* provides a brief overview of the Oracle Financial Services Framework and its components.
- Chapter 2, *Managing User Administration and Security Configuration,* covers the required day-to-day operations and maintenance of OFSBD users, groups, and organizational units.
- *Chapter 3,* *Managing Data,* describes the operation and process flow of data management subsystem components.
- Chapter 4, *Behavior Detection Jobs,* provides an overview of the BDF job protocol and procedures for performing various tasks that relate to starting, stopping, and recovering jobs.
- Chapter 5, *Post-Processing Tasks,* explains how to customize the OFSBD features that affect presentation of user information on the desktop.
- Chapter 6, *Managing Personal Trading Approval,* explains how to perform Personal Trading Approval (PTA) application-specific administrative tasks.
- Chapter 7, *Managing Batch Processing Utilities,* provides information about the OFSBD utilities related to the batch process.
- Chapter 8, *Managing Administrative Utilities,* provides information about the OFSBD utilities that are independent of the batch process.
- Chapter 9, *Posting External Alerts through Batches,* provides information about how to post alerts from an external system into the OFSBD.

- Chapter 10, *Alert Generation from IPE Assessment Results,* provides information about how to generate alerts using the Inline Processing Engine (IPE) Assessments.
- Appendix A, *Logging*, describes the OFSBD logging features.
- Appendix B, *OFSBD Software Updates*, describes the application of OFSBD software updates (hotfix) and their impact on customization.
- Appendix C, *User Administration*, describes the user administration of the Oracle Financial Services Behavior Detection.
- Appendix D, *Managing Data*, describes the BDF file parameters, the FSDF datamaps, the Data Quality group names and related T2T names, the BDF interface files, and the directory structures.
- Appendix E, *Processing Derived Tables and Fields*, describes the additional data processing activities that can be performed in the BD applications.
- Appendix F, *BD Datamap Details,* lists the Datamap XML and their use in OFSBD.
- Appendix G, *Datamaps Matrix* lists which datamaps are required for each solution set.
- Appendix H, *Configuring Administration Tools* describes how to configure the Administration Tools feature.
- Appendix I, *Mapping Compliance Regulatory Reports Actions* provides information about integration of OFSRRS.
- Appendix J, *Alerts from IPE and External System - Run/Process/Tasks* provides information about integration of the Inline Processing Engine.
- 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, refer to the following Behavior Detection application documents, which can be found at http://docs.oracle.com/cd/E60570_01/homepage.htm:

- *Scenario Manager User Guide*
- *Administration Tools User Guide*
- *Services Guide*
- *Data Interface Specification (DIS)*
- *BD Configuration Guide*
- *BD Installation Guide*
- *KYC Administration Guide*

Additionally, you may find pertinent information in the OFSAAI documentation, found at the following link: http://docs.oracle.com/cd/E60058_01/homepage.htm:

- *Oracle Financial Services Analytical Applications Infrastructure User Guide*
- *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration*

For installation and configuration information about Sun Java System, BEA, and Apache software, refer to the appropriate documentation that is available on the associated websites.

Conventions Used in this Guide

This table lists the conventions used in this guide and their associated meanings.

Table 1. Conventions Used in this Guide

Convention	Meaning
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

Abbreviations Used in this Guide

This table lists the abbreviations used in this guide and their associated descriptions.

Table 2. Abbreviations Used in this Guide

Abbreviation	Description
OFSBD	Oracle Financial Services Behavior Detection
AML	Anti-Money Laundering
TC	Trader Compliance
BC	Broker Compliance
ECTC	Energies, Commodities and Trading Compliance
T2T	Table to Table
H2T	Hive to Table
T2H	Table to Hive
AAI	Analytical Applications Infrastructure
CSA	Common Staging Area
FSDM	Financial Services Data Model
BD	Behavior Detection
OFS	Oracle Financial Services
KYC	Know Your Customer

Table 2. Abbreviations Used in this Guide (Continued)

Abbreviation	Description
FATCA	Foreign Account Tax Compliance Act
DQ	Data Quality
DT	Data Transformation

CHAPTER 1

About Oracle Financial Services Behavior Detection (OFSBD)

This chapter provides a brief overview of the Oracle Financial Services Behavior Detection (OFSBD) in terms of its architecture and operations.

This chapter focuses on the following topics:

- Behavior Detection Architecture
- Operations
- Utilities

Behavior Detection 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 architecture:

- **Tiers:** 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.

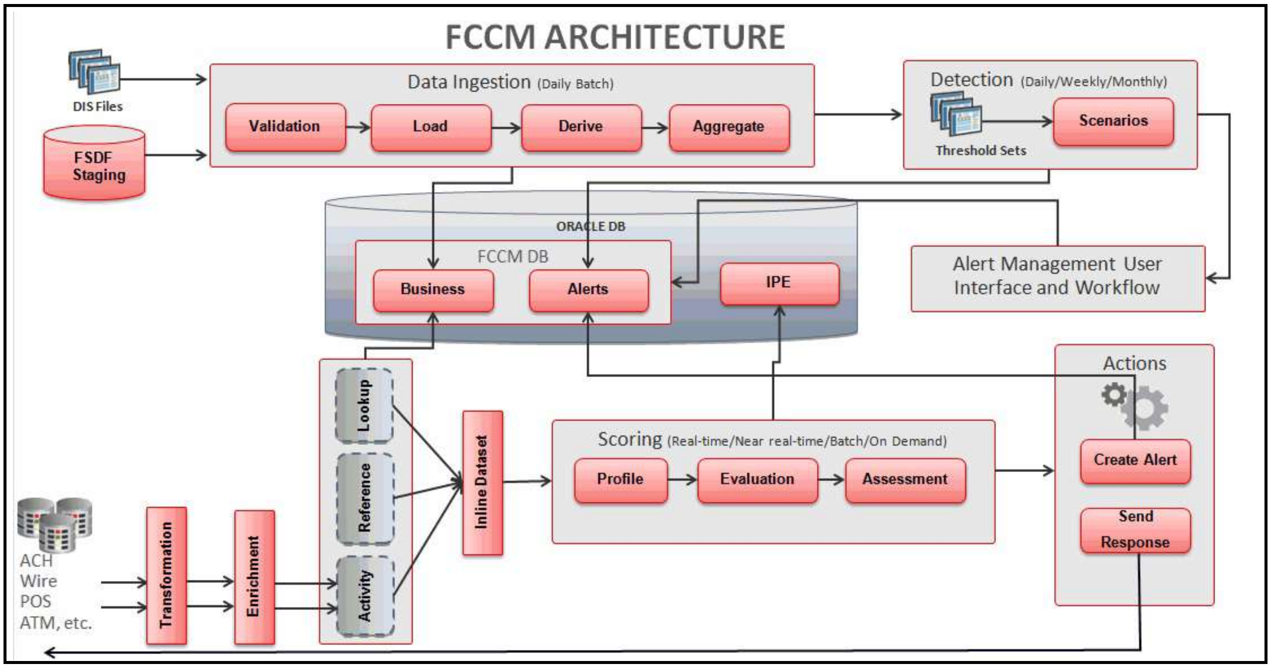


Figure 1. OFSBD Architecture

The architecture is composed of a series of tiers and components. Each tier can include one or more components that are divided into small installable units. A solution set requires installation of the associated components.

Tiers

Tiers represent a product or logical grouping of products under which there may be common components and subsystems. The following image is a graphical representation of the tiers:

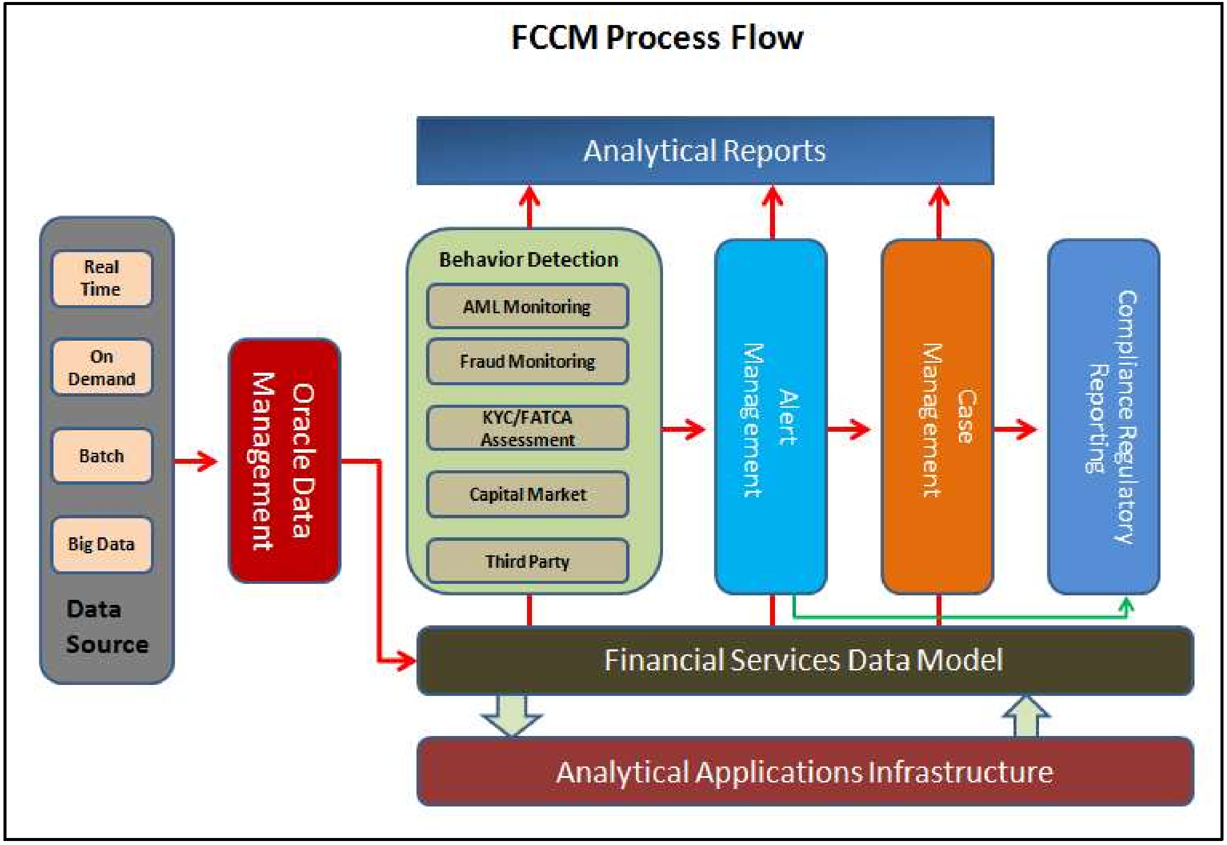


Figure 2. Tiers

The following are the tiers:

- Oracle Financial Services Analytical Applications Infrastructure (OFSAAI)
- Oracle Financial Services Behavior Detection (OFSBD)
- Oracle Financial Services Behavior Detection Applications

The following sections describe the tiers and their components.

Oracle Financial Services Analytical Applications Infrastructure (OFSAAI)

Oracle Financial Services Analytical Applications Infrastructure is the complete end-to-end Business Intelligence solution that allows you to tap your organization's vast store of operational data to track and respond to business trends. It also facilitates analysis of the processed data. Using OFSAAI, you can query and analyze data that is complete, correct, and consistently stored at a single place. It can filter data that you are viewing and using for analysis.

Oracle Financial Services Behavior Detection (OFSBD)

Oracle Financial Services Behavior Detection (OFSBD) contains the following subsystems:

- **Data Management:** Provides data preparation logical functions, which include adapters for files and messages. The functions also include datamap XML for data derivations and aggregations.
- **Behavior Detection:** Provides data access, behavior detection, and job services, which include Oracle Financial Services Behavior Detection (OFSBD), Financial Services Data Model (FSDM), and scenarios specific to a particular solution set.
- **Alert Management**: Provides a user interface and workflow for managing alerts, reporting, and searching business data.

A set of components further divides each OFSBD subsystem. Components are units of a tier that can be installed separately onto a different server. Table 3 outlines the tiers and components. When installed, contents and files related to these components can be located in the folder listed in the Directory Name column. The location and paths to these folders may vary depending on your specific implementation. In some cases, individual deployments can add subsystems to meet a client's custom requirements.

Table 3. Data Management Components

Component	Directory Name	Contents
	ingestion_manager	Java components, scripts, and stored procedures
Financial Services Data Model	database	Database utilities and database creation scripts
BDF Datamaps	bdf	Datamap XML and configuration parameters.

Table 4. Behavior Detection Components

Component	Directory Name	Contents
Behavior Detection	behavior_detection	(Subsystem)
Behavior Detection	bdf	Datamap XML and configuration parameters.
Detection Algorithms	algorithms	C++ behavior detection algorithms
Scenario Manager	toolkit	Job and scenario editors

Table 5. Alert Management Components

Component	Directory Name	Contents
Alert Management Web	solution\am	JSPs used in Alert Management
Alert Management UI	ftpshare\< alert infodom>\erwin\forms	XMLs for rendering the UI
Web Services	services	Web services for watch list scanning and for the alert management supervisor (used when posting alerts to Behavior Detection)
Correlation		
Administration Tools	admin_tools	Web-enabled Administration Tools
Trade Blotter		
Manage Security Restriction		

Table 5. Alert Management Components (Continued)

Component	Directory Name	Contents
Manage Controlling Customer		
Watch List Management		

Data Management

The Oracle Financial Services Ingestion Manager receives, transforms, and loads Market data, Business data (such as, Transactions or Orders and Trades), and Reference data (such as Account and Customer and Employee information) that alert detection processing requires. The template for receiving this information is defined in the *Data Interface Specification (DIS)*. 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 Management subsystem transforms Market, Business, and Reference data to create derived attributes that the detection algorithms require (much of the loaded data is as is). 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 processing datamaps to derive and aggregate data. The Data Management 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 AAI Batch Scheduler to control batch processing of Ingestion Manager.

Behavior Detection

OFSBD uses sophisticated pattern recognition techniques to identify behaviors of interest, or scenarios, that are indicative of potentially interesting behavior. A *pattern* is a specific set of detection logic and match generation criteria for a particular type of behavior. These behaviors can take multiple representations in a firm’s data.
OFSBD detection modules are divided into scenarios that typify specific types of business problems or activities of interest. The scenarios are grouped into scenario classes that represent categories of behaviors or situations that have common underlying characteristics. The scenario class dictates the action choices available and the data that is displayed when an alert is created.

Alert Management

An alert represents a unit of work that is the result of the detection of potentially suspicious behavior by Oracle Scenarios. OFSBD routinely generates alerts as determined by the configuration of the application in your environment, typically nightly, weekly, monthly, and quarterly. Alerts can be automatically assigned to an individual or group of users and can be reassigned by a user. Alert Management contains the following components:

- Alert Management screens, actions, and workflows to support triage of an alert
- Trusted Pairs
- Correlations
- Trade Blotter (This is an optionally licensed component and will depend on your implementation.)
- Controlling Customers
- Security Restrictions

- Suppression Rules
- Watch List Management

Oracle Financial Services Behavior Detection Applications

Oracle solutions, such as Trading Compliance, Anti-Money Laundering, Broker Compliance, Energy and Commodity Trading Compliance, Fraud Detection, Trade Finance, Currency Transaction Reporting, and FATCA Management, extend the Oracle Financial Services Behavior Detection Applications pack. 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.

Oracle Financial Services Enterprise Case Management (OFSECM) enables your firm to manage and track the investigation and resolution of cases related to one or more business entities involved in potentially suspicious behavior. Cases can be manually created within OFSECM or may represent a linked collection of alerts generated by OFSBD that have been promoted to a case (if your firm has implemented Oracle Financial Services Behavior Detection). When used in conjunction with OFSBD, based on your roles and permissions, you can link or unlink additional alerts to a case during the investigation.

Deployment View

The OFSBD architecture from the perspective of its deployment illustrates deployment of the major subsystems across servers. Additionally, the deployment view shows the primary communications links and protocols between the processing nodes.

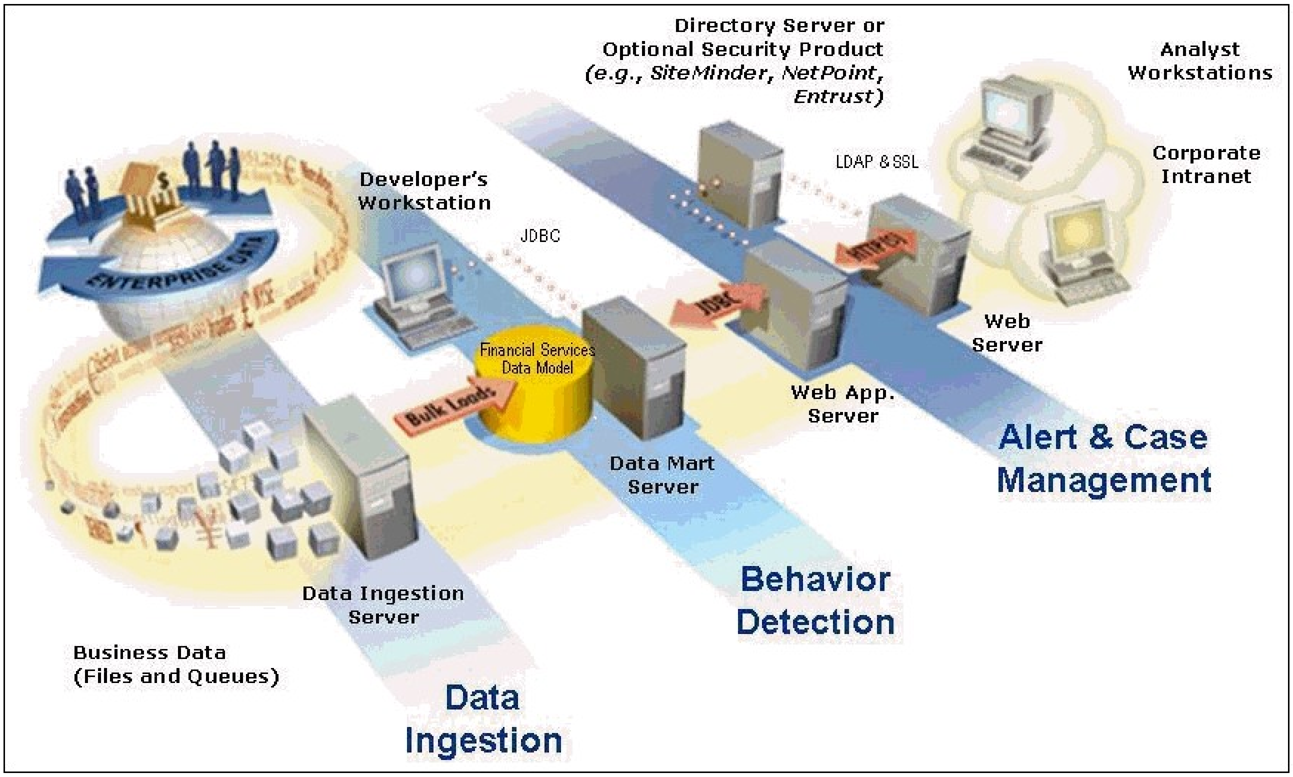


Figure 3. OFSBD Architecture - Deployment View

The complex interactions between the components of the Alert Management and Enterprise Case Management tiers becomes apparent in the deployment view. The Alert Management and Enterprise Case Management tiers require the following:

- Web browser
- Web server
- Web application server

Oracle Financial Services Alert Management and Enterprise Case Management tiers use OFSAAI 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.

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

Security View

The security view describes the architecture and use of security features of the network in a Behavior Detection architecture deployment. Behavior Detection uses an inbuilt Security Management System (SMS) for its authentication and authorization. The SMS has a set of database tables which store information about user authentication.

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

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

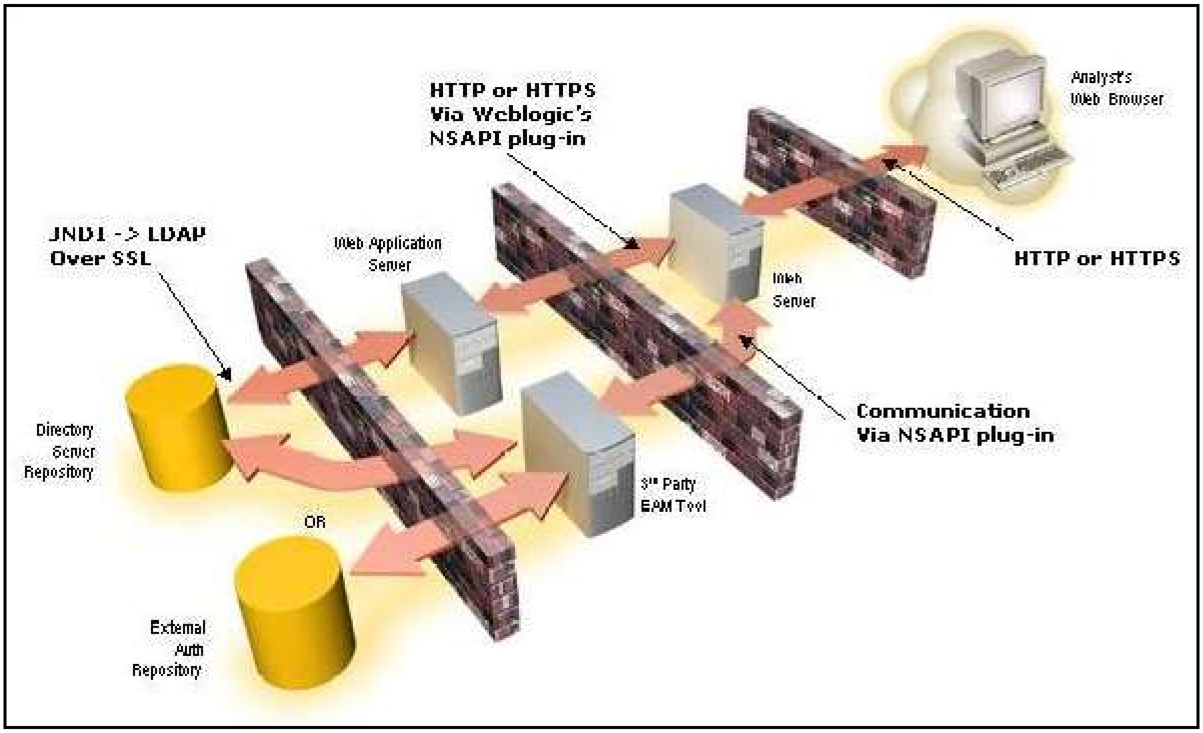


Figure 4. Security View

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 OFSBD directory server uses.

Operations

As the OFSBD administrator, you coordinate the overall operations of OFSBD: Data Management, Behavior Detection, and Post-Processing.

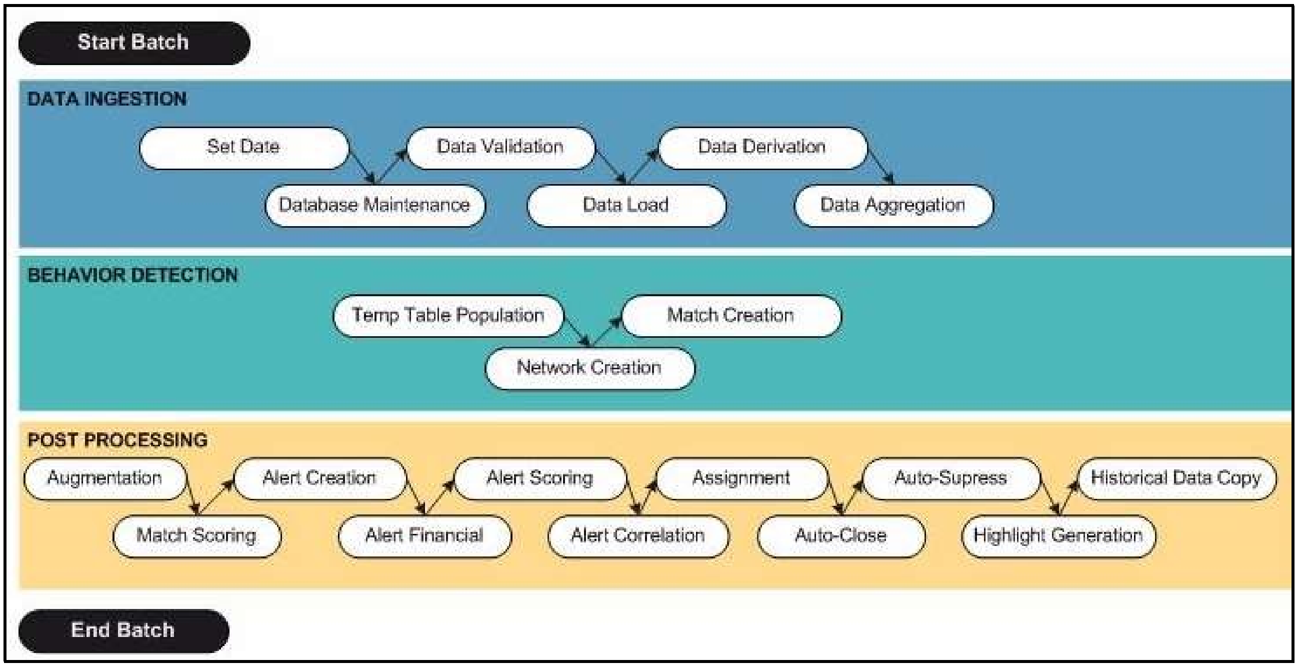


Figure 5. OFSBD Architecture—Behavior Detection Framework Processing

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

As Figure 5 illustrates, each cycle of OFSBD process begins with Data Management, 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
- Managing Data
- Behavior Detection
- Post-Processing
- End Batch

Start Batch

Using the Batch Control Utility, you can manage the beginning of the OFSBD batch process (see *Chapter 6 - Managing Batch Processing Utilities* for more information).

Managing Data

The OFSBD Ingestion Manager controls the Data Management process. The *Data Interface Specification (DIS)* contains specific definition of the types and format of business data that can be accepted for ingestion.

The Ingestion Manager supports files and messages for the ingestion of data. Data Management 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 OFSBD database with the results (see *Chapter 3 - Managing Data* for more information).

Behavior Detection

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

OFSBD 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. This is only relevant for certain scenarios.
3. A match is created by executing scenarios. These scenarios are used to detect the behaviors of interest 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 of detection results, Behavior Detection prepares the detection results for presentation to users. Preparation of the results depends upon the following processes:

- **Match Scoring**: Computes a ranking for scenario matches indicating a degree of risk associated with the detected event or behavior.
- **Alert Creation**: Packages the scenario matches as units of work (that is, alerts), potentially grouping similar matches together, for disposition by end users. This is applicable when multiple matches with distinct scores are grouped into a single alert.
- **Update Alert Financial Data**: Records additional data for alerts such as the related Investment Advisor or Security involved in the alert which may be useful for display and analysis.
- **Alert Scoring**: Ranks the alerts (including each match within the alerts) to indicate the degree of risk associated with the detected event or behavior.
- **Alert Assignment**: Determines the user or group of users responsible for handling each alert.
- **Auto-Close**: Based on configurable rules, closes alerts which are considered to be of lower priority based on attributes of the alert or the alert focus.
- **Automatic Alert Suppression**: Suppresses alerts that share specific scenario and focal entity attributes for a particular time frame. This process will only impact alerts which match suppression logic defined for a specific scenario and focal entity combination.
- **Highlight Generation**: Generates highlights for alerts that appear in the alert list in the Alert Management subsystem and stores them in the database.
- **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.
- **Default Augmentation**:
- **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.
- **Historical Data Copy**: Identifies the records against which the current batch's scenario runs generated alerts and copies them to archive tables. This allows for the display of a snapshot of information as of the time the alert behavior was detected.
- **Alert Correlation:** Uncovers relationships among alerts by correlating alerts to business entities and subsequently correlating alerts to each other based on these business entities. The relationships are discovered based on configurable correlation rule sets.
- **Case Assignment**: Determines the user or group of users responsible for handling each case.
- **Alert Notification:** Sends e-mail to assignees about the alerts that are assigned to them.

End Batch

The system ends batch processing when processing of data from the Oracle client is complete (see *Ending a Batch Process,* for more information). The Alert & Case Management subsystem then controls the alert and case management processes. See *Alert Management User Guide* and *Enterprise Case Management User Guide* for more information.

Utilities

OFSBD database utilities enable you to configure and perform pre-processing and post-processing activities. The following sections describe these utilities.

- Batch Utilities
- Administrative Utilities

Batch Utilities

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

- **Alert Purge Utility**: Provides the capability to remove erroneously generated matches, alerts, and activities.
- **Batch Control Utility**: Manages the start and termination of a batch process (from Data Management to alert post-processing) and enables access to the currently running batch.
- **Calendar Manager Utility**: Updates calendars in the system based on pre-defined business days, holidays, and *days off*, or non-business days.
- **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.
- **Database Statistics Management**: Manages statistics in the database.
- **Flag Duplicate Alerts Utility**: 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.
- **Push Email Notification**: Enables you to configure users of the Alert Management subsystem to receive email when alerts are assigned to them.
- **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.
- **Refreshing Temporary Tables**: Refreshes temporary tables that the Behavior Detection process uses and estimates statistics for the newly populated tables.
- **Truncate Manager**: Truncates tables that require complete replacement of their data.

For more information on Administrative Utilities, see *Managing Batch Processing Utilities*.

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.
- **Get Dataset Query with Thresholds Utility:** Enables you to extract dataset SQL complete with substituted thresholds for analysis of the SQL outside of the Behavior Detection application.
- **Scenario Migration Utility:** Extracts scenarios, datasets, networks, and associated metadata from a database to flat files and loads them into another environment.
- **Alert Correlation Rule Migration Utility**: Enables you to move correlation rules and their audit trails from a source environment to a target environment.
- **Investigation Management Configuration Migration Utility**: Enables you to load data related to alerts and cases into the OFSBD.
- **Watch List Services**: Enables you to query the BD watch lists to find a specific or a partial match.
- **Alert Processing Web Services**: Enables you to execute additional processing steps in an existing service operation.
- **Password Manager Utility**: Enables you to change a password for a specific user in a subsystem apart from alert management and administration tools.
- **Oracle Sequences**: Enables you to update and maintain the Oracle sequences used in OFSBD.

  For more information on Administrative Utilities, see *Managing Administrative Utilities*.

CHAPTER 2

Managing User Administration and Security Configuration

This chapter provides instructions for setting up and configuring the Security Management System (SMS) to support Behavior Detection (BD) applications, user authentication, and authorization.

This chapter focuses on the following topics:

- About User Administration
- Administrator User Privileges
- User Provisioning Process Flow
- Managing User Administration
- Adding Security Attributes
- Mapping Security Attributes to Organizations and Users

About User Administration

User administration involves creating and managing users and providing access rights based on their roles. This section discusses the following:

- Administrator permissions
- Creating and mapping users and user groups
- Loading and mapping security attributes

Administrator User Privileges

The following table lists the access permissions of the administrators depending on the different product suite under BD:

Table 1. Access Permissions for Administrators

Privileges	Alert Management Administrator
User Security Administration	x
Excel Upload	x
Alert Assigner Editor	x
Alert Creator Editor	x
Alert Scoring Editor	x
Web Service Configuration	x
Common Web Service	x
Reports	x

Table 1. Access Permissions for Administrators

Privileges	Alert Management Administrator
Preferences	x
User Administration	x
Security Management System	x
Security Attribute Administration	x
Manage Common Parameters	x
Case Management Configuration	NA
Case Assigner Editor	NA
Unified Metadata Manager	x

Note: If KYC/FATCA is deployed with BD, the respective Administrator must be mapped with the KYC/FATCA Administrator group, as well for other BD-related access.

Note: An AM administrator also has the role of Personal Trading Approval (PTA) Administrator.

User Provisioning Process Flow

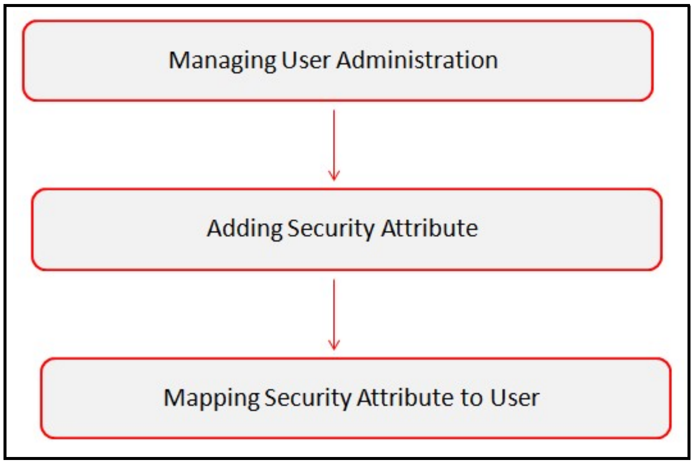

Managing User Administration
Adding Security Attribute
Mapping Security Attribute to User

Figure 1. User Provisioning Process Flow

The following table lists the various actions and associated descriptions of the user administration process flow:

Table 2. User Provisioning Process Flow

Action	Description
Managing User Administration	Create users and map users to user groups. This allows Administrators to provide access, monitor, and administer users.
Adding Security Attributes	Load security attributes. Security attributes are loaded using either Excel or SQL scripts.
Mapping Security Attributes to Organizations and Users	Map security attributes to users. This is done to determine which security attributes control the user's access rights.

Requirements to Access BD Applications

A user gains access to BD applications based on the authentication of a unique user ID and password.

To access the BD applications, you must fulfill the following conditions:

Table 3. Requirements

Applications	Conditions
Alert Management	● Set of privileges 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.
Watch List Management	● Set of policies that associate functional roles with access to specific system functions. ● Access to one or more jurisdictions. ● Access to one or more business domains.
Administration Tools	Set of policies that associate the admin functional role with access to specific system functions.

Managing User Administration

This section allows you to create, map, and authorize users defining a security framework which has the ability to restrict access to the respective BD applications.

Managing Identity and Authorization

This section explains how to create a user and provide access to BD applications.

This section covers the following topics:

- Managing Identity and Authorization Process Flow
- Creating and Authorizing Users and User Groups
- Mapping Users with User Groups

Managing Identity and Authorization Process Flow

The following figure shows the process flow of identity management and authorization:

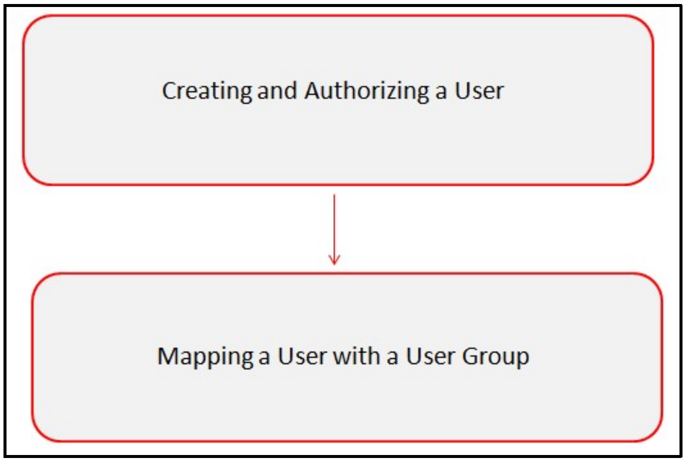

Creating and Authorizing a User
Mapping a User with a User Group

Figure 2. Managing Identity and Authorization Process Flow

The following table lists the various actions and associated descriptions of the user administration process flow:

Table 4. Administration Process Flow

Action	Description
Creating and Authorizing Users and User Groups	Create a user. This involves providing a user name, user designation, and the dates between which the user is active in the system.
Mapping Users with User Groups	Map a user to a user group. This enables the user to have certain privileges that the mapped user group has.

Creating and Authorizing Users and User Groups

The SYSADMN and SYSAUTH roles can be provided to users in the BD application. User and role associations are established using Security Management System (SMS) and are stored in the config schema. User security attribute associations are defined using Security Attribute Administration.

For more information on creating and authorizing a user, see *Chapter 9*, *Oracle Financial Services Analytical Applications Infrastructure 8.0.4 User Guide.*

Mapping Users with User Groups

This section explains how to map Users and User Groups. With this, the user will have access to the privileges as per the role. The SYSADMN user maps a user to a user group in the BD application.The following table describes the predefined Alert Management User Roles and corresponding User Groups.

Table 5. Alert Management (AM) 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

Table 5. Alert Management (AM) Roles and User Groups (Continued)

Role	Group Name	User Group Code
AM Scenario Group	AM Scenario Group User Group	AMDATAMNRGRP
Alert Management Administrator	Mantas Administrator User Group	AMMANADMNGR

The following table describes the KYC and FATCA Case Management User Roles and corresponding User Groups.

Table 6. KYC and FATCA Case Management Roles and User Groups

Role	Group Name	User Group Code
KYC Relationship Manager	KYC Relationship Manager User Group	CMKYCRMUG
KYC Supervisor	KYC Investigator User Group	CMKYCINVSTGTRUG
KYC Analyst	KYC Analyst User Group	CMKYCANALYSTUG
KYC Administrator	KYC Administrator User Group	KYCADMNGRP
FATCA Supervisor	FATCA Supervisor User Group	FTCASUPERVISRUG
FATCA Analyst	FATCA Analyst User Group	FTCAANALYSTUG
FATCA Auditor	FATCA Auditor User Group	FTCAAUDITORUG
FATCA Administrator	FATCA Admin User Group	FTCAADMINUG

The following table describes the Watch List User Roles and corresponding User Groups.

Table 7. Watch List Roles and User Groups

Role	Group Name	User Group Code
Watch List Supervisor	Watchlist Supervisor Group	WLSUPERVISORUG

The following table describes the Personal Trading Approval User Roles and corresponding User Groups.

Table 8. Personal Trading 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
CRQUSTROL	CRQUESTIONUG	CRQUESTIONUG
Alert Management Administrator	Mantas Administrator User Group	AMMANADMNGR

Note: A user with the Alert Management Administrator role will have administrative rights to Alert Management and Personal Trading Approval.

Note: If you want to change the user group mapping for users who are already mapped to one or more groups, you must deselect the preferences for the Home page if it has been set. To change the preferences, follow these steps:

1. In the Home page, click the user name. A drop-down list appears.
2. Click **Preferences**. The Preferences page appears.
3. Select the appropriate Property Value.
4. Click **Save**.

Users should not be mapped to both the CR Supervisor/Analyst role and IP Manager/Manager Supervisor role. The only acceptable role combinations for a user are the Employee role and one of the following four roles:

- CR Supervisor
- CR Analyst
- IP Manager
- Manager Supervisor

  The maximum role combinations should be limited to two. For more information on mapping User with User Groups, see *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

Note: For any customized user group creation and user group-role mapping, see *Appendix C, User Administration*.

Adding Security Attributes

This section explains about security attributes, the process of uploading security attributes, and mapping security attributes to users in the BD application.

This section covers the following topics:

- About Security Attributes
- Loading Security Attributes

About Security Attributes

Security Attributes help an organization classify their users based on their geography, jurisdiction, and business domain, in order to restrict access to the data that they can view.

You need to map the roles with access privileges, and since these roles are associated with user groups, the users associated with the user groups can perform activities throughout various functional areas in the BD application.

Types of Security Attributes

The following are the security attributes:

- Jurisdiction
- Business Domain
- Scenario Group
- Organization

Jurisdiction

OFSFCCM solutions use 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 and users of the system must be associated with one or more jurisdictions. In the Alert Management system, users can view only data or alerts 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, states, and so on.
- **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. The list of jurisdictions in the system reside in the `KDD_JRSDCN` table.

Business Domain

Business domains are used for data access controls similar to jurisdiction but have a different objective. The business domain can be used to identify records of different business types such as Private Client verses 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. The system tags each data record provided through the Ingestion Manager to one or more business domains. It 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 assign this business domain to 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.

Scenario Group

Scenario groups are used for data access controls. A scenario group refers to a group of scenarios in the BD applications that identify a set of scenario permissions and to which a user has access rights. The list of scenario groups in the system resides in the `KDD_SCNRO_GRP` table.

Organization

Organizations are used for data access controls. Organizations are user group to which a user belongs. The list of Organizations in the system resides in the KDD_ORG table.

Loading Security Attributes

This section covers the following topics:

- Loading Security Attributes through Excel
- Loading Security Attributes through SQL Scripts

Loading Security Attributes through Excel

The Excel Upload process inserts the data into the appropriate dimension tables based on the pre-configured Excel Upload definitions installed during the application installation.

Note: Data which already exists must not be loaded again, as this results in failure of the upload. When uploading additional records, only the incremental records should be maintained in the Excel template with the correct unique identifier key.

- All template Excel files for Excel Upload are available in `ftpshare/STAGE/ExcelUpload/AMCMLookupFiles`
- All date values should be provided in `MM/DD/YYYY` format in the Excel worksheet.
- Whenever a record is deleted from the Excel worksheet, the complete row should be deleted (no blank active record should exist in the Excel worksheet).
- After selecting the Excel template, preview it before uploading.

Security attributes are loaded through Excel using the following templates:

Table 9. Security Attributes and Excel Templates

Security Attribute	Excel Template
Jurisdiction	`KDD_JRSDCN.xls`
Business Domain	`KDD_BUS_DMN.xls`
Scenario Group	`KDD_SCNRO_GRP.xls`
Scenario Group Member	`KDD_SCNRO_GRP_MEMBERSHIP.xls`
Organization	`KDD_ORG.xls`

Uploading Excel

To load the security attributes using excel, follow these steps:

1. Login as the Alert Management Administrator. The BD application home page is displayed.
2. Click **Alert Management**. The *Anti Money Laundering* page is displayed.
3. Mouse over the Administration menu and click **Excel Upload**. The *Excel Upload* dialog box is displayed.
4. Click **Excel Upload**.
5. Browse your system and select the Excel file.
6. Select **Sheet** from Sheet drop-down list.
7. Go to the Excel-Entity Mappings section. Click Arrow icon to select one or more Mapping IDs from the dialog box. The Excel is updated.

Loading Security Attributes through SQL Scripts

This section covers the following topics:

- Loading Jurisdictions
- Loading Business Domains
- Loading Scenario Groups
- Loading Scenario Group Memberships
- Loading Organizations

Loading Jurisdictions

To load jurisdictions in the database, follow these steps:

1. Add the appropriate record to the KDD_JRSDCN database table as mentioned in *Table 10*.

Table 10. KDD_JRSDCN Table Attributes

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

Note: The data in the KDD_JRSDCN database table is loaded through the ATOMIC schema.

2. Add records to the table using an SQL script similar to the sample script in following figure:

```
INSERT INTO KDD_JRSDCN (JRSDCN_CD, JRSDCN_NM,JRSDCN_DSPLY_NM,JRSDCN_DESC_TX)
VALUES ('E', 'East', 'East', 'Eastern')
```

Note: The KDD_JRSDCN table is empty after system initialization and needs to be populated before the system can operate.

Loading Business Domains

To load a business domain, follow these steps:

1. Add the appropriate user record to the KDD_BUS_DMN database table as mentioned in the *Table 11.*

Table 11. KDD_BUS_DMN Table Attributes

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

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

2. Add more records to the table using a SQL script similar to the sample script in the following figure:

```
INSERT INTO KDD_BUS_DMN (BUS_DMN_CD, BUS_DMN_DESC_TX, BUS_DMN_DSPLY_NM, MAN-
TAS_DMN_FL) VALUES (‘a’, ‘Compliance Employees’, ‘COMP’, ‘N’);

INSERT INTO KDD_BUS_DMN (BUS_DMN_CD, BUS_DMN_DESC_TX, BUS_DMN_DSPLY_NM, MAN-
TAS_DMN_FL) VALUES (‘b’, ‘Executives’
‘EXEC’, ‘N’);

COMMIT;
```

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'
```

Loading Scenario Groups

To load a Scenario Group, follow these steps:

1. Add the appropriate value in the KDD_SCNRO_GRP database table as mentioned in the *Table 12*.

Table 12. KDD_SCNRO_GRP Table Attributes

Column Name	Description
SCNRO_GRP_ID	Scenario group identifier
SCNRO_GRP_NM	Scenario Group Name

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

```
INSERT INTO KDD_SCNRO_GRP(SCNRO_GRP_ID,SCNRO_GRP_NM) VALUES (66,'BEX');

INSERT INTO KDD_SCNRO_GRP(SCNRO_GRP_ID,SCNRO_GRP_NM) VALUES (77,'CST');

COMMIT;
```

Loading Scenario Group Memberships

To load a Scenario Group Membership, follow these steps:

1. Add the appropriate value in the KDD_SCNRO_GRP_MEMBERSHIP database table as mentioned in *Table 13*.

Table 13. KDD_SCNRO_GRP_MEMBERSHIP Table Attributes

Column Name	Description
SCNRO_ID	Scenario Identifier
SCNRO_GRP_ID	Scenario Group Identifier
SCNRO_GRP_NM	Scenario Group Name

2. Add more records to the table using a SQL script similar to the sample script in the following figure.

```
INSERT INTO KDD_SCNRO_GRP_MEMBERSHIP (SCNRO_ID,SCNRO_GRP_ID,SCNRO_GRP_NM) VAL-
UES (113000016,66,'BEX') ;
INSERT INTO KDD_SCNRO_GRP_MEMBERSHIP (SCNRO_ID,SCNRO_GRP_ID,SCNRO_GRP_NM) VAL-
UES (113000016,77,'CST') ;
```

Loading Organizations

To load an organization in the database, follow these steps:

1. Add the appropriate user record to the KDD_ORG database table as mentioned in *Table 14*.

Table 14. KDD_ORG Table Attributes

Column Name	Description
ORG_CD	Unique identifier for this organization.
ORG_NM	Short name for this organization that is used for display purposes.
ORG_DESC_TX	Description of this organization.
PRNT_ORG_CD	Parent organization of which this organization is considered to be a child. **NOTE:** This should reference an ORG_CD in the KDD_ORG table.
MODFY_DT	Last modified date and time for this organization record.
MODFY_ID	User ID of the user who last modified this organization data. **NOTE:** This should reference a user in the Investigation Owner table (KDD_REVIEW_OWNER.OWNER_SEQ_ID). You can also set the value to owner_seq_id 1, which is SYSTEM, if another suitable ID is not available.
COMMENT_TX	Additional remarks added by the user.

2. Add more records to the table using a SQL script similar to the sample script in the following figure.

```
INSERT INTO KDD_ORG (ORG_CD,ORG_NM,ORG_DESC_TX,PRNT_ORG_CD,MODFY_DT,MOD-
FY_ID,COMMENT_TX) VALUES ('ORG1','COMPLIANCE ORG','DEPARTMENT FOR INVESTIGA-
TION','ORG1 PARENT ORG','01-JUN-2014',1234,'ADDING KDD_ORG ENTRIES')
```

Mapping Security Attributes to Organizations and Users

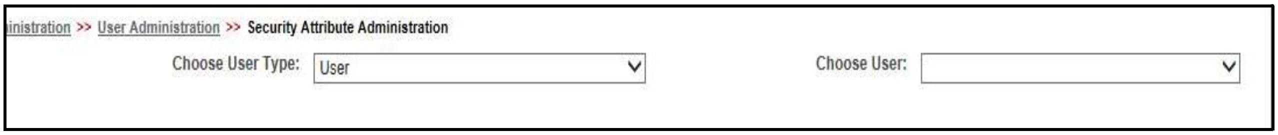
The Mapping Security Attributes to Users functionality/section enables you to determine which security attribute controls a user's access. Using this UI, an Administrator can map both Organizations and Users to different Security attributes.

To map a Security Attribute, follow these steps:

1. Login as the Alert Management Administrator. The BD applications home page is displayed.
2. Click **Alert Management**. The *Anti Money Laundering* page is displayed.
3. Mouse over the Administration menu, select the User Administration sub-menu, and click **Security Attribute Administration**. The *Security Attribute Administration* page is displayed.
4. Select user type from Choose User Type drop-down list. The following options are available:

- Organization
- User.

Note: Before proceeding with providing a user access through this UI, ensure that you have created a user and all necessary data is available in the appropriate database tables.



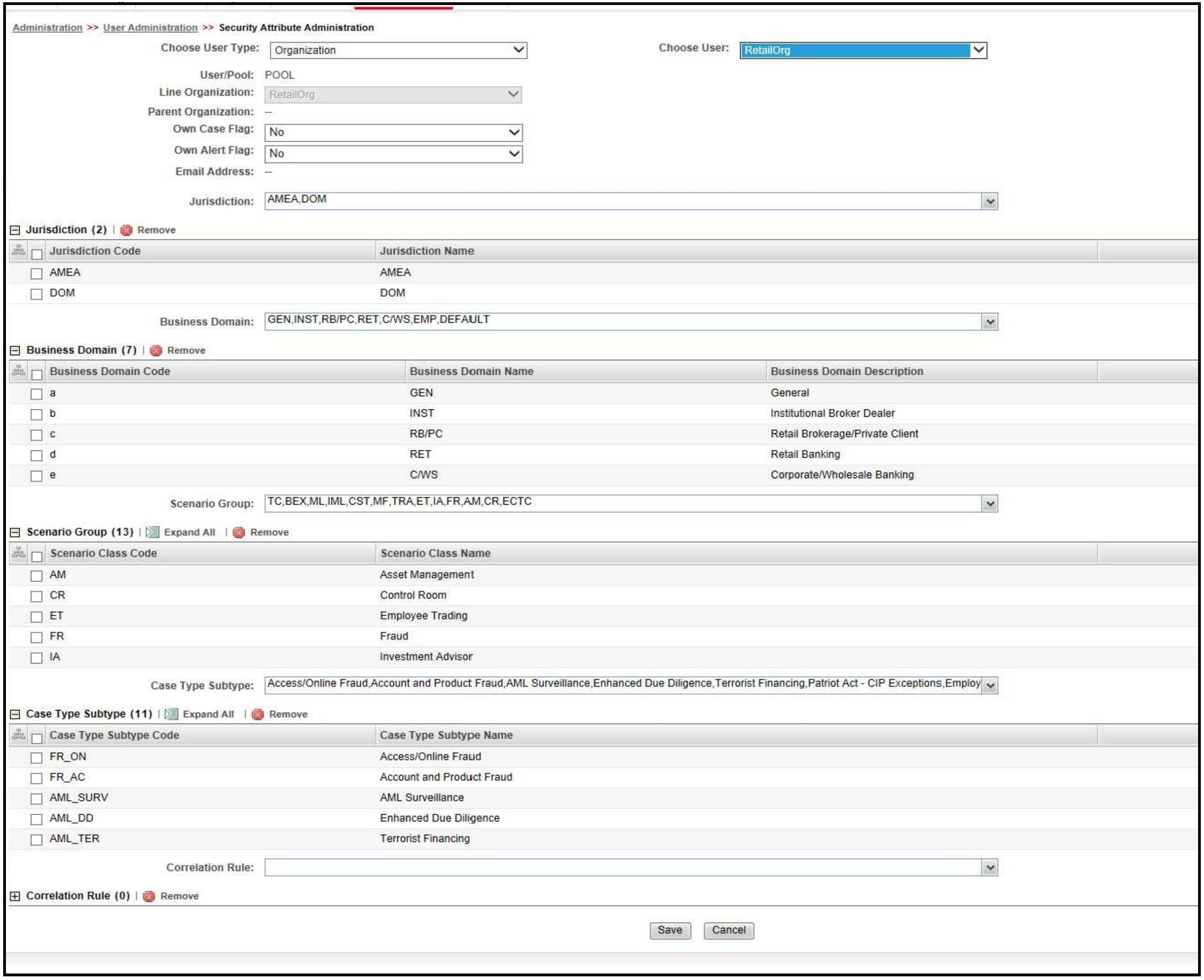
inistration >> User Administration >> Security Attribute Administration

Choose User Type: User

Choose User:

Figure 3. Security Attribute Administration

Depending on the User Type you have selected, the available options in the Choose User drop down list is updated. Select the user from Choose User drop-down list. The relevant *Security Attribute Administration* page is displayed.



Administration >> User Administration >> Security Attribute Administration

Choose User Type: Organization

Choose User: RetailOrg

User/Pool: POOL

Line Organization: RetailOrg

Parent Organization: --

Own Case Flag: No

Own Alert Flag: No

Email Address: --

Jurisdiction: AMEA,DOM

⊟ Jurisdiction (2) | Remove

□	Jurisdiction Code	Jurisdiction Name
□	AMEA	AMEA
□	DOM	DOM

Business Domain: GEN,INST,RB/PC,RET,C/WS,EMP,DEFAULT

⊟ Business Domain (7) | Remove

□	Business Domain Code	Business Domain Name	Business Domain Description
□	a	GEN	General
□	b	INST	Institutional Broker Dealer
□	c	RB/PC	Retail Brokerage/Private Client
□	d	RET	Retail Banking
□	e	C/WS	Corporate/Wholesale Banking

Scenario Group: TC,BEX,ML,IML,CST,MF,TRA,ET,IA,FR,AM,CR,ECTC

⊟ Scenario Group (13) | Expand All | Remove

□	Scenario Class Code	Scenario Class Name
□	AM	Asset Management
□	CR	Control Room
□	ET	Employee Trading
□	FR	Fraud
□	IA	Investment Advisor

Case Type Subtype: Access/Online Fraud,Account and Product Fraud,AML Surveillance,Enhanced Due Diligence,Terrorist Financing,Patriot Act - CIP Exceptions,Employ

⊟ Case Type Subtype (11) | Expand All | Remove

□	Case Type Subtype Code	Case Type Subtype Name
□	FR_ON	Access/Online Fraud
□	FR_AC	Account and Product Fraud
□	AML_SURV	AML Surveillance
□	AML_DD	Enhanced Due Diligence
□	AML_TER	Terrorist Financing

Correlation Rule:

⊞ Correlation Rule (0) | Remove

Figure 4. Security Attribute Administration

Note: In order to update the user profiles before proceeding with mapping any security attributes, select **User** from the **Choose User Type** drop-down list. When selected, all the updates made to all the user profiles through User Maintenance UI are imported from the `CSSMS_USR_PROFILE` table of the ATOMIC schema to the `KDD_REVIEW_OWNER` table of the ATOMIC schema.
If you delete a user through the *Security Management System* screen, you should come back to the *Security Attribute Administration* screen and select the value **User** from the **Choose User Type** drop-down list. Then the deleted user will be updated in the `KDD_REVIEW_OWNER` table against the column `actv_flg` as *N*, and that user is inactive.

Table 15. Security Attributes

Fields	Description
Organization	Select an organization from the drop-down list. A User or Organization's access to other Organizations depends on the selection(s) made for this organization parameter, such as, if a user is mapped to Org1 and Org2, it implies that this user can access alerts which belong to these two organizations, provided other security attributes are also matching.
Own Case Flag	Select whether this user type will own a case flag from the drop-down list.
Own Alert Flag	Select whether this user type will own a alert flag from the drop-down list.
Note: The Own Alert and Case flag is required for taking ownership of the alerts and cases. If an alert user must perform a Promote To Case action, then the following prerequisites should be fulfilled. The user should be mapped to any one of the following user groups: ● Case Supervisor ● Case Analyst1 ● Case Analyst2	
Business Organization	The default Business Organization is displayed, but you can select the business organization from the drop-down list.
Jurisdictions	Select the jurisdictions from the drop-down list. Mapping of one or more jurisdictions to a user or organization allows this user or organization to access cases, alerts, watch lists, and watch list members that belong to the mapped jurisdiction. The selected jurisdictions are displayed in Jurisdictions section after you save your selection.
Business Domain	Select the business domains from the drop-down list. Mapping of one or more business domains to a user or organization allows this user or organization to access cases, alerts, watch lists, and watch list members that belong to the mapped business domains. The selected jurisdictions are displayed in Jurisdictions section after you save your selection.
Scenario Group	Select the scenario group from the drop-down list. Mapping of one or more Scenario Groups to a user or organization allows this user or organization to access alerts that belong to the mapped scenario Group. The selected jurisdictions are displayed in Jurisdictions section after you save your selection.
Case Type/Subtype	Select the case type/subtype from the drop-down list. Mapping of one or more Case Types/Subtypes to a user or organization allows this user or organization to access cases that belong to the mapped Case Type/Subtype. The selected jurisdictions are displayed in Case Types/Subtypes section after you save your selection. This is only applicable if your firm has implemented Enterprise Case Management.
Correlation Rule	Select the correlation rule from the drop-down list. Mapping of one or more correlation rules allows the user to view the correlations generated based on the mapped correlation. The selected jurisdictions are displayed in correlation section after you save your selection.

5. Click **Save**. The following confirmation message displays: *Would you like to save this action?*
6. Click **OK**. The following confirmation message displays: *The update operation successful.*
7. Click **OK**. The updated *Security Attribute* page is displayed.

Removing Security Attributes

This section allows you to delete the mapped security with Users.

To remove security attributes, follow these steps:

1. Navigate to the *Security Attributes* page.
2. Select one or more check boxes in the respective security attributes such as Business Domain, Jurisdictions, and so on.
3. Click Remove. The following confirmation message displays: *Are you sure you want to delete this records*?
4. Click **OK**. The selected record is deleted from the list.
5. Click **Save**. The changes are updated.

CHAPTER 3 Managing Data

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

This chapter focuses on the following topics:

- About Data Management
- Data Loading and Processing Flow Overview
- Managing Data Loading
- Managing Data Processing
- Managing Data For BD Applications

About Data Management

Data Management consists of two main activities:

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

Data Loading and Processing Flow Overview

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

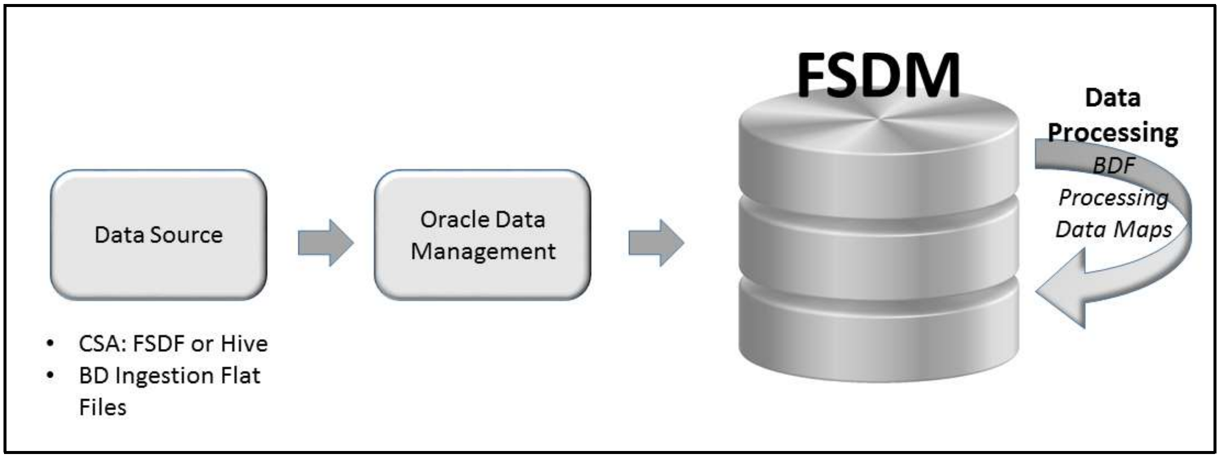


Figure 10. Data Loading and Processing Flow Overview

In BD applications, data is loaded into the FSDM from the following data sources:

- Common Staging Area (CSA) in either FSDF or Hive
- BD Flat File Interface

Data stored in the FSDM is then processed using BD processing datamaps where additional data derivations and aggregations are stored in the FSDM.

CSA

The CSA provides a single repository for data storage for multiple functional areas and applications having the Common Staging Area Model and Reporting Data Model. The Common Staging Area Model provides a simplified, unified data sourcing area for inputs required by FCCM using BD.

Flat Files

The flat files contain data provided by the client. This data is loaded into the Financial Services Data Model (FSDM).

FSDM

The FSDM is a database which consists of well organized business data for analysis. It determines the structured data which stores persistent information in a relational database and is specified in a data modeling language.

BD Datamaps

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

Managing Data Loading

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

The following approaches are used to load the data:

- FSDF CSA Data Load
- Hive CSA Data Load
- BD Ingestion Flat File Data Load
- Managing Data Processing

FSDF CSA Data Load

This section covers the following topics:

- Overview
- Using Table-to-Table (T2T) in the AAI Data Management Framework
- Using Behavior Detection Datamaps

Overview

The CSA Model provides a simplified, unified data sourcing area for inputs required by FCCM. It is the common data sourcing layer across all OFSAA applications and the OFSDF. In the CSA approach, you can load data using the Oracle Analytical Application Infrastructure (AAI) Table-to-Table (T2T) Data Management Framework and BD. The following figure provides an overview of the data loading flow using CSA:

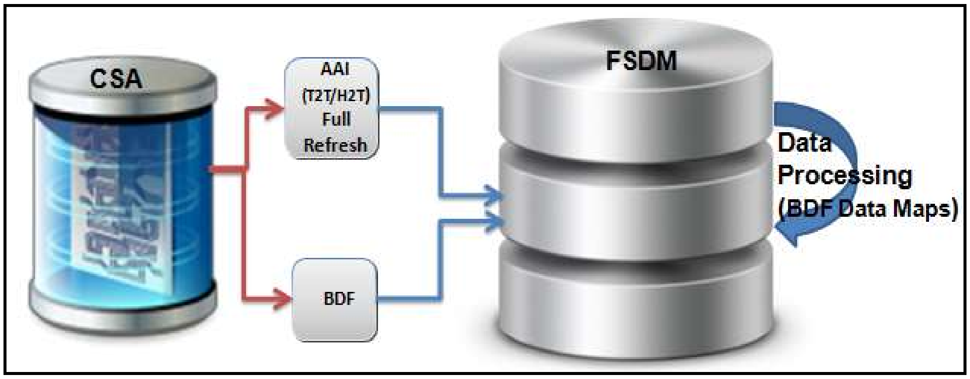


Figure 11. Data Management Flow Using CSA

Using Table-to-Table (T2T) in the AAI Data Management Framework

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

About AAI T2T Data Loading

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

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

This section covers the following topics:

- Process Flow for AAI T2T
- Setup Using AAI Batch
- Running Data Quality Batch
- Executing Data Transformation using DT
- Moving Data through T2T
- Executing Behavior Detection Jobs
- Ending the Batch

Process Flow for AAI T2T

The following figure shows the process flow for AAI T2T:

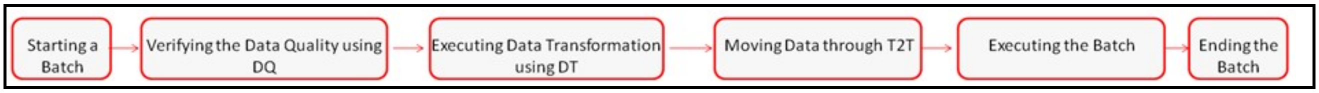


Figure 12. Process Flow for AAI T2T

Setup Using AAI Batch

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

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

Running Data Quality Batch

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

The following data quality checks are done:

- **Length Validation Check**: If the length of data in source column is more than the length of target column, then an error message is generated. For example, if the `ACCT_INTRL_ID` column, which has a column length of 50 characters, needs to be populated from the source table column `V_ACCOUNT_NUMBER`, which has a few data with length more than 50 characters. An error message is raised.

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

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

Executing Data Transformation using DT

The Data Transformation (DT) functionality allows you to delete the existing data in the AAI. For more information, see *Adding Tasks to a BD Batch*.

Moving Data through T2T

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

For the table to be loaded, the list of T2Ts are in *Managing Data*.

Executing Behavior Detection Jobs

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

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

Ending the Batch

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

Using Behavior Detection Datamaps

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

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

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

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

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

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

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

Hive CSA Data Load

Hive-to-Table (H2T) is used in the AAI Framework for data loading. The source for H2T data is the hive database.

About AAI H2T Data Loading

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

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

This section covers the following topics:

- Process Flow for AAI H2T
- Starting a Batch
- Verifying the Data Quality Using DQ
- Moving Data through T2H
- Executing Data Transformation using DT
- Moving Data through H2T
- Executing Behavior Detection Jobs
- Ending the Batch

Process Flow for AAI H2T

The following figure shows the process flow for AAI H2T:

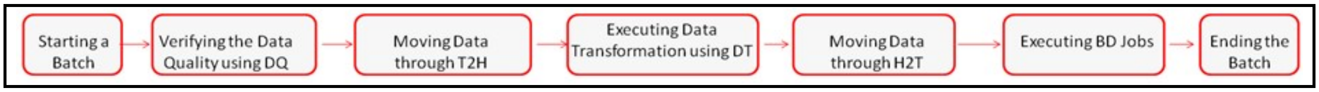


Figure 13. Process Flow for AAI H2T

Starting a Batch

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

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

Verifying the Data Quality Using DQ

Note: Execute the `Update_DQ_Tables.sql` file available in the FIC_HOME path on the ATOMIC schema database.

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

The following data quality checks are done:

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

Note: In addition to the above data checks, another data check is done for duplicate data during data loading through AAI H2T.

Moving Data through T2H

Data is exported or moved from the Oracle RDBMS to the hive database using AAI T2H. For moving data through T2H, the T2H batch must be executed from `RefData_Source` with the following parameters:

1. `[EXEC_ENV_TARGET]=<hive_infodom_name>,` which is the ICC batch parameter
2. `"EXEC_ENV_TARGET","<hive_infodom_name>",` which is the batch parameter if the batch is running through RRF.

For more information on moving data through T2H, see *Chapter 2*, *Section - Data Mapping* of the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

Executing Data Transformation using DT

The Data Transformation (DT) functionality allows you to delete the existing data in the AAI. For more information, see *Adding Tasks to a BD Batch*.

Moving Data through H2T

Data is exported or moved from the hive database to Oracle RDBMS using AAI H2T. For more information on moving data through H2T, see *Chapter 2*, *Section - Data Mapping* of the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

For the table to be loaded, the list of h2ts are in *Managing Data*.

Executing Behavior Detection Jobs

After the data quality check, data transformation, and data movement through T2H and H2T is completed, execute the following jobs:

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

Ending the Batch

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

BD Ingestion Flat File Data Load

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

This section covers the following topics:

- Overview
- Using Behavior Detection Datamaps (Known as BD datamaps)
- Using Pre-processing and Loading (Known as runDP- runDL)

Overview

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

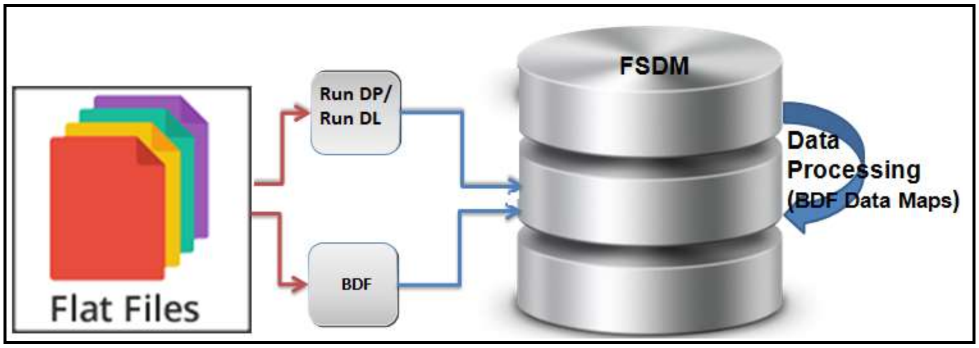


Figure 14. Data Loading Flow Using Flat File Interface

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

Using Behavior Detection Datamaps

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

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

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

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

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

Using Pre-processing and Loading

The pre-processor component (runDP) 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

The loader component (runDL) receive pre-processed Reference data and business data. The components then load this data into the database.

Note: The Pre-processor 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. Oracle clients must only supply file types required by the solution sets on their implementation.

To load data in the FSDM using Pre-processing and Loading, follow these steps:

1. Place the `ASCII.dat` flat files in the `<OFSAAI Installed Directory>/ingestion_manager/inbox` directory. The component then performs data validation and prepares the data for further processing.
2. Execute runDP and runDL using the following sample scripts:

- For runDP: `<OFSAAI Installed Directory>/ingestion_manager/scripts/runDP.sh AccessEvents`
- For runDL:`<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh AccessEvents`

Pre-processors place output files in the directories that *Table 33* lists. The following figure summarizes Pre-processing input and output directories.

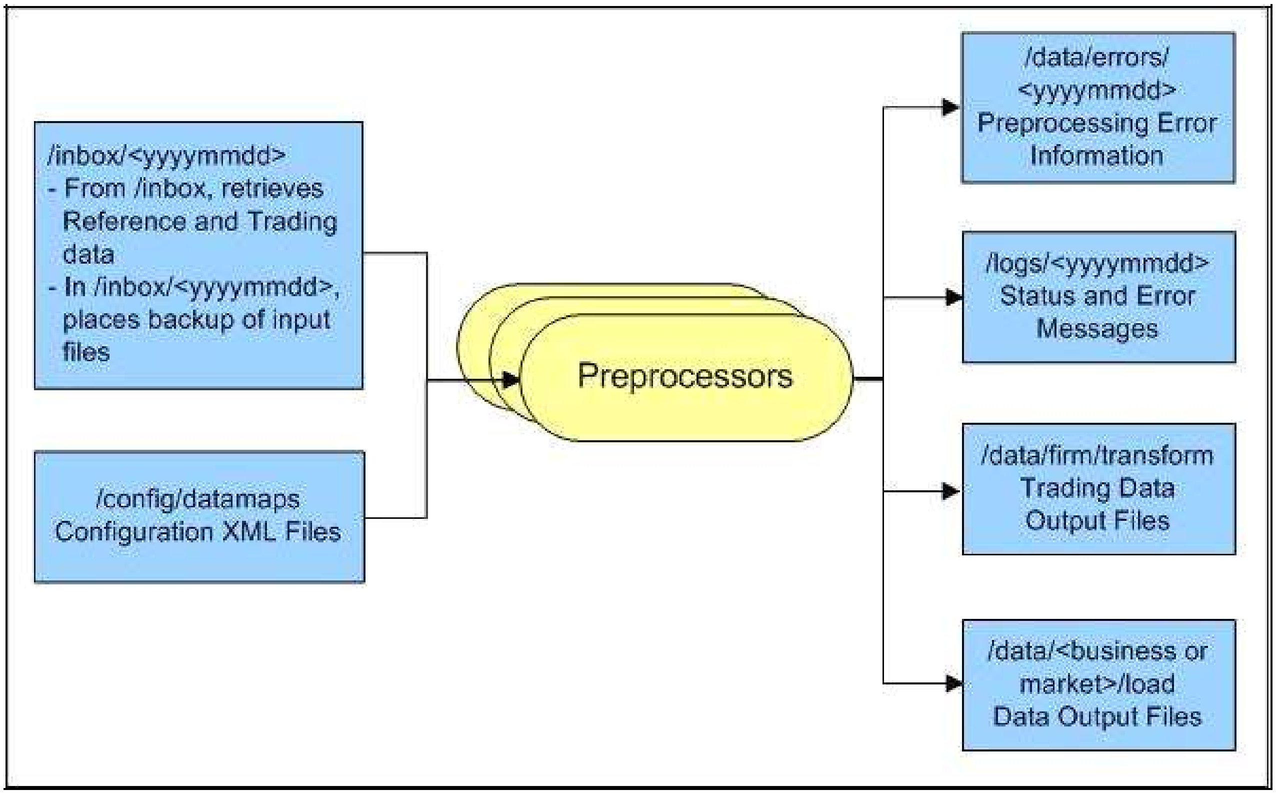

/inbox/<yyyymmdd>
- From /inbox, retrieves Reference and Trading data
- In /inbox/<yyyymmdd>, places backup of input files
/config/datamaps
Configuration XML Files
Preprocessors
/data/errors/<yyyymmdd>
Preprocessing Error Information
/logs/<yyyymmdd>
Status and Error Messages
/data/firm/transform
Trading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

Figure 15. Input and Output Directories

The following figure illustrates the Trading Compliance Solution data loading process.

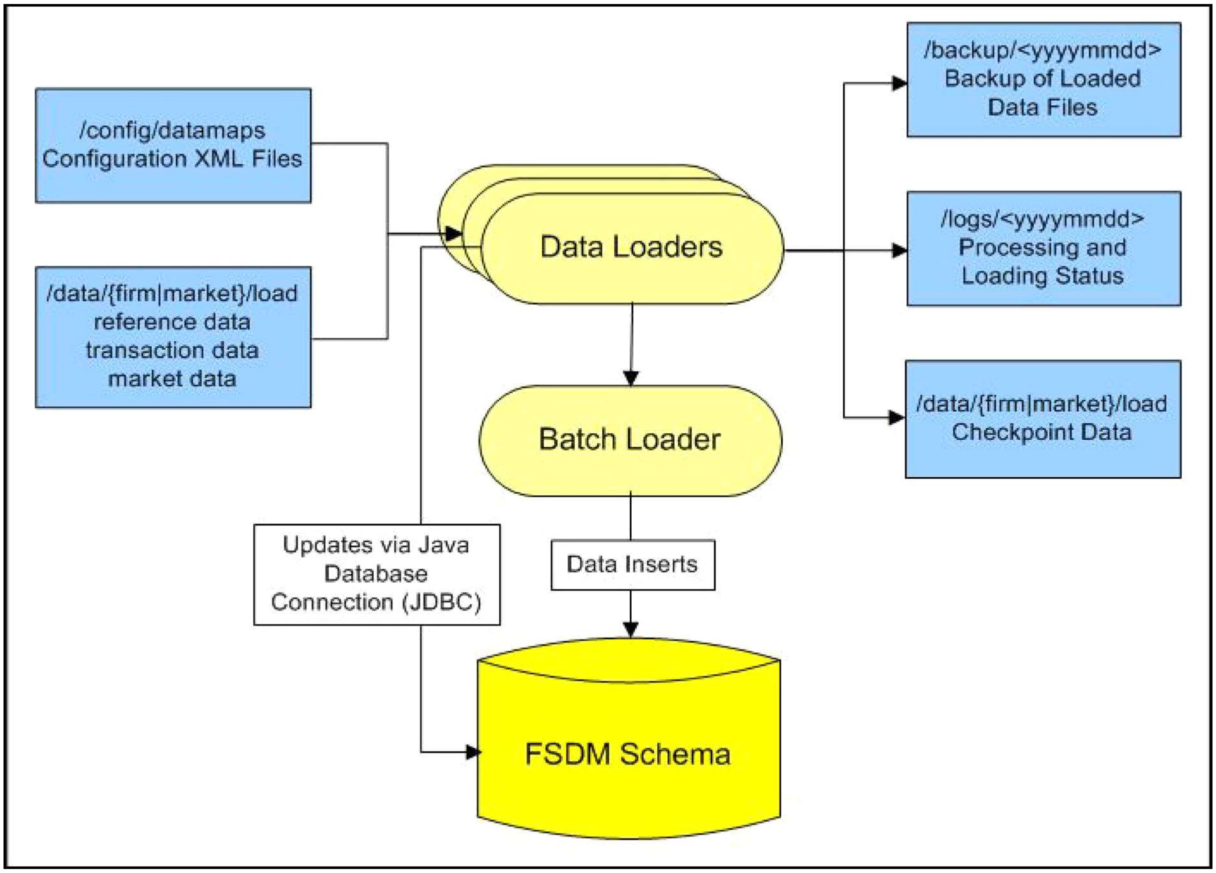


Figure 16. TCS Data Loading Process

Note: For more information on the directory structure, see *Appendix D, Managing Data*.

Configuring RunDP/RunDL

For flat files, Behavior Detection receives firm data in `ASCII.dat` flat files, which an Oracle client's data extraction process places in the `/inbox` directory.

Ways of Data Loading

This section covers the following topics:

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

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

Full Refresh Data Loading

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

To do a full refresh data load, set `load.fullrefresh` to true in the `<OFSAAI Installed Directory>/bdf/config/BDF.xml` path. For more information, see *BDF.xml Configuration Parameters*.

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

Incremental (Delta) Data Loading

For incremental data loading, the following can be done:

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

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

To do an incremental data load, set `load.fullrefresh` to false in the `<OFSAAI Installed Directory>/bdf/config/BDF.xml` path. For more information, see *BDF.xml Configuration Parameters*.

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

Managing Data Processing

This section explains the concept of data processing and various methods of data processing.

This section covers the following topics:

- Generating Change Logs with T2T
- Generating Change Logs with Hive
- Generating Change Logs with BD
- Processing Data Using BD
- Processing Data Using FDT and MDT

Generating Change Logs with T2T

The change log captures data which is added, deleted, or modified in the FSDM table. Data is initially moved from the staging table to the FSDM table through T2T. Once this is done, any modifications made such as adding new data, changing the data, or deleting the existing data are recorded in the change log table.

For example:

- Records are moved from the staging area into the FSDM table through T2T on day 1.
- A row is deleted from the FSDM table.
- Records are moved from the staging area into the FSDM table through T2T on day 2.
- A row is inserted into the FSDM table.

  Both the deleted and added rows are displayed in the change log table.

  The above example mentions two scenarios for displaying data in the change log table: a deleted value and an inserted value. A third scenario is if a value is changed in the column of a table. In this case, the old and new values are both captured in the change log.

  To generate the change log, you must provide CHGLOG_CAPTURE as the DT name in the Rule Name field and the name of the table that you want captured in the change log in the Parameter List field as mentioned in *Adding Tasks to a BD Batch*. Additionally, to delete a table, the DT name must be TRUNC_FSDM_TBL.

Components of the Change Log

The following table describes the functions of the different components in the change log:

Table 21. Change Log Components

Component	Description
Wrapper	The Wrapper contains logic that is required to record all changes (inserted, updated, and deleted data) in the log table.
Metadata table	The metadata table contains the metadata to support the wrapper.
FSDM table	The table in which the data is added, deleted, or modified.
AAI	AAI creates the DT which helps to execute the wrapper and run the T2T.
Change log table	The change log table captures the change log data after the wrapper is executed.

For the Metadata table component, the following values are available as metadata and can be captured in the change log:

Table 22. T2T Change Log Metadata Table Component Values

Value	Description
Table Name	This value is the name of the FSDM table in which a change has been made.
Table’s Column Name	This value is the name of the FSDM column in the table in which a change has been made.
User defined primary key	This value is the primary key of the column in which a change has been made.
Source Table Name	This value is the name of the source table in which a change has been made.
Source Table’s Column name	This value is the name of the column in the source table in which a change has been made.
Table Enable Flag	This value must be changed to N if you do not want to capture the changes made in a table in the change log. The default value is Y.
Table’s Column Enable Flag	This value must be changed to N if you do not want to capture the changes made in the column of a table in the change log. The default value is Y.

Table 22. T2T Change Log Metadata Table Component Values

Value	Description
Customer Notification Suppress Flag	This value indicates whether the customer is notified of the change through email or not.
Additional Columns	This value is used to mention any additional columns of the table whose changes must be captured in the change log.

For the Change log table component, the following values are available:

Table 23. T2T Change Log Table Component Values

Value	Description
Table Name	This value is the name of the FSDM table in which a change has been made.
Column Name	This value is the name of the FSDM column in the table in which a change has been made.
Source File Name	This value is the name of the source extract file from which the field with the changed value was loaded.
Source Field Name	This value is the name of the field with the changed value in the source extract file where the value is changed.
Format	This value is a textual representation of the colum or field format or data type.
Change Entry User Identifier	This value is the identifier of the person who entered the change. This is a unique identifier for the user who makes the change.
Change Entry User System Logon Identifier	This value is the user name of the user who makes the change.
Change Date	This value is the date on which the change was made.
Change Time	This value is the time at which the change was made.
Old Value	This value is the old value which was assigned to the specified table column.
New Value	This value is the new value which is assigned to the specified table column.
Key 1	This value is the textual representation of the value associated with the first column in the Primary Key or the user-defined primary key of the table containing the changed record.
Key 2	This value is the textual representation of the value associated with the second column in the Primary Key or the user-defined primary key of the table containing the changed record.
Key 3	This value is the textual representation of the value associated with the third column in the Primary Key or the user-defined primary key of the table containing the changed record.
Key 4	This value is the textual representation of the value associated with the fourth column in the Primary Key or the user-defined primary key of the table containing the changed record.
Change Type	This value is the code that indicates whether the change is an insert (add), a delete (removal), or an update.
Customer Notification Suppression Indicator	This value indicates whether the customer is notified of the change through email or not.
Source System	This value is the source system from which this data content is extracted.

Generating Change Logs with Hive

The change log captures any modifications made such as adding new data, changing the data, or deleting the existing data. To generate change logs in Hive, follow these steps:

1. Configure the CHG_LOG_REF table. For the Change log table component, the following values are available:

Table 24. H2T Change Log Component Values

Value	Description
Table Name	This value is the name of the FSDM table in which a change has been made.
Column Name	This value is the name of the FSDM column in the table in which a change has been made.
Source File Name	This value is the name of the source extract file from which the field with the changed value was loaded.
Source Field Name	This value is the name of the field with the changed value in the source extract file where the value is changed.
Format	This value is a textual representation of the colum or field format or data type.
Change Entry User Identifier	This value is the identifier of the person who entered the change.This is a unique identifier for the user who makes the change.
Change Entry User System Logon Identifier	This value is the user name of the user who makes the change.
Mantas Change Log Identifier	This value is the Oracle-specific identifier for this change log record that is unique across the FSDM.
Change Date	This value is the date on which the change was made.
Change Date - UTC	This value is the date in UTC on which this data change was made.
Change Time	This value is the time at which the change was made.
Change Time - UTC	This value is the time in UTC on which this data change was made.
Change Time Offset	This value is the Number of Hours Offset for Change Log Time.
Old Value	This value is the old value which was assigned to the specified table column.
New Value	This value is the new value which is assigned to the specified table column.
Key 1	This value is the textual representation of the value associated with the first column in the Primary Key or the user-defined primary key of the table containing the changed record.
Key 2	This value is the textual representation of the value associated with the second column in the Primary Key or the user-defined primary key of the table containing the changed record.
Key 3	This value is the textual representation of the value associated with the third column in the Primary Key or the user-defined primary key of the table containing the changed record.
Key 4	This value is the textual representation of the value associated with the fourth column in the Primary Key or the user-defined primary key of the table containing the changed record.
Change Type	This value is the code that indicates whether the change is an insert (add), a delete (removal), or an update.
Customer Notification Suppression Indicator	This value indicates whether the customer is notified of the change through email or not.

Table 24. H2T Change Log Component Values

Value	Description
Source System	This value is the source system from which this data content is extracted.
Processing Batch	This value is the Ingestion batch in which Oracle processed this data record.
Submission Date	This value is theBusiness date for which the data record is provided to Oracle.

2. Run Ingestion. This will set a baseline for the table you wish to capture in the change log.
3. After Ingestion is done, create a script to create a back up table for the table to be captured in the change log. For example, if you are generating a change log for the ACCT table, create the backup table ACCT_BKP.
4. The next day, run Ingestion after running H2T.
5. Use the following DT to capture the change log:
   `CHGLOG_CAPTUREHIVE`
6. Associate a task to the new DT. For more information about OFSAAI Data Transformation (DT), *Post Load Changes* in the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

The following tables provide the columns found in the CHG_LOG_REF table with an example of what will appear in each column.

Table 25. CHG_LOG_REF Table Example for Hive

Column Name	Example
CHG_TBL_NM	ACCT
CHG_COL_NM	PRMRY_CUST_INTRL_ID,ACCT_STAT_CD,INSTN_CNTRY_CD,STMT_SUPR_FL,NOTFY_LTR_SUPR_FL,RECALCITRANT_FL,US_POA_SIGN_FL,STNDG_INSTR_US_ACCT_FL
CHG_QUERY_KEY	ACCT_INTRL_ID
CHG_OUTPUT_KEY	ACCT_INTRL_ID
CHG_LOG_TBL_ENBL_FL	Y
SRC_TBL_NM	**Note:** This field must contain the back up table name (created in Step 3) for which the Change log will be created. ACCT_BKP
SRC_COL_NM	PRMRY_CUST_INTRL_ID,ACCT_STAT_CD,INSTN_CNTRY_CD,STMT_SUPR_FL,NOTFY_LTR_SUPR_FL,RECALCITRANT_FL,US_POA_SIGN_FL,STNDG_INSTR_US_ACCT_FL
SRC_TBL_QUERY_KEY	ACCT_INTRL_ID
SRC_OUTPUT_KEY	ACCT_INTRL_ID
CHG_LOG_DATASET	<CLOB>

Table 26. CHG_LOG_REF Table Example for T2T

Column Name	Example
CHG_TBL_NM	ACCT
CHG_COL_NM	PRMRY_CUST_INTRL_ID,ACCT_STAT_CD,INSTN_CNTRY_CD,STMT_SUPR_FL,NOTFY_LTR_SUPR_FL,RECALCITRANT_FL,US_POA_SIGN_FL,STNDG_INSTR_US_ACCT_FL
CHG_QUERY_KEY	ACCT_INTRL_ID
CHG_OUTPUT_KEY	ACCT_INTRL_ID
CHG_LOG_TBL_ENBL_FL	Y
SRC_TBL_NM	**Note:** This field must contain the back up table name (created in Step 3) for which the Change log will be created. ACCT_BKP
SRC_COL_NM	PRMRY_CUST_INTRL_ID,ACCT_STAT_CD,INSTN_CNTRY_CD,STMT_SUPR_FL,NOTFY_LTR_SUPR_FL,RECALCITRANT_FL,US_POA_SIGN_FL,STNDG_INSTR_US_ACCT_FL
SRC_TBL_QUERY_KEY	ACCT_INTRL_ID
SRC_OUTPUT_KEY	ACCT_INTRL_ID
CHG_LOG_DATASET	<CLOB>

As part of the product, the CHG_LOG_REF table is pre-populated with along with fields of interest. Clients can add additional tables and and their respective fields to the CHG_LOG_REF table in order to capture any modifications that occur in those tables in the Change Log.

Generating Change Logs with BD

Change log and Change log summary records with BD will be generated through BD.

When loading referential DIS files that are defined as Overwrite, it is possible for BD to generate Change Log records which signify when certain fields associated with a reference data entity have changed. This is done by comparing the contents of the DIS file with the current contents of the associated database table. For performance reasons, this change log processing can be done when external tables are used to load the DIS files, so it is a requirement that `DIS.Source=FILE-EXT`. This requires an external directory, which is created during installation. In order to give acccess to an oracle user, place the .dat files in the external directory. The change log records can also be derived with DIS.Source = 'FSDW' (CSA Ingestion).

Note: To derive the change log records the change log parameters in `<OFSAAI Installed Directory>/BDF/config/BDF.xml` should be uncommented.

Change log records can be generated in the following ways:

- 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 process only accounts for updates to already existing records. Change Log records are 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. If 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. Furthermore, 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 parameters in `<OFSAAI Installed Directory>/bdf/config/BDF.xml`. All of these parameters have been commented out, which means change log processing is turned off by default. To derive the change log records if DIS.Source = 'FILE-EXT', the relevant parameters for the DIS files of interest should be copied to `<OFSAAI Installed Directory>/bdf/config/custom/BDF.xml` and uncommented.

Table 27. Change Log Parameters

Parameter	Description
ChangeLog.<DIS File Type>.Fields	The fields of this particular DIS file type which will be monitored for changes.
ChangeLog.<DIS File Type>.IsSet	Whether change log records are generated based on mechanism 1 above (false) or mechanism 2 (true). The default is false.
ChangeLog.<DIS File Type>.QueryKey	This is only relevant when IsSet=true. 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 business key located in the given DIS file type's data map (for example `bdf/datamaps/AccountAddress.xml`) is used.
ChangeLog.<DIS File Type>.OutputKey	This is only relevant when IsSet=true. This defines the set of fields that are mapped to the Key1, Key2, Key3, and Key4 fields of a Change Log record. This can be different from the QueryKey and business key in order to match what is expected in Change Log DIS file records, and also to support the Change Log Summary data maps. If this parameter is not present, then the business key located in the given DIS file type's data map (for example, `bdf/datamaps/AccountAddress.xml`) is used.

To turn on Change Log processing for a given DIS file type, all the parameters for that file type must be uncommented. The values of the `ChangeLog.<DIS File Type>.Fields` parameter are preset based on the needs of the KYC application. If different fields are required, then this parameter should be changed. It is not necessary to change any of the other parameters.

For Example: If Address Street line fields are to be considered for change log generation, then the `ChangeLog.<DIS File Type>`. Fields parameter should be changed for that particular table as shown below.

```
<Parameter name="ChangeLog.AccountAddress.Fields" type ="STRING"
value="Country,Region,State,City,PostalCode,MailHandlingInstruction" list="true"/>
```

should be changed to

```
<Parameter name="ChangeLog.AccountAddress.Fields" type ="STRING"
value="Country,Region,State,City,PostalCode,MailHandlingInstruction,StreetLine1,StreetLine2
,StreetLine3,StreetLine4,StreetLine5,StreetLine6" list="true"/>
```

As in the example above, StreetLine1,StreetLine2,StreetLine3,StreetLine4,StreetLine5 and StreetLine6 will also be considered for change log generation. Similar steps can be followed for other change log related tables well.

Change Log records are written to the CHG_LOG table as the DIS file is being loaded. There are no additional scripts to be run. As soon as the parameters are uncommented, Change Log records are generated the next time DIS files are loaded.

Processing Data Using BD

This section covers the following topics:

- About BD Datamaps
- BD Derived Datamap Types
- Datamap Categories
- Processing Datamaps
- Configuring Risk Zones
- Customizing Review Reason Text
- DataMaps

About BD Datamaps

The BD datamap component is responsible for taking data from one or more source files or staging tables, transforming and enhancing it, and then loading it into a target database table.

The following types of datamaps are available:

- **DIS datamaps**: DIS datamaps are used to ingest client provided data, either through DIS files as specified in the DIS or through tables in the FSDF.
- **Derived datamaps**: Derived datamaps are used to transform the client provided data and populate other tables for use by scenarios and/or UI functionality.

BD datamaps can perform the following activities:

- Update summaries of trading, transaction, and instruction activity
- Assign transaction and entity risk through watch list processing
- Update various Balances and Positions derived attributes
- Update data related to trade finance attributes

For a complete list of the BD datamaps used in OFSAAI and a brief explanation of the each datamap, see *Appendix F, BD Datamap Details*

BD Derived Datamap Types

The Oracle solution implemented determines the required BD datamaps, or a subset thereof:

- AML Brokerage Datamaps
- AML Banking Datamaps
- Broker Compliance Datamaps
- Fraud Detection Datamaps
- Insurance Datamaps
- Trade Finance Datamaps
- Market Derived Datamaps

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

The following table describes the columns in the datamap tables that each section provides.

Table 28. Datamap Table Descriptions

Column	Description
Datamap Number	Unique, five-digit number that represents a particular datamap.
Datamap Name	Unique name of each datamap.
Predecessor	Indicator that processing of datamaps cannot begin until completion of predecessor datamaps.

Datamap Categories

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

- Optional
- Pre-Watch List
- Watch List
- Post-Watch List
- Summary
- Balances and Positions

Note: The Datamap categories may or may not be required for all solutions.

Processing Datamaps

This section provides the required datamaps for deriving and aggregating data based on the solution. Discussions of the datamaps appear in the order that processing must execute them during data loading, and include tables that describe each datamap. Datamap numbers that the accompanying tables provide also reflect this order.

Where predecessors exist, processing of datamaps cannot begin until completion of *predecessor* datamaps. These dependencies, or predecessors, may be internal to the datamap type, or external to the datamap type such as Summary datamaps dependent on watch list datamaps.

Note: If there is any performance issue with the running sequence of datamaps, it can be re-arranged. However. the predecessor for the datamap must be completed before running the datamap.

Example: The following is the order for the datamap to run:

```
FrontOfficeTransactionParty_InstnSeqID
FrontOfficeTransactionParty_HoldingInstnSeqID
```

If there is any performance issue with the datamap `FrontOfficeTransactionParty_HoldingInstnSeqID`, the datamap position can be rearranged in the batch script. Since there is the possibility that the previous process (`FrontOfficeTransactionParty_InstnSeqID`) is still running, the current datamap is waiting for the resources to be released.

Example for Internal Dependency

For example, processing can run the `FrontOfficeTransactionParty_InstnSeqID` datamap immediately after completion of `FinancialInstitution_FOTPSPopulation` and `AccountToClientBank_FOTPSInstitutionInsert`.

Example for External Dependency

Processing cannot run the `AccountProfile_Trade` datamap until and unless the `FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert` datamap is run.

AML Brokerage Datamaps

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

- AML Brokerage - Pre-Watch List Datamaps
- AML Brokerage - Watch List Datamaps
- AML Brokerage - Post-Watch List Datamaps
- AML Brokerage - Summary Datamaps
- AML Brokerage - Balances and Positions Datamaps

Each section provides a table that illustrates the datamaps and order of each datamap. This table describes the process by datamap number, datamap name, and internal or external predecessors, if any.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Trusted Pair

The Trusted Pair DIS file is different from typical DIS file. In a typical DIS file, 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:

- `execute.sh TrustedPair`
- `execute.sh TrustedPairMember`

Note: BD 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.

Configuring Risk Zones

Risk Zones are the threshold value by which an increase in a party's effective risk will trigger a review of the trusted pair is configurable. 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 default risk zones are configured as:

```
RiskZone1Lower=1
RiskZone1Upper=3
RiskZone2Lower=4
RiskZone2Upper=5
RiskZone3Lower=6
RiskZone3Upper=7
RiskZone4Lower=8
RiskZone4Upper=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.

Note: Ensure that the trusted pair file is run before the risk zones.

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 `TrustedPairReviewReasonText1` and `TrustedPairReviewReasonText2` parameters.

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

TrustedPairReviewReasonText1=Recommend Cancel - risk of <Party1> increased from <A> to <B>

TrustedPairReviewReasonText2= 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 `<OFSAAI Installed Directory>/bdf/config/BDF.xml` file. Risk review only happens if `managing_tp_from_ui` is set to Y in the `installMantas.properties.sample` properties file.

Note: Datamaps 10970,10980,10990, 11000,11010,11020 can be run in parallel.

DataMaps

This section displays the different BD datmap types and covers the following topics:

- AML Banking Datamaps
- Broker Compliance Datamaps
- Fraud Detection Datamaps
- Insurance Datamaps
- Trade Finance Datamaps
- Trusted Pair

AML Banking Datamaps

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

- AML Banking - Pre-Watch List Datamaps
- AML Banking - Watch List Datamaps
- AML Banking - Post-Watch List Datamaps
- AML Banking - Summary Datamaps

Broker Compliance Datamaps

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

- Broker Compliance - Pre-Watch List Datamaps
- Broker Compliance - Post-Watch List Datamaps
- Broker Compliance - Balances and Positions Datamaps
- Broker Compliance - Summary Datamaps

Fraud Detection Datamaps

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

- Fraud Detection - Pre-Watch List Datamaps
- Fraud Detection - Watch List Datamaps
- Fraud Detection - Post-Watch List Datamaps
- Fraud Detection - Summary Datamaps Detection

Insurance Datamaps

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

- Insurance - Pre-Watch List Datamaps
- Insurance - Watch List Datamaps
- Insurance - Post-Watch List Datamaps
- Insurance - Summary Datamaps

Trade Finance Datamaps

The following sections describe the datamaps that are required for deriving and aggregating data for the Trade Finance Solution:

- Trade Finance - Pre-Watch List Datamaps
- Trade Finance- Post-Watch List Datamaps

Market Derived Datamaps

The following are the datamaps that are required for deriving and aggregating market data:

- InsideQuote_Derived
- MarketCenterQuote_Derived
- ReportedMarketSale_Derived

  For information on the predecessors for the datamaps, see *Table 107*.

Processing Data Using FDT and MDT

The following sections describe how Ingestion Manager processes trade-related data, orders and executions, and trades through the Firm Data Transformer (FDT). This section covers the following topics:

- FDT Process Flow
- Populating Summary Information for Market data

FDT Process Flow

The following figure illustrates the FDT process flow:

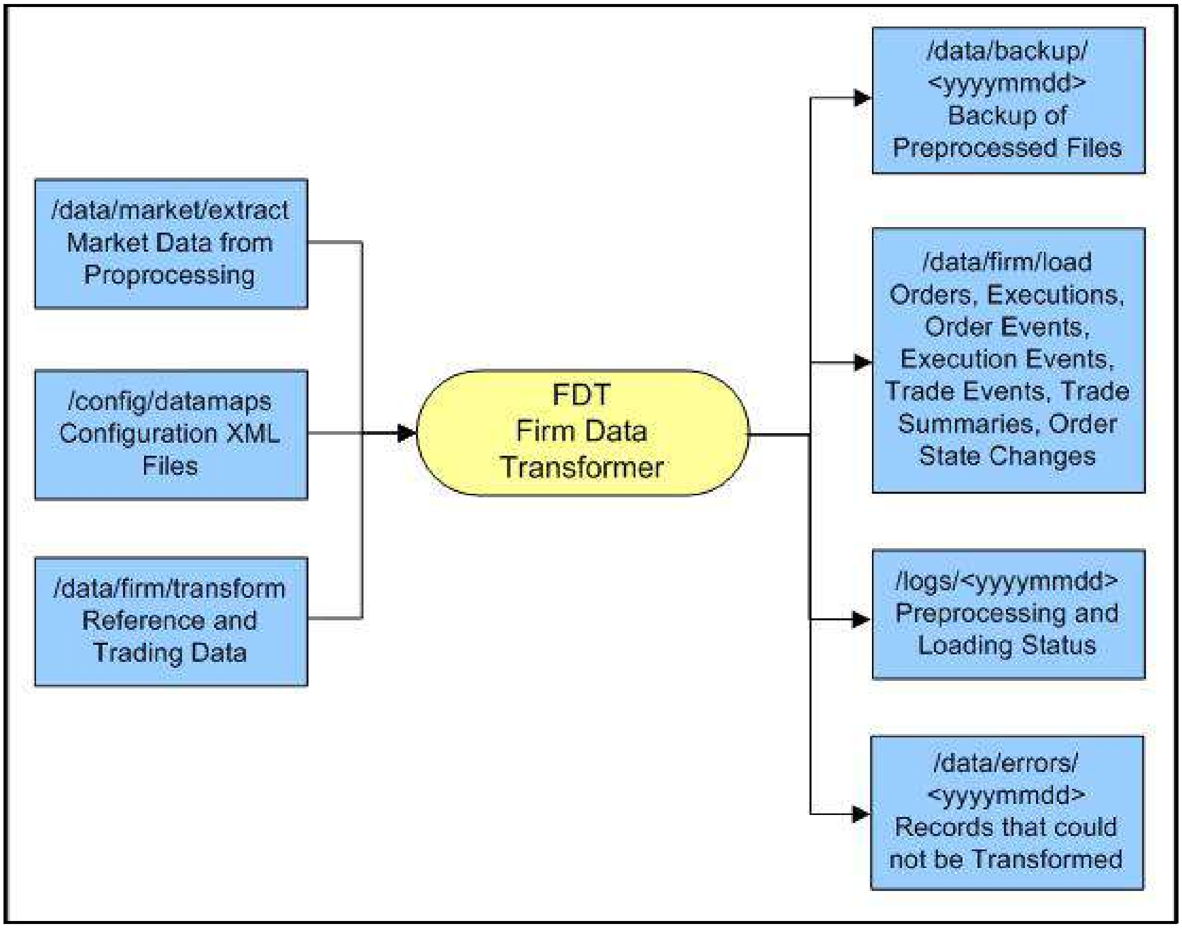

/data/market/extract Market Data from Proprocessing
/config/datamaps Configuration XML Files
/data/firm/transform Reference and Trading Data
FDT Firm Data Transformer
/data/backup/ <yyyymmdd> Backup of Preprocessed Files
/data/firm/load Orders, Executions, Order Events, Execution Events, Trade Events, 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 performs the following actions:

- 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.

Order and Trade Execution files are processed through the Firm Data Transformer (FDT). Before running runFDT.sh, Pre-processor has to be executed, using the following commands:

```
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDP.sh TradeExecution
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDP.sh Order
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDP.sh OpenOrder
```

During execution of the `runFDT.sh` script, the FDT performs the following actions:

- 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.

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

```
<OFSAAI Installed Directory>/ingestion_manager/scripts/runFDT.sh
```

When Ingestion Manager executes `runFDT.sh`, it places output files in the directories in *Table 29*.

Table 29. `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 Pre-processed input files.
`/data/firm/load`	Transformed output files for loading into the database.

After running runFDT, the system executes data loaders using the runDL.sh script; the following provides a sample command:

```
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh Order
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh OrderSummary
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh TradeExecution
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh Execution
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh Trade
<OFSAAI Installed Directory>/ingestion_manager/scripts/runDL.sh DerivedTrade
```

FDT processes are also available with BDF. To perform this action, you must execute the following datamaps in the order given:

1. OpenOrderStage
2. OrderStage
3. TradeExecutionEventStage
4. Scrty_TradeExecutionStageInsert
5. Scrty_OrderStageInsert
6. MktCntr_OrderStageInsert
7. OrderStage_DQupdate
8. TradeExecutionEventStage_DQupdate
9. OrderStage_FDTupdate
10. OrderStage_RmngQtupdate

11. OrderSummary
12. OrderSummary_OpenOrdrInsrt
13. OrderSummary_QtyUpdate
14. OrderStage_OpenOderUpd
15. OrderSummary_Update
16. OrderStage_OrdrSeqUpd
17. OrderEvent_OrderStage
18. Execution_NewEvents
19. Execution_CancelAndReplace
20. Execution_CancelEvents
21. Execution_CorrectionEvents
22. Trade_NewEvents
23. Trade_CancelAndReplace
24. Trade_CorrectionEvents
25. Trade_CancelEvents
26. Trade_DerivedTrade
27. Trade_OrigSeqIDUpd
28. Trade_ParentSeqIDUpd
29. Trade_RplcngSeqIDUpd
30. TradeExecutionEvent_Trade
31. TradeExecutionEvent_Execution
32. TradeExecutionEvent_CancelReplaceTrade
33. TradeExecutionEvent_FirmRefTrade
34. TradeExecutionEvent_MktRefTrade
35. Trade_RefData
36. Execution_Update

Populating Summary Information for Market data

As part of end of day processing, Market and Trade data summary information gets updated in the following path of the Java Utility:

```
<OFSAAI Installed Directory>/ingestion_manager/scripts/process_firm_summary.sh
```

Managing Data For BD Applications

This section explains different methods used to load and process data in various BD applications. Figure 18 shows the sequence for data loading:.

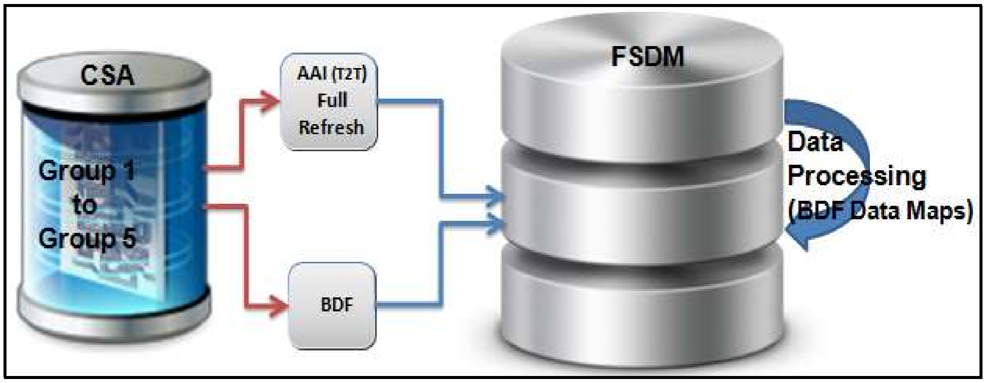


Figure 18. Data Loading For AML/Fraud/KYC/FATCA/CTR Applications

The following table provides the steps required to load data for Behavior Detection Applications. .

Table 30. Managing Application Data

Application	Steps	Group
AML Fraud KYC FATCA CTR	1. Execute Group 1 through Group 5 in sequence in the CSA using AAI T2T/H2T. For more information, see *Loading T2T using the AAI Framework*. For more information on the interface files available in Group 1 to Group 5, see *Behavior Detection Flat File Interface*. 2. Process the loaded data using BD datamaps in FSDM. For more information, see *Managing Data Processing*. 3. Interface files in the same group loaded through different loading method can be executed in parallel. 4. Run AML BD transformation. For more information on the AML datamaps, see *AML Brokerage Datamaps* and *AML Banking Datamaps*. 5. For network scenarios, refresh the temporary tables.	Group 1 Group 2 Group 3 Group 4 Group 5
TC BC PTA	1. Execute Group 1 and Group 2 in sequence. For more information on the interface files available in Group 1 and Group 2, see *Table 102* and *Table 103* respectively. 2. Run scripts for populating Market Data (MDT). For more information, see *Processing Data Using FDT and MDT*. 3. Execute Group 6 followed by the derived datamaps. For more information on the interface files available in Group 6, see *Table 107*. 4. Execute Group 3 through Group 5 of both runDP/runDL and BD sequentially. For more information on the interface files available in Group 3 and Group 5, see *Table 104* and *Table 106* respectively. 5. Run scripts for populating Firm Data (FDT). For more information, see *Processing Data Using FDT and MDT*. 6. These dependencies are as follows. Prior to executing the runDP.sh, TradeExecution, and runDL.sh scripts, run the following datamaps in BD Ingestion: ■ Security - Group 2 ■ MarketCenter - Group 1 ■ CorporateAction - Group 5 ■ StructuredDeal - Group 4 ■ SettlementInstruction - Group 5 7. Run BD scripts for populating Security Market Daily and Security Firm Daily. 8. Run Java Utilities (process_market_summary.sh, process_firm_summary.sh) for updating Security Market Daily and Security Firm Daily. 9. Process the loaded data using BC BD Transformation.	Group 1 Group 2 Group 6 Group 3 Group 4 Group 5

Table 30. Managing Application Data

Application	Steps	Group
Trade Finance	1. Execute Group 7using FDT/MDT. For more information, see *Table 108*. 2. Process the derived datamaps for trade finance. For more information, see *Trade Finance Datamaps*.	Group 7
ECTC	1. Execute Group 1 through Group 3 using runDP/runDL and BD. For more information on the interface files available, see *Behavior Detection Flat File Interface.* 2. Execute Group 4 using runDP/runDL. For more information on the interface files available in Group 4, see *Table 105.* **Note:** The usage of runDP/runDL or BD usage depends on the ingestion manager.	Group 1 Group 2 Group 3 Group 4

CHAPTER 4

Behavior Detection Jobs

This chapter provides an overview of the OFSBD Job Protocol and 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 OFSBD jobs. This chapter focuses on the following topics:

- About the OFSBD Job Protocol
- Performing Dispatcher Tasks
- Performing Job Tasks
- Clearing Out the System Logs
- Recovering Jobs from a System Crash
- Executing Batches Through the OFSAAI User Interface

Note: If you are using a job script that allows for multiple parameters, the values for the parameters must be separated by spaces ( ) and not commas (,).

About the OFSBD Job Protocol

The system initiates all OFSBD jobs by using a standard operational protocol that utilizes each job's metadata, which resides in a standard set of database tables. OFSBD 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).

The OFSBD administrator invokes the `dispatcher` and MANTAS processes by running the shell scriptsthat are mentioned in Table 31:

Table 31. OFSBD Job Protocol Shell Scripts

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

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

This section covers the following topics:

- Understanding the OFSBD Job Protocol
- Understanding the Dispatcher Process
- Understanding the MANTAS Process
- Applying a Dataset Override

Understanding the OFSBD Job Protocol

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

The following table provides an example of a job template group with two job templates.

Table 32. `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 must 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.

Note: If you observe job failures when running on the AIX operating system, it may be due to resource constraints of the AIX system. In this case, you must try reducing the number of jobs you are attempting to run in parallel or try running the jobs sequentially.

Refer to *Starting the Dispatcher* and *Stopping the Dispatcher* 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

The dataset override feature permits 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

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:

   ```
   <OFSAAI Installed Directory>/behavior_detection/algorithms/MTS/mantas_cfg/
   install.cfg
   ```

   The dataset override is specified with this property:

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

> **Note:** Specify the directory for the above given property 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 override automatically.

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

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

   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 such as `@Min_Indiv_Trxn_Am`.

Performing Dispatcher Tasks

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

This section covers the following topics:

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

- *Stopping the Dispatcher*
- *Monitoring the Dispatcher*

Setting Environment Variables

Environment variables are set up during the OFSBD 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

The following table describes environment variables in the `system.env` file. This file can be found at `<OFSAAI Installed Directory>/behavior_detection/algorithms/MTS/share`

Table 33. OFSBD Environment Variables in system.env File

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

The following table describes database environment variables in the `system.env` file.

Table 34. 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.

The following table shows operating system variables in the `system.env` file.

Table 35. Operating System Environment Variables in system.env File

Variable	Description
PATH	Augmented to include `<OFSAAI Installed Directory>/behavior_detection/algorithms/MTS/bin` and the `$ORACLE_HOME, $ORACLE_HOME/bin` pair (for Oracle).
LD_LIBRARY_PATH, LIBPATH, SHLIB_PATH (based on operating system)	Augmented to include `<OFSAAI Installed Directory>/behavior_detection/algorithms/MTS/lib` and `$ORACLE_HOME/lib` (for Oracle)

Starting the Dispatcher

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

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

To start the `dispatcher`, follow these steps:

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

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

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

   The `dispatcher` queries the database to check for any new jobs that must 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.

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.

Note: The dispatcher which started from the Behavior Detection jobs in the UI should be stopped before restarting servers.

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

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

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

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

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

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

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

   This script shuts down the dispatcher.

Monitoring the Dispatcher

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

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

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

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

   The log file scrolls down the screen.

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

Caution: The dispatch.log file can be a lengthy printout.

Performing Job Tasks

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

This section cover the following topics:

- Understanding the Job Status Codes
- Starting Behavior Detection 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 OFSBD administrator sets these codes through an OFSBD 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 Behavior Detection Jobs

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

To start a new job in OFSBD, follow these steps:

1. Create the new job and job description through an OFSBD Job Editor in the Scenario Manager.

   OFSBD 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 OFSBD Job Protocol* on page 62 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 OFSBD metadata tables. Each job consists of one or more runs.
4. The MANTAS process handles the job runs.

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

Starting Jobs Without the Dispatcher

Clients who use multiple services to run jobs for one OFSBD 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, as follows:

start_mantas.sh <template id> -nd

Doing so 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 in Behavior Detection Framework without the **`dispatcher`**, follow these steps:

1. Create the new job and job description through an OFSBD Job Editor.

   OFSBD 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 in OFSBD, 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 OFSBD 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 Behavior Detection Jobs* on page 67 for more information).

   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 such as `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 in OFSBD, execute the `restart_mantas.sh` script with the following parameters:

```
restart_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.

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 be necessary 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, follow these steps:

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 `<OFSAAI Installed Directory>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg` file.

Note: `$KDD_PRODUCT_HOME is the path of <OFSAAI Installed Directory>/behavior_detection/algorithms/MTS`

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 must resolve it. If you believe that the job errors were generated due to incorrect setups in OFSBD, 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. In order to find the underlying error, you must refer to the job log.

To monitor a specific job or to look at the job log history for diagnostic purposes, follow these 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 must clear out the dispatch and job log files. Otherwise, the files become so large that they are difficult to use as diagnostic tools and their size can impact the performance of the system.

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

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

This section covers the following topics:

- Clearing the Dispatch Log
- Clearing the Job Logs

Clearing the Dispatch Log

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

1. Shut down the `dispatcher` by following the procedure for Stopping the `dispatcher` (Refer to section *Stopping the Dispatcher* 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`.

   Refer to *Starting the Dispatcher* for more information.

Clearing the Job Logs

To clear the job logs, follow these steps:

1. Stop the `dispatcher.` (Refer to section *Stopping the Dispatcher* 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 `<OFSAAI Installed Directory>/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-MM-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.

Executing Batches Through the OFSAAI User Interface

System Administrator users can run Behavior Detection jobs and Post Processing jobs from the OFSAAI UI. Activities can be performed through a batch process that can be executed once a year or periodically such as Daily, Weekly, Monthly, Quarterly, and Half-yearly depending on a firm's requirement.

Note: For the batches to start, iccserver, router, AM and message server must be started in the same sequence as mentioned. For more information on starting servers, refer to the *Oracle Financial Services Advanced Analytical Applications Infrastructure (OFS AAAI) Applications Pack Installation and Configuration Guide*.

This section includes the following topics:

- Adding Behavior Detection Batches
- Adding Tasks to a BD Batch
- Setting Task Precedence
- Running a Single Task Using a Batch
- Scheduling a Batch Once
- Scheduling a Daily Batch
- Scheduling a Weekly Batch
- Configuring a Monthly Batch
- Monitoring a Batch After Execution
- Cancelling a Batch After Execution
- Re-starting a Batch
- Re-running a Batch

Note: Available cursors in database should be set to a minimum of 1000.
Before restarting the Webserver, dispatcher should be ended.

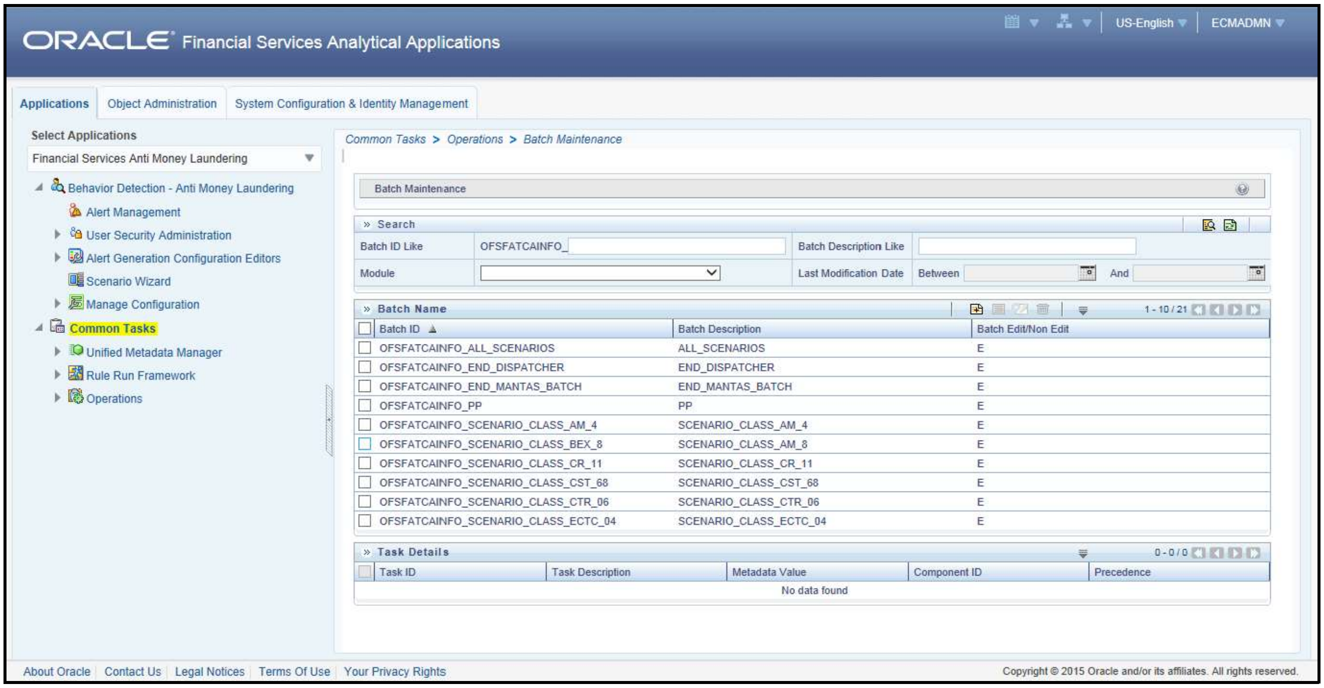
Adding Behavior Detection Batches

To add a batch, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.

   For more information, see *Alert Management user guide*.

2. In the Common Tasks section, click **Operations**.
3. Click **Batch Maintenance** in the RHS menu. The Batch Maintenance page is displayed.



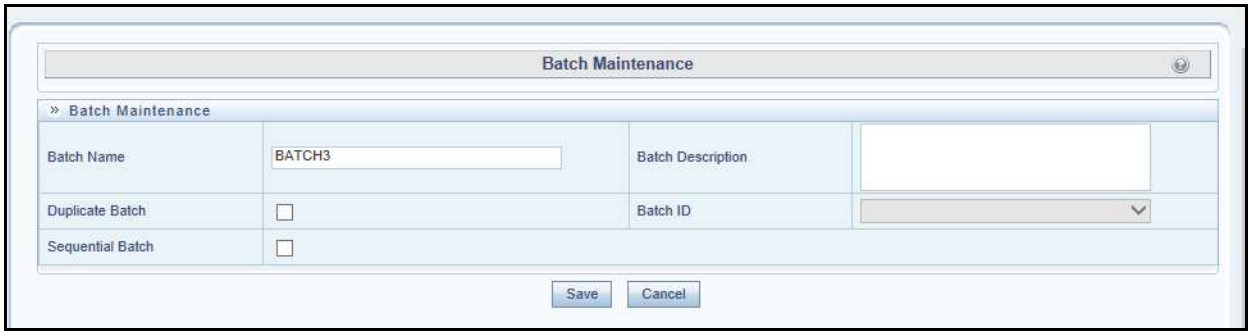
Batch ID Like: OFSFATCAINFO_

Batch ID	Batch Description	Batch Edit/Non Edit
OFSFATCAINFO_ALL_SCENARIOS	ALL_SCENARIOS	E
OFSFATCAINFO_END_DISPATCHER	END_DISPATCHER	E
OFSFATCAINFO_END_MANTAS_BATCH	END_MANTAS_BATCH	E
OFSFATCAINFO_PP	PP	E
OFSFATCAINFO_SCENARIO_CLASS_AM_4	SCENARIO_CLASS_AM_4	E
OFSFATCAINFO_SCENARIO_CLASS_BEX_8	SCENARIO_CLASS_AM_8	E
OFSFATCAINFO_SCENARIO_CLASS_CR_11	SCENARIO_CLASS_CR_11	E
OFSFATCAINFO_SCENARIO_CLASS_CST_68	SCENARIO_CLASS_CST_68	E
OFSFATCAINFO_SCENARIO_CLASS_CTR_06	SCENARIO_CLASS_CTR_06	E
OFSFATCAINFO_SCENARIO_CLASS_ECTC_04	SCENARIO_CLASS_ECTC_04	E

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

Figure 19. Batch Maintenance Page

4. In the Batch Name section, click.⊞. The Add Batch Definition page is displayed.



Batch Name	BATCH3	Batch Description	
Duplicate Batch	□	Batch ID	
Sequential Batch	□		

Figure 20. Add Batch Definition page

5. Enter the batch details as described in the following table:

Table 36. New Batch Details

Field	Description
Batch Name	Enter the name for the new batch.
Batch Description	Enter a description for this batch.
Duplicate Batch	Select this check box if the batch is a duplicate batch.
Sequential Batch	Select this check box if the batch must be run sequentially to another batch.
Batch ID	The Batch ID will be auto-populated.

6. Click **Save**. The added batch appears in the Batch Name section of the Batch Maintenance page.

Setting up Ingestion through AAI

Ingestion through AAI can be achieved by calling the customized shell scripts from the OFSAA Framework Batch Operations Module. The following scripts can be customized through OFSAAI:

- `set_mantas_date.sh`
- `start_mantas_batch.sh`
- `runDP.sh`
- `runDL.sh`
- `execute.sh`
- `runFDT.sh`
- `end_mantas_batch.sh`
- `process_firm_summary.sh`
- `process_market_summary.sh`

The custom shell script must be kept under `<FIC_HOME>/ficdb/bin` and associated to an OFSAAI Data Transformation (DT).

The following Custom shell scripts are present in `<FIC_HOME>ficdb/bin,` which can be used directly in OFSAAI Data Transformation (DT).

- `SetMantasDate.sh`
- `StartMantasBatch.sh`
- `AlertAssignment.sh`
- `PTC_Auto_Case_Assignment.sh`
- `EndMantasBatch.sh`

For more information about OFSAAI Data Transformation (DT), refer to *Post Load Changes* in the *Oracle Financial Services Analytical Applications Infrastructure User Guide.*

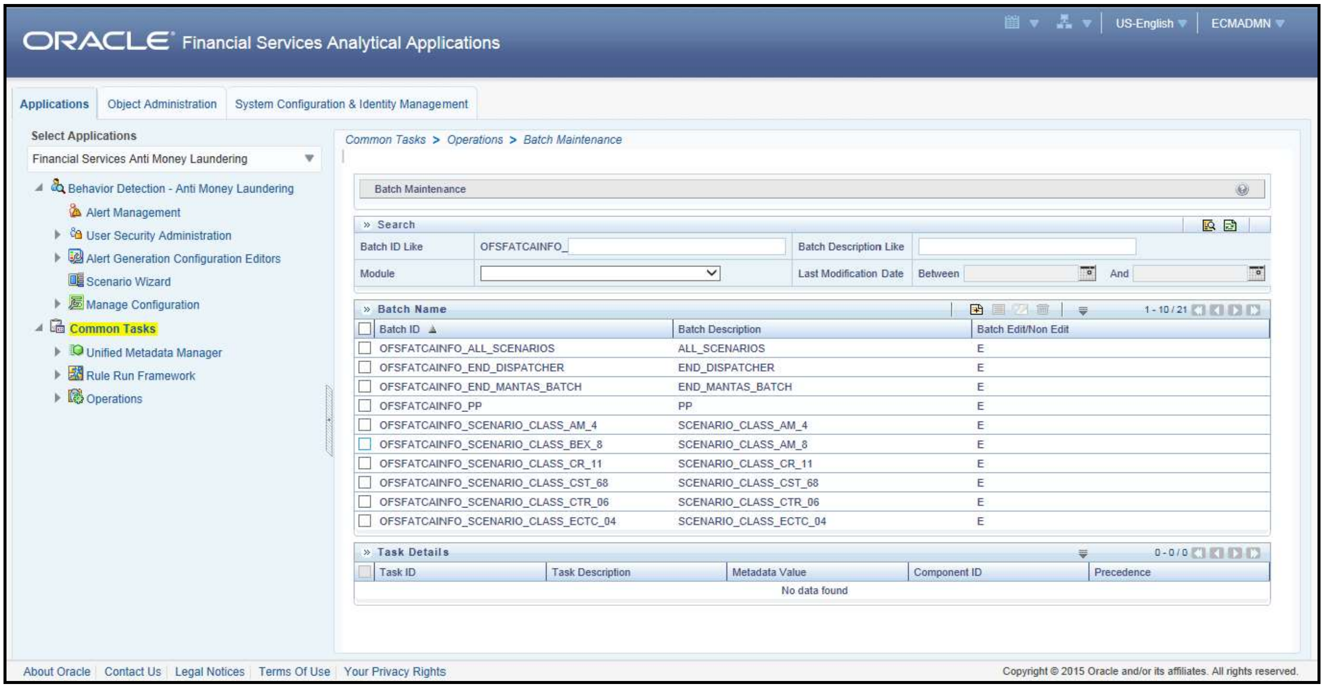
Similarly, you must create custom shell scripts for the following and associate them to an OFSAAI Data Transformation (DT).

- `runDP.sh`
- `runDL.sh`
- `execute.sh`
- `runFDT.sh`
- `process_firm_summary.sh`
- `process_market_summary.sh`

Adding Tasks to a BD Batch

To add tasks to an existing batch or newly created batch definition, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, select **Operations**.
3. Click **Batch Maintenance**. The Batch Maintenance page is displayed.


ORACLE Financial Services Analytical Applications
US-English
ECMADMN
Applications
Object Administration
System Configuration & Identity Management
Select Applications
Financial Services Anti Money Laundering
Behavior Detection - Anti Money Laundering
Alert Management
User Security Administration
Alert Generation Configuration Editors
Scenario Wizard
Manage Configuration
Common Tasks
Unified Metadata Manager
Rule Run Framework
Operations
Common Tasks > Operations > Batch Maintenance
Batch Maintenance
About Oracle | Contact Us | Legal Notices | Terms Of Use | Your Privacy Rights
Copyright © 2015 Oracle and/or its affiliates. All rights reserved.

» Search

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

» Batch Name 1 - 10 / 21

□	Batch ID ▲	Batch Description	Batch Edit/Non Edit
□	OFSFATCAINFO_ALL_SCENARIOS	ALL_SCENARIOS	E
□	OFSFATCAINFO_END_DISPATCHER	END_DISPATCHER	E
□	OFSFATCAINFO_END_MANTAS_BATCH	END_MANTAS_BATCH	E
□	OFSFATCAINFO_PP	PP	E
□	OFSFATCAINFO_SCENARIO_CLASS_AM_4	SCENARIO_CLASS_AM_4	E
□	OFSFATCAINFO_SCENARIO_CLASS_BEX_8	SCENARIO_CLASS_AM_8	E
□	OFSFATCAINFO_SCENARIO_CLASS_CR_11	SCENARIO_CLASS_CR_11	E
□	OFSFATCAINFO_SCENARIO_CLASS_CST_68	SCENARIO_CLASS_CST_68	E
□	OFSFATCAINFO_SCENARIO_CLASS_CTR_06	SCENARIO_CLASS_CTR_06	E
□	OFSFATCAINFO_SCENARIO_CLASS_ECTC_04	SCENARIO_CLASS_ECTC_04	E

» Task Details 0 - 0 / 0

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

Figure 21. Batch Maintenance Page

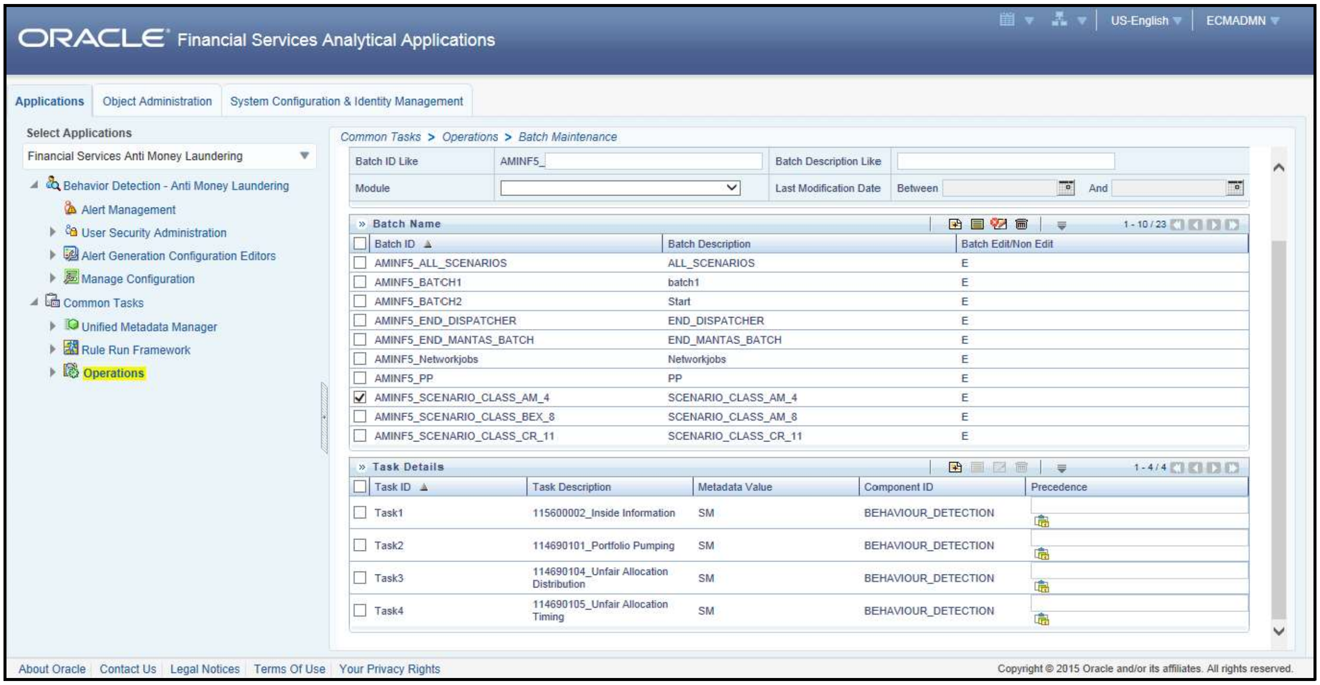
For further instructions on how to add a new batch or add tasks to an existing batch, see the *Batch Maintenance* section in the *Operation* chapter of the *Oracle Financial Services Advanced Analytical Applications Infrastructure(OFSAAAI) User Guide*.

Setting Task Precedence

After you have created a task, you must indicate which tasks must be executed prior to the newly created task in a batch.

To set task precedence, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, select **Operations**.
3. Click **Batch Maintenance**. The Batch Maintenance page is displayed.


ORACLE Financial Services Analytical Applications
US-English
ECMADMN
Applications
Object Administration
System Configuration & Identity Management
Select Applications
Financial Services Anti Money Laundering
Behavior Detection - Anti Money Laundering
Alert Management
User Security Administration
Alert Generation Configuration Editors
Manage Configuration
Common Tasks
Unified Metadata Manager
Rule Run Framework
Operations
Common Tasks > Operations > Batch Maintenance
About Oracle
Contact Us
Legal Notices
Terms Of Use
Your Privacy Rights
Copyright © 2015 Oracle and/or its affiliates. All rights reserved.

Batch ID Like: AMINF5_

Batch Description Like:

Module:

Last Modification Date: Between And

» Batch Name 1 - 10 / 23

□	Batch ID ▲	Batch Description	Batch Edit/Non Edit
□	AMINF5_ALL_SCENARIOS	ALL_SCENARIOS	E
□	AMINF5_BATCH1	batch1	E
□	AMINF5_BATCH2	Start	E
□	AMINF5_END_DISPATCHER	END_DISPATCHER	E
□	AMINF5_END_MANTAS_BATCH	END_MANTAS_BATCH	E
□	AMINF5_Networkjobs	Networkjobs	E
□	AMINF5_PP	PP	E
☑	AMINF5_SCENARIO_CLASS_AM_4	SCENARIO_CLASS_AM_4	E
□	AMINF5_SCENARIO_CLASS_BEX_8	SCENARIO_CLASS_AM_8	E
□	AMINF5_SCENARIO_CLASS_CR_11	SCENARIO_CLASS_CR_11	E

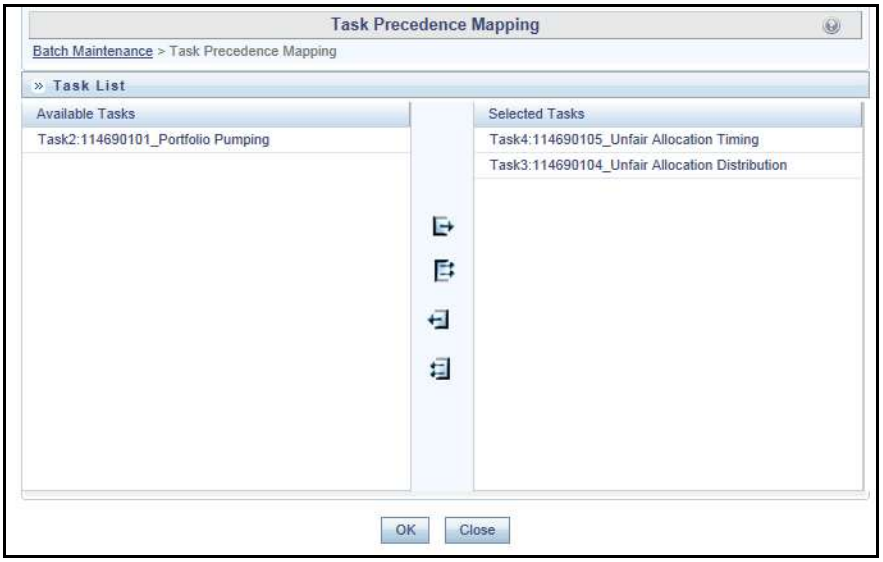
» Task Details 1 - 4 / 4

□	Task ID ▲	Task Description	Metadata Value	Component ID	Precedence
□	Task1	115600002_Inside Information	SM	BEHAVIOUR_DETECTION	
□	Task2	114690101_Portfolio Pumping	SM	BEHAVIOUR_DETECTION	
□	Task3	114690104_Unfair Allocation Distribution	SM	BEHAVIOUR_DETECTION	
□	Task4	114690105_Unfair Allocation Timing	SM	BEHAVIOUR_DETECTION	

Figure 22. Batch Maintenance page

4. In the Batch Name section, select the batch that you want to set task precedence for.

5. In the Task Details section, click ▣. The Task Precedence Mapping window is displayed.



Available Tasks	Selected Tasks
Task2:114690101_Portfolio Pumping	Task4:114690105_Unfair Allocation Timing
	Task3:114690104_Unfair Allocation Distribution

Figure 23. Task Precedence Mapping

6. Move the tasks which must be executed prior to this task from the Available Tasks pane to the Selected Tasks pane.

7. Click **OK** after you have selected all tasks which must precede the task. The selected tasks are listed in the Precedence column of the Task Details section.

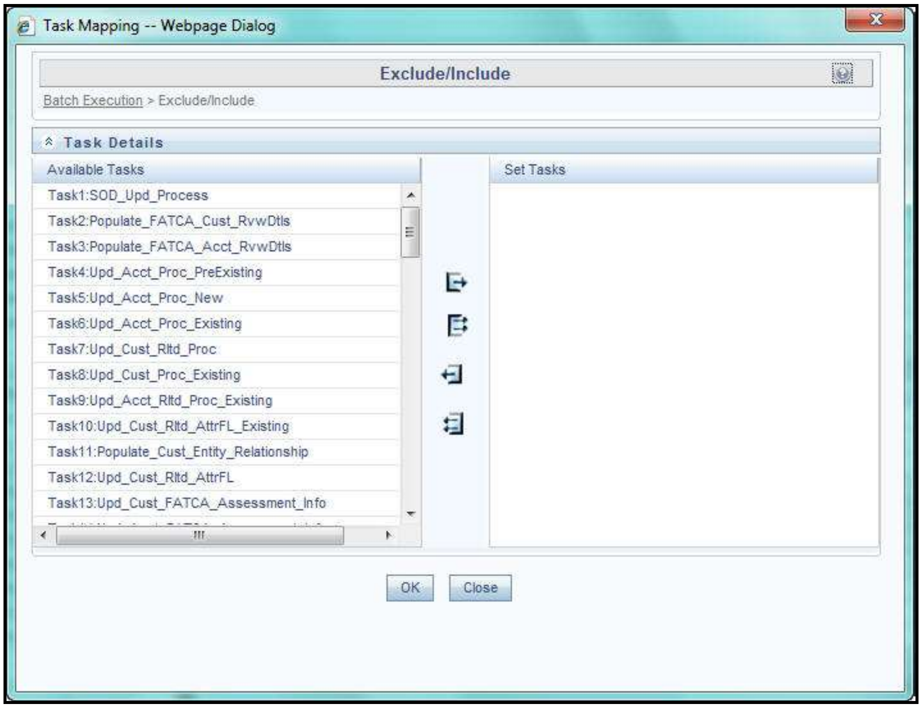
Running a Single Task Using a Batch

From the Batch Execution page, you can also run a single task from a batch.

Note: Running a single task using a batch is not a recommended approach and should be done only for debugging a particular task.

To run a single task using a batch, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Execution**. The Batch Execution page is displayed.
4. In the Batch Details section, select the particular batch that you want to execute.
5. In the Task Details section, click . The Task Mapping window is displayed.


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

- Task1:SOD_Upd_Process
- Task2:Populate_FATCA_Cust_RvwDtls
- Task3:Populate_FATCA_Acct_RvwDtls
- Task4:Upd_Acct_Proc_PreExisting
- Task5:Upd_Acct_Proc_New
- Task6:Upd_Acct_Proc_Existing
- Task7:Upd_Cust_Rltd_Proc
- Task8:Upd_Cust_Proc_Existing
- Task9:Upd_Acct_Rltd_Proc_Existing
- Task10:Upd_Cust_Rltd_AttrFL_Existing
- Task11:Populate_Cust_Entity_Relationship
- Task12:Upd_Cust_Rltd_AttrFL
- Task13:Upd_Cust_FATCA_Assessment_Info

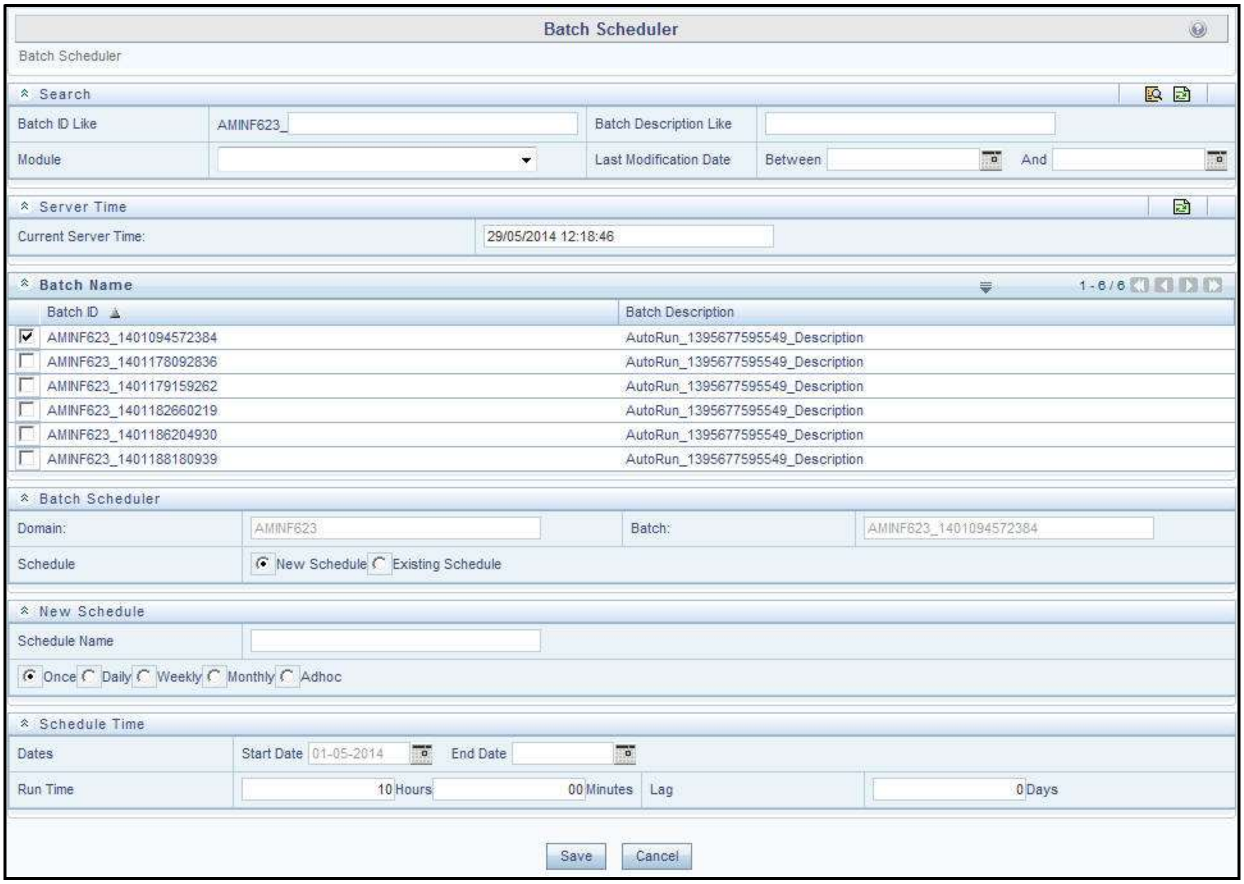
Figure 24. Running a Single Task Using a Batch

6. Retain the tasks that you want to execute under Available Tasks section and move the rest to the Set Tasks section.
7. Click **OK**. The following warning message is displayed: *If you exclude a task, it will be skipped when executing the batch but, the precedence will not be altered. Do you want to exclude the selected tasks)?*
8. Click **OK**.
9. Click **Execute Batch**.

Scheduling a Batch Once

To schedule a batch that you want to run only once, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the **Common Tasks** section, click **Operations**.
3. Click **Batch Scheduler**. The Batch Scheduler page is displayed.
4. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section is expanded and displays additional options.
5. Click **New Schedule**.
6. Set the frequency of the new schedule as **Once**.
7. Enter the schedule time of the batch by specifying the **Start Date** and the **Run Time**.



Batch Scheduler

Batch Scheduler

Search

Batch ID Like	AMINF623_	Batch Description Like			
Module		Last Modification Date	Between	And	

Server Time

Current Server Time: 29/05/2014 12:18:46

Batch Name — 1-6 / 6

	Batch ID	Batch Description
☑	AMINF623_1401094572384	AutoRun_1395677595549_Description
□	AMINF623_1401178092836	AutoRun_1395677595549_Description
□	AMINF623_1401179159262	AutoRun_1395677595549_Description
□	AMINF623_1401182660219	AutoRun_1395677595549_Description
□	AMINF623_1401186204930	AutoRun_1395677595549_Description
□	AMINF623_1401188180939	AutoRun_1395677595549_Description

Batch Scheduler

Domain:	AMINF623	Batch:	AMINF623_1401094572384
Schedule	● New Schedule ○ Existing Schedule		

New Schedule

Schedule Name

● Once ○ Daily ○ Weekly ○ Monthly ○ Adhoc

Schedule Time

Dates	Start Date 01-05-2014	End Date	
Run Time	10 Hours 00 Minutes	Lag	0 Days

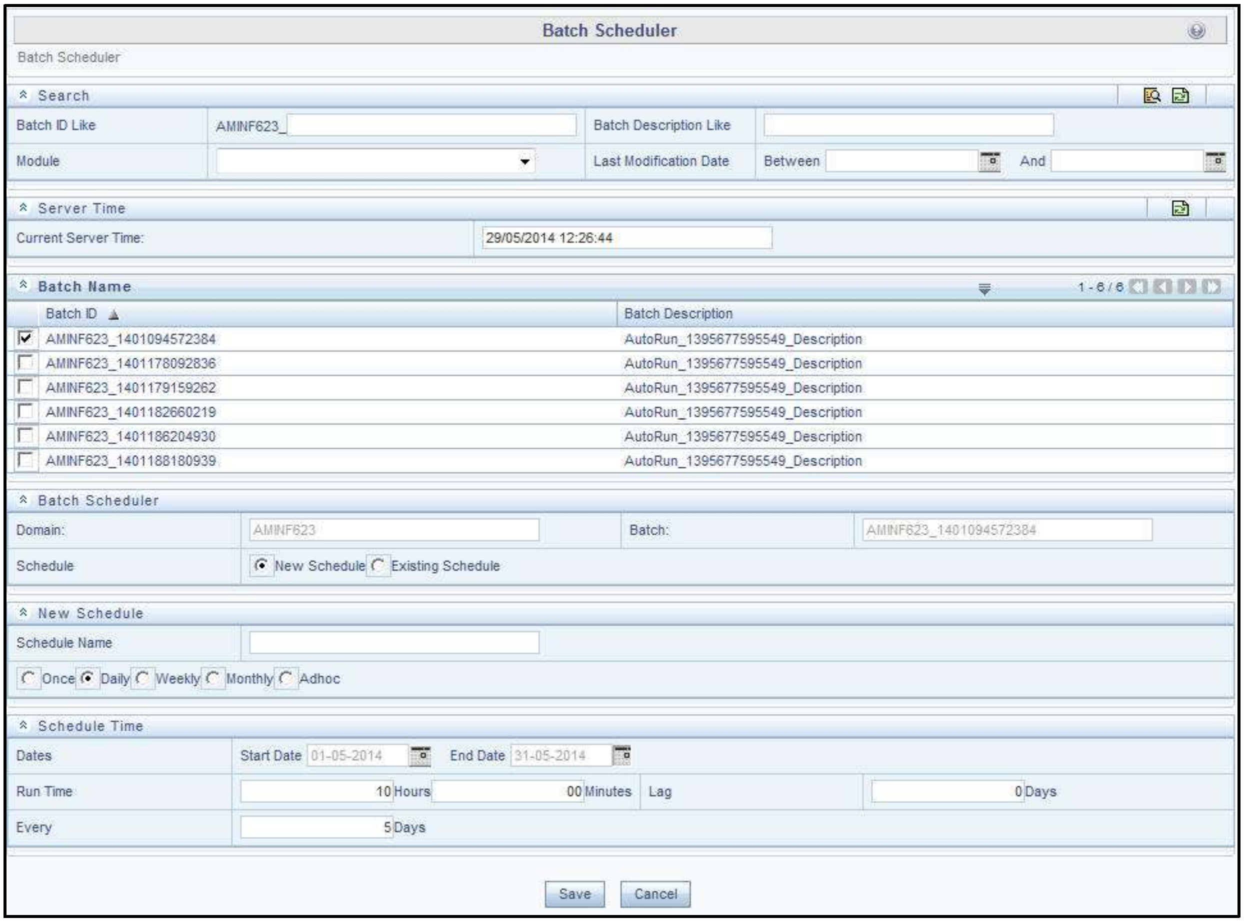
Figure 25. Scheduling a Batch Once

8. Click **Save**. The batch will run at the specified date and time.

Scheduling a Daily Batch

To schedule a batch that you want to run daily, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the **Common Tasks** section, click **Operations**.
3. Click **Batch Scheduler**. The Batch Scheduler page is displayed.
4. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section is expanded and displays additional options.
5. Click **New Schedule**.
6. Set the frequency of the new schedule as **Daily**.
7. Enter the schedule time of the batch by specifying the **Dates**, **Run Time**, and **Every** information.



Batch Scheduler

Batch Scheduler

Search

Batch ID Like: AMINF623_
Batch Description Like:
Module:
Last Modification Date: Between And

Server Time

Current Server Time: 29/05/2014 12:26:44

Batch Name 1 - 6 / 6

	Batch ID	Batch Description
☑	AMINF623_1401094572384	AutoRun_1395677595549_Description
□	AMINF623_1401178092836	AutoRun_1395677595549_Description
□	AMINF623_1401179159262	AutoRun_1395677595549_Description
□	AMINF623_1401182660219	AutoRun_1395677595549_Description
□	AMINF623_1401186204930	AutoRun_1395677595549_Description
□	AMINF623_1401188180939	AutoRun_1395677595549_Description

Batch Scheduler

Domain: AMINF623
Batch: AMINF623_1401094572384
Schedule: ● New Schedule ○ Existing Schedule

New Schedule

Schedule Name:
○ Once ● Daily ○ Weekly ○ Monthly ○ Adhoc

Schedule Time

Dates: Start Date 01-05-2014 End Date 31-05-2014
Run Time: 10 Hours 00 Minutes Lag 0 Days
Every: 5 Days

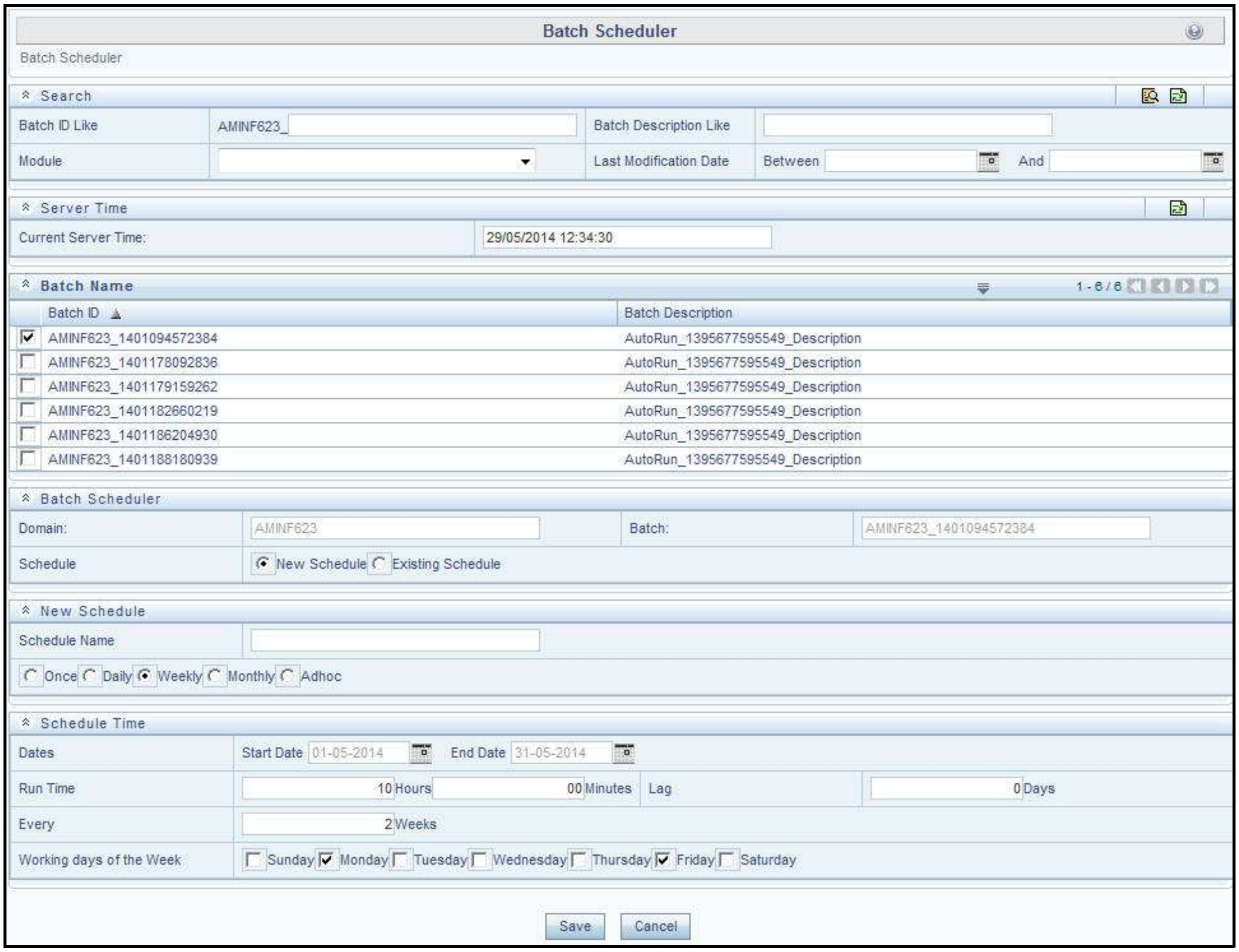
Figure 26. Scheduling a Daily Batch

8. Click **Save**. The batch will run at the specified date and time.

Scheduling a Weekly Batch

To schedule a batch that you want to run weekly, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Scheduler**. The Batch Scheduler page is displayed.
4. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section is expanded and displays additional options.
5. Click **New Schedule**.
6. Set the frequency of the new schedule as **Weekly**.
7. Enter the schedule time of the batch by specifying the **Dates**, **Run Time**, **Every, Working days of the Week** information.



Batch Scheduler

Batch Scheduler

Search

Batch ID Like	AMINF623_	Batch Description Like	
Module		Last Modification Date	Between And

Server Time

Current Server Time: 29/05/2014 12:34:30

Batch Name   1 - 6 / 6

	Batch ID	Batch Description
☑	AMINF623_1401094572384	AutoRun_1395677595549_Description
□	AMINF623_1401178092836	AutoRun_1395677595549_Description
□	AMINF623_1401179159262	AutoRun_1395677595549_Description
□	AMINF623_1401182660219	AutoRun_1395677595549_Description
□	AMINF623_1401186204930	AutoRun_1395677595549_Description
□	AMINF623_1401188180939	AutoRun_1395677595549_Description

Batch Scheduler

Domain:	AMINF623	Batch:	AMINF623_1401094572384
Schedule	● New Schedule ○ Existing Schedule		

New Schedule

Schedule Name

○ Once ○ Daily ● Weekly ○ Monthly ○ Adhoc

Schedule Time

Dates	Start Date 01-05-2014 End Date 31-05-2014	
Run Time	10 Hours 00 Minutes	Lag 0 Days
Every	2 Weeks	
Working days of the Week	□ Sunday ☑ Monday □ Tuesday □ Wednesday □ Thursday ☑ Friday □ Saturday	

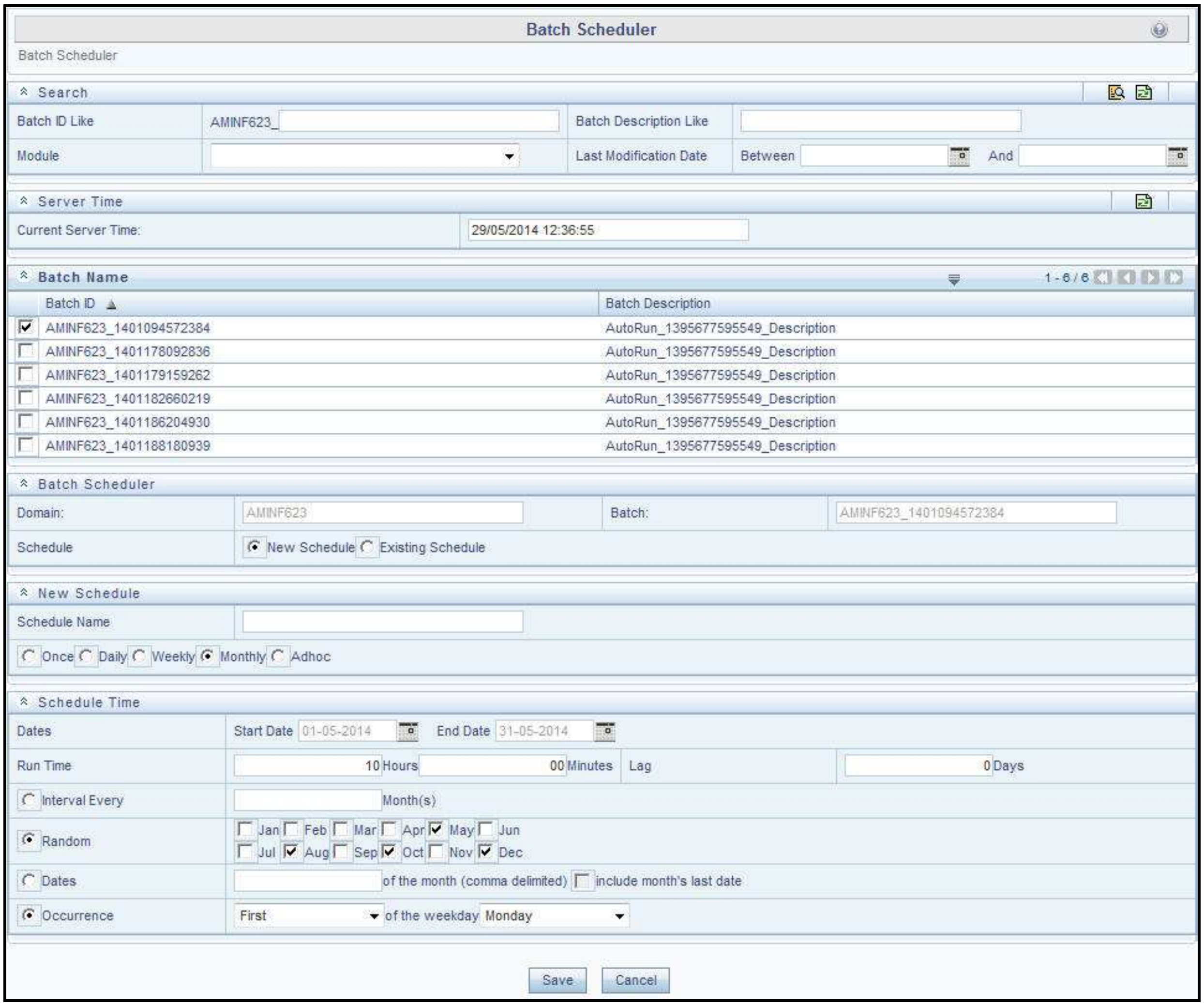
Figure 27. Scheduling a Weekly Batch

8. Click **Save**. The batch will run at the specified date and time.

Configuring a Monthly Batch

To schedule a batch that you want to run monthly, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Scheduler**. The Batch Scheduler page is displayed.
4. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section is expanded and displays additional options.
5. Click **New Schedule**.
6. Set the frequency of the new schedule as **Monthly**.
7. Enter the schedule time of the batch by specifying the **Dates**, and **Run Time** information.



Batch Scheduler

Batch Scheduler

Search

Batch ID Like	AMINF623_	Batch Description Like	
Module		Last Modification Date	Between And

Server Time

Current Server Time: 29/05/2014 12:36:55

Batch Name 1-6/6

	Batch ID	Batch Description
☑	AMINF623_1401094572384	AutoRun_1395677595549_Description
☐	AMINF623_1401178092836	AutoRun_1395677595549_Description
☐	AMINF623_1401179159262	AutoRun_1395677595549_Description
☐	AMINF623_1401182660219	AutoRun_1395677595549_Description
☐	AMINF623_1401186204930	AutoRun_1395677595549_Description
☐	AMINF623_1401188180939	AutoRun_1395677595549_Description

Batch Scheduler

Domain: AMINF623 Batch: AMINF623_1401094572384

Schedule ● New Schedule ○ Existing Schedule

New Schedule

Schedule Name

○ Once ○ Daily ○ Weekly ● Monthly ○ Adhoc

Schedule Time

Dates: Start Date 01-05-2014 End Date 31-05-2014

Run Time: 10 Hours 00 Minutes Lag 0 Days

○ Interval Every Month(s)

● Random ☐ Jan ☐ Feb ☐ Mar ☐ Apr ☑ May ☐ Jun ☐ Jul ☑ Aug ☐ Sep ☑ Oct ☐ Nov ☑ Dec

○ Dates of the month (comma delimited) ☐ include month's last date

● Occurrence First of the weekday Monday

Figure 28. Configuring a Monthly Batch

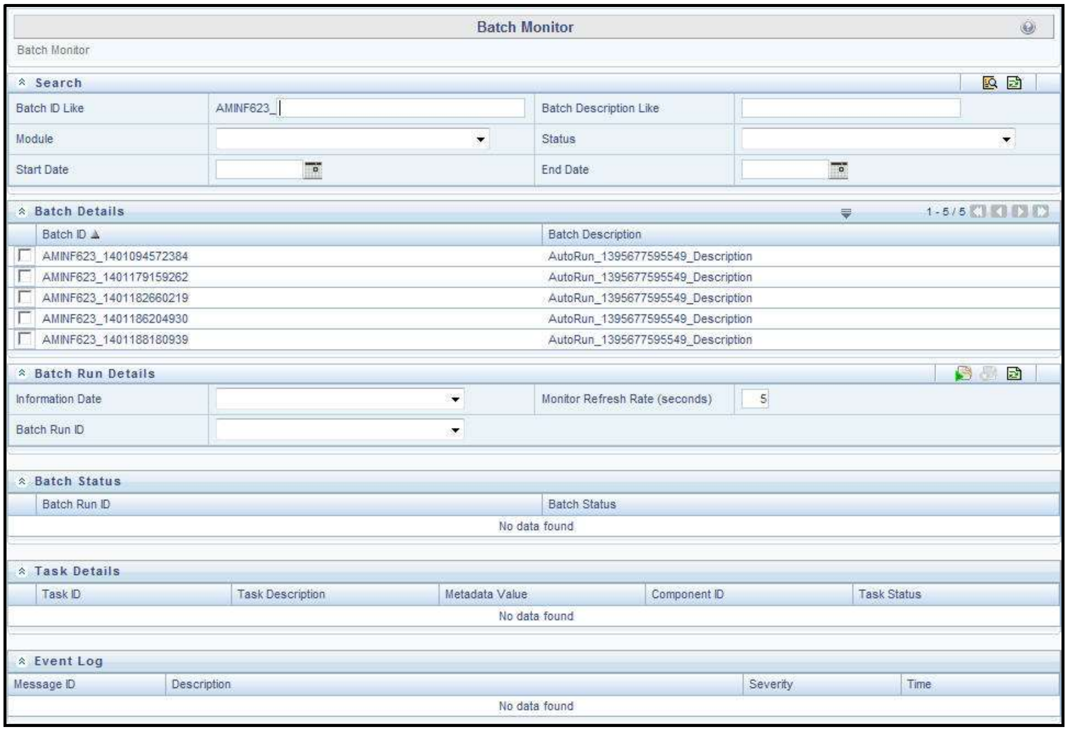
8. Click **Save**. The batch will run at the specified date and time.

Monitoring a Batch After Execution

Monitoring a batch helps you track the status of execution of an individual task that was included in the batch. Through monitoring, you can also track the batch status which in turn helps you in debugging.

To monitor a batch after it is executed, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Monitor**. The Batch Monitor page is displayed in the RHS.



Batch Monitor

Batch Monitor

Search

Batch ID Like	AMINF623_	Batch Description Like	
Module		Status	
Start Date		End Date	

Batch Details 1 - 5 / 5

	Batch ID	Batch Description
□	AMINF623_1401094572384	AutoRun_1395677595549_Description
□	AMINF623_1401179159262	AutoRun_1395677595549_Description
□	AMINF623_1401182660219	AutoRun_1395677595549_Description
□	AMINF623_1401186204930	AutoRun_1395677595549_Description
□	AMINF623_1401188180939	AutoRun_1395677595549_Description

Batch Run Details

Information Date		Monitor Refresh Rate (seconds)	5
Batch Run ID			

Batch Status

Batch Run ID	Batch Status
No data found	

Task Details

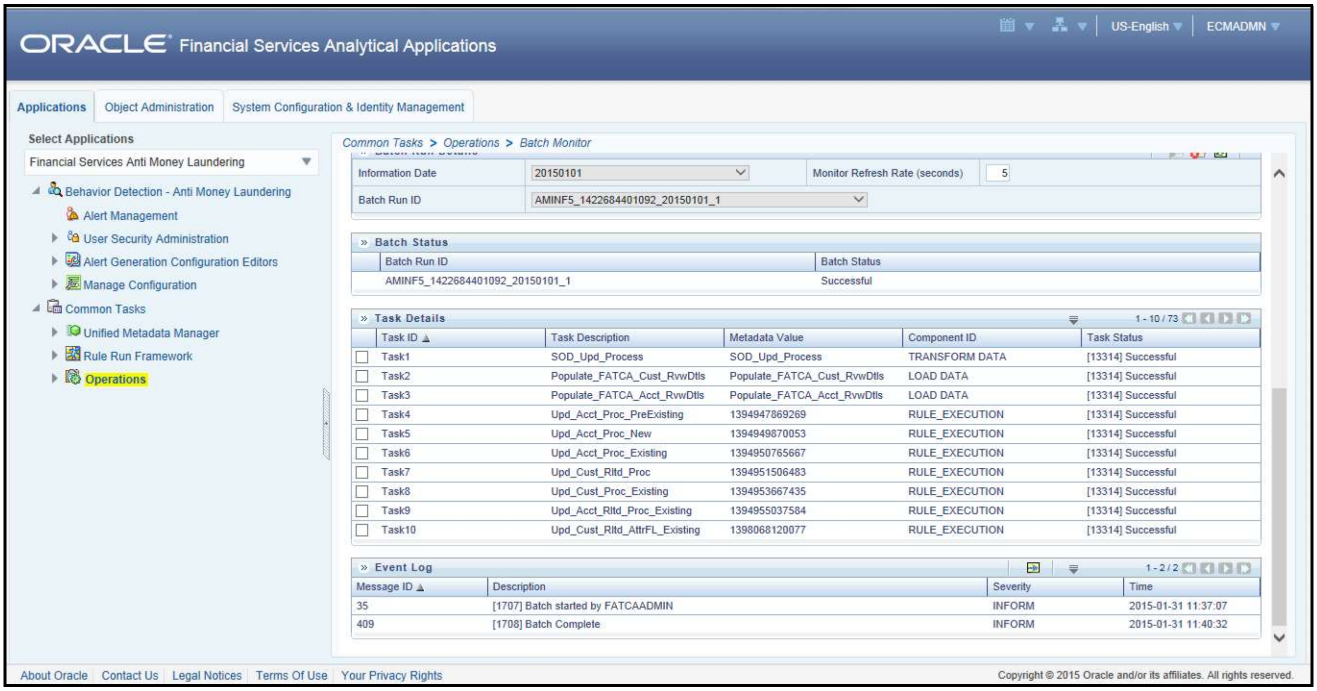
Task ID	Task Description	Metadata Value	Component ID	Task Status
No data found				

Event Log

Message ID	Description	Severity	Time
No data found			

Figure 29. Batch Monitor Page

4. Select a batch from the Batch Details lists that you want to monitor.
5. From Batch Run Details section, select an Information Date and the Batch Run ID from the drop-down list.
6. Click to start the monitoring. The Batch Status, Task Details, and Event Log sections are populated with information about this batch's execution.


ORACLE Financial Services Analytical Applications
US-English
ECMADMN
Applications
Object Administration
System Configuration & Identity Management
Select Applications
Financial Services Anti Money Laundering
Behavior Detection - Anti Money Laundering
Alert Management
User Security Administration
Alert Generation Configuration Editors
Manage Configuration
Common Tasks
Unified Metadata Manager
Rule Run Framework
Operations
About Oracle
Contact Us
Legal Notices
Terms Of Use
Your Privacy Rights
Copyright © 2015 Oracle and/or its affiliates. All rights reserved.

Common Tasks > Operations > Batch Monitor

Information Date	20150101	Monitor Refresh Rate (seconds)	5
Batch Run ID	AMINF5_1422684401092_20150101_1		

» Batch Status

Batch Run ID	Batch Status
AMINF5_1422684401092_20150101_1	Successful

» Task Details 1 - 10 / 73

	Task ID ▲	Task Description	Metadata Value	Component ID	Task Status
☐	Task1	SOD_Upd_Process	SOD_Upd_Process	TRANSFORM DATA	[13314] Successful
☐	Task2	Populate_FATCA_Cust_RvwDtls	Populate_FATCA_Cust_RvwDtls	LOAD DATA	[13314] Successful
☐	Task3	Populate_FATCA_Acct_RvwDtls	Populate_FATCA_Acct_RvwDtls	LOAD DATA	[13314] Successful
☐	Task4	Upd_Acct_Proc_PreExisting	1394947869269	RULE_EXECUTION	[13314] Successful
☐	Task5	Upd_Acct_Proc_New	1394949870053	RULE_EXECUTION	[13314] Successful
☐	Task6	Upd_Acct_Proc_Existing	1394950765667	RULE_EXECUTION	[13314] Successful
☐	Task7	Upd_Cust_Rltd_Proc	1394951506483	RULE_EXECUTION	[13314] Successful
☐	Task8	Upd_Cust_Proc_Existing	1394953667435	RULE_EXECUTION	[13314] Successful
☐	Task9	Upd_Acct_Rltd_Proc_Existing	1394955037584	RULE_EXECUTION	[13314] Successful
☐	Task10	Upd_Cust_Rltd_AttrFL_Existing	1398068120077	RULE_EXECUTION	[13314] Successful

» Event Log 1 - 2 / 2

Message ID ▲	Description	Severity	Time
35	[1707] Batch started by FATCAADMIN	INFORM	2015-01-31 11:37:07
409	[1708] Batch Complete	INFORM	2015-01-31 11:40:32

Figure 30. Batch Execution Details

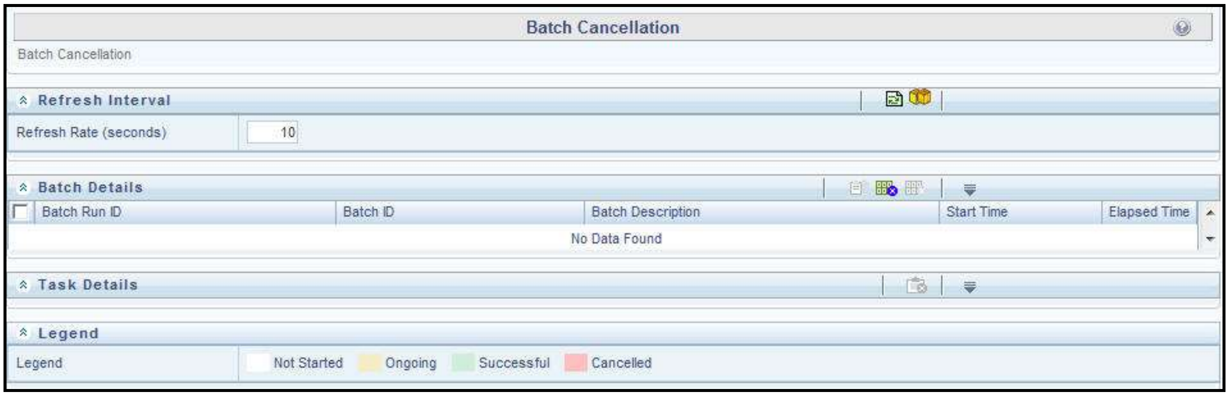
Cancelling a Batch After Execution

Cancellation of a batch cancels a current batch execution.

Note: This is not recommended and should be done only when the batch was fired accidentally or when a particular is taking too long to execute.

To cancel a batch after it is executed, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Cancellation**. The Batch Cancellation page is displayed in RHS.



Batch Cancellation

Batch Cancellation

Refresh Interval

Refresh Rate (seconds): 10

Batch Details

□	Batch Run ID	Batch ID	Batch Description	Start Time	Elapsed Time
			No Data Found		

Task Details

Legend

Legend: ⬜ Not Started 🟨 Ongoing 🟩 Successful 🟥 Cancelled

Figure 31. Batch Cancellation Page

4. Under the Batch Details section, select the batch whose execution you want to cancel.
5. Click **Cancel Batch**.

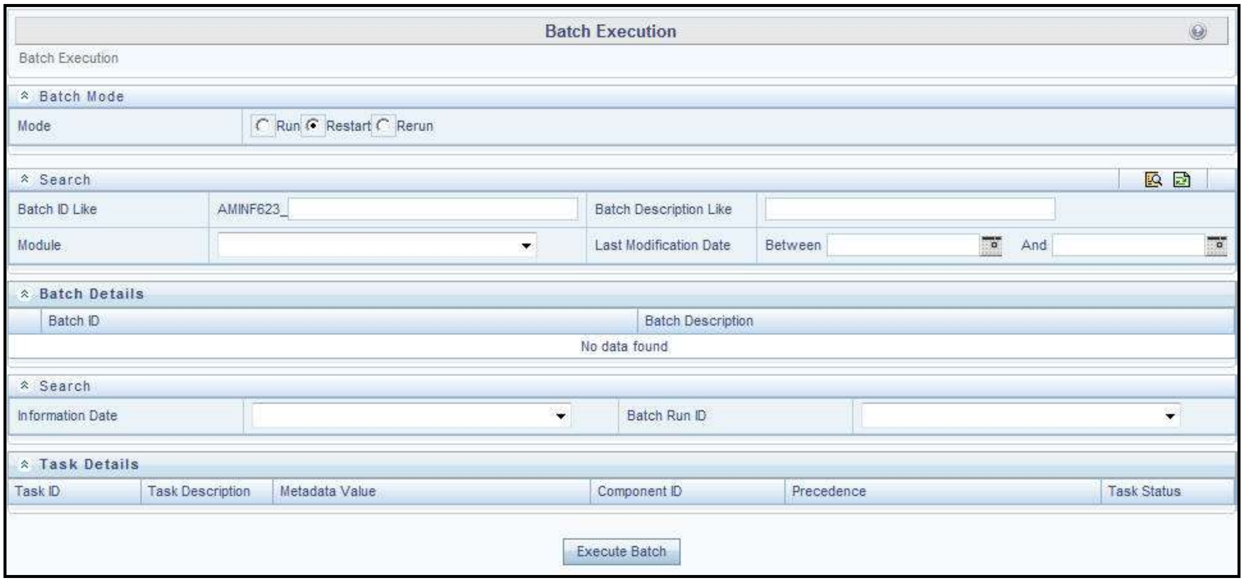
Re-starting a Batch

You can restart a batch execution when they have fail in their execution. When you restart a batch, it starts from the task at which it had failed. This happens when the failed task issue is debugged and resolved.

Note: It is recommended that you debug and resolve a failed task before restarting the batch execution.

To restart a batch execution, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Execution**. The Batch Execution page is displayed.
4. Select **Restart** radio button from the Batch Mode section.



Batch Execution

Batch Execution

Batch Mode

Mode ○ Run ● Restart ○ Rerun

Search

Batch ID Like	AMINF623_	Batch Description Like	
Module		Last Modification Date	Between And

Batch Details

Batch ID	Batch Description
No data found	

Search

Information Date		Batch Run ID	

Task Details

Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status

Figure 32. Re-starting a Batch

5. Select the batch from the Batch Details section that you want to restart.
6. Select the Information Date and Batch Run ID for the selected batch from the drop-down list.
7. Click **Execute Batch**.

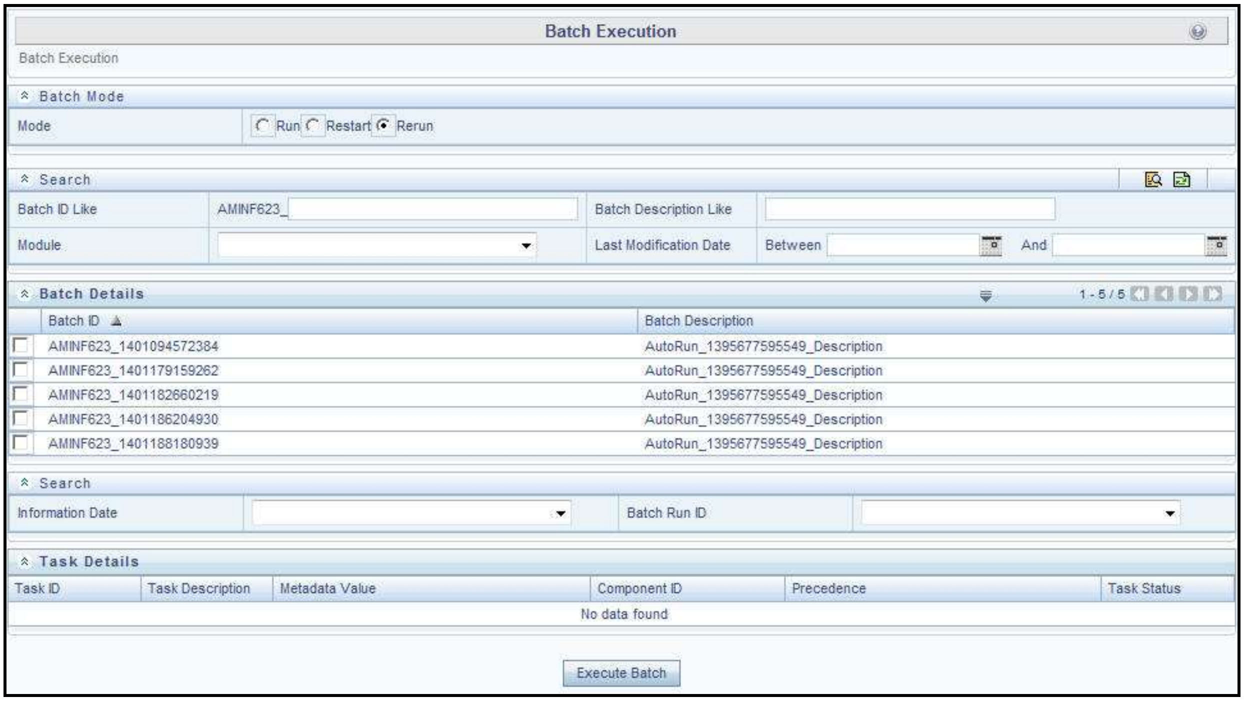
Re-running a Batch

You can rerun a batch execution when you want all the tasks from a successful batch execution to be executed again from the beginning. When a successfully executed batch is rerun, a different Batch Run ID is created for each instance for the same Information Date.

Note: Creation of different Batch Run ID for each rerun of a batch is optional depending upon a firm's requirement.

To rerun a batch, follow these steps:

1. Navigate to the Oracle Financial Services Analytical Applications Infrastructure page.
2. In the Common Tasks section, click **Operations**.
3. Click **Batch Execution**. The Batch Execution page is displayed.
4. Select **Rerun** from the Batch Mode section.



Batch Execution

Batch Execution

Batch Mode

Mode: ○ Run ○ Restart ● Rerun

Search

Batch ID Like	AMINF623_	Batch Description Like	
Module		Last Modification Date	Between And

Batch Details 1 - 5 / 5

	Batch ID	Batch Description
□	AMINF623_1401094572384	AutoRun_1395677595549_Description
□	AMINF623_1401179159262	AutoRun_1395677595549_Description
□	AMINF623_1401182660219	AutoRun_1395677595549_Description
□	AMINF623_1401186204930	AutoRun_1395677595549_Description
□	AMINF623_1401188180939	AutoRun_1395677595549_Description

Search

Information Date		Batch Run ID	

Task Details

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

Figure 33. Re-running a Batch

5. Select the batch from the Batch Details section that you want to rerun.
6. Select the Information Date and Batch Run ID for the selected batch from the drop-down list.
7. Click **Execute Batch**.

CHAPTER 5

Post-Processing Tasks

This chapter defines the following post-processing administrative tasks:

- About Post-Processing
- Augmentation
- Match Scoring
- Alert Creation
- Update Alert Financial Data
- Alert Scoring
- Alert Assignment
- Case Assignment
- Auto-Close
- Automatic Alert Suppression
- Highlight Generation
- Augment Trade Blotter
- Score Trade Blotter
- Historical Data Copy
- Alert Correlation

- Personal Trading 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. This process 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* 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 *Alert Creation* for more information).
- **Update Alert Financial Data:** Records additional data for alerts such as the related Investment Advisor or Security involved in the alert.(Refer to *Update Alert Financial Data* for more information).
- **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* for more information).

- **Alert Assignment:** Determines the user or group of users responsible for handling each alert (Refer to *Alert Assignment* for more information).
- **Case Assignment**: Determines the user or group of users responsible for handling each case. (Refer to *Case Assignment* for more information). This is only applicable if your firm has implemented Enterprise Case Management.
- **Auto-Close (optional):** Closes alerts that are of a lower priority to the business (Refer to *Auto-Close* 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* 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* 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* 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* 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* 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 *Alert Correlation* 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. Multi Match Alert Creation (502)
3. Single Match Alert Creation (503)
4. Update Alert Financial Data
5. Alert Scoring (504)
6. Alert Assignment
7. Auto-Close (506)
8. Automatic Alert Suppression (507)
9. Highlight Generation
10. Augment Trade Blotter
11. Score Trade Blotter
12. Historical Data Copy
13. Alert Correlation (508)
14. Case Assignment

Note: For all the post processing jobs MANTAS batch should be up and running.

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 *Administration Tools User Guide* 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, 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 group, and possibly share other common attributes. Each focus type has a separate job template.
- Single-match Alert Creator: Generates one alert per match.

Note: The `KDD_JRSDCN` table is empty after system initialization and requires populating before the system can operate. If a new jurisdiction is to be added, it should be added to `KDD_JRSDCN` table.

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 *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 group 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

OFSBD 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 `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/logs` directory. The name of the file is `run_stored_procedure.log`.

Alert Scoring

OFSBD 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 *Administration Tools User Guide* 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 OFSBD provides to run the Alert Scoring algorithm.

Alert Assignment

OFSBD provides a mechanism to assign alerts to a predefined owner (either an individual user or a pool of users). When performing alert assignment, the module fetches new, unowned alerts 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 *Administration Tools User Guide*, for more information.

The assignment framework allows customers to write their own Java code to replace the product functionality with their own customized functionality. The modules that can be replaced include the assignment-eligible objects (currently Alerts and Cases), the assignment rule processing logic, and the manner in which the assignment results are output (currently results are written out to the database for batch assignment, or passed back in a SOAP XML response for the assignment web services call). For more information on how to take advantage of this feature, please contact Oracle Support.

Running the Alert Assignment Job

The Alert Assignment Job is part of the OFSBD subsystem.

To run an Alert Assignment job, follow these steps:

1. Run the execute.sh script as follows:

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

By default, Behavior Detection writes log messages for this script in the `<OFSAAI Installed Directory>/bdf/logs/<Processing Date>/AlertAssignment.log` file.

Auto-Close

OFSBD 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 such as from the Relationship tab in the OFSBD UI) and processing , such as 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 such as alert score > 50, age > 30. A set of operations based on the same attribute, such as 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. Enter `NULL` if the 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`. While using the `IN` operator, the right-hand side variables should be separated by| such as `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 OFSBD 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. This 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 OFSBD 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.

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.

   The following table describes commonly used Alert Closing Attributes.

Table 37. 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 Group	113000064
Scenario Group	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 such as >, <, 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. An error occurs if two operations have the same precedence. All operations must have 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. Inserting an ID into the NEXT_OPRTN_SET column previously called creates a loop and results in an error.
- DMN_CD is the OFSBD product code.
- The CLS_ACTIVITY_TYPE_CD column specifies the activity type code of the closing action. The activity type code that this column 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.
- The CMMNT_TX column specifies an optional text comment.

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.

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

Table 38. `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 39 shows how to set it to the next operation's ID within the operation set.

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

Table 39. 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. OFSBD 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 *Administration Tools User Guide* for more information.

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 OFSBD 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, such as through the Relationship tab, and processing, such as 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 Alert Management User Guide* for detailed information about initiating and canceling Alert Suppression.

Running the Suppression Job

The suppression job is part of the Behavior Detection subsystem. OFSBD 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 *Administration Tools User Guide* for more information.

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 OFSBD 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 OFSBD UI. The system calculates and stores these highlights in the database as part of the batch cycle using the following shell script:

```
run_highlights.ksh
```

This script is part of the Database Tools that resides in the `<OFSAAI Installed Directory>/database/db_tools/bin` directory. This script attaches to the database using the user that the `utils.database.username` property identifies in the `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/logs/highlights.log` file.

Augment Trade Blotter

OFSBD provides the ability to differentiate between various types of trades , such as Client age 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 `<OFSAAI Installed Directory>/database/db_tools/logs` directory.

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

Note: This utility can be run anytime after data management 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 `<OFSAAI Installed Directory>/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 , such as 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 `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file. This property should identify the *Atomic Schema user*, a user in the database with write access to tables in Behavior detection Atomic schema.

To improve performance, you can adjust two configurable parameters in the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file.

Table 40. 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 `<OFSAAI Installed Directory>/database/db_tools/logs/hdc.log` file.

Alert Correlation

OFSBD 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. OFSBD 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 Behavior Detection Framework Services Guide*).

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 OFSBD 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 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 Enterprise 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.

Note: To assist with performance tuning, the *filter_by_batch* job parameter can optionally be added to the `KDD_PARAM_BINDING` table. A value of *true* causes a filter to be appended to all queries retrieving alerts/correlations/cases by the Alert Correlation algorithm based on the current batch name. A value of *false* (default behavior) does not include this filter. For example, if your organization varies batches by country, and only needs to pull in data for a specific country in each batch, turning this filter on prevents them from pulling in unnecessary data (that is, from other countries) in each batch. This parameter is added as a job parameter instead of the `install.cfg` parameter because the requirement to filter by batch may vary from job to job.

Note: Execute the below-mentioned query in order to run the 508 Job for the *filter_by_batch* parameter:

```
insert into kdd_param_binding values ('filter_by_batch', 'Alert Correlation',
113000023, 'true').
```

This query must be manually executed.

In addition to the job parameters, there is a certain metadata that must 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 41. 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. To do this, you must define whether the origin of the path should be the focus of an alert or a matched record, by populating either. 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 to execute to establish the relationship between the alert's focus or matched record and this business entity defined by a "query definition" represented in the KDD_QUERY_DEFS table as follows:

- The QUERY_DEF_NM column provides a name for the query definition.
- The FILTER_TABLE_NM provides the name of the source data table containing the data for the business entity we are trying to correlate to.
- The FILTER_ATTR_NM provides the column name from the FILTER_TABLE_NM that defines the focal attribute or matched record attribute (path origin) that we are filtering business entity source data records by (path destination).
- The FILTER_ATTR_TYPE_CD provides the type code of this attribute (L for long/numeric, S for string).

Finally, the SQL_TX provides the actual query where we must select three columns:

- origin key ID(focal/matched-attribute key ID)
- destination key id (business entity key ID)
- display id (business entity display ID)

For example, if we are trying to establish an alert-to-business-entity path/correlation from an alert's focal account to primary customer, the record in KDD_QUERY_DEFS would be defined as follows: QUERY_DEF_NM of "AC to CU-Prmry", FILTER_TABLE_NM of "ACCT", FILTER_ATTR_NM of "ACCT_INTRL_ID", FILTER_ATTR_TYPE_CD of "S", and SQL_TX of "SELECT ACCT_INTRL_ID, PRMRY_CUST_INTRL_ID, PRMRY_CUST_INTRL_ID FROM BUSINESS.ACCT WHERE ACCT.PRMRY_CUST_INTRL_ID is NOT NULL" The Alert Correlation engine will add a filter to this query at run-time based on the FILTER_TABLE_NM and FILTER_ATTR_NM (In this example it would add "AND ACCT.ACCT_INTRL_ID in (?)" where "?" would be replaced with the alert's focal entity ID).

The `PATH_ID` and `PATH_NM` in the table above are 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 42. 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 group. 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 group. The "importance" or "strength" of a correlation determined by this path may vary depending on the scenario or scenario group 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. 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.

Note: You can set the precedence for each rule in the KDD_CORR_RULE table by providing the appropriate precedence number in the PRECEDENCE_NB column.

For more information on validating/loading correlation rules, refer to the *Managing Alert Correlation Rule Migration Utility* section. 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[ (CORR.SCORE_CT >= 0) OR (CORR.ALERT_CT > 2) ]]>
  </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.SCNRO_ID = 114000074 ) AND (CORR.SCNRO_CT) >= 3]]>
      </AlertCorrAttrOperations>
      <ExistingCasePromoteLossRcvryData>TRUE
      </ExistingCasePromoteLossRcvryData>
      <Case type="AML" subtype="SAR" 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 numerical value , such as *from alert* and *to alert* which can be correlated if they both have `SCORE_CT >= 0`, represented by `CORR.SCORE_CT >= 0`, or a *from alert* and *to alert* which can be correlated if `CORR.ALERT_CT >2`. 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 , such as > 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 , such as if the current date is October 14, we will look back 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:

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

`<OFSAAI Installed Directory>/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 , such as 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.

Note: This is only applicable if your firm has implemented Enterprise Case Management.

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 need to 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.

Custom Scoring Rules

The custom scoring rules enhancement allows users to configure rules which define the attributes that are evaluated before arriving at a defined score. Rules are PL/SQL procedures, so these attributes can be bindings, matched attributes or any other alert data.

The following is a sample rule with a temporary table called @corr_score_temp, which is a dynamic table:

```
DECLARE
   c_corr_id @corr_score_temp.corr_id%type;
   c_score_ct @corr_score_temp.score_ct%type;
   CURSOR c_corr_scoring is

select z.corr_id,sum (z.score_ct_curr)score_ct from
(

  select distinct a.corr_id,a.score_ct,a.score_ct+06 score_ct_curr
  from @corr_score_temp a
  INNER JOIN kdd_review kr ON kr.review_id =a.review_id
  INNER JOIN kdd_break kb ON kr.review_id =kb.prnt_break_id
  INNER JOIN kdd_break_binding kbb ON kb.break_id= kbb.break_id and
kbb.bindg_nm='Tot_Trxn_Am' and kbb.value_tx>=30000

  union

  select distinct b.corr_id,b.score_ct,b.score_ct+16 score_ct_curr
  from @corr_score_temp b
  INNER JOIN kdd_review_scnro krs ON krs.review_id = b.review_id
  group by b.corr_id, b.score_ct
  having count(distinct(krs.scnro_id))>=2

  union

  select distinct c.corr_id,c.score_ct,c.score_ct+26 score_ct_curr
  from @corr_score_temp c,cust cu,kdd_review kr
  where c.review_id= kr.review_id
  and kr.focal_ntity_dsply_id = 'CUMLNOAAC-701'
  and kr.creat_ts BETWEEN TO_DATE('2016/11/09', 'yyyy/mm/dd') AND
       TO_DATE('2016/11/09', 'yyyy/mm/dd')

  union

  select distinct d.corr_id, d.score_ct,d.score_ct+36 score_ct_curr
  from @corr_score_temp d
  INNER JOIN kdd_review_scnro krs ON krs.review_id=d.review_id and krs.scnro_id =
'114000081'

  union

  select distinct e.corr_id,e.score_ct,e.score_ct+46 score_ct_curr
  from @corr_score_temp e
  INNER JOIN kdd_review_bus_ntity krbn on krbn.review_id = e.review_id and
krbn.bus_ntity_id='113000004'
  INNER JOIN cust cu ON krbn.bus_ntity_key_id = cu.cust_intrl_id and
cu.cust_bus_risk_nb >= 6



) z
group by z.corr_id,
         z.score_ct;

BEGIN
   OPEN c_corr_scoring;
   LOOP
```

```
      FETCH c_corr_scoring into c_corr_id, c_score_ct;
      EXIT WHEN c_corr_scoring%notfound;
      UPDATE @corr_score_temp z SET z.SCORE_CT = c_score_ct WHERE z.CORR_ID = 
c_corr_id;
   END LOOP;
   CLOSE c_corr_scoring;
   COMMIT;
END;
```

Note: An entry is created in the log file after the correlation job is executed and the `@corr_score_temp` placeholder is replaced with the temporary table that is created when the job is executed.

Note: If the job is executed successfully, the temporary table is truncated. If the job fails, the temporary table remains in the database.

The following is the rule with the `@corr_score_temp` table replaced with a `CORR_SCORE_TMP` table:

```
DECLARE
   c_corr_id CORR_SCORE_TMP_560201.corr_id%type;
   c_score_ct CORR_SCORE_TMP_560201.score_ct%type;
   CURSOR c_corr_scoring is

select z.corr_id,sum (z.score_ct_curr)score_ct from
(

  select distinct a.corr_id,a.score_ct,a.score_ct+06 score_ct_curr
  from CORR_SCORE_TMP_560201 a
  INNER JOIN kdd_review kr ON kr.review_id =a.review_id
  INNER JOIN kdd_break kb ON kr.review_id =kb.prnt_break_id
  INNER JOIN kdd_break_binding kbb ON kb.break_id= kbb.break_id and 
kbb.bindg_nm='Tot_Trxn_Am' and kbb.value_tx>=30000

  union

  select distinct b.corr_id,b.score_ct,b.score_ct+16 score_ct_curr
  from CORR_SCORE_TMP_560201 b
  INNER JOIN kdd_review_scnro krs ON krs.review_id = b.review_id
  group by b.corr_id, b.score_ct
  having count(distinct(krs.scnro_id))>=2
  union


  select distinct c.corr_id,c.score_ct,c.score_ct+26 score_ct_curr
  from CORR_SCORE_TMP_560201 c,cust cu,kdd_review kr
  where c.review_id= kr.review_id
  and kr.focal_ntity_dsply_id = 'CUMLNOAAC-701'
  and kr.creat_ts BETWEEN TO_DATE('2016/11/09', 'yyyy/mm/dd') AND
       TO_DATE('2016/11/09', 'yyyy/mm/dd')

  union


  select distinct d.corr_id, d.score_ct,d.score_ct+36 score_ct_curr
  from CORR_SCORE_TMP_560201 d
  INNER JOIN kdd_review_scnro krs ON krs.review_id=d.review_id and krs.scnro_id =
'114000081'

  union


  select distinct e.corr_id,e.score_ct,e.score_ct+46 score_ct_curr
  from CORR_SCORE_TMP_560201 e
  INNER JOIN kdd_review_bus_ntity krbn on krbn.review_id = e.review_id and
krbn.bus_ntity_id='113000004'
  INNER JOIN cust cu ON krbn.bus_ntity_key_id = cu.cust_intrl_id and
cu.cust_bus_risk_nb >= 6



) z
group by z.corr_id,
         z.score_ct;


BEGIN
   OPEN c_corr_scoring;
   LOOP
      FETCH c_corr_scoring into c_corr_id, c_score_ct;
      EXIT WHEN c_corr_scoring%notfound;
     UPDATE CORR_SCORE_TMP_560201 z SET z.SCORE_CT = c_score_ct WHERE z.CORR_ID =
c_corr_id;
   END LOOP;
   CLOSE c_corr_scoring;
```

```
    COMMIT;
END;
```

Configuring Rules

This section covers the following topics:

- Structure of the Configuration Table
- Structure of the Temporary Table
- Configuring Custom Rules

Structure of the Configuration Table

The KDD_CORR_SCRNG_CSTM_RULE is a metadata table which is added to configure rules.

Table 43. Structure of the Configuration Table

Column Name	Primary Key	Column Type	Nullable?
RULE_ID	*	NUMBER(10)	NO
STATUS_CD		VARCHAR2(50)	NO
SQL_TX		CLOB	NO

Following are the description for each column name:

- RULE_ID: This column contains a unique number which identifies the alert correlation scoring rule.
- STATUS_CD: This column contains either ACT or INACT, which stands for active or inactive respectively.
- SQL_TX: This column contains the PL/SQL procedure that contains the score evaluation logic.

Structure of the Temporary Table

The CORR_SCORE_TMP_<run id> is a temporary table which is created when the correlation job is run

Table 44. Structure of the Temporary Table

Column Name	Primary Key	Column Type	Nullable?
CORR_ID	*	NUMBER(10)	NO
REVIEW_ID	*	NUMBER(10)	NO
SCORE_CT		NUMBER(10)	NO
CORR_RULE_ID	*	NUMBER(10)	NO

Following are the description for each column name:

- CORR_ID: This column contains a unique number which identifies the alert correlation that is scored.
- REVIEW_ID: This column contains a unique number that identifies an alert in the correlation.
- SCORE_CT: This column contains the correlation score.
- CORR_RULE_ID: This column contains the conditions which need to be satisfied in the CLOB column in order to promote the alert to a case.

Configuring Custom Rules

The following are the steps to configure custom rules:

1. Create a Pl/SQL procedure that has the logic for scoring with a place holder for the temporary table.
2. Make an entry in the KDD_CORR_SCRNG_CSTM_RULE with STATUS_CD as ACT.
3. Change the score strategy to CUSTOM_SCORE in the `correlation rule.xml` file. (KDD_CORR_RULE.RULE_OP_TX) Example: <Scoring strategy="CUSTOM_SCORE" incStrategy="ALERT_COUNT" />
4. A new `install.cfg` property called `alert_alert_corr.ptc.threshold`, is added. This is an integer value. Correlations are promoted to a case only if the correlation score exceeds this threshold.

Sample Alert Correlation Rules

OFSBD 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.

OFSBD installs these sample alert correlation rules in the `<OFSAAI Installed Directory>/database/db_tools/data` directory.

Displaying Alert-to-Business Entity Path Details on the User Interface

To view Alert-to-Business Entity Path rules in the UI (in addition to the default rules), you must add entries to KDD_CODE_SET_TRNLN with a CODE_SET value of "Relationship", a CODE_VAL value that corresponds to the KDD_BUS_NTITY_PATH.PATH_NM of the new rule, and a CODE_DISP_TX with the desired display text to appear on the UI for the new rule. All correlation rules must also be added to the User/Organization under the Security Attribute Administration screen.

Case Assignment

OFSBD provides a mechanism to assign cases to a predefined owner (either an individual user or a pool of users). When performing case assignment, the module fetches new, unowned 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 *Administration Tools User Guide*, for more information.

The assignment framework allows for customers to write their own Java code to replace the product functionality with their own customized functionality. The modules that can be replaced include the assignment-eligible objects (currently Alerts and Cases), the assignment rule processing logic, and the manner in which the assignment results are output (currently results are written out to the database for batch assignment, or passed back in a SOAP XML response for the assignment web services call). For more information on how to take advantage of this feature, please contact Oracle Support.

Running the Case Assignment Job

The Case Assignment Job is part of the OFSBD subsystem.

To run an Case Assignment job, follow these steps:

1. Run the execute.sh script as follows:

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

By default, Behavior Detection writes log messages for this script in the `<OFSAAI Installed Directory>/bdf/logs/<Processing Date>/CaseAssignment.log` file.

CHAPTER 6

Managing Personal Trading Approval

This chapter provides an overview of the Personal Trading Approval (PTA) application-specific admin tasks.

This chapter focuses on the following topics:

- Mapping the Employee Role in PTA
- Personal Trading Approval Tasks
- Mapping the Employee Role in PTA

These are applicable and required only if the Oracle client has deployed the Personal Trading Approval application.

Mapping the Employee Role in PTA

Employees of the Oracle client must be mapped to the Employee role to be able to access the Personal Trading Approval application. Due the high number of employees within the Oracle client that must be mapped to this role, the following script captures the employees along with their system logon and maps each employee to the Employee User Group, which gives each employee access to the Account Approval, Pre-Trade Approval, and Attestation functionality within Personal Trading Approval to allow employees to submit all personal investment accounts for approval, pre-trade approval requests, and attestation, respectively:

Table 45. Load Employee User Groups

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 Financial Services client to the Personal Trading Approval 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 Personal Trading Approval application. To identify all employees that must have access to Personal Trading Approval, the Employee and System Logon files must be provided. The system logon IDs for all employees within the Oracle Financial Services client captured in the System Logon file must have the Source System (`SRC_SYS_CD`) field value set to MTS. See *Oracle Financial Service Behavior Detection Data Interface Specification* for details on how to populate the Employee and System Logon files.

Before running this script, all data for the Employee and System Logon files as specified in the *Data Interface Specification (DIS)* must be loaded using the BD Ingestion Manager. For more information on data management, see *Chapter 3, "Managing Data."*.

Migrating External Investment Accounts into the Account Table

In the Personal Trading Approval application, employees of the Oracle Client will use the Account Approval functionality to submit their own personal investment accounts held outside of their firm for approval.

If the Oracle Client intends to run the Brokerage Compliance scenarios for Personal Trading Approval, all newly submitted personal investment accounts of employees of the Oracle client must be migrated into the FSDM Account table at the end of the day, after all the DIS files have been completely loaded by executing the following command:

```
runUtility.sh ExternalInvestmentAccountToAccount
```

Personal Trading Approval Tasks

If the Oracle client has implemented the Personal Trading Approval application, the following processes must be executed to successfully utilize Personal Trading Approval:

Table 46. Personal Trading Approval Database Procedures

Task	Script	Database Procedure / Description
Request Expiration	`run_pta_set_rqst_expired.sh`	`P_PTA_BA_RQST_EXPR` The procedure updates all open pre-trade approval requests not approved by end of the day to "Request Expired" status.

Update Derived Account Identifier field in the External Investment Account table	`run_update_exna_drvd_acct.sh`	The script `run_update_exna_drvd_acct.sh` internally calls the `P_UPD_ExNA_DRVD_ACCT_INTRL_ID` procedure. This procedure will update Derived Account Identifier field in the External Investment Account table and the External Investment Account Arc table for legacy accounts uploaded into this table using the Excel Upload feature by the Oracle client.
		The script `run_update_exna_drvd_acct.sh` can either be run without any arguments or it can take either [`EXTRL_NVSMT_ACCT`] OR [`EXTRL_NVSMT_ACCT_ARC`] as its input.
		Usage:
		`run_update_exna_drvd_acct.sh`
		OR
		`run_update_exna_drvd_acct.sh EXTRL_NVSMT_ACCT`
		OR
		`run_update_exna_drvd_acct.sh EXTRL_NVSMT_ACCT_ARC`
		When `run_update_exna_drvd_acct.sh` is run without any arguments it will process records for both `EXTRL_NVSMT_ACCT and EXTRL_NVSMT_ACCT_ARC` tables.

After these scripts have been successfully executed upon initialization of the Personal Trading Approval application, how frequent each script is run is determined by the Oracle client. For more information about running tasks, see Chapter 5, *Post-Processing Tasks,* on page 87.

CHAPTER 7 Managing Batch Processing Utilities

OFSBD 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 Batch Processing Utilities
- Managing Common Resources for Batch Processing Utilities
- Managing Annual Activities
- Managing Alert and Case Purge Utility
- Managing Batch Control Utility
- Managing Calendar Manager Utility.
- Managing Data Retention Manager
- Database Statistics Management
- Managing Flag Duplicate Alerts Utility
- Managing Notification
- Managing Push E-mail Notifications
- Refreshing Temporary Tables
- Managing Truncate Manager
- Managing ETL Process for Threshold Analyzer Utility
- Managing Deactivate Expired Alert Suppression Rules

About Batch Processing Utilities

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

- **Managing Alert and Case Purge Utility**: Provides the capability to remove alerts (along with their matches and activities) generated erroneously or which have exceeded the retention policies of the organization.
- **Managing Batch Control Utility**: Manages the start and termination of a batch process (from data management to alert post-processing) and enables access to the currently running batch.
- **Managing Calendar Manager Utility.**: Updates calendars in the OFSBD system based on predefined business days, holidays, and days off or non-business days.
- **Managing 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.
- **Database Statistics Management**: The system uses a script to manage Oracle database statistics. These statistics determine the appropriate execution path for each database query.

- **Managing Flag Duplicate Alerts Utility**: 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.
- **Push E-mail Notification**: Enables you to configure users of the Alert Management subsystem to receive e-mail when alerts are assigned to them.
- **Managing Notification**: Enables you to configure users of Alert 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.
- **Refreshing Temporary Tables**: Refreshes temporary tables that the behavior detection process uses and estimates statistics for the newly populated tables.
- **Managing Truncate Manager**: Truncates tables that require complete replacement of their data.

Figure 34 illustrates the frequency with which you use these batch-related database utilities when managing activities: daily, weekly, monthly, annually, or as needed.

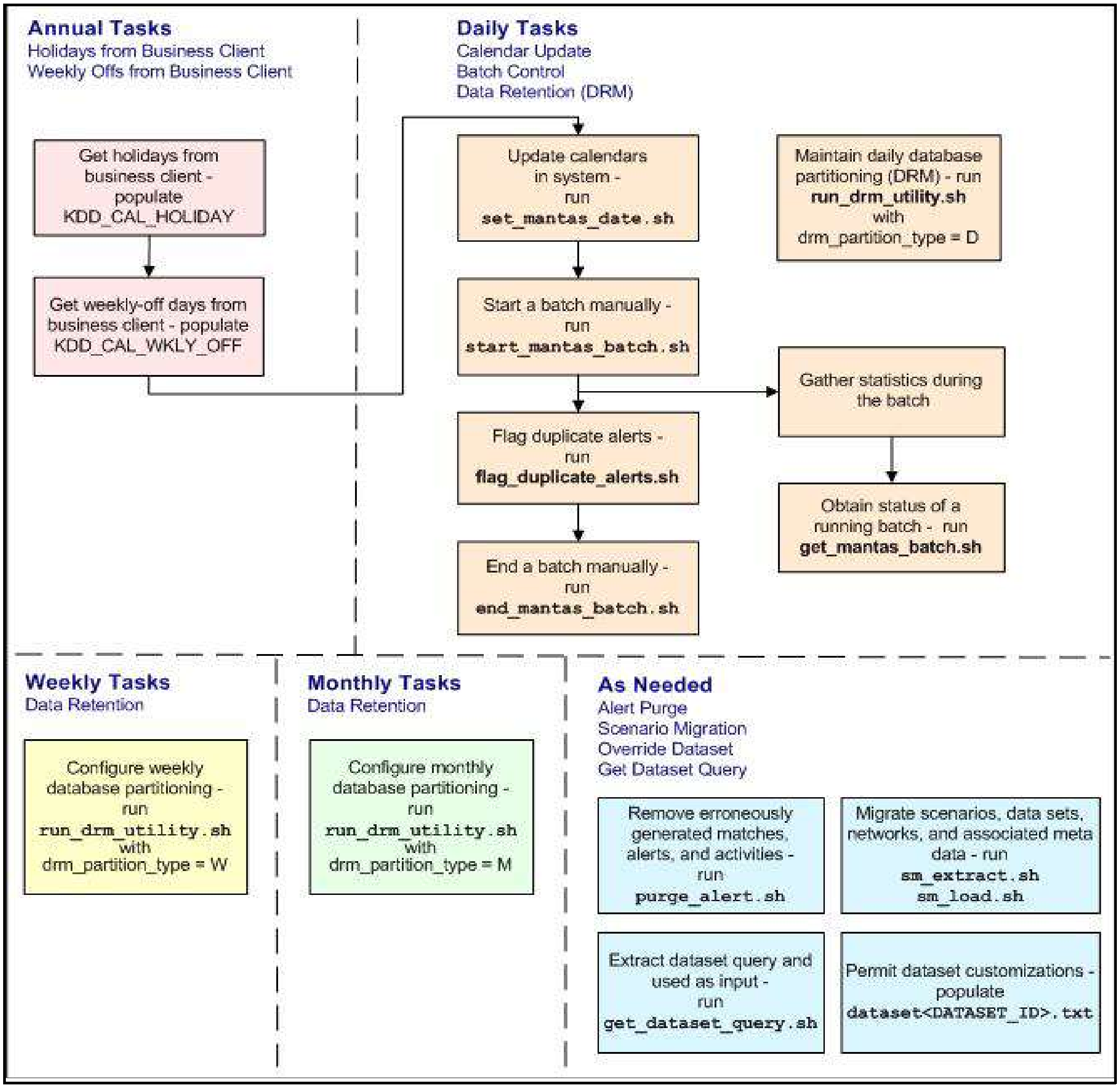

Annual Tasks
Holidays from Business Client
Weekly Offs from Business Client
Get holidays from business client - populate KDD_CAL_HOLIDAY
Get weekly-off days from business client - populate KDD_CAL_WKLY_OFF
Daily Tasks
Calendar Update
Batch Control
Data Retention (DRM)
Update calendars in system - run 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 34. Managing Database Activities with Utilities

Figure 34 illustrates the following:

- Daily tasks are initially dependent on the annual tasks that you perform, such as obtaining holiday and weekly off-days from an Oracle client.
- Daily tasks can include updating Behavior Detection calendars and managing batch processes. You may must 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 , such as from test to production.

Managing Common Resources for Batch Processing 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 Configuration
- Categories Configuration

Install Configuration

Configuration information resides in the <OFSAAI Installed Directory>/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 must 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 , such as activate or deactivate particular logging levels and specify locations for logging entries.

Figure 35 (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. You should ensure that the ATOMIC schema name is in uppercase.

```
# @(#)Copyright (c) 2016 Oracle Finanacial Services Software Inc. All Rights
Reserved.
# @(#) $Id: install.cfg $
#
# 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
```

(Continued on next page)

(Continued from previous page)

```
# The file contains some properties that are common and specific properties for
each
# of the tools.

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

NLS_LENGTH_SEMANTICS=CHAR
database.driverName=oracle.jdbc.driver.OracleDriver
utils.database.urlName=jdbc:oracle:thin:@ofss2221324.in.oracle.com:1521:Ti5O12L64
utils.database.username=f802_fccm
utils.database.password=NzBXdzslR43hh0nWkaqYvA==
schema.algorithms.owner=f802_fccm
schema.algorithms.password=NzBXdzslR43hh0nWkaqYvA==
schema.web.owner=f802_fccm
schema.web.password=NzBXdzslR43hh0nWkaqYvA==
schema.report.owner=f802_fccm
schema.report.password=NzBXdzslR43hh0nWkaqYvA==

schema.mantas.owner=f802_fccm
schema.mantas.password=NzBXdzslR43hh0nWkaqYvA==
utils.miner.user=f802_fccm
utils.miner.password=NzBXdzslR43hh0nWkaqYvA==
schema.business.owner=f802_fccm
schema.business.password=NzBXdzslR43hh0nWkaqYvA==
schema.market.owner=f802_fccm
schema.market.password=NzBXdzslR43hh0nWkaqYvA==
utils.data.directory=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/data
ingest.user=f802_fccm
ingest.password=NzBXdzslR43hh0nWkaqYvA==

schema.kdd.owner=f802_fccm
schema.kdd.password=NzBXdzslR43hh0nWkaqYvA==
casemng.schema.owner=f802_fccm
casemng.schema.password=NzBXdzslR43hh0nWkaqYvA==
```

(Continued on next page)

(Continued from previous page)

```
################ 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

############### 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

#If custom patterns exist for a product scenario, set to 'Y' when loading a
scenario hotfix.
#This should normally be set to 'N'.
load.ignore.custom.patterns=N
```

(Continued on next page)

(Continued from previous page)

```
#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=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/mantas_cfg/dep.cfg

sm.release=5.7.1

#### EXTRACT

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

# Specify the case schema name for both extraction and load .
caseschema.schema.owner=f802_fccm

# Specify the jdbc driver details for connecting to the source database
extract.conn.driver=${database.driverName}
extract.conn.url=jdbc:oracle:thin:@ofss2221324.in.oracle.com:1521/Ti5O12L64

#Source System Id
extract.system.id=

# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
extract.schema.case=f802_fccm
extract.schema.business=${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
```

(Continued on next page)

(Continued from previous page)

```
#Specify the full path in which to place extracted scenarios
extract.dirname=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/data

#Specify the full path of the directory where the backups for the extracted
scripts would be maintained
extract.backup.dir=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/data/te
mp

#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=Ti5O12L64
# Specify the schema names for Load
load.schema.mantas=${schema.mantas.owner}
load.schema.case=f802_fccm
load.schema.business=${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=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/data
# Specify whether threshold can be updated
load.threshold.update=Y
# Specify whether score can be updated
load.score.update=Y
```

(Continued on next page)

(Continued from previous page)

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

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

# Specify whether or not to consider Matches
limit_matches=N

# Specify whether or not to purge the data
purge=Y

# Specify batch size for which commit should perform
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

# Specify purge db user
purge.database.user=f802_fccm

# Specify purge db user password.
purge.database.password=

# Specify whether alerts has to be purged or not.
purge_alert_flag=Y
```

(Continued on next page)

(Continued from previous page)

```
# Specify whether fatca cases/assessments has to be purged or not.
purge_fatca_flag=Y

# Specify whether case has to be purged or not.
purge_case_flag=Y

# Specify defualt rule set.
purge_default_rule_set=

# Specify total number of threads should be used for the process.
purge_threads_no=10

# Specify report directory for report on process performed.
purge_report_directory=

# Specify product version
purge_product_version=
#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=/scratch/ofsaadb/BD802_Final/BD802FL/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
```

(Continued on next page)

```
#################################################################
```

(Continued from previous page)

```
# file must be placed in the mantas_cfg file, and the jec.jar file must be
# copied to the db_tools/lib directory. Then, edit the file
# db_tools/bin/run_push_email.ksh, uncomment the JEC_JARS= line.
#
#########################################################################
# Currently only smtp, smtps, or exchange
email.type=smtp

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

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

# From address for sent mails. This is ignored in Exchange mode. If omitted in SMTP mode, the mail account associated
# with the Unix/Linux account is used.
email.from=
# SMTP settings
email.smtp.host=mailhost.us.oracle.com
# 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
```

(Continued on next page)

(Continued from previous page)

```
email.exchange.useNTLM=false
email.exchange.draftsfoldername=drafts
email.exchange.useHTML=true

#HTML email styles
email.style.header=font-family:Arial, Helvetica, sans-serif;font-size:10pt;
color:black;
email.style.hr=color: #555; background-color: #f00; height: 1px;
email.style.title=font-family:Arial, Helvetica, sans-serif;font-style:
bold;font-size:12pt;
email.style.message=font-family:Arial, Helvetica, sans-serif;font-size:11pt;
email.style.table=font-family:Arial, Helvetica, sans-serif;border:1px solid #000;
border-collapse:collapse;
email.style.th=font-style: bold;border:1px solid #000; border-collapse:collapse;
padding: 4px; background:#C7DAED
email.style.tr=font-size:10pt
email.style.td=border:1px solid #000; border-collapse:collapse; padding: 4px
email.style.footer=font-family:Arial, Helvetica, sans-serif;font-size:10pt;
color:black;
email.style.disclaimer=font-style: italic;

######### PDF ARCHIVE CONFIGURATION #########################
# 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=@ALERT_MANAGEMENT_SERVICE_URL@
######### 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 #########################
```

(Continued on next page)

(Continued from previous page)

```
#
# 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=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/mantas_cfg/analysis_aml.xml

# Output file and file format control
dat.analysis.output=/scratch/ofsaadb/BD802_Final/BD802FL/database/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=,
######### 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=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/mantas_cfg/db_variables.cfg
```

(Continued on next page)

(Continued from previous page)

```
Oracle.DuplicateRow=1
Oracle.ObjectExists=955,2260,2275,1430,1442,1451,957,1408,2261,1543
Oracle.ObjectDoesNotExist=942,1418,1434,2441,904,4043,1927,2443

dbscript.execution.users=(system|business|mantas|market|miner|ingest|report|kdd|a
lgorithms|case|config|fatca|ctr|kyc|fsdf|dbutil|web)

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

############# 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=true
```

(Continued on next page)

(Continued from previous page)

```
log.trace=true
log.time.zone=US/Eastern

# Specify whether logging for a particular level should be performed
# synchronously or asynchronously.
log.fatal.synchronous=true
log.warning.synchronous=true
log.notice.synchronous=true
log.diagnostic.synchronous=true
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=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/mantas
_cfg/etc/mantas_database_message_lib_en.dat
# Specify the full path to the categories.cfg file
log.categories.file.path=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/m
antas_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=/scratch/ofsaadb/BD802_Final/BD802FL/database/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.
```

(Continued on next page)

(Continued from previous page)

```
# Logging will go to the console if mantaslog was selected and this property is
# not given a value.
log.mantaslog.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/log
s/mantaslog.log

# 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
```

Figure 35. Sample `install.cfg` **File**

Categories Configuration

In the `<OFSAAI Installed Directory>/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 36 provides a sample `categories.cfg` file.

```
# @(#)Copyright (c) 2016 Oracle Finanacial Services Software Inc. All Rights Reser
# @(#) $Id: categories.cfg $
# Common Logging categories configuration for Mantas 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 Mantas does not currently route by
# 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.PURGE_UTIL.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_
/logs/purge.log
log.category.BATCH_CONTROL.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/
ols/logs/batch_control.log
log.category.CALENDAR_MANAGER.location=/scratch/ofsaadb/BD802_Final/BD802FL/databa
_tools/logs/calendar_manager.log
log.category.DATA_RETENTION_MANAGER.location=/scratch/ofsaadb/BD802_Final/BD802FL/
ase/db_tools/logs/DRM_Utility.log
log.category.TRUNCATE_MANAGER.location=/scratch/ofsaadb/BD802_Final/BD802FL/databa
_tools/logs/truncate_manager.log
log.category.COMMON_UTILITIES.location=/scratch/ofsaadb/BD802_Final/BD802FL/databa

db_tools/logs/common_utilities.log
```

(Continued on next page)

(Continued from previous page)

```
log.category.EXTRACT.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log
log.category.LOAD.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log
log.category.REFRESH_TEMP_TABLE.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/refresh_temp_table.log
log.category.RUN_STORED_PROCEDURE.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/run_stored_procedure.log
log.category.GET_DATASET_QUERY.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/get_dataset_query.log
log.category.HDC.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/hdc.log
log.category.PUSH_EMAIL.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/push_email.log
log.category.HIGHLIGHT_GENERATOR.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/highlight_generator.log
log.category.REPORT.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/report.log
log.category.DATA_ANALYSIS_TOOL.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/data_analysis_tool.log


log.category.DB_BUILDER.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/db_builder.log
log.category.DB_BUILDER_SQL.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/db_builder.log,/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/db_builder_sql.log


log.category.CORRRULEMIGRATIONUTIL_EXTRACT.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log
log.category.CORRRULEMIGRATIONUTIL_LOAD.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log
log.category.CONFIGURATIONMIGRATIONUTIL_EXTRACT.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log
log.category.CONFIGURATIONMIGRATIONUTIL_LOAD.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log
log.category.ARCHIVE_PDF.location=/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/pdf_archive.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.
```

(Continued on next page)

(Continued from previous page)

```
# 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

log.category.DB_BUILDER.notice.location=console
log.category.ARCHIVE_PDF.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/pdf_archive.log

log.category.CORRRULEMIGRATIONUTIL_LOAD.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log

log.category.CORRRULEMIGRATIONUTIL_LOAD.fatal.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log
log.category.CORRRULEMIGRATIONUTIL_EXTRACT.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log
log.category.CORRRULEMIGRATIONUTIL_EXTRACT.fatal.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log
log.category.CONFIGURATIONMIGRATIONUTIL_LOAD.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log
log.category.CONFIGURATIONMIGRATIONUTIL_LOAD.fatal.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/load.log
log.category.CONFIGURATIONMIGRATIONUTIL_EXTRACT.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log
log.category.CONFIGURATIONMIGRATIONUTIL_EXTRACT.fatal.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/extract.log

log.category.PURGE_UTIL.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.warning.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.fatal.location=/scratch/ofsaadb/BD802_Final/BD802FL/datab
ase/db_tools/logs/purge.log
log.category.PURGE_UTIL.trace.location=/scratch/ofsaadb/BD802_Final/BD802FL/datab
ase/db_tools/logs/purge.log
```

Figure 36. Sample Logging Information in the `categories.cfg` File

Configuring Console Output

Figure 37 displays a section of the sample `categories.cfg` file from Figure 36.

Note: The log routing information is in bold text.

```
log.category.PURGE_UTIL.notice.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.warning.location=console,
/scratch/ofsaadb/BD802_Final/BD802FL/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.fatal.location=/scratch/ofsaadb/BD802_Final/BD802FL/databa
b_tools/logs/purge.log
log.category.PURGE_UTIL.trace.location=/scratch/ofsaadb/BD802_Final/BD802FL/databa
b_tools/logs/purge.log
```

Figure 37. 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 37, Alert and Case Purge and Calendar Manager display relevant utility information in addition to logging it.

Note: If an entry in the `categories.cfg` file does not already include this information, you must add it manually, including the comma.

Managing Annual Activities

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

This section covers the following topics:

- Loading Holidays
- Loading Non-business Days

Loading Holidays

On an annual basis, you must 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 38 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( '01/01/2017',
'MM/DD/YYYY'), 'New Year''s Day - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '01/16/2017',
'MM/DD/YYYY'), 'Martin Luther King Jr.''s Birthday - 2017',
'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '02/20/2017',
'MM/DD/YYYY'), 'President''s Day - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '04/14/2017',
'MM/DD/YYYY'), 'Good Friday - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '05/29/2017',
'MM/DD/YYYY'), 'Memorial Day - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '07/04/2017',
'MM/DD/YYYY'), 'Independence Day - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '09/04/2017',
'MM/DD/YYYY'), 'Labor Day - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '11/22/2017',
'MM/DD/YYYY'), 'Thanksgiving Day - 2017', 'C');

INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '12/25/2017',
'MM/DD/YYYY'), 'Christmas Day - 2017', 'C');
```

Figure 38. Sample `KDD_CAL_HOLIDAY` Table Loading Script

The following table describes the contents of the KDD_CAL_HOLIDAY table.

Table 47. **KDD_CAL_HOLIDAY**

Column Name	Description
CLNDR_NM	Specific calendar name.
CLNDR_DT	Date that is a holiday.
HLDY_NM	Holiday name , such as 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.

Note: If the maximum date is less than 90 days ahead of the provided date, the process logs a warning message that the specific calendar's future holidays need updating. If any calendars have no holiday records, the system logs a Warning message that the specific calendar has no recorded holidays for the appropriate date range.

Loading Non-business Days

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

The following text provides an example of an SQL script for loading the table.:

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

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

COMMIT;
```

Figure 39. Sample KDD_CAL_WKLY_OFF Table Loading Script

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

The following table describes the contents of the KDD_CAL_WKLY_OFF table.

Table 48. **KDD_CAL_WKLY_OFF**

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,Wednesday=4, Thursday=5, Friday=6, Saturday=7.

Note: 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.

Managing Alert and Case Purge Utility

The ingestion of certain data can result 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.

There may also be cases when the alerts or cases have been residing in the database due to the retention policies imposed by the regulatory bodies, or the internal policies of the respective organization.

The Alert and Case Purge Utility enables you to identify and remove such matches, alerts and cases, and activities selectively, based on a number of parameters (like the Behavior Detection Job ID, Behavior Detection Scenario ID, Behavior Detection Scenario Class, or 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, a XML File and a configuration file in which you define the process parameters to use in the purge processing. The system directs output to a configurable log file; processing appends this log with information about subsequent executions of the scripts.

This section covers the following topics:

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

Directory Structure

The following table describes the directory structure for the Alert and Case Purge Utility.

Table 49. Alert and Case 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 , such as `install.cfg` and `categories.cfg`, in which you can configure properties and logging attributes.
`logs/`	Keeps the `<OFSAAI Installed Directory>/database/db_tools/logs/purge.log file` that the utility generates during execution.
`data/`	Keeps `.sql` files for execution.
`.xml`	Contains the Purge Rules Configuration File (`PurgeRules.xml`), which is used for configuring the Alert and Case purge rules.

Logs

As the Alert and Case Purge Utility performs alert detection activities, it generates a log that it enters in the `<OFSAAI Installed Directory>/database/db_tools/logs/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 and Case Purge Utility in the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg` files. For more information about logging in these configuration files, refer to *Managing Common Resources for Batch Processing Utilities* on page 122 and Appendix A, *Logging,* on page 245 for more information.

Precautions

You use the utility to rid the system of falsely-generated matches and alerts or cases. Other than recorded information in the `<OFSAAI Installed Directory>/database/db_tools/logs/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: The utility also purges any alert or case which is used to trigger Auto Suppression or establish Trusted Parties. However, this would not affect the Suppression Rule or the Trusted Pair except that the `kdd_auto_suppr_alert.trgr_alert_id,kdd_trusted_pair.trgr_alert_id,`or `kdd_trusted_pair.trgr_case_id` columns are set to a null value

Note: Run the Alert and Case Purge Utility 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 and Case Purge Utility

The Alert and Case Purge Utility is not part of an automated batch process. You run this manual process only when necessary (refer to Figure 34). The following sections describe configuring and executing the utility, as well as the utility's process flow:

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

Configuring the Alert and Case Purge Utility

To configure the Alert and Case Purge Utility, follow these steps:

1. Navigate to the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg`.
2. Edit the parameters in the install.cfg file to the desired settings. This file contains common configuration information that the Alert and Case Purge Utility and other utilities require for processing (refer to Figure 35). The following is a sample section from the `install.cfg` file for configuration information specific to this utility:

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

# Specify whether or not to consider Matches
limit_matches=N

# Specify whether or not to purge the data
purge=Y

# Specify batch size for which commit should perform
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

# Specify purge db user
purge.database.user=f802_fccm

# Specify purge db user password.
purge.database.password=

# Specify whether alerts has to be purged or not.
purge_alert_flag=Y

# Specify whether fatca cases/assessments has to be purged or not.
purge_fatca_flag=Y
```

(Continued on next page)

(Continued from previous page)

```
# Specify whether case has to be purged or not.
purge_case_flag=Y

# Specify defualt rule set.
purge_default_rule_set=

# Specify total number of threads should be used for the process.
purge_threads_no=10

# Specify report directory for report on process performed.
purge_report_directory=

# Specify product version
purge_product_version=


#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=/scratch/ofsaadb/BD804_Final/BD804FL/database/db_tools/data
```

Figure 40. Configuration Information

Note: Not specifying a value of *null* , such as leaving a value blank, in this section of the `install.cfg` file causes undesirable results.

The following table describes required and optional parameters for this utility.

Table 50. Alert and Case 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.
`purge_alert_flag`	Determines whether or not the utility would purge alerts, depending on the specified value: ● N: Does not purge the alerts irrespective of whether or not they identified according to the purge rule being used. This may be used when purging only the cases. ● Y (default): Purges the alerts as identified by the purge rule used to perform the purge operation.
`purge_case_flag`	Determines whether or not the utility would purge cases, depending on the specified value: ● N: Does not purge the cases irrespective of whether or not they identified according to the purge rule being used. This may be used when purging only the cases. ● Y (default): Purges the cases as identified by the purge rule used to perform the purge operation.
`purge_default_rule_set`	*(Optional)* Indicates the default set of rules to be used for purging alerts/cases. You may either specify the purge rules to be used against this parameter, or pass the name of the specific purge rules) as command line parameters You may specify a single purge rule, or a comma separated list of purge rules to be used as default when no other purge rule is provided from the command line.
`purge_threads_no`	*(Optional)* Identifies the number of concurrent threads to create for purging the alerts to optimize the performance. Specifying a non-positive value or specifying no value uses the default of 10 threads.
`purge_report_directory`	Identifies the absolute path to the directory where the purge activity report should be generated. The report file name has a name similar to `Purge_<YYYYMMDD.HH.MM.SS>.txt`. Here `<YYYYMMDD.HH.MM.SS>` represents current timestamp when the utility was executed.
`purge_product_version`	Identifies the OFSBD Product Version installed by the client.

The `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/etc/xml/PurgeRules.xml` file contains purge rules configuration information that the Alert and Case Purge Utility requires for processing. The following sample section from the `PurgeRules.xml` file provides configuration information for this utility.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:RuleSet xmlns:xs="http://namespaces.mantas.com/RuleSet">
 <Alert>
  <Rule id="1">
  <IdentifierList>286,4565,4537</IdentifierList>
 <ScenarioIdList>114697002</ScenarioIdList>
   <ScenarioClassList>CR</ScenarioClassList>
   <CreateDate>
    <StartDate>2011-05-25</StartDate>
    <EndDate>2011-05-25</EndDate>
   </CreateDate>
   <DomainCode>MTS</DomainCode>
   <BatchId>2</BatchId>
   <ThresholdSetIds>118745206,118710066</ThresholdSetIds>
   <LastActionDate>
    <StartDate>2016-05-25</StartDate>
    <EndDate>2016-05-25</EndDate>
   </LastActionDate>
   <Status>CL</Status>
   <JobIds>102202</JobIds>
  </Rule>
 </Alert>
 <Case>
  <Rule id="2">
   <IdentifierList>CA51300004,CA3773,CA3757,CA3766</IdentifierList>
   <CaseTypeList>FR_EE,FR_ON</CaseTypeList>
   <CreateDate>
    <Age>1Y</Age>
   </CreateDate>
   <LastActionDate>
    <StartDate>2016-06-22</StartDate>
    <EndDate>2016-06-22</EndDate>
</LastActionDate>
  </Rule>
  </Case>
</xs:RuleSet>
```

Figure 41. Configuration Information

The following table describes the Purge Rules Configuration Parameters.

Table 51. Alert and Case Purge Utility Parameters

Parameter	Description
Alert/Case	Identifies and encapsulates the purge rules for Alerts/Cases. You may define any number of purge rules for both alerts and cases.
Rule	Identifies a set of rules to be used for purging Alert/Case Information. All Alert and Case purge rules defined in this file must be provided a unique positive integer ID (as specified against the ID attribute). The value provided against the ID attribute is used by the utility to identify the rules to be used for carrying out the purge operations. **Note:** Not specifying a unique value for the ID attribute may lead to undesirable results.
IdentifierList	Identifies a list of Alert and Case IDs to be purged. You may specify more than one alert or case ID by separating them by comma.
ScenarioIdList	Identifies a list of Scenario IDs for which the alerts are to be purged. You may specify more than one Scenario ID by separating them by comma. **Note:** This property is specific to alerts only. This should not be specified for cases
ScenarioClassList	Identifies a list of Scenario Class for which the alerts are to be purged. You may specify more than one Scenario Class by separating them by comma. **Note:** This property is specific to alerts only. This should not be specified for cases

Table 51. Alert and Case Purge Utility Parameters (Continued)

Parameter	Description
`CreateDate`	Identifies the dates to be considered for purging the alerts or cases by their creation date. The date range may be provided in terms of Start Date or End Date, or the Age of the Alert or Case calculated from the current day/month/year. ● **StartDate**: Identifies the date from when the alerts/cases are to be considered for purging. The date should be provided in the format YYYY-MM-DD. ● **EndDate**: Identifies the date up to which the alerts are to be purged. The date should be provided in the format YYYY-MM-DD ● **Age**: Identifies the age of the Alert/Case to be purged relative to the current date/month/year. Acceptable values for this parameter constitutes a non-negative number followed by D (Days), M (Months) or Y (Years). If we specify age of a record is 1 Day means it should complete 1 day in the database. That is from current day to yesterday. The example below gives more details: (Assume Current date: 21 NOV 2012) Case1: (i) if age = 1Y: Date range would be considered: 21 NOV 2012 to 21 NOV 2011 (includes both days) (ii) if age = 5Y: Date range would be considered: 21 NOV 2012 to 21 NOV 2007 (includes both days) Case2: (i) if age = 1M: Date range would be considered: 21 NOV 2012 to 21 OCT 2012 (includes both days) (ii) if age = 5M: Date range would be considered: 21 NOV 2012 to 21 JUN 2012 (includes both days) Case3: (i) if age = 1D: Date range would be considered: 21 NOV 2012 to 20 NOV 2012 (includes both days) (ii) if age = 5D: Date range would be considered: 21 NOV 2012 to 16 NOV 2012 (includes both days) (iii) if age = 0D: Date range would be considered: 21 NOV 2012 to 21 NOV 2012 (that is, current date only) **Note:** If only EndDate is specified, utility would consider it as on or before that date, in case of only StartDate being provided, utility would consider it as on or after that date. In-case both dates are specified utility would consider both the dates and the dates in between them.
`BatchId`	Identifies the list of Batch IDs for which the alerts should be purged. **Note:** This property is specific to alerts only. This should not be specified for cases.
`DomainCode`	Identifies the list of domains for which the alerts should be purged. Acceptable values include: ● MTS ● TST ● PFM ● NVZ **Note:** This property is specific to alerts only. This should not be specified for cases.

Table 51. Alert and Case Purge Utility Parameters (Continued)

Parameter	Description
`LastActionDate`	Identifies the dates to be considered for purging the alerts and cases by he date on which last action was taken on them. The date range may be provided in terms of Start Date or End Date, or the Age of the Alert or Case calculated from the current day/month/year. ● **StartDate**: Identifies the date from when the alerts/cases are to be considered for purging.The date should be provided in the format YYYY-MM-DD ● **EndDate**: Identifies the date up to which the alerts are to be purged. The date should be provided in the format YYYY-MM-DD ● **Age**: Identifies the age of the Alert or Case to be purged relative to the current date/month/year. Acceptable values for this parameter constitutes a non-negative number followed by D (Days), M (Months) or Y (Years). If we specify age of a record is 1 Day means it should complete 1 day in the database. That is from current day to yesterday. The example below gives more details: (Assume Current date: 21 NOV 2012) Case1: (i) if age = 1Y: Date range would be considered: 21 NOV 2012 to 21 NOV 2011 (includes both days) (ii) if age = 5Y: Date range would be considered: 21 NOV 2012 to 21 NOV 2007 (includes both days) Case2: (i) if age = 1M: Date range would be considered: 21 NOV 2012 to 21 OCT 2012 (includes both days) (ii) if age = 5M: Date range would be considered: 21 NOV 2012 to 21 JUN 2012 (includes both days) Case3: (i) if age = 1D: Date range would be considered: 21 NOV 2012 to 20 NOV 2012 (includes both days) (ii) if age = 5D: Date range would be considered: 21 NOV 2012 to 16 NOV 2012 (includes both days) (iii) if age = 0D: Date range would be considered: 21 NOV 2012 to 21 NOV 2012 (that is, current date only) **Note:** If only EndDate is specified, utility would consider it as on or before that date, in case of only StartDate being provided, utility would consider it as on or after that date. If both dates are specified utility would consider both the dates and the dates in between them.
`Status`	Identifies a list of Status Codes against which the Alert or Case should be purged. You may specify more than one Status Code by separating them by `comma`.
`JobIds`	Identifies the list of Job IDs for which the alerts should be purged. You may specify more than one Job ID by separating them by `comma`. **Note:** This property is specific to alerts only. This should not be specified for cases.
`ThresholdSetIds`	Identifies the list of Threshold Set IDs for which the alerts should be purged. You may specify more than one Threshold Set ID by separating them by `comma`. **Note:** This property is specific to alerts only. This should not be specified for cases.

Executing the Alert and Case Purge Utility

To execute the Alert and Case Purge Utility, follow these steps:

1. Verify that the Behavior Detection database is operational:

   tnsping `<database instance name>`

2. Verify that the `<OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/database/db_tools/bin
   ```

4. Start the alert and case purge shell script:

   ```
   run_alert_purge.sh -purge
   ```

   Executing this command sets the environment classpath and starts the utility. You may also pass command line arguments to the utility, and execute the utility in any of the following ways:

- You may pass a list of purge rules (as configured in `PurgeRules.xml` file) separated by a comma (,) following the convention of `alert_rule_<i0>` for alert-related rules and `case_rule_<i0>` for case-related rules; here `i0` is an integer representing the corresponding rule number in the `purgeRules.xml` file.

  ```
  ./run_alert_purge.sh  -purge  alert_rule_<i0>,alert_rule_<i1>,case_rule_<i2>….
  ```

- You may instruct the utility not to purge any alerts, but only cases, and vice-versa. If the value passed is 'alert=N' the utility considers this as no to purge alerts

  ```
  ./run_alert_purge.sh  -purge  alert=N
  ```

  If the value passed is `'case=N'` the utility considers this as no to purge cases

  ```
  ./run_alert_purge.sh  -purge  case=N
  ```

  You may instruct the utility only to simulate the purge process and not purge the alerts and cases by passing a command line parameter '`test=Y`'. In this case, the utility considers this as running in test mode and generates the report of alerts and cases that would have purged.

  ```
  ./run_alert_purge.sh  -purge  test=Y
  ```

  You can provide all these parameters or a combination of these parameters irrespective of order, once at a time, to the utility as shown in the example below:

  ```
  ./run_alert_purge.sh   -purge  case=N alert_rule_<i0>,alert_rule<i1> test=Y
  ```

Note: If the utility is executed without any command line arguments, the utility considers purging the alerts and cases as configured in the `install.cfg` file.

Processing for Purging

The process for purging is as follows:

1. Once you execute the `run_alert_purge.sh` script, the Alert and Case Purge Utility generates a listing of actions, matches, and alerts or cases that it must purge according to the rules specified at the command line, or the default rule set configured in the `install.cfg`file.

2. After the script is executed, the actions, alerts, and cases are recorded in the `<OFSAAI Installed Directory>/database/db_tools/logs/purge.log` file.

Note: 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`.

3. 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.
4. The utility captures purging results and any errors in the `purge.log` and a report (having the naming convention `Purge_<YYYYMMDD.HH.MM.SS>.txt`) files.

Note: The Alert and Case Purge Utility purges 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 and Case 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 log information about the interruption in the `<OFSAAI Installed Directory>/database/db_tools/logs/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 And Case Purge Processes

This section includes examples of the Purge Alerts process based on input parameters. These example patterns are also applicable for filtering cases.

Example 1

If user specifies only one rule 'xyz' for purging alerts and assume it as follows:

```
<Alert>
………. .
    <Rule id="xyz">
      <IdentifierList>3775,3731,3669,3663</IdentifierList>
<Status>CL</Status>
</Rule>
……. .
</Alert>
```

The utility filters in the existing alerts for IDs 3775,3731,3669,3663 and* status having Closed (CL).

Here and* specifies the logical and operation specified by sql.

In this case, the alert has closed status among the existing alert IDs of (3775, 3731, 3669, and 3663).

```
<Alert>
```

```
………. 
<Rule id="xyz">
<IdentifierList>3775,3731,3669,3663</IdentifierList>
<Status>CL</Status>
<ScenarioIdList>114697002, 114690106</ScenarioIdList>
<JobIds>456789</JobIds>
</Rule>
……….. 
</Alert>
```

The utility filters in the existing alerts for IDs 3775,3731,3669,3663 and* having status Closed (CL) and* having Scenario IDs 114697002,114690106 and having Job Id 456789.

Example 2

If user specifies multiple rules for purging:

```
<Alert>
………. 
<Rule id="pqr">
<IdentifierList>3775, 3731,3669,3663</IdentifierList>
<Status>CL</Status>
<JobIds>456789</JobIds>
</Rule>
<Rule id="xyz">
<ScenarioIdList>114697002,114690106</ScenarioIdList>
<CreateDate>
<StartDate>2011-05-25</StartDate>
<EndDate>2011-05-29</EndDate>
</CreateDate>
</Rule>
……….. 
</Alert>
```

The utility prepares a query to filter alerts so that rule 'pqr' (fetches alerts as per the single rule de-scribed above) or* rule 'xyz' (fetches alerts as per the single rule described above) or*... That is, union of the alerts from all the rules would be filtered.

Here or* specifies the logical or operation specified by sql.

Managing Batch Control Utility

The Batch Control Utility enables you to manage and record the beginning and ending of a 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 `<OFSAAI Installed Directory>/database/db_tools/lib` and UNIX script files that reside in `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` (refer to *Install Configuration* on page 122 for more information).

This section covers the following topics:

- 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 `<OFSAAI Installed Directory>/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 management 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 , such as 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 52 provides the directory structure for the Batch Control Utility, in `<OFSAAI Installed Directory>/database/db_tools/`:

Table 52. 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 , such as `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 `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg`. For more information about logging in these configuration files, refer to *Managing Common Resources for Batch Processing Utilities* on page 122, and Appendix A, *Logging,* on page 245, 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.

You can use the Batch Control Utility to run the following 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 , such as 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.

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
- Obtaining a Batch Name

Configuring the Batch Control Utility

To configure the batch control utility, follow these steps:

1. Navigate to the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file.This file contains common configuration information that Batch Control and other utilities require for processing (see Figure 35).
2. Use the following sample section from the `install.cfg` file to input 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.
```

Figure 42. 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 61 on page 164, 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

OFSBD 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. The DLY processing batch with ALL as the source origin is reserved for instances where one batch load is required, ignoring source systems. If you wish to associate specific source systems to DLY, then the DLY/ALL record must be deleted from the `KDD_PRCSNG_BATCH_SRC` table.

The following table provides the contents of the `KDD_PRCSNG_BATCH` table.

Table 53. `KDD_PRCSNG_BATCH` Table Contents

Column Name	Description
`PRCSNG_BATCH_NM`	Name of the batch , such as DLY.
`PRCSNG_BATCH_DSPLY_NM`	Readable name for the batch, such as Daily.
`PRCSNG_ORDER`	Relative order of a batch run within processing.

Table 53. KDD_PRCSNG_BATCH Table Contents

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.
PRCSNG_BATCH_NM	Description of this processing 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 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 54.

Table 54. 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 55 describes.

Table 55. 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	MORN

In this configuration, run the Calendar Manager Utility only during the MORN batch. Refer to *Managing Calendar Manager Utility.* on page 162, for more information. You can run the Data Retention Manager either in 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 *Managing Data Retention Manager*, for more information.

Multiple Countries

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

Table 56. KDD_PRCSNG_BATCH_SRC FSDM Columns

Column	Data Type	Null	Primary Key	Default Value
PRCSNG_BATCH_NM	VARCHAR2(20)	Not Null	Yes	DLY To load only the US source for a batch, for example, Batch1, another record, Batch1, needs to be added.
SRC_ORIGIN	VARCHAR2(3)	Not Null	Yes	ALL To load only the US source for a batch, for example, Batch1, another record, US, needs to be added.
SRC_DESC	VARCHAR2(255)	Null	No	Productized Daily Processing Batch for all Source Systems

If you want to load only the US source for a batch, for example, Batch1, then another record, US Source System Load, needs to be added.

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 57 describes.

Table 57. 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 `<OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/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.

Note: 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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/categories.cfg` file. Refer to *"Configuring Console Output*, 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 57).
2. The utility determines whether a batch is running by querying the `KDD_PRCSNG_BATCH_CONTROL` table. The following table describes the `KDD_PRCSNG_BATCH_CONTROL` table.

Table 58. `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.

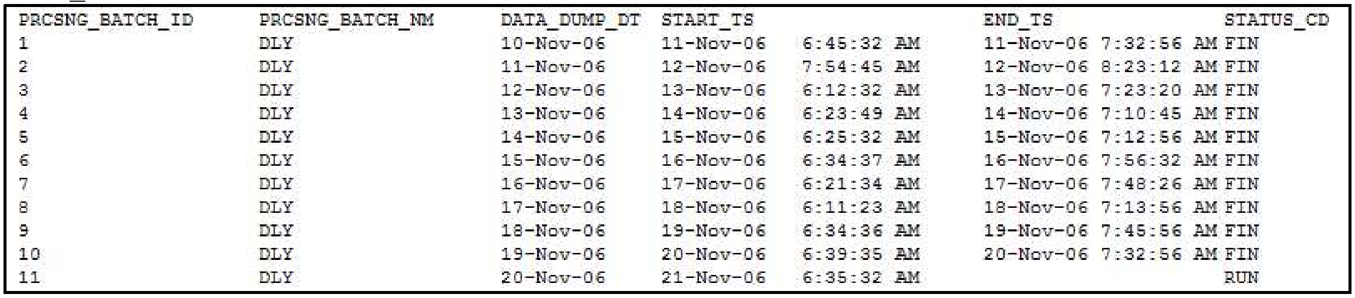
   The following table describes the `KDD_PRCSNG_BATCH_HIST` table.

Table 59. `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 `<OFSAAI Installed Directory>/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 43. 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   6:45:32 AM    11-Nov-06 7:32:56 AM FIN
2                  DLY                11-Nov-06     12-Nov-06   7:54:45 AM    12-Nov-06 8:23:12 AM FIN
3                  DLY                12-Nov-06     13-Nov-06   6:12:32 AM    13-Nov-06 7:23:20 AM FIN
4                  DLY                13-Nov-06     14-Nov-06   6:23:49 AM    14-Nov-06 7:10:45 AM FIN
5                  DLY                14-Nov-06     15-Nov-06   6:25:32 AM    15-Nov-06 7:12:56 AM FIN
6                  DLY                15-Nov-06     16-Nov-06   6:34:37 AM    16-Nov-06 7:56:32 AM FIN
7                  DLY                16-Nov-06     17-Nov-06   6:21:34 AM    17-Nov-06 7:48:26 AM FIN
8                  DLY                17-Nov-06     18-Nov-06   6:11:23 AM    18-Nov-06 7:13:56 AM FIN
9                  DLY                18-Nov-06     19-Nov-06   6:34:36 AM    19-Nov-06 7:45:56 AM FIN
10                 DLY                19-Nov-06     20-Nov-06   6:39:35 AM    20-Nov-06 7:32:56 AM FIN
11                 DLY                20-Nov-06     21-Nov-06   6:35:32 AM                         RUN
```

Figure 43. 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 53).To stop a batch process manually, follow these steps:

1. Verify that the Behavior Detection database is operational.

   tnsping <database instance name>

2. Verify that the <OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/database/db_tools/bin
   ```

4. Start the batch shell script:

   ```
   end_mantas_batch.sh
   ```

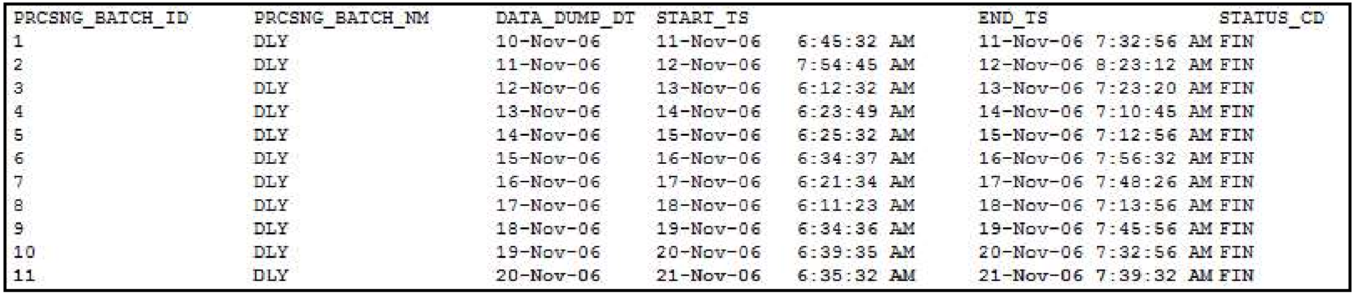
   If you enter an invalid batch name, the utility terminates and logs a message that describes the error. The error message appears on the console only if you have output to the console enabled in the <OFSAAI Installed Directory>/database/db_tools/mantas_cfg/categories.cfg configuration file.

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 58 on page 159).

2. Records information about the batch in the KDD_PRCSNG_BATCH_ HIST table (refer to Table 59 on page 159). This table contains a history of all batches that appear by start date and end date. Figure 44 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	6:45:32 AM	11-Nov-06 7:32:56 AM	FIN
2	DLY	11-Nov-06	12-Nov-06	7:54:45 AM	12-Nov-06 8:23:12 AM	FIN
3	DLY	12-Nov-06	13-Nov-06	6:12:32 AM	13-Nov-06 7:23:20 AM	FIN
4	DLY	13-Nov-06	14-Nov-06	6:23:49 AM	14-Nov-06 7:10:45 AM	FIN
5	DLY	14-Nov-06	15-Nov-06	6:25:32 AM	15-Nov-06 7:12:56 AM	FIN
6	DLY	15-Nov-06	16-Nov-06	6:34:37 AM	16-Nov-06 7:56:32 AM	FIN
7	DLY	16-Nov-06	17-Nov-06	6:21:34 AM	17-Nov-06 7:48:26 AM	FIN
8	DLY	17-Nov-06	18-Nov-06	6:11:23 AM	18-Nov-06 7:13:56 AM	FIN
9	DLY	18-Nov-06	19-Nov-06	6:34:36 AM	19-Nov-06 7:45:56 AM	FIN
10	DLY	19-Nov-06	20-Nov-06	6:39:35 AM	20-Nov-06 7:32:56 AM	FIN
11	DLY	20-Nov-06	21-Nov-06	6:35:32 AM	21-Nov-06 7:39:32 AM	FIN

Figure 44. Sample KDD_PRCSNG_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 <OFSAAI Installed Directory>/database/db_tools/logs/batch_control.log file, stating that the end batch process has begun.

Identifying a Running Batch Process

Caution: At times, you may must 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 must identify one or more by name. 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 identify a running batch process, follow these steps:

1. Access the directory where the shell script resides:

   ```
   cd <OFSAAI Installed Directory>/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 138, for more information).

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 58 on page 159).

The utility returns the batch name to standard output.

Managing 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 OFSBD system calendars. The <OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg configuration file contains modifiable inputs that you use to run the utility (refer to *Install Configuration* for more information).

This section contains the following topics:

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

Directory Structure

The following table provides the directory structure for the Calendar Manager Utility in <OFSAAI Installed Directory>/database/db_tools/.

Table 60. 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 , such as install.cfg and categories.cfg, in which you can configure properties and logging attributes.
logs/	Keeps the calendar_manager.log log file that the utility generates during execution.

Logs

As the utility updates the calendars in the OFSBD system, it generates a log that it enters in the <OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg and categories.cfg. For more information about logging in these configuration files, refer to *Managing Common Resources for Batch Processing Utilities* on page 122, and Appendix A, *Logging,* on page 245, for more information.

Calendar Information

The Calendar Manager Utility obtains all holidays and weekly off-days for loading into the OFSBD calendars by retrieving information from the KDD_CAL_HOLIDAY and KDD_CAL_WKLY_OFF tables (refer to Table 47 and Table 48). These tables contain calendar information that an Oracle client has provided regarding observed holidays and non-business days.

Using the Calendar Manager Utility

The Calendar Manager Utility runs as part of automated processing that a job scheduling tool such as Maestro or Unicenter AutoSys controls. The utility runs through a shell script, using values in parameters that the install.cfg file contains. The utility then populates the KDD_CAL database table with relevant OFSBD 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 <OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg file contains common configuration information that Calendar Manager and other utilities require for processing (refer to Figure 35). 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 must be more than 10 days. This is used when calculating projected settlement dates during data management.

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

You can 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.

Starting 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 `<OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/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 , such as *`20161130`* for November 30, 2016. 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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/categories.cfg` configuration file. refer to *Configuring Console Output,* on page 138, for more information.

Updating the `KDD_CAL` Table

The Calendar Manager Utility retrieves information that it needs for updating OFSBD business calendars from the `KDD_CAL_HOLIDAY` and `KDD_CAL_WKLY_OFF` database tables. It then populates the `KDD_CAL` table accordingly. 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`.

The following table provides the contents of the `KDD_CAL` table.

Table 61. `KDD_CAL` Table Contents

Column Name	Description
CLNDR_NM	Specific calendar name.
CLNDR_DT	Date in the range between the lookback and lookforward periods.

Table 61. KDD_CAL Table Contents (Continued)

Column Name	Description
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.
DAY_OF_WK	Value that represents the day of the week: Sunday=1, Monday=2, Tuesday=3, ... Saturday=7.
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.
BUS_DAY_TYPE_ CD	Indicates the type of business day: ● N = Normal ● C = Closed ● S = Shortened
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.

Table 61. KDD_CAL Table Contents (Continued)

Column Name	Description
SESSN_TM_OFFST_TX	Indicates the timezone offset for SESSN_OPN_TM and SESSN_CLS_TM. The format is HH:MM.
QRTR_BNDRY_CD	Quarter's start day (SD) and end day (ED). ● If this is the last business day for this calendar name for the quarter (that is, next business day in a different quarter), set ED to <y>, where y is a numeric counter with the start/end of the quarter that the provided date is in = 0. ● If it is the first business day for this calendar name for this quarter (that is, previous business day is in a different quarter), set SD to <y>. Quarters before the provided date increment the counter, and quarters 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_PRCSNG_BATCH_CONTROL.DATA_DUMP_DT field.

Managing 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 34 on page 121). 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

The following table provides the directory structure for the Data Retention Manager.

Table 62. 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 , such as `install.cfg` and `categories.cfg`, in which you can configure properties and logging attributes.
`logs/`	File `<OFSAAI Installed Directory>/database/db_tools/logs/DRM_Utility.log` that the utility generates during execution.

Logs

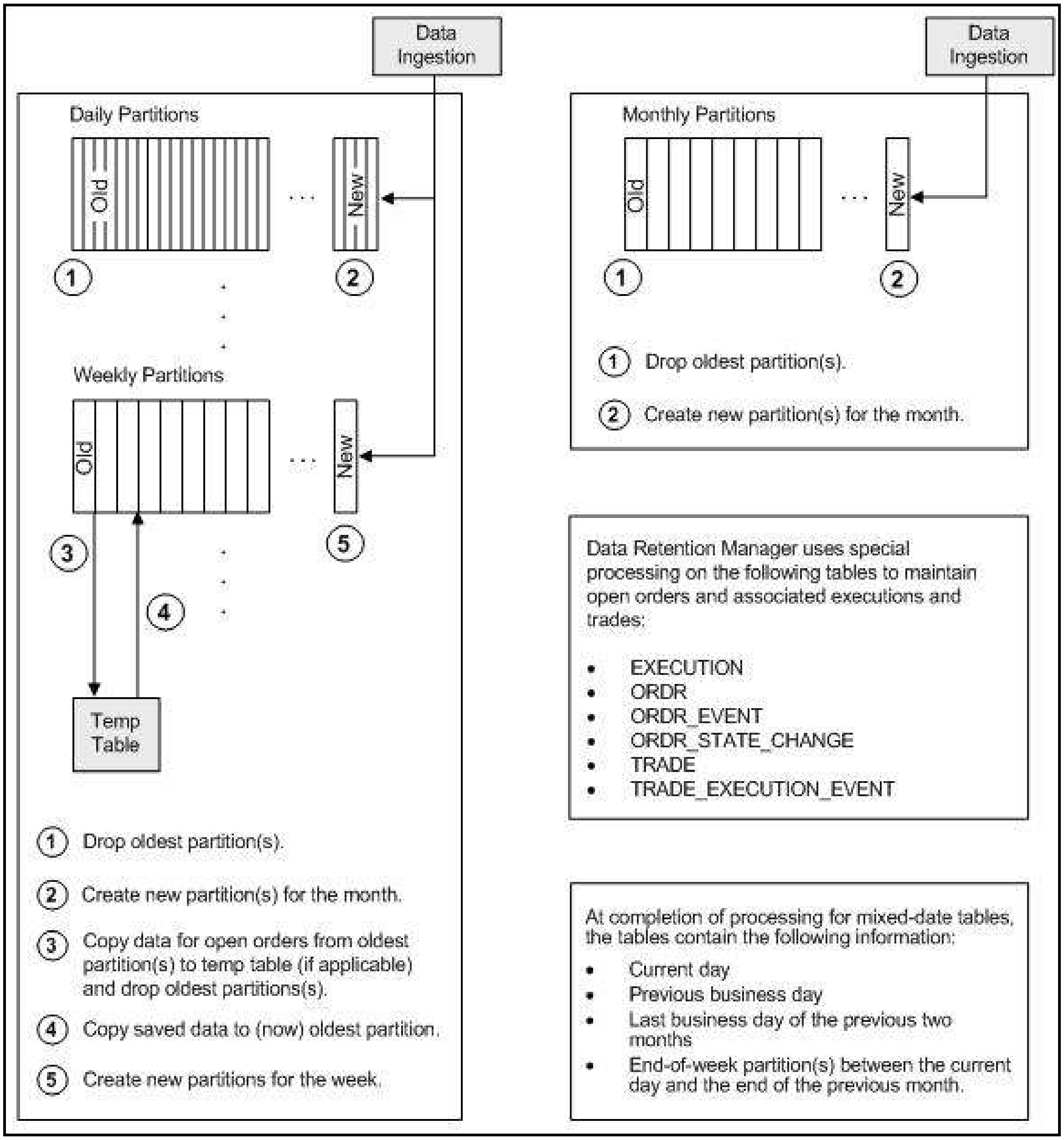
Oracle stored procedures implement Data Retention Manager and conducts some logging on the database server. A configuration parameter in the `install.cfg` file controls the path to which you store the logs on the database server.

As the Data Retention Manager performs partitioning and indexing activities, it generates a log that it enters in the `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg`. For more information about logging in these configuration files, refer to *Managing Common Resources for Batch Processing Utilities,* on page 122, and Appendix A, *Logging,* on page 245, for more information.

Processing Flow

Figure 45 illustrates the Data Retention Manager's process flow for daily, weekly, and monthly partitioning. Based on a table's retention period, the utility drops the oldest partition and then adds a new partition.



1. Drop oldest partition(s).
2. Create new partition(s) for the month.
3. Copy data for open orders from oldest partition(s) to temp table (if applicable) and drop oldest partitions(s).
4. Copy saved data to (now) oldest partition.
5. Create new partitions for the week.

1. Drop oldest partition(s).
2. Create new partition(s) for the month.

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

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

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

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

Figure 45. 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 how to configure and execute 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

To configure the Data Retention Manager, follow these steps:

1. Navigate to the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file.This file contains common configuration information that Data Retention Manager and other utilities require for processing
2. Use the sample install.cfg file in Figure 35 to do a configuration.

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 , such as 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
```

Figure 46. install.cfg Data Retention Manager Configuration

This example shows default values that the system uses only when calling the utility with no command line parameters. The following table describes these parameters.

Table 63. 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 (Atomic 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

Before you execute the Data Retention Manager, ensure that users are not working on the system. To avoid conflicts, Oracle recommends that you use this utility as part of the end-of-day activities.

The Data Retention Manager should be executed nightly for Daily partitioned and Mixed-date partitioned tables, after the calendar has been set for the next business day. For weekly and monthly partitioned tables, the Data Retention Manager should be executed prior to the end of the current processing period.

Note: Oracle recommends running the Data Retention Manager on Thursday or Friday for weekly partitioned tables and on or about the 23rd of each month for monthly partitioned tables.

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

Running the Data Retention Manager

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

1. Verify that the Behavior Detection database is operational:

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

2. Verify that the `<OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/database/db_tools/bin
   ```

4. Start the batch shell script with the parameters in Table 63:

   ```
   run_drm_utility.sh <drm_operation> <drm_partition_type> <drm_owner> <drm_object_name>
   <drm_weekly_proc_fl>
   ```

The following are examples of running the script:

- To run the utility for all daily tables in the ATOMIC 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

To create partition names, use the formats in the following table.

Table 64. 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.

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

Maintaining Partitions

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

To maintain Partitions, follow these steps:

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

Managing Daily Partitioning Alternative

The Data Retention Manager also enables you to build five daily partitions on a weekly basis. To build partitions, follow these steps:

1. Execute the `run_drm_utility.sh` shell script
2. Set the `drm_weekly_proc_flg` parameter to Y.For more information, refer to Table 63.

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

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

Note: You must set the `WEEKLY_ADD_FL` parameter in the `KDD_DR_MAINT_OPRTN` table to `Y` so that the procedure works correctly. For more information about this parameter, refer to Table 65 on page 174, for more information.

Partition Structures

The structures of business data partitions and market data partitions differ in the following ways:

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

  You can use the Calendar Manager Utility to configure a business calendar as desired. For more information about this utility, refer to *Managing Calendar Manager Utility.* on page 162, for more information.

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

Recommended Partition Maintenance

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

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

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

Note: If you ingest data belonging to a date less than the current date, you should run the DRM utility till current date. This avoids the error *Partition Not Found* while accessing trade records in Trade Blotter UI.

Managing Alternative Monthly Partition

As part of an alternative method of monthly partition maintenance, you can either add or drop a monthly database partition. as described in the following section:

Adding a Monthly Database Partition

To add a monthly partition, run the utility's shell script as follows (refer to Table 63 for parameters):

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

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

Dropping a Monthly Database Partition

To drop a monthly partition, run the utility's shell script as follows (refer to Table 63 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 the following work tables during database partitioning:

- 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. The following table provides these details.

Table 65. `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 65. 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.
NEXT_PARTN_DATE	Indicates starting date of the next partition that may get created, based on the current partitioned date.

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. The following table provides these details.

Table 66. 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. The following table provides these details.

Table 67. **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 61 for contents of the KDD_CAL table.

Database Statistics Management

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 <OFSAAI Installed Directory>/database/db_tools/mantas_cfg/categories.cfg file.

Using Database Statistics Management

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

- analyze_mantas.sh <analysis_type> [TABLE_NAME]

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 related tables).
- 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) that the Behavior Detection UI uses to display alerts. It is recommended that you do not modify this table. The Behavior Detection Historical Data Copy procedures uses this table to archive alert-related data.
- DLY_PRE_HDC: Use this value to update statistics of the Mantas related tables that contain the alert-related information. It is recommended that you do not modify this table. The Behavior Detection Historical Data Copy procedures uses this table to archive alert-related data.

- `DLY_POST_LINK`: Use this value to update statistics of the Mantas related 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 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 now based on `DBMS_STATS.AUTO_SAMPLE_SIZE`.

- `IMP_STATS`: Imports statistics that were previously calculated. When running an ALL analysis, the system exports statistics for the tables for later use.

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 `<OFSAAI Installed Directory>/ database/db_tools/mantas_cfg/install.cfg` file. The `install.cfg` file also contains the following properties:

- `schema.mantas.owner`

The system derives schema name from this property.

For the ATOMIC 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.

Managing Flag Duplicate Alerts Utility

This section covers the following topics:

- Using Flag Duplicate Alerts Utility
- Executing 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 , such as 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 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 and with different focuses , such as if one alert is an ID and the other is a Name. The utility flags these alerts as possible duplicates and adds a system comment which can be viewed on the Audit tab of the alert (each alert cross-references the other). For example:

```
Possible duplicate of alert xxxxx.
```

Executing 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:

```
<OFSAAI Installed Directory>/database/db_tools/bin/flag_duplicate_alerts.sh
```

The system writes log information for this process to the following location:

```
<OFSAAI Installed Directory>/database/db_tools/logs/run_stored_procedure.log
```

Managing 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:

- **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 the data transfer during promotion of an alert to a case or linking of an alert to a case is Unsuccessful, then a notification is generated to the user who is taking the action, the owner of the alert, and the owner of the case, and then assigned to the user of the case.

Batch Based

These notifications are the 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.

These 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 notifications which are relevant to them.

Note: You can set the near due date and display of notification parameters from the Manage Parameters screen. (Refer to the *Configuration Guide* for more information).

Managing Push E-mail Notifications

Alert Management provides the Push E-mail Notification utility to send e-mail to users about activity that is pending for them or about activity that has occurred on their alerts. The user sets a preference on the Preferences page to indicate whether they wish to receive notification messages or not. The system is delivered with two notification sets:

- Activity: This notification tells users of any actions that have occurred on alerts that they own since the last time this notification job was run. This notification also identifies any alerts that have been assigned to the user that the user has not yet opened.
- OverDue: This notification identifies alerts that are either past their due date or are nearing their due date (within 4 days).

Notifications can be run individually, in groups, or all at once. Notification jobs can be run at any time of the day as is appropriate for the information that is to be provided. For example, it is appropriate to run the OverDue notification at the beginning of each day, whereas it may be appropriate to run the Activity notification multiple times per day. If there is no information to provide to a user, no e-mail is sent. If sections of a notification contain no information, that section is suppressed , such as the Reassignment section may be populated, but there may not be a section for Actions taken on your alerts.

For a user to receive notification, the user must have an e-mail address identified through their user configuration.

Using Push E-mail Notification

To run this utility, follow these steps:

1. `Navigate to <OFSAAI Installed Directory>`/database/db_tools/bin

2. Run the following shell script:
   `run_push_email.ksh [notification list]`

Note: If a notification list value is not provided in the command-line parameter, the system runs all notifications. You can provide one or more notifications as command line arguments. The notification names are case-sensitive.

The script runs a java class that attaches to the database using the user that the `utils.database.username` property identifies in the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file. The java process then runs the queries associated with the desired notifications and sends e-mail to the users. By default, the system sends e-mails using unauthenticated SMTP, however it also supports authenticated SMTP, authenticated or unauthenticated SMTPS and Microsoft Exchange.

Note: When the notification runs, the date and timestamp for the notification is stored in the `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/notification.config file.`

Warning: The `notification.config` file must not be edited.

To avoid corruption of this file, do not run two instances of the `run_push_email.ksh` script at once.

The script returns a status code to indicate whether it was successful. The following table lists the status codes returned.

Table 68. Return Codes for `run_push_email.ksh` script

Return Code	Meaning
0	Success
1	The process failed, check the log for reasons.
100-200	The process succeeded, but not all e-mails were delivered the percent not delivered is calculated using (return code – 100).

Configuring Push E-mail Notification

The following files are used to configure Push E-mail notification:

- Configuration for connectivity and mail format parameters are modified in:
  `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg`
- The definition of notification types and the sections of each notification (including headers, footers and disclaimers) is configured in:
  `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/NotificationDetails.xml`
- The queries that are run against the database for notification are configured in:
  `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/etc/xml/QBD_Notification.xml`

The following sections provide configuration guidance for each of these files.

Configuring General Notification Properties

Table 69 identifies the configurable parameters associated to Push E-mail Notification in the database tools `install.cfg` file.

Note: The Password Manager Utility should be used to set the `email.smtp.password` and `email.exchange.password` properties in the `install.cfg` file. These properties should never be modified directly in the file. Run the following commands and enter the appropriate passwords:

```
<OFSAAI Installed Directory>/changePasswords.sh email.smtp.password
```

```
<OFSAAI Installed Directory>/changePasswords.sh email.exchange.password
```

Table 69. Push E-mail Notification Configurable parameters

Property	Description	Sample Value
`email.type`	The type of e-mail connection to use. The valid values are `smtp`, `smtps` and `exchange`. This defaults to `smtp`.	smtp
`notification.threads`	The number of threads to use for sending out notification e-mails. Default value is 4.	2
`utils.database.max_connectios`	The maximum number of database connections to open to run notification queries in parallel. Default value is 4.	2
`email.from`	The e-mail address shows as the From address on the notification e-mail. This value is only used for SMTP and SMTPS mail. If it is omitted, then the e-mail address associated with the Unix or Linux user running the process will appear as the From address.	ofsbd@yourdomain.com
`email.smtp.host`	The host name for SMTP or SMTPS server.	mailhost.yourdomain.com
`email.smtp.port`	The port on which the SMTP or SMTPS server listens. For SMTP, this is 25, for SMTPS, this is typically 465.	25
`email.smtp.auth`	To connect to the SMTP or SMTPS server using a username/password, set this value to `true`. To connect unauthenticated, set to `false`.	true
`email.smtp.user`	The username for authenticated connections.	User
`email.smtp.password`	The password for authenticated connections. This is set by the Password Manager Utility.	
`email.smtp.useHTML`	If set to `true`, e-mail is sent with an HTML body. If set to `false`, e-mail is sent in plain text only. This defaults to `true`.	true
`email.exchange.server`	If using Exchange, this is your Exchange server.	webmail.yourdomain.com
`email.exchange.domain`	Domain for the user.	YourDomain
`email.exchange.user`	Username to connect to Exchange.	ofsbd

Table 69. Push E-mail Notification Configurable parameters (Continued)

Property	Description	Sample Value
email.exchange.password	Password to connect to Exchange. This is set by the Password Manager Utility.	
email.exchange.prefix	The prefix used for Exchange. Consult your Exchange administrator for this value. This defaults to `exchange`.	exchange
email.exchange.mailbox	The mailbox for the user. Consult your Exchange administrator for this value.	ofsbd.System
email.exchange.useSSL	To connect using SSL, set this value to `true`.	true
email.exchange.useFBA	To use Form Based Authentication, set this value to `true`. This value defaults to `true`.	true
email.exchange.useNTLM	To use NTLM authentication, set this value to `true`. This value defaults to `false`.	false
email.exchange.draftsfolder	The name of the Drafts folder within the mailbox. This defaults to `drafts`.	drafts
email.exchange.useHTML	If set to `true`, e-mail is sent with an HTML body. If set to `false`, e-mail is sent in plain text only. This defaults to `true`.	true

The connectivity to Microsoft Exchange is implemented using a third party product called Java Exchange Connector (JEC). Oracle does not provide a copy of this product, it must be purchased separately. After you have purchased JEC, place the jec.jar in the `<OFSAAI Installed Directory>/database/db_tools/lib` directory, and copy your jeclicense file into `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg`.

In addition to the configuration parameters identified in Table 69 above, there are series of configuration parameters that are used to control the formatting of the HTML e-mail messages. These parameters use HTML style syntax to control styles for different sections of the generated e-mail message.

Configuring Notifications

To configure notifications, follow these steps:

1. Navigate to `OFSAAI Installed Directory>/database/db_tools/mantas_cfg/NotificationDetails.xml`. The list of notifications to be delivered are configured in the `NotificationDetails.xml` file.
2. Use the following sample to configure notifications.

```
<NotificationDetails>

    <Disclaimer>This message is for the designated recipient only and may 
contain privileged or confidential information.</Disclaimer>
    <HTMLDisclaimer><![CDATA[This message is for the designated recipient only 
and may contain privileged or <B>confidential</B> 
information.]]></HTMLDisclaimer>

    <Notification name="Activity" userQueryDef="AllActiveUsers">
        <Subject>Mantas Activity Notification</Subject>
        <Header>*** This message was system-generated. Do not reply to this 
message. ***</Header>
        <HTMLHeader><![CDATA[*** This message was system-generated. Do not reply 
to this message. ***]]></HTMLHeader>
        <Footer>*** This message was system-generated. Do not reply to this 
message. ***</Footer>
        <HTMLFooter><![CDATA[*** This message was system-generated. Do not reply 
to this message. ***]]></HTMLFooter>

        <Section queryDef="ReassignedAlerts">
            <Title>Reassigned Alerts:</Title>
            <HTMLTitle><![CDATA[Reassigned Alerts:<BR>]]></HTMLTitle>
            <Message>The following alerts have been recently assigned to 
you:</Message>
            <HTMLMessage><![CDATA[<BR>The following alerts have been recently 
assigned to you:]]></HTMLMessage>
            <Column text="Alert ID" key="REVIEW_ID"/>
            <Column text="Assigned By" key="ASSIGNED_BY"/>
            <Column text="Assigned On" key="ASSIGNED_DATE" format="datetime"/>
        </Section>
   </Notification>
</NotificationDetails>
```

Figure 47. Sample `NotificationDetails.xml` file

The file starts with disclaimer configuration. The disclaimer is included in all e-mail sent by the system. The `<DisclaimerHTML>` tag permits use of HTML within the disclaimer section for emphasis, embedded links, etc. In general, where there is a tag and a tag with the same name but HTML added, the message will include the appropriate element based on whether the message is being sent in Text or HTML mode. If a message is sent in HTML mode, and there is no HTML tag, the basic element is used for that element.

Each Notification begins with a name and identifies the queryDef used to select the candidate users. A queryDef is a configurable query, and will be discussed in detail the next section. The queryDef `AllActiveUsers` selects all active users who have an e-mail address configured and have not specified in their user preferences that they do not want notifications. A notification has a Subject, which is the subject of the delivered e-mail and a header and footer. These are used for introductory text of the e-mail message.

After the header and footer, a number of sections are defined. Each section specifies a queryDef to run to find the records that are reflected in the section. The following table contains the additional elements.

Table 70. Additional Elements of `NotificationDetails.xml` file

Element	Description
Title HTMLTitle	The title is a title for the section.
Message HTMLMessage	The Message appears between the title and the table of results.
Column	There is a Column element for each column that your query displays. The column element has a text attribute, which is the column header in the rendered table, and a key, which refers to a column in the query results that is displayed in this section.

The queries that are run for identifying users and for populating each section of the notification are configured in queryDefs. The queryDefs for the default notifications are configured in `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/etc/xml/QBD_Notification.xml.`

Configuring Notification Queries

To configure notification queries, follow these steps:

1. Navigate to `OFSAAI Installed Directory>/database/db_tools/mantas_cfg/etc/xml/QBD_Notification.xml.` The queryDefs for the default notifications are configured in the `QBD_Notification.xml` file.
2. Use the following sample to configure notification queries:

Note: The Notification process reads all QBD files in this directory, so custom notifications can be placed in the existing file or in a new file , such as `QBD_CustomNotification.xml`.

Figure 48 shows the sample structure of the `QBD_CustomNotification.xml` file.

```
<queries>
    <ReassignedAlerts>
        <baseQuery>
            select
              kdd_activity.new_review_owner_id owner_seq_id,
              kdd_activity.review_id,
              old_owner.owner_id assigned_by,
              kdd_activity.start_dt assigned_date
            from
              mantas.kdd_review inner join mantas.kdd_activity
                on kdd_review.review_id = kdd_activity.review_id
                and kdd_activity.new_review_status_cd = 'RA'
              inner join mantas.kdd_review_owner old_owner
                on kdd_activity.creat_id = old_owner.owner_seq_id
            <usingColumns/>
            <groupingColumns/>
        </baseQuery>
        <filterProperties>
            <property name="_filter_MIN_DATE" operator="&gt;="
table="kdd_activity" columnName="start_dt" type="Timestamp"/>
            <property name="_filter_MAX_DATE" operator="&lt;"
table="kdd_activity" columnName="start_dt" type="Timestamp"/>
        </filterProperties>
        <sortProperties>
            <sort name="default">
                <property table="kdd_activity" columnName="start_dt"
direction="ASC" order="1"/>
            </sort>
        </sortProperties>
    </ReassignedAlerts>
<queries>
```

Figure 48. Sample Structure of `QBD_CustomNotification.xml`

Note: This is an example, and may not represent what is in the deployed product.

Each query is defined as an XML element (in this example, `ReassignedAlerts` is the element). The following table lists the other sub-elements of the `QBD_CustomNotification.xml` file.

Table 71. Sub-Elements of the Sample File

Element	Description
`baseQuery`	The base query is where the query is defined. The query can be any query, however it must return a column `OWNER_SEQ_ID` that identifies the owner to whom the notification should be sent. Other columns depend on what is appropriate for the query. The query may not contain a group by or order by clause, it must either end after the `FROM` clause or after the `WHERE` clause. If the query contains any XML-reserved characters, be sure to surround the query with `<[!CDATA[ ]]>`. The base query has two sub-elements defined below.
`usingColumns`	You can either include all of the join conditions in the `FROM` clause, or you can have a using clause appended to the `FROM` clause by identifying columns in this section. If you do use this element, then each column you will include in the `USING` clause is specified as follows: `<column name="OWNER_SEQ_ID"/>`

Table 71. Sub-Elements of the Sample File

Element	Description
groupingColumns	If you wish to have a query that performs a group by, then the GROUP BY clause is specified in this element. Each column that is part of the clause is specified using the same notation specified under usingColumns above.
filterProperties	The filterProperties element allows you to specify filters that are applied to the query. The filters are provided programmatically. The two filters provided in the sample above are the only filters that are accepted for Notification. The filter _filter_MIN_DATE is replaced by the date of the last execution of the notification. The filter _filter_MAX_DATE is replaced by the current system date. When specifying the filter properties, identify the table (or table alias) and column against which the filter is applied.
sortProperties	The sortProperties element allows you to define sorts for the query. Only the sort with the name *default* is used by Notifications. When specifying a sort, identify the table (or table alias) and column for the sort. You can specify more than one sort column (distinguishing them with the order attribute). You can specify either ASC for ascending sorts or DESC for descending sorts.

QueryDefs are used broadly in the Behavior Detection Framework user interface definition. Only the subset of queryDef capabilities that are used by Notification have been addressed in this section.

Logs

The log.category.PUSH_EMAIL_NOTIFICATION.location property in the <OFSAAI Installed Directory>/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:

```
<OFSAAI Installed Directory>/database/db_tools/logs/push_email.log
```

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 `<OFSAAI Installed Directory>/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:

```
<OFSAAI Installed Directory>/database/db_tools/logs/refresh_temp_table.log
```

Using Refreshing Temporary Tables

The BD ATOMIC 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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file.

Populating Temporary Tables for Scenarios

Scenarios typically depend on data management to complete processing. However the following scenarios depend on population of Temp Tables to populate data.

1. (IML/CU) Hidden Relationships
2. (FR/AC) Networks of Accounts, Entities, and Customers
3. (ML/AC) Networks of Accounts, Entities, and Customers
4. (CST/AC) Customers Who Have Experienced a Large Loss Recently
5. (CST/HH) Customers Who Have Experienced a Large Loss Recently

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 these steps:

1. Execute the following refresh temporary table processes (these commands can be run in parallel).

- If you run a scenario with the Include records for active batch parameter = 'N' (All records loaded during lookback period will analyzed regardless of the name of the batch process which means it will include records from other batches in a multi-country installation)

```
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_JRNL TMP_HIDREL_NT_JRNL_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIRE TMP_HIDREL_NT_WIRE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACTAXID TMP_HIDREL_NT_ACTAXID_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACADDR TMP_HIDREL_NT_ACADDR_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACPHONE TMP_HIDREL_NT_ACPHONE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACEMAIL TMP_HIDREL_NT_ACEMAIL_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACPSWRD TMP_HIDREL_NT_ACPSWRD_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_INST TMP_HIDREL_NT_INST_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIREACBENE TMP_HIDREL_NT_WIREACBENE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIREACORIG TMP_HIDREL_NT_WIREACORIG_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACTAXID TMP_HIDREL_NT_CUACTAXID_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACADDR TMP_HIDREL_NT_CUACADDR_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACPHONE TMP_HIDREL_NT_CUACPHONE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACEMAIL TMP_HIDREL_NT_CUACEMAIL_VW
```

-) If you run scenario with parameter Include records for active batch = 'Y' Only records loaded during the lookback period with batch name which is currently active will be analyzed which means it will not include records from other batches in a multi-country installation).

```
<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_JRNL TMP_HIDREL_NT_JRNL_BATCH_VW
```

```
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIRE TMP_HIDREL_NT_WIRE_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACTAXID TMP_HIDREL_NT_ACTAXID_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACADDR TMP_HIDREL_NT_ACADDR_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACPHONE TMP_HIDREL_NT_ACPHONE_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACEMAIL TMP_HIDREL_NT_ACEMAIL_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_ACPSWRD TMP_HIDREL_NT_ACPSWRD_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_INST TMP_HIDREL_NT_INST_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIREACBENE TMP_HIDREL_NT_WIREACBENE_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIREACORIG TMP_HIDREL_NT_WIREACORIG_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACTAXID TMP_HIDREL_NT_CUACTAXID_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACADDR TMP_HIDREL_NT_CUACADDR_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACPHONE TMP_HIDREL_NT_CUACPHONE_ BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACEMAIL TMP_HIDREL_NT_CUACEMAIL_BATCH_ BATCH_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698616.

   - If you ran a scenario where the Include records for active batch parameter = 'N' (All records loaded during lookback period will analyzed regardless name of batch process) then insert the record to KDD_PARAM_BINDING following these steps:

   ```
   insert into KDD_PARAM_BINDING values ('filter_by_batch', 'Link Analysis', 
   <param_set_id>, <true or false>)
   ```

   For example:

   ```
   insert into KDD_PARAM_BINDING values ('filter_by_batch', 'Link Analysis', 
   114698653, 'false')
   ```

   Run the Link Analysis IGN job which has a 'false' value in KDD_PARAM_BINDING

3. Execute the scenario job with appropriate value in parameter Include records for active batch . The product job template ID is 116200024.

ML-NetworkOfAcEn-fAC

To populate the temporary tables for ML-NetworkOfAcEn-fAC scenario, follow these steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel):

- If you run a scenario with parameter Include records for active batch = 'N' (All records loaded during lookback period will analyzed regardless of the name of the batch process which means it will include records from other batches in a multi-country installation)

```
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTADDR TMP_NETACENCU_NT_ACCTADDR_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTEMAIL TMP_NETACENCU_NT_ACCTEMAIL_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTPHONE TMP_NETACENCU_NT_ACCTPHONE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTPSWRD TMP_NETACENCU_NT_ACCTPSWRD_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTTAXID TMP_NETACENCU_NT_ACCTTAXID_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACADDR TMP_NETACENCU_NT_CUACADDR_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACEMAIL TMP_NETACENCU_NT_CUACEMAIL_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACPHONE TMP_NETACENCU_NT_CUACPHONE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACTAXID TMP_NETACENCU_NT_CUACTAXID_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_JRNL TMP_NETACENCU_NT_JRNL_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIREACBENE TMP_NETACENCU_NT_WIREACBENE_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIREACORIG TMP_NETACENCU_NT_WIREACORIG_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIRETRXN TMP_NETACENCU_NT_WIRETRXN_VW
```

- If you run a scenario with parameter Include records for active batch = 'Y' Only records loaded during the lookback period with batch name which is currently active will be analyzed[which means it will not include records from other batches in a multi-country installation)

```
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTADDR TMP_NETACEN_ACCTADDR_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTEMAIL TMP_NETACEN_ACCTEMAIL_BATCH_VW
```

```
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTPHONE TMP_NETACEN_ACCTPHONE_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTPSWRD TMP_NETACEN_ACCTPSWRD_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTTAXID TMP_NETACEN_ACCTTAXID_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACADDR TMP_NETACEN_CUACADDR_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACEMAIL TMP_NETACEN_CUACEMAIL_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACPHONE TMP_NETACEN_CUACPHONE_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACTAXID TMP_NETACEN_CUACTAXID_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_JRNL TMP_NETACEN_JRNL_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIREACBENE TMP_NETACEN_WIREBENE_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIREACORIG TMP_NETACEN_WIREORIG_BATCH_VW
<OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIRETRXN TMP_NETACEN_WIRETRXN_BATCH_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698120.

- If you run a scenario with parameter Include records for active batch = 'N', then insert a record to KDD_PARAM_BINDING using the following instructions:

  ```
  insert into KDD_PARAM_BINDING values ('filter_by_batch', 'Link Analysis',
  <param_set_id>, <true or false>)
  ```

  For Example

  ```
  insert into KDD_PARAM_BINDING values ('filter_by_batch', 'Link Analysis',
  118745109, 'false')
  ```

  Run the Link Analysis IGN job which has a 'false' value in KDD_PARAM_BINDING

- If you run a scenario with parameter Include records for active batch = 'Y' then insert a record to KDD_PARAM_BINDING using the following instructions:

  ```
  insert into KDD_PARAM_BINDING values ('filter_by_batch', 'Link Analysis',
  <param_set_id>, <true or false>)
  ```

  For Example

  ```
  insert into KDD_PARAM_BINDING values ('filter_by_batch', 'Link Analysis',
  118745110, 'true')
  ```

  Run the Link Analysis IGN job which has a 'true' value in KDD_PARAM_BINDING

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.):

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_NT_ACCTADDR TMP_FRNTWRK_NT_ACCTADDR_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_ACCTEMAIL TMP_FRNTWRK_ACCTEMAIL_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_ACCTPHONE TMP_FRNTWRK_ACCTPHONE_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_ACCTPSWRD TMP_FRNTWRK_ACCTPSWRD_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_ACCTTAXID TMP_FRNTWRK_ACCTTAXID_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_CUACADDR TMP_FRNTWRK_CUACADDR_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_CUACEMAIL TMP_FRNTWRK_CUACEMAIL_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_CUACPHONE TMP_FRNTWRK_CUACPHONE_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_CUACTAXID TMP_FRNTWRK_CUACTAXID_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_JRNL TMP_FRNTWRK_JRNL_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_WIREACBENE TMP_FRNTWRK_WIREACBENE_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_WIREACORIG TMP_FRNTWRK_WIREACORIG_VW
   ```

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_FRNTWRK_WIRETRXN TMP_FRNTWRK_WIRETRXN_VW
   ```

2. Execute the link analysis/network generation job. The product job template ID is 118745091.
3. Execute the scenario job. The product job template ID is 117350084.

CST-Losses

To populate the temporary tables for CST-LOSSES scenario, follow these steps:

1. Execute this refresh temporary table process:

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   VWCST_lOSSES_AC_ASM_TMP VWCST_lOSSES_AC_ASM
   ```

2. Execute the scenario job.

CST-UncvrdLongSales-dRBPC

To populate the temporary table UNCVRD_LONG_TRADE_TEMP for CST-UncvrdLongSales-dRBPC scenario, follow these steps:

Note: This should be run after the ingestion is completed, just before the scenario job runs.

1. Execute this to refresh temporary table process:
   `<OFSAAI Installed Directory>/database/db_tools/run_p_uncvrdlongsales_ew.sh`
2. Execute the scenario job.

Managing Truncate Manager

The data management 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 `<OFSAAI Installed Directory>/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:

```
<OFSAAI Installed Directory>/database/db_tools/logs/truncate_manager.log
```

Using the Truncate Manager

For the `run_truncate_manager.sh` script to take the table name as an argument, the table must exist in the BD ATOMIC schema. The script logs into the database using the user that the `truncate.database.username` property specifies in the `<OFSAAI Installed Directory>/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.

Managing 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`).
- `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`.

Note: The log for these scripts is written in the `run_stored_procedure.log` file under the `<OFSAAI Installed Directory>/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 `<OFSAAI Installed Directory>/database/db_tools/bin` directory.

You can run this utility anytime, that is, it is not necessary to run this utility during specific processing activities.

Running Threshold Analyzer

To run the threshold analyzer, follow these steps:

1. Go to ATOMIC schema and execute below query:

   ```
   select distinct (creat_ts)
     from kdd_review t
    where t.review_type_cd = 'AL'
      and SCNRO_DISPL_NM <> 'User Defined'
      and PRCSNG_BATCH_NM = 'DLY';
   ```

2. Set date as per dates returned from above SQL. Say CREATE_TS is 05/21/2013 in kdd_review table than we will set a date 05/17/2013 (Friday of last week) from the $FICHOME/database/db_tools/bin folder.
3. Execute the following command:

   ```
   start_mantas_batch.sh DLY
   set_mantas_date.sh 20130517 --(Friday of last week)
   ```

4. Execute DRM utility to create partitions, refer to Table -63 for parameter values:

   ```
   run_drm_utility.sh <Partition> <Weekly> <schema> <Table name>
   <drm_weekly_proc_fl>
   ```

There should be different variations for each Oracle product. For example:

```
run_drm_utility.sh P W ATOMIC KDD_TA_ML_DATA N
run_drm_utility.sh P W ATOMIC KDD_TA_BC_DATA N
run_drm_utility.sh P W ATOMIC KDD_TA_TC_DATA N
```

5. Execute the following Insert and Update Threshold Analyzer scripts from $FICHOME/database/db_tools/bin folder:

   ```
   run_insert_ta_utility.sh
   run_update_ta_utility.sh
   ```

6. Repeat the above process if you have more than one date retuned from query in point 1.

Managing 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). Also, after the batch is executed, it makes an audit entry

CHAPTER 8 Managing Administrative Utilities

OFSBD 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
- Managing Data Analysis Tool
- Managing Get Dataset Query with Thresholds Utility
- Managing Scenario Migration Utility
- Managing Alert Correlation Rule Migration Utility
- Investigation Management Configuration Migration Utility
- Managing Watch List Service
- Managing Alert Processing Web Services
- Managing Password Manager Utility
- Updating Oracle Sequences

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:

- **Managing Data Analysis Tool:** Assists a Data Miner or Data Analyst in determining how well a customer has populated the Production Data Model.
- **Managing Get Dataset Query with Thresholds Utility:** Enables the extraction of dataset SQL complete with substituted thresholds for analysis of the SQL outside of the Behavior Detection application.
- **Managing Scenario Migration Utility:** Extracts scenarios, datasets, networks, and associated metadata from a database to flat files and loads them into another environment.

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.

Managing 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 assumptions 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 following are the key features of 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 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 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.

This section covers the following topics:

- Configuring General Tool Properties
- Configuring the Analysis XML File

Configuring General Tool Properties

Use the Data Analysis Tool to configure the general tool properties.

To access the Data Analysis Tool, follow these steps:

1. Navigate to the `install.cfg` file that resides in `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg.`
2. Refer to the table below. The table provides the configuration instructions for the properties that the Data Analysis Tool uses in the `install.cfg` file.

Table 72. **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 ATOMIC schema.	`schema.business. owner=ATOMIC`
`schema.market. owner`	Database user for the ATOMIC schema.	`schema.market.owner= ATOMIC`
`dat.database. username`	User name for the database. The Data Analysis Tool connects to the database as the ATOMIC USER for the appropriate privileges.	`dat.database.username= ATOMIC`
`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 `<OFSAAI Installed Directory>/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 *Sample `install.cfg` File*.

Configuring 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:

- Analysis Constraints
- Analyzing Distinct Values for Fields of Interest
- Analyzing Null and Padded Space Count
- Analyzing Join Counts
- Other Queries

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

  The following table lists valid code operators:

Table 73. 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 73 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

Note: Oracle recommends `DD-MON-YYYY` as the default format

Analyzing Distinct Values for Fields of Interest

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
```

Analyzing Null and Padded Space Count

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
```

Analyzing Join Counts

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 204, for more information).
- Simple join between two or more tables with filter restriction (Refer to *Simple Join with Filter Restriction* on page 205, for more information).
- Join count of distinct values for specific column (Refer to *Join Count by Distinct Column* on page 206, 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 73, 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>
```

Note: 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>
```

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
, such as > 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 , such as <![CDATA[...) and the closing tag , such as ...]]>.

- 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 , such as ATOMIC, has aliases for most tables you may must analyze. Thus, running the tool as ATOMIC 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.

The following table lists the XML input files delivered for use with the Data Analysis Tool.

Table 74. 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.

Running the Data Analysis Tool

To run the Data Analysis Tool, follow these steps:

1. Navigate to the `<OFSAAI Installed Directory>/database/db_tools/bin` directory.
2. Execute the following command:

```
run_data_analysis_tool.sh [bg] [-i input_file.xml] [-o outputfile]
```

The following table describes the command line arguments that the Data Analysis Tool uses.

Table 75. 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 74) 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:

```
<OFSAAI Installed Directory>/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 209, 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 recommends that you view the output report using Microsoft Excel because this HTML file has specific HTML formatting for Excel.

The following table describes sections of the output report.

Table 76. 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.

Table 76. Data Analysis Report Output (Continued)

Section	Description
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 77 lists common Data Analysis Tool errors and their solutions.

Table 77. 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.

Managing 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.

This section covers the following topics:

- Using the Get Dataset Query With Thresholds Utility
- Executing the Get Dataset Query with Thresholds Utility

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.

The following table describes the parameters to provide with the `get_dataset_query.sh` script:

Table 78. 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 Utility, follow these steps:

1. After the Alert Creator process completes, execute the `get_dataset_query.sh` script as follows:

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/get_dataset_query.sh <Dataset 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:

   ```
   <OFSAAI Installed Directory>/database/db_tools/bin/get_dataset_query.sh <Dataset ID>
   <Threshold Set ID> query.sql
   ```

Managing Scenario Migration Utility

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 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.

This section covers the following topics:

- Logs
- Using the Scenario Migration Utility
- Scenario Migration Best Practices

Logs

The Scenario Migration Utility produces two log files (Figure 49 on page 215): `load.log` and `extract.log`. These files reside in the following location:

```
<OFSAAI Installed Directory>/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

To configure the Scenario Migration Utility, follow these steps:

Navigate to `OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg.` Theinstall.cfg file contains common configuration information that Scenario Migration and other utilities require for processing. Figure 49 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 must 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

#If custom patterns exist for a product scenario, set to 'Y' when loading a scenario hotfix.
#This should normally be set to 'N'.
load.ignore.custom.patterns=N

#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=/scratch/ofsaaapp/OFSBD 8.0.2/OFSBD
8.0.2_B06/BDP62_B06/database/db_tools/mantas_cfg/dep.cfg

sm.release=5.7.1

#### EXTRACT

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

# Specify the case schema name for both extraction and load .
caseschema.schema.owner=ATOMIC

# Specify the jdbc driver details for connecting to the source database
extract.conn.driver=${database.driverName}
extract.conn.url=jdbc:oracle:thin:@ofss220074.in.oracle.com:1521:Ti1O11L56
#Source System Id
extract.system.id=
# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
extract.schema.case=ATOMIC
extract.schema.business=${schema.business.owner}
```

(Continued on next page)

(Continued from previous page)

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

# File Paths for Extract

#Specify the full path in which to place extracted scenarios
extract.dirname=/scratch/ofsaaapp/OFSBD 8.0.2/OFSBD
8.0.2_B06/BDP62_B06/database/db_tools/data

#Specify the full path of the directory where the backups for the extracted scripts would be
maintained
extract.backup.dir=/scratch/ofsaaapp/OFSBD 8.0.2/OFSBD
8.0.2_B06/BDP62_B06/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 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

# 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=Ti1O11L56

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

(Continued on next page)

(Continued from previous page)

```
#Directory where scenario migration files reside for loading
load.dirname=/scratch/ofsaaapp/OFSBD 8.0.2/OFSBD
8.0.2_B06/BDP62_B06/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
```

Figure 49. 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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` shows. The tables in the following sections describe the parameters specific to the Scenario Migration Utility.

Configuring General Scenario Migration

The following table describes general scenario migration parameters.

Table 79. 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, <OFSAAI Installed Directory>/database/db_tools/mantas_cfg/dep.cfg.
`sm.release`	Version of the Scenario Migration Utility.

Caution: Oracle 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

The following table describes scenario extraction parameters.

Table 80. Scenario Extraction Parameters

Parameter	Description
`extract.database.username`	User used to connect to the database when extracting scenarios (ATOMIC).
`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`	ATOMIC schema owner in the database into which extraction of the scenarios occurs (ATOMIC).
`extract.schema.business`	ATOMIC schema owner in the database into which extraction of the scenarios occurs (ATOMIC).
`extract.schema.market`	ATOMIC schema owner in the database into which extraction of the scenarios occurs (ATOMIC).
`extract.user.miner`	ATOMIC schema owner in the database into which extraction of the scenarios occurs (ATOMIC).
`extract.miner.password`	Password for the above user.
`extract.dirname`	Full path to the target directory where the utility writes extracted metadata (<OFSAAI Installed Directory>/database/ db_tools/data).
`extract.backup.dir`	Full path to the target directory where the utility writes backups of the extracted metadata (<OFSAAI Installed Directory>/ 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

The following table describes scenario load parameters.

Table 81. Scenario Load Parameters

Parameter	Description
`load.conn.driver`	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
`load.conn.url`	Database connection string that the Scenario Migration Utility is to use.
`load.ignore.custom.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 deactivated. When set to Y, any custom patterns will be ignored by the load process, and should continue to operate.
`load.schema.mantas`	ATOMIC schema owner in the database in which loading of the scenario occurs (ATOMIC).
`load.schema.business`	ATOMIC schema owner in the database in which loading of the scenario occurs (ATOMIC).
`load.schema.market`	ATOMIC schema owner in the database in which loading of the scenario occurs (ATOMIC).
`load.user.miner`	ATOMIC schema owner in the database in which loading of the scenario occurs (ATOMIC).
`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 target environment. ● Selecting Y updates thresholds in the target environment to values from the incoming file.
`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, dataset, and network metadata, follow these steps:

1. Navigate to the
   `cd <OFSAAI Installed Directory>/db_tools directory`
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 avoid corrupting the Behavior Detection process, never load scenarios while the process is running.

To load scenario, dataset, and network metadata, follow these steps:

1. Navigate to the following directory:

   ```
   cd <OFSAAI Installed Directory>/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_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.
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 OFSBD 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 must 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 ATOMIC 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 OFSBD. Once the application is installed you should execute these steps to avoid scenario migration problems.

Making 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.

Separating Sequence ID ranges

1. Review the `ATOMIC.KDD_COUNTER` table, which contains all sequence ID ranges and current values.
2. Start your sequence ID ranger at 10,000,000 and separate each environment by 10,000,000. The OFSBD 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 82 lists sequences involved and sample values for the Development environment.

Table 82. 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	200000000	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_NTWRK	NTWRK_ID_SEQ	10000000	10000000	19999999
KDD_PARAM_SET	PARAM_SET_ID_SEQ	200000000	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

Table 82. Environment 1 (Development)

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 83 lists sequences involved and sample values for the Test/UAT environment.

Table 83. 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_ANLYS_NTWRK_DEFN	NTWRK_DEFN_ID_SEQ	20000000	20000000	29999999
KDD_LINK_ANLYS_TYPE_CD	TYPE_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 84 lists sequences involved and sample values for the Production environment.

Table 84. 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_ANLYS_NTWRK_DEFN	NTWRK_DEFN_ID_SEQ	30000000	30000000	39999999

Table 84. Environment 3 (PROD) (Continued)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
KDD_LINK_ANLYS_TYPE_CD	TYPE_ID_SEQ	30000000	30000000	39999999
KDD_NTWRK	NTWRK_ID_SEQ	20000000	20000000	29999999
KDD_PARAM_SET	PARAM_SET_ID_SEQ	30000000	30000000	39999999
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_DEFN_ID_SEQ',
 'TYPE_ID_SEQ',
 'TAB_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.

Managing 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 OFSBD 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.

This section covers the following topics:

- Logs
- Using the Alert Correlation Rule Migration Utility

Logs

The Alert Correlation Rule Migration Utility produces two log files (Figure 50 on page 225): `load.log` and `extract.log`. These files reside in the following location:

```
<OFSAAI Installed Directory>/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 Alert Correlation Rules Migration Utility
- Extracting Alert Correlation Rule
- Loading Alert Correlation Rule

Configuring Alert Correlation Rules Migration Utility

To configure Alert Correlation Migration Utility, folow these steps:

1. Navigate to `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg.The install.cfg` file contains common configuration information that Alert Correlation Rule Migration and other utilities require for processing.
2. Refer the following sample section from the install.cfg file for configuration information 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:thin:@ofss220074.in.oracle.com:1521:Ti1O11L56

# Specify the case schema name for both extraction and load .
caseschema.schema.owner=ECM62_B06_CASE

#Source System Id
extract.system.id= ENVIORNMENT

# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
extract.schema.case=ECM62_B06_CASE

# 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

#### LOAD (These properties are shared by Correlation Rule Migration Utility with the
Scenario Migration Utility)
```

(Continued on next page)

(Continued from previous page)

```
# 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}
load.schema.case=ECM62_B06_CASE

#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://localhost:8060/mantas/services/AlertProcessingService
aps.service.user=ECM62_B06_WEB_SERVICE
aps.service.user.password=
```

Figure 50. 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 `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg` shows (refer to Table 85). The tables in the following sections describe the parameters specific to the Alert Correlation Rule Migration Utility.

Configuring General Alert Correlation Rule Migration

The following table describes general alert correlation rule migration parameters.

Table 85. General Alert Correlation Rule Migration Parameters

Parameter	Description
corrRuleMig.CorrRuleFileNm	Location of the file containing the list of Alert Correlation Rule names to be migrated. `<OFSAAI Installed Directory> /database/db_tools/mantas_cfg/<FileName>.cfg`
aps.service.user	User name used for authenticating the web service call to the Alert Processing Service
aps.service.user.password	Password used for authenticating the web service request to the Alert Processing Service.
aps.service.url	Web service URL of the AlertProcessing service to be used for refreshing the correlation rules.
create_single_alert_correlation	Parameter to identify whether the correlation is created for a single alert. By Default, the value is set to *False*. This means that the system creates the correlation only when the alert count is more than 1.

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: `<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/corrRule.cfg`

Configuring Alert Correlation Rule Extraction

The following table describes alert correlation rule extraction parameters.

Table 86. Alert Correlation Rule Extraction Parameters

Parameter	Description
extract.database.username	User to use to connect to the database when extracting alert correlation rules (ATOMIC schema 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	ATOMIC schema owner in the database into which extraction of the alert correlation rule occurs (ATOMIC).
extract.dirname	Full path to the target directory where the utility writes extracted metadata (<OFSAAI Installed Directory>/database/ db_tools/data).
extract.backup.dir	Full path to the target directory where the utility writes backups of the extracted metadata (<OFSAAI Installed Directory>/ database/db_tools/data/temp).

Table 86. Alert Correlation Rule Extraction Parameters (Continued)

Parameter	Description
`caseschema.schema.owner`	Points to the ATOMIC schema for validating the case-related data defined in the correlation rules while extraction/loading of the rules into the database. This is only applicable if your firm has implemented Enterprise Case Management.

Configuring Alert Correlation Rule Load

The following table describes alert correlation rule load parameters.

Table 87. Alert Correlation Rule Load Parameters

Parameter	Description
`load.database.username`	User to use to connect to the database when loading alert correlation rules (ATOMIC schema 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.case`	ATOMIC schema name. This information is not only used by Correlation Migration Utility, but also by database utilities which interact with the ATOMIC schema.
`load.schema.mantas`	ATOMIC schema owner in the database in which loading of the alert correlation rule occurs.
`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. **Note:** Irrespective of whether you specify N or Y, 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 <OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/database/db_tools/bin
   ```
5. Execute the sm_extract.sh script as follows:

   ```
   sm_extract.sh correlation
   ```

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 ATOMIC 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 ATOMIC schema table KDD_CASE_TYPE_SUBTYPE.
- The attribute value for <subClassTagLevel1/> should exist in the CASE_SUB_CLASS_LVL1_CD column of the ATOMIC schema table KDD_SUBCLASS1.
- The attribute value for <subClassTagLevel2/> should exist in the CASE_SUB_CLASS_LVL2_CD column of the ATOMIC 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 ATOMIC 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 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 expression 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 ATOMIC 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 <OFSAAI Installed Directory>/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 <OFSAAI Installed Directory>/database/db_tools/bin
   ```

6. Execute the `sm_load.sh` script as follows:

   ```
   sm_load.sh correlation
   ```

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 ATOMIC 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 ATOMIC schema table KDD_CASE_TYPE_SUBTYPE.
- The attribute value for <subClassTagLevel1/> should exist in the CASE_SUB_CLASS_LVL1_CD column of the ATOMIC schema table KDD_SUBCLASS1.
- The attribute value for <subClassTagLevel2/> should exist in the CASE_SUB_CLASS_LVL2_CD column of the ATOMIC 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 ATOMIC 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 ATOMIC schema tables `KDD_ALERT_CORR` and `KDD_ALERT_CORR_SCNRO`.

Investigation Management Configuration Migration Utility

Use the Investigation Management Configuration Migration Utility to migrate Alert/Case investigation configuration metadata between environments. This utility provides a means to load alert and case configuration metadata into OFSBD as well as allows you to move configuration metadata between installations of OFSBD. Configuration metadata is considered to be that metadata associated with the alert and case workflows, such as actions, action categories, standard comments, case types, case workflows, and case statuses. The migration process handles ONLY database metadata and is executed using two separate procedures—extraction and loading. The extraction process pulls metadata from an environment into a file that can be can be moved, configuration controlled, and loaded into another environment. The load process loads these extracted files into the target environment.

To avoid accidental loading of Investigation Metadata into the incorrect environment, the Investigation Management Configuration 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 Investigation Metadata. If you attempt to load it to a different environment, the system displays a warning prompt.

Note: Because not all configuration metadata lies within the database it may be necessary to manually copy over XML files associated with configuration. This manual process is not handled by the Investigation Management Configuration Migration Utility. Specifically, if you are running Enterprise Case Management it will be necessary to migrate the `CaseWorkflowModel.xml` file. Basically, any customized XML file pertaining to configuration will must be manually migrated.

This section covers the following topics:

- Logs

- Using the Investigation Management Configuration Migration Utility

Logs

The Investigation Management Configuration Migration Utility produces two log files—`load.log` and `extract.log`. These files reside at the following location:

```
<OFSAAI Installed Directory>/database/db_tools/logs
```

Using the Investigation Management Configuration Migration Utility

This section covers the following topics, which describe configuring and executing the Investment Configuration Metadata Migration Utility, including extracting and loading metadata:

- Configuring the Investment Configuration Metadata Migration Utility
- Extracting Investigation Metadata
- Loading Alert/Case Investigation Metadata

Configuring the Investment Configuration Metadata Migration Utility

The <OFSAAI Installed Directory>`/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Investment Configuration Metadata Migration Utility and other utilities require for processing. Figure 51 provides sample information from the `install.cfg` file that is specific to this utility.

This utility migrates data for the following tables:

- KDD_CASE_TYPE_SUBTYPE
- KDD_ACTIVITY_TYPE_CD
- KDD_ACTVY_TYPE_REVIEW_STATUS
- KDD_SCNRO_CLASS_ACTVY_TYPE
- KDD_ACTVY_TYPE_RSTRN
- KDD_ACTVY_CAT_CD
- KDD_CMMNT
- KDD_SCNRO_CLASS_CMMNT
- KDD_CMMNT_CAT_CD
- KDD_REVIEW_STATUS
- KDD_ACTIVITY_RESULT_STATUS
- KDD_CASE_TYPE_CMMNT
- KDD_CASE_TYPE_ACTIVITY
- KDD_EXTRL_REF_SRC
- KDD_FOCUS_ALERT_ASGMT
- KDD_AUTO_CLOSE_ALERT
- KDD_BUS_DMN
- KDD_JRSDCN
- KDD_SUBCLASS1

- KDD_SUBCLASS2
- KDD_TYPE_CLASS_MAP
- KDD_COUNTER
- KDD_CAL_HOLIDAY
- KDD_CAL_WKLY_OFF
- KDD_REPORT_TEMPLATE
- KDD_REPORT_TEMPLATE_PARAM
- KDD_REPORT_DEFN
- KDD_REPORT_DEFN_PARAM
- KDD_REPORT_TEMPLATE_JRSDCN
- KDD_AVERTED_LOSS_TYPE
- KDD_REG_REPORT_TYPE
- KDD_REG_REPORT_STATUS

```
#### EXTRACT (These properties are shared by IMCM 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= TEST_ENVIORNMENT
# File Paths for Extract
#Specify the full path in which to place extracted Correlation Rules
extract.dirname=/users/oriont/Mantas5.8/database/db_tools/data
#### LOAD (These properties are shared by IMCM Utility with the Scenario Migration Utility)
#Target System ID
load.system.id= PROD_ENVIRONMENT
# Specify whether or not to verify the target environment on load
verify.target.system=Y
# Specify the prefix for the file that would be created by IMCM Utility during extract. This
property is specific to Investigation Management Configuration Migration Utility
config.filenm.prefix=Config
```

Figure 51. Sample `install.cfg` **File for Investigation Management Configuration 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 Investigation Metadata Migration, modify the parameters that the sample `install.cfg` file shows (refer to Table 88). The tables in the following sections describe the parameters specific to the Investigation Management Configuration Migration Utility.

Configuring General Investigation Metadata Migration

The following table describes the general Investigation Metadata migration parameters.

Table 88. General Investigation Metadata Migration Parameters

Parameter	Description
config.filenm.prefix	Prefix used by the utility for naming the extracted file,

Configuring Investigation Metadata Extraction

The following table describes Investigation Metadata extraction parameters.

Table 89. Investigation Metadata Extraction Parameters

Parameter	Description
extract.database.username	User to connect to the database when extracting Investigation Metadata (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 Investigation Metadata Migration Utility is to use.
extract.system.id	System from which the Investigation Metadata was extracted.
extract.dirname	Full path to the target directory where the utility writes extracted metadata (`$FIC_WEB_HOME`/database/ db_tools/data).

Configuring Alert Investigation Metadata Load

The following table describes the Investigation Metadata load parameters.

Table 90. Investigation Metadata Load Parameters

Parameter	Description
utils.database.username	User to connect to the database when loading Investigation Metadata (DB_UTIL_USER).
utils.database.password	Password for the above user.
database.driverName	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
utils.database.urlName	Database connection string that the Investigation Metadata Migration Utility is to use.

Table 90. Investigation Metadata Load Parameters (Continued)

Parameter	Description
load.system.id	Name that is assigned to the system into which this instance of Investigation Metadata Migration loads metadata. The system compares the value for this setting to the target system in the metadata file.
verify.target.system	Check target name upon loading metadata files. ● Setting to N prevents Investigation Metadata Migration from checking the load.system.id against the target system specified when the Investigation Metadata 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 Investigation Metadata Migration or allow the utility to continue prompting on each XML file that has a mismatch.

Extracting Investigation Metadata

Investigation metadata includes XML files that contain the table data for the Alert/Case Investigation. The `sm_extract.sh` script invokes a Java tool, which creates these files. You start the script as follows:

```
sm_extract.sh investconfig
```

To extract Alert/Case Investigation metadata, execute the `sm_extract.sh` file.

Loading Alert/Case Investigation Metadata

The `sm_load.sh` script loads translated XML table data files into the target database.

To load the Alert/Case Investigation metadata, execute the `sm_load.sh` file as follows:

```
sm_load.sh investconfig
```

Managing 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 *Services Guide*, for more details on how the service can be called and the results that are returned.

Managing 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 *Services Guide*.

Managing Password Manager Utility

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

<OFSAAI Installed Directory>/`changePassword.sh`.:

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

- Data Ingest User
- Database Utility User
- Data Miner User
- Purge Utility User
- Patch Database User
- Algorithm User
- Web Application User
- Web Service User

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

```
    <OFSAAI Installed Directory>/changePasswords.sh <property name>
```

For example,

```
    <OFSAAI Installed Directory>/changePasswords.sh email.smtp.password
```

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

```
        <OFSAAI Installed Directory>/changePassword.sh all
```

For changing password for admin tools subsystem, execute the command n `$FIC_HOME/ficweb/webroot/solution/bdf/scripts/changePasswords.sh`This prompts for the passwords of the following users:

- Web Application User
- Data Miner User

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

Updating Oracle Sequences

The OFSBD framework uses Oracle sequences for BD datamap component. To this end, OFSBD provides the ability to maintain the Oracle sequences used in Behavior Detection. This utility must be compulsorily run by clients who are upgrading from Informatica to OFSBD at least one time at the end of the stage 1 upgrade process. This utility also doubles up as a maintenance utility for these Oracle sequences.

The shell script which must be executed for invoking this utility is `run_update_ora_seq.sh`. This script in turn calls a database procedure by the name of `P_UPDATE_ORACLE_SEQUENCE`. The database procedure `P_UPDATE_ORACLE_SEQUENCE` contains the logic to set the correct start value of Oracle sequences. The procedure internally drops and re-creates Oracle sequences by getting the max value +1 of the `seq_id` column from the base table as specified in the `TABLE_NM` column of metadata table `KDD_ORACLE_SEQUENCE`.

Clients upgrading from previous version of OFSBD to 6.2.1 version just must run the script `run_update_ora_seq.sh` without any parameters.

For maintenance work the script can be executed either by not passing any parameter or by passing either the table name or the Oracle sequence name as its optional parameter.

For example:

1. Without any parameter: `run_update_ora_seq.sh`
2. Passing table name or Oracle sequence name as parameter: `run_update_ora_seq.sh<TABLE_NAME> OR run_update_ora_seq.sh<ORACLE_SEQUENCE_NM>`

If the table name OR the sequence name is not specified, then the utility performs the maintenance activity for all sequences mentioned in the `KDD_ORACLE_SEQUENCE` metadata table. If the script is called by passing the table name or the Oracle sequence name as its parameter, then the maintenance activity is done only for that particular table / Oracle sequence.

Note: Do not modify the `KDD_ORACLE_SEQUENCE` metadata table unless specifically requested by the Oracle support team.

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

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

Clients who are upgrading from Informatica to OFSBD must run this utility at the end of the stage 1 upgrade process. Also, this utility can be run anytime there is a maintenance work on the database affecting the Oracle sequences. Additionally, there can be scenarios when the database is recovered due to some fault in the database requiring run of this utility. Failure to comply with this may result in Unique Constraints violation errors when datamaps are executed.

Note: When executing `run_update_ora_seq_sh`, it may fail and display the following error: *ORA-04006: START WITH cannot be less than MINVALUE*. To fix this error, update dim_country set N_COUNTRY_SKEY = 0 where N_COUNTRY_SKEY = -999

CHAPTER 9

Posting External Alerts through Batches

Alerts which are created by external systems can be posted into the Behavior Detection system for further investigation through batch mode. The data from external sources should be made available in the processing tables using the Excel Upload functionality. Once the data is available in the processing tables, the system will post the external alerts.

This chapter discusses the following topics:

The user must be mapped to the AMMANADMNGR (Mantas Administrator User Group) user group to post external alert data into the processing tables and execute the batch which moves the data into the Alert Investigation table.

Batch Execution

Once the external data is loaded into the processing tables, the BD_EXTRL_ALERT_GENERATION batch has to be executed. The following tasks should be configured with valid values for the batch date and batch name in the BD batch before triggering the BD_EXTRL_ALERT_GENERATION batch. The BD batch should be configured with the batch name and the batch date before triggering the batch:

- BD_SET_BATCH_DATE_FOR_IPE
- BD_START_BATCH_FOR_IPE

For more information about how to execute a batch, refer to the *Oracle Financial Services Analytical Applications Infrastructure User Guide.*

Note:
a)Values for the tasks should be enclosed within double quotes.
b)Batch date should be in the YYYYMMDD format.
c) The application is pre-packaged with one BD batch. The BD batch should be triggered once a day. If there is a need to trigger the BD batch more than once a day, then insert a record into the KDD_PRCSNG_BATCH.
d)The processing table updates from the External Sources System and from IPE. The BD_EXTRL_ALERT_GENERATION batch and BD_GENERATE_ALERTS_FROM_IPE batch should not be executed in parallel.

Posting Alert from External Source

The tasks mentioned in Table 91 are used for generating alerts from an external source system except for Tasks 3 and 4.

Posting Alerts from External Source System

Alerts can be posted into the BD system from the external source system using OFSAAI's Excel Upload functionality. The Excel templates provided as part of the installer can be used to populate data into common processing tables. These files are available at `<ftpshare>/STAGE/ExcelUpload/TEMPLATES`. For more information about the Excel Upload feature, refer to the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

The following Excel Upload Templates are provides the external alerts data:

- `kdd_extrl_batch_last_run.xlsx`
- `kdd_extrl_mtch.xlsx`
- `kdd_extrl_break_mtchs.xlsx`

To upload the data into the processing tables, follow these steps::

1. Populate the Excel templates as per the data instructions available. For more information, refer to *Oracle Financial Services FSDM Reference Guide: Volume 2*.
   - **`kdd_extrl_batch_last_run.xlsx:`**This template populates the External Batch table, which is the processing table for capturing batches that must be considered for alert generation process. The Batch Run ID populated in this table must be unique. The suggested batch run ID is the External Source System Type followed by the date and time in MM/DD/YYY HHMM format on which the data is being uploaded.
   - **`kdd_extrl_mtch.xlsx:`**This template populates the External Alert Match table, which is the processing table for posting external alerts data into the Alerts table structure through batch mode. Each record in the table contains information that can be used to create an external alert to be posted to the Alert Management Data Model. Information can include attributes such as Scenario ID, Focal Entity ID, Class, Focus Type, and identification of the source of this record (internal, external) and so on. Scenario Class, Scenario ID, Pattern ID and Focus must be among the available values. If the Scenario ID and Pattern ID of the external source system are not available in the BD system, then as a onetime activity the scenario or pattern of the alert which will be posted must be populated into the KDD_SCNRO and KDD_PTTRN table. Refer to section *Scenario Migration Best Practices,* on page 219.
   - **`kdd_extrl_break_mtchs.xlsx:`** This template populates the External Alert Matched Data table, which is the processing table for posting data associated with an external alert into the Alerts table structure through batch mode. Currently, the break matches can be generated only through the External Alert Matched Data. Matched Entity Sequence Identifier (`KDD_EXTRL_BREAK_MTCHS.ENTITY_KEY_ID`).The Business data is available in the BD instance for moving the associated alert data into the respective archive tables.

2. Log in to the OFSBD application as an Admin user.

3. Navigate to **Unified Metadata Manager > Data Entry Forms and Queries > Excel Upload**. The Excel Upload page is displayed.

   Note: After logging into the application, make sure the OFSBD application Information Domain is selected from the drop-down list at the left hand corner of the page.

4. Click **Browse** under **Excel File to Upload**.
5. Select the Excel template which should be uploaded.
6. Click the **Arrow** button next to **Browse**.
7. Preview the data created under the Preview section.
8. In the Excel - Entity Mappings section, click **Arrow**.
9. Select the table name with the same name as that of the Excel sheet.
10. Click **Upload**. The following message is displayed: *Successfully uploaded data*
11. Click **OK**.
12. If Excel Upload is not successful, refer to logs available in the `<ftpshare>/STAGE/ExcelUpload/logs` folder.

    Note: Scan the web application server log for any errors after uploading each Excel sheet.For example, for Oracle Weblogic, the web application server log is wls.out, for Websphere, the web application server log is Systemout.log and Systemerr.log.

13. Close the Excel Upload page.

CHAPTER 10

Alert Generation from IPE Assessment Results

This chapter provides information about executing the batch for IPE assessments, Alert Generation and discusses the following topics:

- Execution
- Alert Generations Batch

In order to configure IPE assessments and execute the batch, a user must be mapped to the IPEADMIN (Inline Processing Admin Group) and AMMANADMNGR (Mantas Administrator User Group) user groups. For more information on user/user group mapping, refer to the *Oracle Financial Services Analytical Applications Infrastructure User Guid***e**. For more information about the IPEADMIN user group, refer to the *Oracle Financial Services Inline Processing Engine User Guide***.**

Execution

Once an Assessment is defined, the assessment must be executed as a task in batch mode using the Run Rule Framework. For more information, refer to the *Oracle Financial Services Inline Processing Engine User Guide*.

As a prerequisite for further processing of assessment results into alerts, the BD_POPULATE_LAST_RUN_BATCH task should be executed after the IPE assessment task.

An example is provided below:

An IPE assessment has been configured using the Run Rule Framework. The Trade task was created for executing IPE assessment in batch mode. The BD_POPULATE_LAST_RUN_BATCH task must be populated in the same process as the IPE assessment task, with the IPE assessment task having precedence.

For more information on how to create and execute the tasks sequentially in Run Rule Framework, refer to the *Oracle Financial Services Analytical Applications Infrastructure User Guid***e**.

APPENDIX A Logging

This appendix describes the mechanism that OFSBD uses when logging system messages.

- About System Log Messages
- Message Template Repository
- Logging Levels
- Logging Message Libraries
- Logging Configuration File
- Alert Generation Logging from IPE and External System

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 , such as `mantas.log`, `mantas.log.1`, `mantas.log.2`.

Note: The log file size is a configurable property; section *Log File Sizes* on page 251 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 the full path to a subsystem's message library file.

Logging Levels

Table 91 outlines the logging levels that the Common Logging component supports.

Table 91. 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 , such as failed user sessions or missing data fields).
Notice (default)	High-level, informational messaging that highlights progress of an application , such as 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_WEB_HOME/AM/admin_tools/WEB-INF/classes/conf/mantas_cfg/etc/
mantas_admin_tools_message_lib_en.dat
```

All message 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:

```
<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/etc/
mantas_database_message_lib_en.dat
```

All message 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:

```
<OFSAAI Installed Directory>/behavior_detection/toolkit/mantas_cfg/etc/
mantas_toolkit_message_lib_en.dat
```

All message 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:

```
<OFSAAI Installed Directory>/services/server/webapps/mantas/WEB-INF/classes/conf/
mantas_cfg/etc/mantas_alert_management_message_lib_en.dat
```

All message 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:

```
$FIC_WEB_HOME/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:

```
<OFSAAI Installed Directory>/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. The following table lists the subsystems and their log files:

Table 92: Logging Configuration Files

Subsytem	File
Administration Tools	$FIC_WEB_HOME/AM/admin_tools/WEB-INF/classes/conf/mantas_cfg/install.cfg
Database	`<OFSAAI Installed Directory>/database/db_tools/mantas_cfg/install.cfg`
Scenario Manager	`<OFSAAI Installed Directory>/behavior_detection/toolkit/mantas_cfg/install.cfg`
Behavior Detection	`<OFSAAI Installed Directory>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg`

Alert Management/Case Management Web Server logs	$FIC_WEB_HOME/conf/RevLog4jConfig.xml
	`<root>`
	`<priority value ="debug" />`
	`<appender-ref ref="ConsoleAppender1"/>`
	`</root>`
	The following logger levels are available:
	● DEBUG
	● INFO
	● WARN
	● SEVERE
	● FATAL

Alert Management/Case Management Application Server logs	$FIC_WEB_HOME/conf/RevLog4jConfig.xml
	`<root>`
	`<priority value ="debug" />`
	`<appender-ref ref="ConsoleAppender1"/>`
	`</root>`
	The following logger levels are available:
	● DEBUG
	● INFO
	● WARN
	● SEVERE
	● FATAL

Services	<OFSAAI Installed Directory>/services/server/webapps/mantas/WEB-INF/classes/conf/mantas_cfg/install.cfg<OFSAAI Installed Directory>/services/mantas_cfg/install.cfg
Ingestion Manager	`<OFSAAI Installed Directory>/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 following sample as they contain constraints that relate to properties and logging.

```
# Specify which priorities are enabled in a hierarchical fashion, i.e., if
# DIAGNOSTIC priority is enabled, NOTICE, WARN, and FATAL are also enabled,
# but TRACE is not.
# Uncomment the desired log level to turn on appropriate level(s).
# Note, DIAGNOSTIC logging is used to log database statements and will slow
# down performance. Only turn on if you must 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
```

(Continued on next page)

(Continued from previous page)

```
# 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
filelog.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
# Specify where messages of a specific severity and category should be logged to.
# The valid values are the same number as for category.
# Multiple locations can be listed for each property using a comma delimiter.
# If an entry for a severity is not listed here, the message is logged to
# the location specified for the category number by the above property, and if that does
not exist, it is logged to the default location configured below.
log.TEST_CATEGORY.warning.location=syslog
log.TEST_CATEGORY.fatal.location=user@domain.com
log.TEST_CATEGORY_2.warning.location=syslog

# # Specify the full path to the external log4j configuration file
log4j.config.file=@WORKFLOW_LOG4J_CONFIG_FILE@

# Specify where a message should get logged for a category for which there is
# no location property listed above.
# This is also the logging location of the default Oracle Financial Services category
unless
# otherwise specified above.
# Note that if this property is not specified, logging will go to the console.
log.default.location=

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

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

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

# Specify the maxfile size of a logfile before the log messages get rolled to
# a new file (measured in bytes).
# If this property is not specified, the default of 10 MB will be used.
log.max.size=2000000000
```

Figure 52. 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 93 identifies the valid values for this property.

Table 93. Logging Location Property Values

Property value	Log location
`console`	Records the logs to the `system.out` or `system.err` file.
`syslog`	Records the logs to a remote UNIX syslog daemon. This is the default location.
`eventviewer`	Records the logs to the Event Log system.
`mantaslog`	Indicates the format of the mantaslog filename as job<job #>-datetimestamp (if running the algorithms). For other subsystems, the format is mantaslog-datetimestamp. The file resides at the location that the `log.mantaslog.location` property specifies in the appropriate `install.cfg` file. If this property is unspecified, the system outputs logs to the console.
`<path>/<filename>`	Records the logs to a file with the filename <filename>, which resides at <path>. For example, `log.message.library=/user/jsmith/message/messages.dat`
`<name@address>`	Records the logs in a message to the e-mail address indicated by `<name@address>`.

Note that category names (that is, property values) cannot contain the following reserved words: fatal, warning, notice, diagnostic, trace, category, or location. You can configure multiple locations for each property using a comma delimiter.

For example:

```
log.category.TEST_CATEGORY.location=console, mantaslog
log.category.TEST_CATEGORY_2.location=console,
/users/jsmith/logs/mylog.log
```

Log File Sizes

If an Oracle 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 94 identifies the configurable properties for logging in an Oracle client's environment.

Table 94. 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 251 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 251 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 251 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.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 level logging should happen asynchronously; true indicates fatal level logging should happen synchronously. **Note**: Setting value to true (synchronous) may have performance impact

Table 94. Configurable Parameters for Common Logging (Continued)

Property	Sample Value	Description
`log.warning`	true	Indicates enabling of warning logging; *false* indicates that warning logging is not enabled.
`log.warning.synchronous`	false	Indicates that warning level logging should happen asynchronously; true indicates warning level logging should happen synchronously. **Note**: Setting value to true (synchronous) may have performance impact
`log.notice`	true	Indicates enabling of notice logging; *false* indicates that notice logging is not enabled.
`log.notice.synchronous`	false	Indicates that notice level logging should happen asynchronously; true indicates notice level logging should happen synchronously. **Note**: Setting value to true (synchronous) may have performance impact
`log.diagnostic`	false	Indicates that diagnostic logging is not enabled; *true* indicates enabling of diagnostic logging.
`log.diagnostic.synchronous`	false	Indicates that diagnostic level logging should happen asynchronously; true indicates diagnostic level logging should happen synchronously. **Note**: Setting value to true (synchronous) may have performance impact
`log.trace`	false	Indicates that trace logging is not enabled; *true* indicates enabling of trace logging.
`log.trace.synchronous`	true	Indicates that trace level logging should happen asynchronously; true indicates trace level logging should happen synchronously. **Note**: Setting value to true (synchronous) may have performance impact
`log.syslog.hostname`	hostname	Indicates the host name of syslog for messages sent to syslog.
`log.time.zone`	US/Eastern	Indicates the time zone that is used when logging messages.

The Ingestion Manager uses common logging by assigning every component , such as FDT or MDT, to a category. You can configure the destination of log messages for each component which Table 93 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 OFSBD 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.

Alert Generation Logging from IPE and External System

The following logs contain the message library for External Alert Generation:

- All the Tasks involved in the batch other than `BD_SET_BATCH_DATE_FOR_IPE,` `BD_START_BATCH_FOR_IPE`, `BD_ALERT_ASSIGNMENT`, `BD_HISTORICAL_DATA_COPY`, `BD_END_BATCH_FOR_IPE` are maintained in table `KDD_LOGS_MSGS`. Logs for these are maintained at `$FIC_HOME/database/db_tools/logs`
- Log files are available at the following locations:
  - `$FIC_DB_HOME/Logs`
  - `$FIC_DB_HOME/bin/nohup.out`

APPENDIX B OFSBD Software Updates

This appendix describes the application of OFSBD software updates in Oracle Financial Services Behavior Detection:

- OFSBD Software Updates - Hotfix
- Hotfix Effect on Customization

OFSBD 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 OFSBD. 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
- Ingestion/BD

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 cannot 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 hotfixes 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, 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 User Administration

This appendix describes the user administration of the Oracle Financial Services Behavior Detection Platform.

- Managing User Groups and User Roles
- Managing User Groups
- Defining User Access Properties and Relationships
- Accessing objects under Metadata Browser

Managing User Groups and User Roles

User Roles are pre-defined in OFSFCCM solutions. 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 pre-defined 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 on creating a new user group and mapping it to an existing role, For more information on mapping user with user groups, see *Oracle Financial Services Analytical Applications Infrastructure User Guide* in Identity Management section.

Note: Different solutions have different pre-defined/pre-occupied precedence of User Groups. Therefore, if a BD Admin/System Admin is creating a new User Group, do not use the following precedence while providing precedence value:

Table 95. Solution with Pre-defined Precedence Range

Solution	Precedence Range Already Occupied
OFS ECM	901 to 1000
OFS OR	1001 to 2000
OFS KYC	2001 to 3000
OFS RR	3001 to 4000
OFS PTA	4001 to 5000

Note: While creating a new User Group, you can set precedence as 5001 or greater.

Managing User Groups

The following sections describe how to manage User Groups:

- Defining User Group Maintenance Details
- Adding New User Group Details
- Mapping Users to User Groups

- Mapping User Group(s) to Domain(s)
- Mapping a User to a Single User Group

Defining User Group Maintenance Details

For more information on defining user group maintenance details, see *Oracle Financial Services Analytical Applications Infrastructure User Guide* in Identity Management section.

Adding New User Group Details

For more information on adding new user group details, see *Oracle Financial Services Analytical Applications Infrastructure User Guide* in Identity Management section.

Mapping Users to User Groups

Note: One user can also be used against multiple roles. If multiple roles are allocated to a single user, then the availability of actions depends on the Four Eyes approval option. If Four Eyes approval is *off*, then the user can take all actions available by the allocated roles, with no duplicates. If Four Eyes approval is *on*, then action linked to a role that does not require Four Eyes approval takes precedence if there is a conflict.

For more information on mapping users to user group, see *Oracle Financial Services Analytical Applications Infrastructure User Guide* in Identity Management section.

Mapping User Group(s) to Domain(s)

This section lists the steps involved in mapping user groups to information domains.

To map user group or groups to domain or domains, follow these steps:

1. Map all Alert Management User Groups to the Alert Management Information Domain (Infodom).
2. Map all Case Management User Groups to the Alert Management Information Domain (Infodom) and Case Management Information Domain (Infodom).
3. Map all Know Your Customer User Groups to the Alert Management Information Domain (Infodom), Case Management Information Domain (Infodom), and Know Your Customer Information Domain (Infodom).
4. Map all FATCA User Groups to the Alert Management Information Domain(Infodom) and Case Management Information Domain (Infodom).
5. Map all Personal Trading Approval User Groups to the Alert Management Information Domain (Infodom).

For more information on mapping user group or groups to domain or domains, see *Oracle Financial Services Analytical Applications Infrastructure User Guide* in Identity Management section.

For more information on configuring FATCA, see *FATCA Administration and Configuration Guide*.

Actions to Role mappings are done through Database tables. Sample action to role mappings are included in the application. For more information on changing the mapping of roles to actions, *Configuration Guide,* and refer to the following sections.

- Working with Alert Action Settings
- Working with Case Action Settings

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.

Mapping a User to a Single User Group

If a user has only one role then that user can be mapped to a single User Group associated with that User Role. For more information on mapping a user to a single user group, see *Oracle Financial Services Analytical Applications Infrastructure User Guide* in Identity Management section.

Mapping a User to Multiple User Groups

If a user have more than one role within FCCM (that is, within both Alert Management and Case Management), then the user must be mapped to the different User Groups associated with the corresponding role. When the user logs into FCCM, the user access permissions are the union of access and permissions across all roles.

Mapping a User to an Organization

If a user is mapped to an organization indicating that it is the line organization for the user and if there exists any child organization for that line organization, then those organizations are implicitly mapped to the user as a business organization. If the same organization is already mapped as the business organization, then the child of the organizations should not be mapped to the user implicitly by the system.

If an organization is implicitly mapped to the user based on line organization association, the user can still be unmapped from that organization if there is a need to limit them from seeing the organization. The organization still shows (I) in the Organization list to show that the organization is a child of the line organization. But the fact that it is not selected will prevent the user from being mapped to it.

The following rules apply:

- Users can have only one organization as the line organization.
- A child organization can have only one parent organization

To map organizations, follow these steps:

1. Select a user from the **Select User** drop-down list.
2. Select the line organization or organizations you want to map the user to from the Line Organization drop-down list.

   Note: If the user is associated with both line and business organizations, then the business organizations associated to the Line Organization must be implicitly mapped and display the organizations as well.

The system visually distinguishes the Implicit (I), which is the system determination based on line organization and Explicit (E), which was manually added by the user mapping, of business organizations. The system displays either I or E in the brackets to indicate that the grid displays two different column, one for Implicit and the other one for Explicit mapping.

3. Click **Save.**

Mapping a Function to a Role

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

The following table provides the function role mapping details.

Table 96. Function to Role Mapping Details

Function	Description
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.

Defining 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, 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 Behavior Detection Framework OFSECM administrator or Auditor).
- **Scenario Group(s):** Group of scenarios that identify a set of scenario permissions and to which a user has access rights.
- **Case Type or Subtype(s):** Case type or subtypes combinations to which, a user has access rights.

The following figure shows the user authorization model.

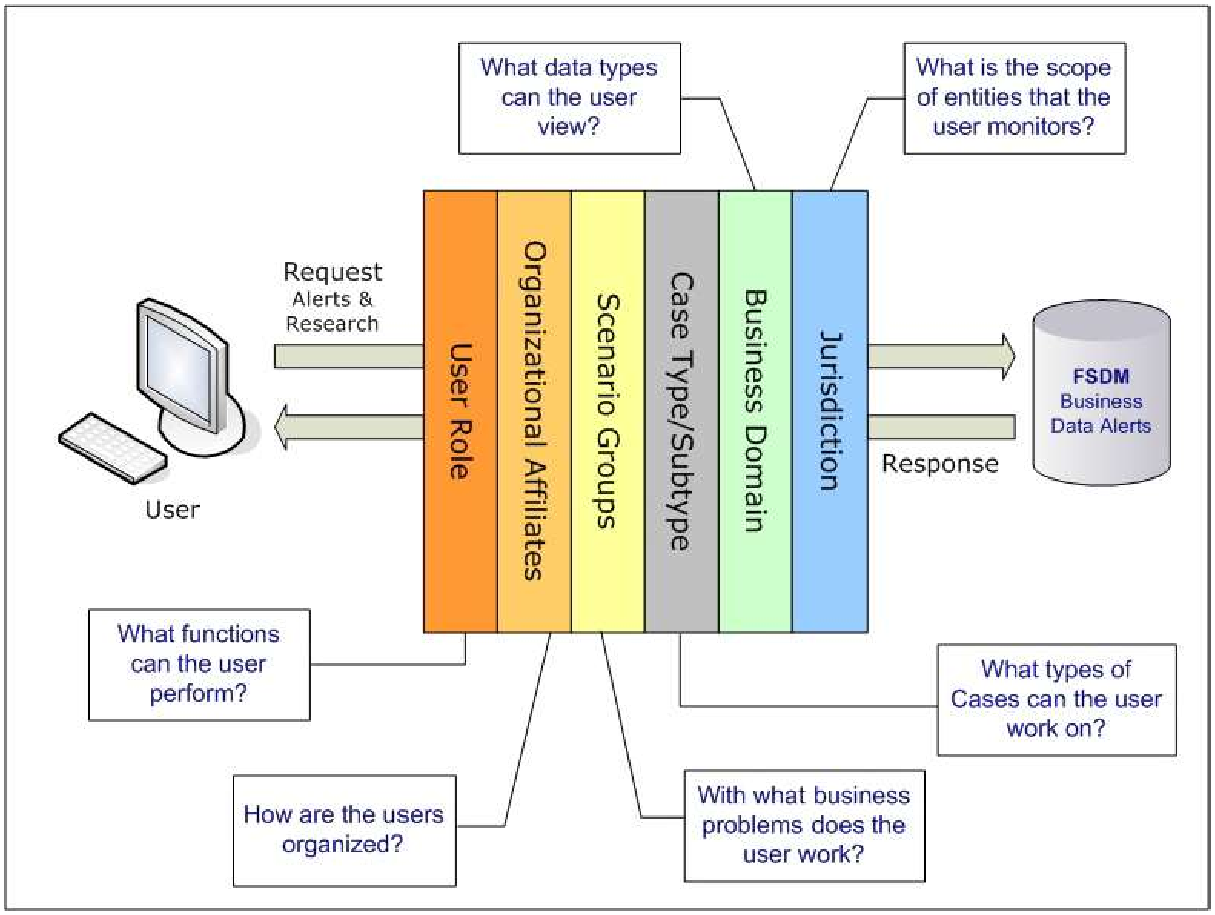

What data types can the user view?
What is the scope of entities that the user monitors?
Request
Alerts & Research
User
User Role
Organizational Affiliates
Scenario Groups
Case Type/Subtype
Business Domain
Jurisdiction
Response
FSDM
Business
Data Alerts
What functions can the user perform?
What types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

Figure 53. User Authorization Model

The following table provides the relationships between the data points that Figure 3 illustrates.

Table 97. Relationships between Data Points

Data Point	Relationship
Organization	Root of a BD 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 Group	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

Accessing objects under Metadata Browser

In order to access objects under the Metadata Browser, the following task must be executed:

`MDBPublishExecution.sh`

With appropriate admin privileges, navigate to **Common Tasks>Operations> Batch Execution** and execute the batch `OFSBDINFO_MDB_Batch.`

Alternatively, this batch can be executed from the putty console by following these steps:

1. Navigate to `$FIC_DB_HOME/bin.`
2. Run the script `MDBPublishExecution.sh`

APPENDIX D Managing Data

This appendix covers the following topics:

- FSDF/Hive CSA Ingestion
- Flat File Ingestion

FSDF/Hive CSA Ingestion

This section refers to FSDF/Hive Common Staging Area (CSA) ingestion and covers the following topics:

- CSA/Hive Datamaps
- List of Data Quality Group names, T2T and H2T names

CSA/Hive Datamaps

The following list of files can be run using FSDF Staging. Files have been grouped in such a way that files in the same group can be executed in parallel to load data. However, you must execute Group 1 through Group 6 in sequence.

Note: Ensure that you run the Country and Customer data files before you run the other files.

Table 98. CSA/Hive Datamaps

Group	Logical Table Name	
1.	Country	
2.	Account Phone Watch List Account Emai lAddress Insurance Product Insurance Policy Insurance Transaction Insurance Policy Balance Front Office Transaction Account Customer Role Organization Insurance Policy Feature Market Center	Insurance Policy To Customer Market Index Daily Loan Issuer Loan Daily Activity Market Index Online Account Service Team Insurance Seller Service Team Member Insurance Seller To License Customer Credit Rating Customer Identification Document

Table 98. CSA/Hive Datamaps

Group	Logical Table Name	
3.	Account To Peer Group Account Group Peer Group Security Firm Daily	Market Index Member Security Security Market Daily Security Customer
4.	Account Watch List Entry Loan Product Employee	Front Office Transaction Party Organization Relationship Restriction List Automated Quote
5.	Managed Account Account To Customer Account Group Member Account To Correspondent Account Balance Account Address Customer To Markets Served Customer To Products Offered Customer To Customer Relationship Anticipatory Profile Customer Phone Customer Email Address Customer Country Customer Address Online Account To Account	Controlling Customer Employee To Account Account Position Security Trading Restriction Employee Trading Restriction Employee Phone Employee Email Address Employee Address Security Group Member Security Investment Rating Structured Deal Account Profit And Loss Account Investment Objective Account Position Pair Mutual Fund Breakpoint Market News Event
6.	Borrower Account Restriction Back Office Transaction	Investment Advisor Settlement Instruction Loan Origination Document Print Log

List of Data Quality Group names, T2T and H2T names

The following table provides the FSDM logical table names, T2T names, H2T names, and the corresponding data quality group names:

Table 99. Data Quality Group Names and Related T2T Names

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Account	ACCT	● t2t_Account.STG_ANNUITY_CONTRACTS ● t2t_Account.STG_BILLS_CONTRACTS ● t2t_Account.STG_BORROWINGS ● t2t_Account.STG_CARDS ● t2t_Account.STG_CASA ● t2t_Account.STG_CORRESPONDENT_ACCOUNT ● t2t_Account.STG_FUTURES ● t2t_Account.STG_FX_CONTRACTS ● t2t_Account.STG_MERCHANT_CARDS ● t2t_Account.STG_MM_CONTRACTS ● t2t_Account.STG_MUTUAL_FUNDS ● t2t_Account.STG_OD_ACCOUNTS ● t2t_Account.STG_OPTION_CONTRACTS ● t2t_Account.STG_PREPAID_CARDS ● t2t_Account.STG_REPO_CONTRACTS ● t2t_Account.STG_RETIREMENT_ACCOUNTS ● t2t_Account.STG_SWAPS_CONTRACTS ● t2t_Account.STG_TD_CONTRACTS ● t2t_Account.STG_TRADING_ACCOUNT ● t2t_Account.STG_TRUSTS	● h2t_Account.STG_ANNUITY_CONTRACTS ● h2t_Account.STG_BILLS_CONTRACTS ● h2t_Account.STG_BORROWINGS ● h2t_Account.STG_CARDS ● h2t_Account.STG_CASA ● h2t_Account.STG_CORRESPONDENT_ACCOUNT ● h2t_Account.STG_FUTURES ● h2t_Account.STG_FX_CONTRACTS ● h2t_Account.STG_MERCHANT_CARDS ● h2t_Account.STG_MM_CONTRACTS ● h2t_Account.STG_MUTUAL_FUNDS ● h2t_Account.STG_OD_ACCOUNTS ● h2t_Account.STG_OPTION_CONTRACTS ● h2t_Account.STG_PREPAID_CARDS ● h2t_Account.STG_REPO_CONTRACTS ● h2t_Account.STG_RETIREMENT_ACCOUNTS ● h2t_Account.STG_SWAPS_CONTRACTS ● h2t_Account.STG_TD_CONTRACTS ● h2t_Account.STG_TRADING_ACCOUNT ● t2t_Account.STG_TRUSTS
Account	ACCT	● t2t_Account.STG_FUTURES ● t2t_Account.STG_FX_CONTRACTS ● t2t_Account.STG_GUARANTEES ● t2t_Account.STG_INVESTMENTS ● t2t_Account.STG_LC_CONTRACTS ● t2t_Account.STG_LEASES_CONTRACTS ● t2t_Account.STG_LOAN_CONTRACTS	● h2t_Account.STG_FUTURES ● h2t_Account.STG_FX_CONTRACTS ● h2t_Account.STG_GUARANTEES ● h2t_Account.STG_INVESTMENTS ● h2t_Account.STG_LC_CONTRACTS ● h2t_Account.STG_LEASES_CONTRACTS ● h2t_Account.STG_LOAN_CONTRACTS
Account Address	ACCT_ADDR	● t2t_AccountAddress	● h2t_Account Address

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Account Balance	ACCT_BAL_POSN_SMRY	● t2t_AccountBalance.STG_ANNUITY_CONTRACTS ● t2t_AccountBalance.STG_CARDS ● t2t_AccountBalance.STG_CASA ● t2t_AccountBalance.STG_CORRESPONDENT_ACCOUNT ● t2t_AccountBalance.STG_INVESTMENTS ● t2t_AccountBalance.STG_LEASES_CONTRACTS ● t2t_AccountBalance.STG_LOAN_CONTRACTS ● t2t_AccountBalance.STG_OD_ACCOUNTS ● t2t_AccountBalance.STG_RETIREMENT_ACCOUNTS ● t2t_AccountBalance.STG_TD_CONTRACTS ● t2t_AccountBalance.STG_TRADING_ACCOUNT ● t2t_AccountBalance.STG_TRUSTS	● h2t_AccountBalance.STG_ANNUITY_CONTRACTS ● h2t_AccountBalance.STG_CARDS ● h2t_AccountBalance.STG_CASA ● h2t_AccountBalance.STG_CORRESPONDENT_ACCOUNT ● h2t_AccountBalance.STG_INVESTMENTS ● h2t_AccountBalance.STG_LEASES_CONTRACTS ● h2t_AccountBalance.STG_LOAN_CONTRACTS ● h2t_AccountBalance.STG_OD_ACCOUNTS ● h2t_AccountBalance.STG_RETIREMENT_ACCOUNTS ● h2t_AccountBalance.STG_TD_CONTRACTS ● h2t_AccountBalance.STG_TRADING_ACCOUNT ● h2t_AccountBalance.STG_TRUSTS
Account Customer Role	CUST_ACCT_ROLE	t2t_AccountCustomerRole	h2t_Account Customer Role
Account Email Address	ACCT_EMAIL_ADDR	t2t_AccountEmailAddress	h2t_Account Email Address
Account Group	ACCT_GRP	t2t_AccountGroup	h2t_Account Group
Account Group Member	ACCT_RLSHP	t2t_AccountGroupMember	h2t_Account Group Member
Account Investment Objective	ACCT_NVSMT_OBJ	t2t_AccountInvestmentObjective	NA
Account Phone	ACCT_PHON	t2t_AccountPhone	h2t_Account Phone
Account Position	ACCT_POSN	t2t_AccountPosition.STG_ACCOUNT_POSITION	NA
Account Position Pair	ACCT_POSN_PAIR	t2t_AccountPositionPair	NA
Account Profit and Loss		t2t_AccountProfitAndLoss	NA

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Account Restriction	ACCT_RSTRN	t2t_AccountRestriction	NA
Account to Correspondent	ACCT_INSTN_MAP_STAGE	t2t_AccountToCorrespondent	h2t_Account to Correspondent
Account to Customer	CUST_ACCT	t2t_AccountToCustomer	h2t_Account to Customer
Account To Peer Group	ACCT_PEER_GRP	t2t_AccountToPeerGroup	h2t_Account To Peer Group
Anticipatory Profile	NTCPTRY_PRFL	● t2t_AnticipatoryProfile.STG_ACCT_ANTICIPATORY_PROFILE ● t2t_AnticipatoryProfile.STG_CUST_ANTICIPATORY_PROFILE	● h2t_AnticipatoryProfile.STG_ACCT_ANTICIPATORY_PROFILE ● h2t_AnticipatoryProfile.STG_CUST_ANTICIPATORY_PROFILE
Automated Quote	AUTO_QUOTE	t2t_AutomatedQuote.STG_AUTOMATED_QUOTE	NA

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Back Office Transaction	BACK_OFFICE_TRXN	● t2t_BackOfficeTransaction.STG_ANNUITY_TXNS ● t2t_BackOfficeTransaction.STG_BILLS_CONTRACTS_TXNS ● t2t_BackOfficeTransaction.STG_BORROWINGS_TXNS ● t2t_BackOfficeTransaction.STG_CARDS_PAYMENT_TXNS ● t2t_BackOfficeTransaction.STG_CARDS_SETTLEMENT_TXNS ● t2t_BackOfficeTransaction.STG_CASA_TXNS ● t2t_BackOfficeTransaction.STG_CORRESPONDENT_ACCT_TXNS ● t2t_BackOfficeTransaction.STG_CUSTODIAN_ACCOUNT_TXNS ● t2t_BackOfficeTransaction.STG_FOREX_TXNS ● t2t_BackOfficeTransaction.STG_INTERBANK_TXNS ● t2t_BackOfficeTransaction.STG_INVESTMENT_TXNS ● t2t_BackOfficeTransaction.STG_LEASES_TXNS ● t2t_BackOfficeTransaction.STG_LOAN_CONTRACT_TXNS ● t2t_BackOfficeTransaction.STG_MERCHANT_CARDS_TXNS	● h2t_BackOfficeTransaction.STG_ANNUITY_TXNS ● h2t_BackOfficeTransaction.STG_BILLS_CONTRACTS_TXNS ● h2t_BackOfficeTransaction.STG_BORROWINGS_TXNS ● h2t_BackOfficeTransaction.STG_CARDS_PAYMENT_TXNS ● h2t_BackOfficeTransaction.STG_CARDS_SETTLEMENT_TXNS ● h2t_BackOfficeTransaction.STG_CASA_TXNS ● h2t_BackOfficeTransaction.STG_CORRESPONDENT_ACCT_TXNS ● h2t_BackOfficeTransaction.STG_CUSTODIAN_ACCOUNT_TXNS ● h2t_BackOfficeTransaction.STG_FOREX_TXNS ● h2t_BackOfficeTransaction.STG_INTERBANK_TXNS ● h2t_BackOfficeTransaction.STG_INVESTMENT_TXNS ● h2t_BackOfficeTransaction.STG_LEASES_TXNS ● h2t_BackOfficeTransaction.STG_LOAN_CONTRACT_TXNS ● h2t_BackOfficeTransaction.STG_MERCHANT_CARDS_TXNS

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
		● t2t_BackOfficeTransaction.STG_MM_TXNS ● t2t_BackOfficeTransaction.STG_OD_ACCOUNTS_TXNS ● t2t_BackOfficeTransaction.STG_OTHER_SERVICES_TXNS ● t2t_BackOfficeTransaction.STG_REPO_TRANSACTIONS ● t2t_BackOfficeTransaction.STG_RETIREMENT_ACCOUNTS_TXNS ● t2t_BackOfficeTransaction.STG_SWAP_ACCOUNT_TXNS ● t2t_BackOfficeTransaction.STG_TERMDEPOSITS_TXNS ● t2t_BackOfficeTransaction.STG_TRADING_ACCOUNT_TXNS ● t2t_BackOfficeTransaction.STG_TRUSTS_TXNS	● h2t_BackOfficeTransaction.STG_MM_TXNS ● h2t_BackOfficeTransaction.STG_OD_ACCOUNTS_TXNS ● h2t_BackOfficeTransaction.STG_OTHER_SERVICES_TXNS ● h2t_BackOfficeTransaction.STG_REPO_TRANSACTIONS ● h2t_BackOfficeTransaction.STG_RETIREMENT_ACCOUNTS_TXNS ● h2t_BackOfficeTransaction.STG_SWAP_ACCOUNT_TXNS ● h2t_BackOfficeTransaction.STG_TERMDEPOSITS_TXNS ● h2t_BackOfficeTransaction.STG_TRADING_ACCOUNT_TXNS ● h2t_BackOfficeTransaction.STG_TRUSTS_TXNS
Borrower	BORROWER	t2t_Borrower	NA
Branch CTR Conductor	BRANCH_CTR_CNDTR	t2t_BranchCTRConductor	NA
Branch CTR Summary	BRANCH_CTR_SMRY	t2t_BranchCTRSummary	NA
Branch CTR Transaction	BRANCH_CTR_TRXN	t2t_BranchCTRTransaction	NA
Controlling Customer	CNTRL_CUST	t2t_ControllingCustomer	NA
Corporate Action	CORP_ACTN	t2t_CorporateAction	NA
Country	GEOGRAPHY	t2t_Country.STG_COUNTRY_MASTER	h2t_Country.STG_COUNTRY_MASTER

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Currency Transaction	CURRENCY_TRXN	● t2t_CurrencyTransaction.STG_ANNUITY_TXNS ● t2t_CurrencyTransaction.STG_BORROWINGS_TXNS ● t2t_CurrencyTransaction.STG_CARDS_PAYMENT_TXNS ● t2t_CurrencyTransaction.STG_CASA_TXNS ● t2t_CurrencyTransaction.STG_CUSTODIAN_ACCOUNT_TXNS ● t2t_CurrencyTransaction.STG_FOREX_TXNS ● t2t_CurrencyTransaction.STG_INTERBANK_TXNS ● t2t_CurrencyTransaction.STG_INVESTMENT_TXNS ● t2t_CurrencyTransaction.STG_LEASES_TXNS ● t2t_CurrencyTransaction.STG_LOAN_CONTRACT_TXNS ● t2t_CurrencyTransaction.STG_OD_ACCOUNTS_TXNS ● t2t_CurrencyTransaction.STG_OTHER_SERVICES_TXNS ● t2t_CurrencyTransaction.STG_RETIREMENT_ACCOUNTS_TXNS ● t2t_CurrencyTransaction.STG_SWAP_ACCOUNT_TXNS ● t2t_CurrencyTransaction.STG_TERMDEPOSITS_TXNS ● t2t_CurrencyTransaction.STG_TRADING_ACCOUNT_TXNS ● t2t_CurrencyTransaction.STG_TRUSTS_TXNS	NA
Customer	CUST	t2t_Customer.STG_PARTY_MASTER	h2t_Customer.STG_PARTY_MASTER
Customer Address	CUST_ADDR	t2t_CustomerAddress	h2t_CustomerAddress
Customer Country	CUST_CNTRY	● t2t_CustomerCountry ● t2t_CustomerCreditRating	h2t_CustomerCountry
Customer E-mail Address	CUST_EMAIL_ADDR	t2t_CustomerEmailAddress	h2t_CustomerEmailAddress

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Customer Identification Document	CUST_ID_DOC	t2t_CustomerIdentificationDocument	h2t_CustomerIdentificationDocument
Customer Import License		t2t_CustomerImportLicense	NA
Customer Import License to Goods		t2t_CustomerImportLicensetoGoods	NA
Customer Phone	CUST_PHON	t2t_CustomerPhone	h2t_CustomerPhone
Customer Supplemental Attribute	CUST_SUPPLEMENTAL_ATTR	t2t_CustomerSupplementalAttribute	h2t_CustomerSupplementalAttribute
Customer to Customer Relationship	CUST_CUST	t2t_CustomerToCustomerRelationship	h2t_CustomerToCustomerRelationship
Customer to Markets Served	CUST_MKT_SERVED	t2t_CustomerToMarketsServed	h2t_CustomerToMarketsServed
Customer to Products Offered	CUST_PRODUCT	t2t_CustomerToProductsOffered	h2t_CustomerToProductsOffered
Documentary Collection Contract Event	DOCMNTRY_COLL_CNTRCT_EVENT	t2t_DocumentaryCollectionContractEvent.STG_DOC_COLLCTN_CNTRCT_EVENT	NA
Documentary Collection Contract Acceptance Stage	DOCMNTRY_COLL_CNTRCT_ACC_STG	t2t_DocumentaryCollectionContractAcceptanceStage.STG_DOC_COLLCTN_CNTRCT_ACC	NA
Documentary Collection Contract Acknowledgement Stage	DOCMNTRY_COLL_CNTRCT_ACK_STG	t2t_DocCollectionContractAcknowlegementStage.STG_DOC_COLLCTN_CNTRCT_ACK	NA

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Documentary Collection Discrepancy Detail	DOCMNTRY_COLL_DISCRP_DTL	t2t_DocumentaryCollectionDiscrepancyDetail.STG_DOC_COLLCTN_DISCPNT_DTL	NA
Employee	EMP	t2t_Employee.STG_EMPLOYEE_MASTER	h2t_Employee.STG_EMPLOYEE_MASTER
Employee Address	EMP_ADDR	t2t_EmployeeAddress	h2t_EmployeeAddress
Employee Email Address	EMP_EMAIL_ADDR	t2t_EmployeeEmailAddress	h2t_EmployeeEmailAddress
Employee Phone	EMP_PHON	t2t_EmployeePhone	h2t_EmployeePhone
Employee to Account	EMP_ACCT	t2t_EmployeeToAccount	h2t_EmployeeToAccount
Employee Trading Restriction	EMP_SCRTY_RSTRN_LIST	t2t_EmployeeTradingRestriction.STG_EMPLOYEE_TRD_RESTRICTION	NA

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Front Office Transaction	FO_TRXN_STAGE	● t2t_FrontOfficeTransaction.STG_ANNUITY_TXNS ● t2t_FrontOfficeTransaction.STG_BILLS_CONTRACTS_TXNS ● t2t_FrontOfficeTransaction.STG_BORROWINGS_TXNS ● t2t_FrontOfficeTransaction.STG_CARDS_PAYMENT_TXNS ● t2t_FrontOfficeTransaction.STG_CARDS_SETTLEMENT_TXNS ● t2t_FrontOfficeTransaction.STG_CASA_TXNS ● t2t_FrontOfficeTransaction.STG_CORRESPONDENT_ACCT_TXNS ● t2t_FrontOfficeTransaction.STG_CUSTODIAN_ACCOUNT_TXNS ● t2t_FrontOfficeTransaction.STG_FOREX_TXNS ● t2t_FrontOfficeTransaction.STG_INTERBANK_TXNS ● t2t_FrontOfficeTransaction.STG_INVESTMENT_TXNS ● t2t_FrontOfficeTransaction.STG_LEASES_TXNS ● t2t_FrontOfficeTransaction.STG_LOAN_CONTRACT_TXNS ● t2t_FrontOfficeTransaction.STG_MERCHANT_CARDS_TXNS ● t2t_FrontOfficeTransaction.STG_MM_TXNS	● h2t_FrontOfficeTransaction.STG_ANNUITY_TXNS ● h2t_FrontOfficeTransaction.STG_BILLS_CONTRACTS_TXNS ● h2t_FrontOfficeTransaction.STG_BORROWINGS_TXNS ● h2t_FrontOfficeTransaction.STG_CARDS_PAYMENT_TXNS ● h2t_FrontOfficeTransaction.STG_CARDS_SETTLEMENT_TXNS ● h2t_FrontOfficeTransaction.STG_CASA_TXNS ● h2t_FrontOfficeTransaction.STG_CORRESPONDENT_ACCT_TXNS ● h2t_FrontOfficeTransaction.STG_CUSTODIAN_ACCOUNT_TXNS ● h2t_FrontOfficeTransaction.STG_FOREX_TXNS ● h2t_FrontOfficeTransaction.STG_INTERBANK_TXNS ● h2t_FrontOfficeTransaction.STG_INVESTMENT_TXNS ● h2t_FrontOfficeTransaction.STG_LEASES_TXNS ● h2t_FrontOfficeTransaction.STG_LOAN_CONTRACT_TXNS ● h2t_FrontOfficeTransaction.STG_MERCHANT_CARDS_TXNS ● h2t_FrontOfficeTransaction.STG_MM_TXNS

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
		● t2t_FrontOfficeTransaction.STG_OD_ACCOUNTS_TXNS ● t2t_FrontOfficeTransaction.STG_OTHER_SERVICES_TXNS ● t2t_FrontOfficeTransaction.STG_REPO_TRANSACTIONS ● t2t_FrontOfficeTransaction.STG_RETIREMENT_ACCOUNTS_TXNS ● t2t_FrontOfficeTransaction.STG_SWAP_ACCOUNT_TXNS ● t2t_FrontOfficeTransaction.STG_TERMDEPOSITS_TXNS ● t2t_FrontOfficeTransaction.STG_TRADING_ACCOUNT_TXNS ● t2t_FrontOfficeTransaction.STG_TRUSTS_TXNS	● h2t_FrontOfficeTransaction.STG_OD_ACCOUNTS_TXNS ● h2t_FrontOfficeTransaction.STG_OTHER_SERVICES_TXNS ● h2t_FrontOfficeTransaction.STG_REPO_TRANSACTIONS ● h2t_FrontOfficeTransaction.STG_RETIREMENT_ACCOUNTS_TXNS ● h2t_FrontOfficeTransaction.STG_SWAP_ACCOUNT_TXNS ● h2t_FrontOfficeTransaction.STG_TERMDEPOSITS_TXNS ● h2t_FrontOfficeTransaction.STG_TRADING_ACCOUNT_TXNS ● h2t_FrontOfficeTransaction.STG_TRUSTS_TXNS
Front Office Transaction Party	FO_TRXN_PARTY_STAGE	t2t_FrontOfficeTransactionParty	h2t_FrontOfficeTransactionParty
Inside Quote	BBO_STAGE	t2t_InsideQuote	NA
Insurance Policy	INSURANCE_POLICY	t2t_InsurancePolicy.STG_LIFE_INS_CONTRACTS	h2t_InsurancPolicy.STG_LIFE_INS_CONTRACTS
Insurance Policy Balance	INSURANCE_POLICY_BAL	t2t_InsurancePolicyBalance.STG_LIFE_INS_CONTRACTS	h2t_InsurancePolicyBalance.STG_LIFE_INS_CONTRACTS
Insurance Policy Feature	INSURANCE_POLICY_FEATURE	t2t_InsurancePolicyFeature	h2t_InsurancePolicyFeature
Insurance Policy To Customer	INSURANCE_POLICY_CUST	t2t_InsurancePolicyToCustomer	h2t_InsurancePolicyToCustomer
Insurance Product	INSURANCE_PRODUCT	t2t_InsuranceProduct	h2t_InsuranceProduct
Insurance Seller	INSURANCE_SELLER	t2t_InsuranceSeller	h2t_InsuranceSeller
Insurance Seller To License	INSURANCE_SELLER_LICENSE	t2t_InsuranceSellerToLicense.STG_INS_SELLER_LICENSE	h2t_InsuranceSellerToLicense.STG_INS_SELLER_LICENSE
Insurance Transaction	INSURANCE_TRXN	t2t_InsuranceTransaction.STG_LIFE_INS_POLICY_TXNS	h2t_InsuranceTransaction.STG_LIFE_INS_POLICY_TXNS
Investment Advisor	NVSMT_MGR	t2t_InvestmentAdvisor	NA

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Issuer	ISSUER	t2t_Issuer.STG_PARTY_MASTER	NA
Loan	LOAN	t2t_Loan.STG_LOAN_CONTRACTS	● h2t_Loan.STG_LOAN_CONTRACTS ● h2t_Loan.STG_CARDS
Loan Daily Activity	LOAN_SMRY_MNTH_STAGE	t2t_LoanDailyActivity.STG_LOAN_CONTRACTS	h2t_LoanDailyActivity.STG_LOAN_CONTRACTS
Loan Origination Document Print Log	LOAN_ORIG_DOC_PRINT_LOG	t2t_LoanOriginationDocumentPrintLog	NA
Loan Product	LOAN_PRODUCT	t2t_LoanProduct	h2t_LoanProduct
Managed Account	MANGD_ACCT	t2t_ManagedAccount	NA
Market Center	MARKET_CENTER	t2t_MarketCenter.STG_MARKET_CENTER_MASTER	NA
Market Center Quote	QUOTE_STAGE	t2t_MarketCenterQuote	NA
Market Index	MKT_IDX	t2t_MarketIndex	NA
Market Index Daily	MKT_IDX_DAILY	t2t_MarketIndexDaily	NA
Market Index Member Security	MKT_IDX_MBR_SCRTY	t2t_MarketIndexMemberSecurity	NA
Market News Event	DOW_JONES_NEWS	t2t_MarketNewsEvent.STG_MARKET_NEWS_EVENT	NA
Mutual Fund Breakpoint	MFUND_BRKPT	t2t_MutualFundBreakpoint	NA
Online Account	ONLINE_ACCT	t2t_OnlineAccount	h2t_OnlineAccount
Online Account To Account	ONLINE_ACCT_ACCT	t2t_OnlineAccountToAccount	h2t_OnlineAccountToAccount

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Organization	ORG	● t2t_Organization.STG_GEOGRAPHY_MASTER ● t2t_Organization.STG_ORG_STRUCTURE_MASTER ● t2t_Organization.STG_ORG_UNIT_MASTER ● t2t_Organization.STG_TRADING_DESK_MASTER	NA
Organization Relationship	ORG_RLSHP	t2t_OrganizationRelationship.STG_ORG_STRUCTURE_MASTER	NA
Peer Group	PEER_GRP	t2t_PeerGroup	h2t_PeerGroup
Reported Market Sale	REPORTED_SALE_STAGE	t2t_ReportedMarketSale	NA
Restriction List	RSTRN_LIST	t2t_RestrictionList	NA
Security	SCRTY	t2t_Security.STG_INSTRUMENT_CONTRACT_MASTER	NA
Security Firm Daily	SCRTY_FIRM_DAILY	t2t_SecurityFirmDaily	NA
Security Group Member	RLTD_SCRTY	t2t_SecurityGroupMember	NA
Security Investment Rating	SCRTY_NVSMT_RTNG	t2t_SecurityInvestmentRating	NA
Security Market Daily	SCRTY_MKT_DAILY	t2t_SecurityMarketDaily.STG_INSTRUMENT_MARKET_PRICES	NA
Security Trading Restriction	SCRTY_RSTRN_LIST	t2t_SecurityTradingRestriction.STG_INST_TRADE_RESTRICTION	NA
Service Team	ACCT_SRVC_TEAM	t2t_ServiceTeam	NA
Service Team Member	ACCT_SRVC_TEAM_MEMBER	t2t_ServiceTeamMember	NA
Settlement Instruction	INSTRUCTION	t2t_SettlementInstruction.STG_SETTLEMENT_INSTRUCTION	NA
Structured Deal	DEAL	t2t_StructuredDeal	NA

Interface File Name	Data Quality Group Name	Corresponding T2T Name	Corresponding H2T Name
Trade Finance Contract	TRADE_FIN_CNTRCT_EVENT	t2t_TradeFinanceContractAmendmentStatusStage.STG_TRD_FIN_CNTRCT_AMD_STATUS	NA
Trade Finance Contract Amendment Status	TRD_FIN_CNTRCT_AMD_STATUS_STG	t2t_TradeFinanceContractEventAcknowledgementStage.STG_TRD_FIN_CNTRCT_EVENT_ACK	NA
Trade Finance Contract Event Acknowledgement	TRD_FIN_CNTRCT_EVENT_ACK_STG	t2t_TradeFinanceContractEvent.STG_TRADE_FIN_CNTRCT_EVENT	NA
Trade Finance Document	TRADE_FIN_DOC	t2t_TradeFinanceDocument.STG_TRADE_FINANCE_DOCUMENT	NA
Trade Finance Draft		t2t_TradeFinanceDraft	NA
Trade Finance Good or Service	TRADE_FIN_GOOD_SRVC	t2t_TradeFinanceGoodorService.STG_TRADE_FINANCE_GOODS	NA
Trade Finance Party	TRADE_FIN_PARTY_STG	t2t_TradeFinanceParty.STG_TRADE_FINANCE_PARTY_EVENT	NA
Trade Finance to Account	TRADE_FIN_ACCT	t2t_TradeFinancetoAccount.STG_TRADE_FINANCE_ACCT	NA
Watch List	WATCH_LIST_SOURCE	t2t_WatchList	h2t_WatchList
Watch List Entry	WATCH_LIST	t2t_WatchListEntry	h2t_WatchListEntry

Group Dependencies

Processing data in Group1 requires no prerequisite information (dependencies) for Pre-processing. Groups 2-5, however, rely on successful pre-processing of the previous group to satisfy any dependencies. For example, the 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 the 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.

Flat File Ingestion

This section refers to Behavior Detection (BD) Ingestion Flat Files and covers the following topics:

- BDF.xml File Parameters
- Behavior Detection Flat File Interface

BDF.xml File Parameters

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

Table 100. Parameters Related to Processing DIS Files

Property Name	Description	Default
DIS.Source	Indicates the source of DIS records. Valid values are: ● FILE for a DIS file ● FSDW for CSA table loading ● FILE-EXT for loading DIS file using an external table	FILE
DIS.ArchiveFlag	Indicates whether a DIS file should be archived after it has been processed.	true
DIS.BufferSize	Indicates the size of a byte buffer (in kilobytes) used to read in a line from a DIS file. This should be set to the maximum possible record size (in kilobytes) of a record in a DIS file.	100
DIS.InputFileCharset	Indicates the character set of a DIS file.	UTF8
DIS.Default.Check.Requirement	Indicates whether the mandatory and conditional checks on a DIS record should be done	true
DIS.Default.Reject.Requirement	Indicates whether a mandatory or conditional check failure for a record should result in the record being rejected. If this is set to FALSE and a missing value is attempted to be inserted into a NOT NULL column, then the record will be rejected anyway.	true
DIS.Default.Check.Domain	Indicates whether the domain value checks on a DIS record should be done.	true
DIS.Default.Reject.Domain	Indicates whether a domain value check failure for a record should result in the record being rejected.	true
DIS.Default.Check.Length	Indicates whether the maximum length checks on a DIS record should be done.	true
DIS.Default.Reject.Length	Indicates whether a maximum length check failure for a record should result in the record being rejected. If this is set to FALSE, then the value will be truncated based on the maximum length of the field.	true
DIS.Default.Check.Threshold	Indicates whether the threshold checks (GREATER_THAN_ZERO, etc) on a DIS record should be done.	true
DIS.Default.Reject.Threshold	Indicates whether a threshold check failure for a record should result in the record being rejected.	true

Property Name	Description	Default
DIS.Default.Check.Lookup	Indicates whether the reference data lookups on a DIS record should be done.	true
DIS.Default.Reject.Lookup	Indicates whether a reference data lookup failure for a record should result in the record being rejected.	true
MITrxnProducttypes	Indicates the parameter which is used to pass a list of product codes for trailing digit purpose (AUG_INSTR_NB derivation).	● CHECK ● CHECK-ACH
CustProfileLookBack	Indicates the parameter which is used to look back at the days in Customer Summary Daily for Customer Summary Month recalculation. **Note:** In order to look back at a specific time period in Customer Summary Daily, you must have partitions available in Customer Summary Month.	31
CustAcctHolderType	Indicates the parameter which is used to identify customer account types to be included in customer summary.	CI

BD Ingest DIS Data Files by Group

Ingestion Manager processes data files in groups (in a specified order) from Oracle client data in the /inbox directory. The following list of files can be run using CSA in FSDF or Hive. Files have been grouped in such a way that files in the same group can be executed in parallel to load data. However, you must execute Group 1 through Group 6 in sequence.The following table lists the data files by group.

Table 101. BD Ingest DIS Data Files By Group

Group	Data Files	
1.	Account Phone Watch List Account Emai lAddress Insurance Product Insurance Policy Insurance Transaction Insurance Policy Balance Front Office Transaction Account Customer Role Organization Insurance Policy Feature Market Center	Insurance Policy To Customer Market Index Daily Loan Issuer Loan Daily Activity Market Index Online Account Service Team Insurance Seller Service Team Member Insurance Seller To License Country
2.	Account To Peer Group Account Group Peer Group Security Firm Daily	Market Index Member Security Security Market Daily Security
3.	Account Customer Watch List Entry Loan Product Employee	Front Office Transaction Party Organization Relationship Restriction List Automated Quote

Table 101. BD Ingest DIS Data Files By Group

Group	Data Files	
4.	Managed Account Account To Customer Account Group Member Account To Correspondent Account Balance Account Address Customer To Markets Served Customer To Products Offered Customer To Customer Relationship Anticipatory Profile Customer Phone Customer Email Address Customer Country Customer Address Online Account To Account	Controlling Customer Employee To Account Account Position Security Trading Restriction Employee Trading Restriction Employee Phone Employee Email Address Employee Address Security Group Member Security Investment Rating Structured Deal Account Profit And Loss Account Investment Objective Account Position Pair Mutual Fund Breakpoint Market News Event
5.	Borrower Account Restriction Back Office Transaction	Investment Advisor Settlement Instruction Loan Origination Document Print Log
6.	OpenOrder Order	TradeExecutionEvent

Behavior Detection Flat File Interface

The following tables describe the Ingestion Flat File details for products within the BD Application Pack. Files have been grouped in such a way that files in the same group can be executed in parallel to load data. However, you must execute Group 1 through Group 5 in sequence. For more information, see *List of Data Quality Group names, T2T and H2T names*

The Staging Representation column indicates whether this file requires a Staging source.

The following table describes the Group 1 Ingestion Flat File details.

Table 102. Group 1 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represent ation	T2T	H2T
Account Phone	X	X	X	X						BD Datamaps	Yes	Yes	Yes
Account Email Address	X	X	X	X						BD Datamaps	Yes	Yes	Yes
Insurance Policy	X	X	X							BD Datamaps	Yes	Yes	Yes
Insurance Policy Balance	X	X								BD Datamaps	Yes	Yes	Yes

Table 102. Group 1 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Account Customer Role	X	X		X	X					BD Datamaps	Yes	Yes	Yes
Insurance Policy Feature	X	X								BD Datamaps	Yes	Yes	Yes
Insurance Policy to Customer	X	X	X							BD Datamaps	Yes	Yes	Yes
Loan	X	X								BD Datamaps	Yes	Yes	Yes
Loan Daily Activity	X	X								BD Datamaps	Yes	Yes	Yes
Online Account	X	X								BD Datamaps	Yes	Yes	Yes
Insurance Seller	X	X								BD Datamaps	Yes	Yes	Yes
Insurance Seller to License	X	X								BD Datamaps	Yes	Yes	Yes
Country	X	X		X						BD Datamaps	Yes	Yes	Yes
Watch List	X	X	X							BD Datamaps	Yes	Yes	Yes
Insurance Product	X	X	X							BD Datamaps	Yes	Yes	Yes
Insurance Transaction	X	X								BD Datamaps	Yes	Yes	Yes
Front Office Transaction	X	X								BD Datamaps	Yes	Yes	Yes
Organization					X	X		X		BD Datamaps	Yes	No	No
Market Center						X				BD Datamaps	Yes	No	No
Market Index Daily						X				BD Datamaps	Yes	No	No
Issuer						X				BD Datamaps	Yes	No	No
Market Index						X				BD Datamaps	Yes	No	No
Service Team Member								X		BD Datamaps	Yes	No	No
Service Team								X		BD Datamaps	Yes	No	No

Table 102. Group 1 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
CTR Transaction	X	X			X					runDP/runDL	No	No	No
Account Realized Profit and Loss								X		runDP/runDL	No	No	No
Letter of Intent								X		runDP/runDL	No	No	No
Collateral Value-Currency								X		runDP/runDL	No	No	No
Collateral Value-Product								X		runDP/runDL	No	No	No
Commission Product								X		runDP/runDL	No	No	No
Compliant Registration								X		runDP/runDL	No	No	No
Complaint Type Rating								X		runDP/runDL	No	No	No
Employee to Insurance Policy								X		runDP/runDL	No	No	No
Investment Guideline								X		runDP/runDL	No	No	No
Investment Guideline to Account								X		runDP/runDL	No	No	No
System Logon Type								X		runDP/runDL	No	No	No
Registered Representative Complaint								X		runDP/runDL	No	No	No
Energy And Commodity Instrument									X	runDP/runDL	No	No	No

The following table describes the Group 2 Ingestion Flat File details.

Table 103. Group 2 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Account to Peer Group	X	X	X							BD Datamaps	Yes	Yes	Yes
Account Group	X	X								BD Datamaps	Yes	Yes	Yes
Peer Group	X	X	X							BD Datamaps	Yes	Yes	Yes
Security Market Daily						X				BD Datamaps	Yes	No	No
Security Firm Daily						X				BD Datamaps	Yes	No	No
Security						X	X			BD Datamaps	Yes	No	No
Market Index Member Security						X				BD Datamaps	Yes	No	No
Security Market State Change						X				BD Datamaps	Yes	No	No
Matched Entity	X	X								runDP/runDL	No	No	No
Trusted Pair	X	X								BD Datamaps	Yes	No	No
Firm Account Position Pair						X		X		runDP/runDL	No	No	No
Natural Gas Flow									X	runDP/runDL	No	No	No

The following table describes the Group 3 Ingestion Flat File details.

Table 104. Group 3 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Account	X	X	X	X	X					BD Datamaps	Yes	Yes	Yes
Customer	X	X	X	X	X					BD Datamaps	Yes	Yes	Yes

Table 104. Group 3 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Watch List Entry	X	X	X							BD Datamaps	Yes	Yes	Yes
Loan Product	X	X								BD Datamaps	Yes	Yes	Yes
Employee	X	X								BD Datamaps	Yes	Yes	Yes
Front Office Transaction Party	X	X								BD Datamaps	Yes	Yes	Yes
Organization Relationship					X	X		X		BD Datamaps	Yes	No	No
Restriction List						X				BD Datamaps	Yes	No	No
Automated Quote						X				BD Datamaps	Yes	No	No
Account Supplemental Attribute			X							runDP/runDL	No	No	No
Customer Supplemental Attribute			X							runDP/runDL	No	Yes	Yes
Market Trading Session						X				runDP/runDL	No	No	No
Account GroupAddress	X	X								runDP/runDL	No	No	No
Account Group Investment Objective								X		runDP/runDL	No	No	No
Account Group IOS Member								X		runDP/runDL	No	No	No
Account Group Member Experience								X		runDP/runDL	No	No	No
Loan Origination Action								X		runDP/runDL	No	No	No
Mail Handling Instruction Activity								X		runDP/runDL	No	No	No

Table 104. Group 3 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Repres entation	T2T	H2T
Banker To Officer								X		runDP/runD L	No	No	No
Reference Table Detail								X		runDP/runD L	No	No	No
General Usage List								X		runDP/runD L	No	No	No
Loan Origination Product								X		runDP/runD L	No	No	No
Organization To Mortgage Type								X		runDP/runD L	No	No	No
Securities License								X		runDP/runD L	No	No	No
Service Vendor								X		runDP/runD L	No	No	No
Energy and Commodity Trade									X	runDP/runD L	No	No	No

The following table describes the Group 4 Ingestion Flat File details.

Table 105. Group 4 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
Market News Event						X				BD Datamaps	Yes	No	No
Managed Account	X	X	X							BD Datamaps	Yes	No	No
Account To Customer	X	X	X	X	X					BD Datamaps	Yes	Yes	Yes
Branch CTR Transaction					X					BD Datamaps	Yes	No	No
Branch CTR Conductor					X					BD Datamaps	Yes	No	No
Branch CTR Summary					X					BD Datamaps	Yes	No	No
Account Group Member	X	X								BD Datamaps	Yes	Yes	Yes

Table 105. Group 4 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Account To Correspondent	X	X								BD Datamaps	Yes	Yes	Yes
Account Balance	X	X	X	X						BD Datamaps	Yes	Yes	Yes
Account Address	X	X	X	X						BD Datamaps	Yes	Yes	Yes
Customer Identification Document	X	X	X	X						BD Datamaps	Yes	Yes	Yes
Customer To Markets Served	X	X	X							BD Datamaps	Yes	Yes	Yes
Customer To Products Offered	X	X	X							BD Datamaps	Yes	Yes	Yes
Customer To Customer Relationship	X	X	X	X						BD Datamaps	Yes	Yes	Yes
Anticipatory Profile	X	X	X							BD Datamaps	Yes	Yes	Yes
Customer Phone	X	X	X	X	X					BD Datamaps	Yes	Yes	Yes
Customer Email Address	X	X	X	X	X					BD Datamaps	Yes	Yes	Yes
Customer Country	X	X	X							BD Datamaps	Yes	Yes	Yes
Customer Address	X	X	X	X	X					BD Datamaps	Yes	Yes	Yes
Online Account to Account	X	X	X							BD Datamaps	Yes	No	No
Controlling Customer	X	X								BD Datamaps	Yes	No	No
Employee To Account	X	X								BD Datamaps	Yes	Yes	Yes
Account Position						X		X		BD Datamaps	Yes	No	No
Security Trading Restriction						X	X			BD Datamaps	Yes	No	No

Table 105. Group 4 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
Employee Trading Restriction						X	X			BD Datamaps	Yes	No	No
Employee Phone						X				BD Datamaps	Yes	Yes	Yes
Employee Email Address						X	X			BD Datamaps	Yes	Yes	Yes
Employee Address						X				BD Datamaps	Yes	Yes	Yes
Outside Business Activity										BD Datamaps	Yes	No	No
Private Security Transaction										BD Datamaps	Yes	No	No
Security Group Member						X				BD Datamaps	Yes	No	No
Security Investment Rating						X				BD Datamaps	Yes	No	No
Structured Deal						X				BD Datamaps	Yes	No	No
Account Profit and Loss								X		BD Datamaps	Yes	No	No
Account Position Pair								X		BD Datamaps	Yes	No	No
Account Investment Objective								X		BD Datamaps	Yes	No	No
Mutual Fund Breakpoint								X		BD Datamaps	Yes	No	No
Account Feature								X		runDP/runDL	No	No	No
Access Events		X								runDP/runDL	No	No	No
Customer Balance		X								runDP/runDL	No	No	No
Front Office Transaction Remittance Document	X	X								runDP/runDL	No	No	No

Table 105. Group 4 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Related Front Office Transaction Information	X	X								runDP/runDL	No	No	No
Account To Organization						X		X		runDP/runDL	No	No	No
Firm Account Position						X		X		runDP/runDL	No	No	No
External Investment Account Position							X	X		runDP/runDL	No	No	No
Employee To Organization							X	X		runDP/runDL	No	No	No
Security Select List Entry						X				runDP/runDL	No	No	No
Account Fees								X		runDP/runDL	No	No	No
Account Profile Stage								X		runDP/runDL	No	No	No
Account Qualification Agreement								X		runDP/runDL	No	No	No
Account Representative Position								X		runDP/runDL	No	No	No
Account Asset Allocation								X		runDP/runDL	No	No	No
Account Scheduled Event								X		runDP/runDL	No	No	No
Account Identifier Change History								X		runDP/runDL	No	No	No

Table 105. Group 4 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
Account Position Profile And Loss								X		runDP/runDL	No	No	No
Uncovered Option Account Position								X		runDP/runDL	No	No	No
Account Collateral								X		runDP/runDL	No	No	No
Mail Handling Instruction								X		runDP/runDL	No	No	No
Mutual Fund Family Letter of Intent								X		runDP/runDL	No	No	No
Employee Disciplinary Action								X		runDP/runDL	No	No	No
Employee Exam History								X		runDP/runDL	No	No	No
Employee Firm Transfer History								X		runDP/runDL	No	No	No
Employee Securities License State Registration								X		runDP/runDL	No	No	No
Employee Supervision List								X		runDP/runDL	No	No	No
Employee To Manager History								X		runDP/runDL	No	No	No
Employee To Securities License								X		runDP/runDL	No	No	No
Employmen t History								X		runDP/runDL	No	No	No
System Logon								X		runDP/runDL	No	No	No

Table 105. Group 4 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
Plan of Solicitation								X		runDP/runDL	No	No	No
Mutual Fund Family Configurati on								X		runDP/runDL	No	No	No
Energy And Commodity Market Daily									X	runDP/runDL	No	No	No
Energy And Commodity Firm Daily									X	runDP/runDL	No	No	No
Energy And Commodity Reported Market Sale									X	runDP/runDL	No	No	No
Energy And Commodity Market Trading Session									X	runDP/runDL	No	No	No
Energy And Commodity Market Center									X	runDP/runDL	No	No	No
Energy And Commodity Location									X	runDP/runDL	No	No	No
Energy Flow Mode									X	runDP/runDL	No	No	No
Energy and Commodity Instrument Position									X	runDP/runDL	No	No	No

The following table describes the Group 5 Ingestion Flat File details.

Table 106. Group 5 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Borrower	X	X								BD Datamaps	Yes	No	No
Back Office Transaction	X	X								BD Datamaps	Yes	Yes	Yes
Account Restriction			X			X				BD Datamaps	Yes	No	No
Investment Advisor						X				BD Datamaps	Yes	No	No
Investment Guideline Override										BD Datamaps	Yes	No	No
Settlement Instruction						X				BD Datamaps	Yes	No	No
Loan Origination Document Print Log								X		BD Datamaps	Yes	No	No
Change Log	X	X	X	X						runDP/runDL	No	No	No
Options Violation								X		runDP/runDL	No	No	No
Loan Origination Condition								X		runDP/runDL	No	No	No
Loan Origination Fee Detail								X		runDP/runDL	No	No	No
Loan Origination Note								X		runDP/runDL	No	No	No
Loan Origination To Service								X		runDP/runDL	No	No	No
Investment Guideline Override								X		runDP/runDL	No	No	No
Loan Origination Condition Type								X		runDP/runDL	No	No	No
System Logon To System Logon Type								X		runDP/runDL	No	No	No

Table 106. Group 5 Interface Ingestion Flat Files

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
System Logon To Organization								X		runDP/runDL	No	No	No
Registered Representative Account Commission								X		runDP/runDL	No	No	No
Registered Representative Account Commission Prior Year								X		runDP/runDL	No	No	No
Registered Representative Commission Monthly Profile								X		runDP/runDL	No	No	No
Registered Representative Commission Product								X		runDP/runDL	No	No	No
Currency Transaction					X					BD Datamaps	Yes	No	No

The following table describes the Group 6 Ingestion Flat File details.

Table 107. Group 6 Interface Ingestion for Market Data

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Representation	T2T	H2T
Inside Quote						X				BD Datamaps	Yes	No	No
Market Center Quote						X				BD Datamaps	Yes	No	No
ReportedMarketSale						X				BD Datamaps	Yes	No	No
InsideQuote_Derived						X				BD Datamaps	Yes	No	No

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
MarketCenterQuote_Derived						X				BD Datamaps	Yes	No	No
ReportedMarketSale_Derived						X				BD Datamaps	Yes	No	No

The following table describes the Group 7 Ingestion Flat File details.

Table 108. Group 7 Interface Ingestion for Trade Finance Data

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
TradeFinanceContractEventAcknowledgement						X				BD Datamaps	Yes	No	No
TradeFinanceContractAmendmentStatus						X				BD Datamaps	Yes	No	No
TradeFinanceContract						X				BD Datamaps	Yes	No	No
TradeFinancetoAccount						X				BD Datamaps	Yes	No	No
TradeFinanceDocument						X				BD Datamaps	Yes	No	No
TradeFinanceGoodorService						X				BD Datamaps	Yes	No	No
TradeFinanceParty						X				BD Datamaps	Yes	No	No
DocCollectionContractAcknowlegementStage						X				BD Datamaps	Yes	No	No
DocumentaryCollectionContractAcceptaceStage						X				BD Datamaps	Yes	No	No

Interface File Name	AML	Fraud	KYC	FATCA	CTR	TC	PTA	BC	ECTC	Current Ingestion	Staging Represe ntation	T2T	H2T
DocumentaryCollectionDiscrepancyDetail						X				BD Datamaps	Yes	No	No
DocumentaryCollectionContractEvent						X				BD Datamaps	Yes	No	No

Note: The AccountAverageNetWorth file is an exceptional case, and is only intended to be run once before any other files have been loaded. The average net worth amount in the account profile table is built up over time as transactions are ingested. This file allows this value to be set as a starting point before any transactions have been ingested. After transactions are ingested, this file should no longer be used.

Note: The following derived datamaps must be run after running the corresponding BD scripts.
CurrencyTransaction_ExemptFlagUpd
SecurityInvestmentRating_PrevInvestmentUpd
AutomatedQuote_SecurityUpd

For Example:
AutomatedQuote_SecurityUpd should be run after <OFSAAI Installed Directory>/BDF/scripts/execute.sh AutomatedQuote as <OFSAAI Installed Directory>/BDF/scripts/execute.sh AutomatedQuote_SecurityUpd

Pre-processing & Loading Directory Structure

Data for Pre-processing & Loading are organized in subdirectories below the ingestion_manager root level. Figure 54illustrates the subdirectories that the ingestion_manager directory contains.

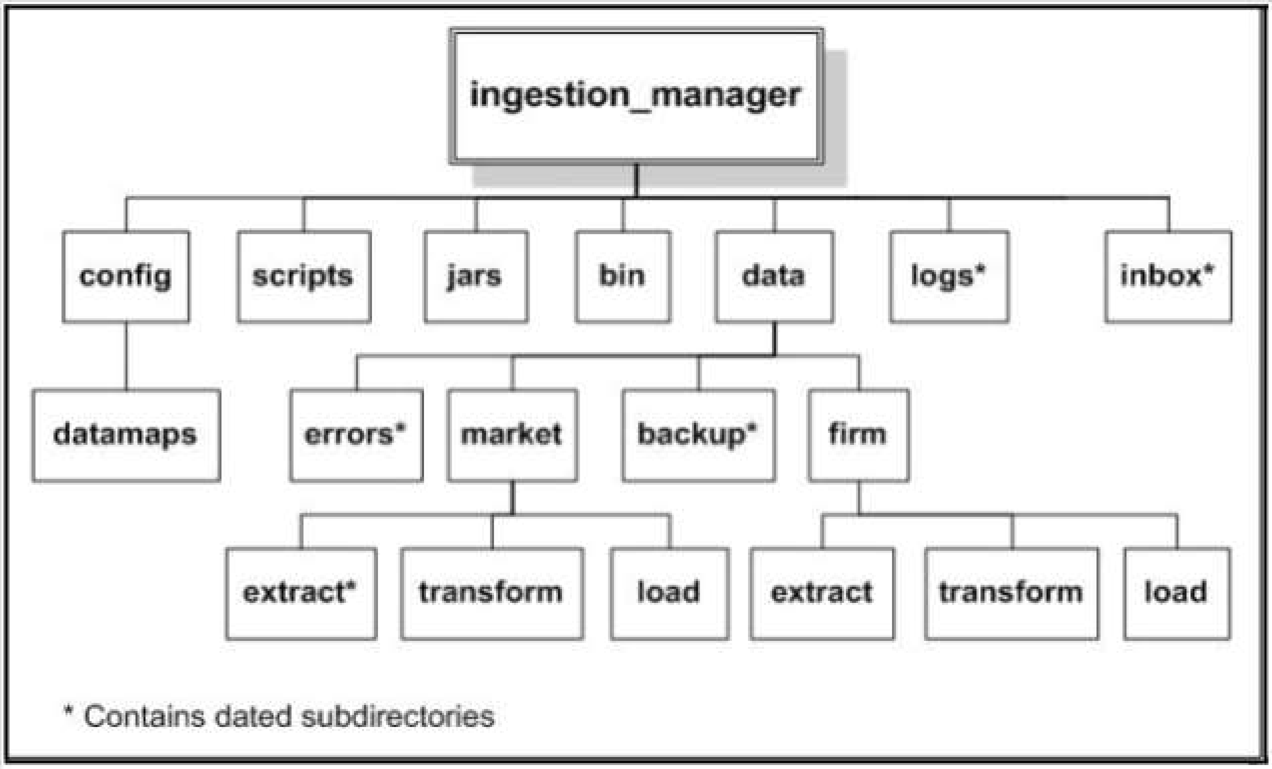


Figure 54. Data Management Subsystem Directory Structure

Directory Structure Descriptions

The following table lists important subdirectories that compose the `<OFSAAI Installed Directory>/ingestion_manager` directory structure.

Table 109. Data Management Directory Structure Description

Directory Name	Description
config	Contains files used to configure the Data Management components (see *config Subdirectory* for more information).
data/backup	Contains backup files for the various Data Management components (see *data/backup Subdirectory* for more information).
data/errors	Contains error files for various Data Management components (see *data/errors Subdirectory* for more information).
data/firm	Contains Oracle client data files that Data Management components write (see *data/firm Subdirectory*for more information).
inbox	Contains data files that the Oracle client provides (see *inbox Subdirectory*for more information).

Table 109. Data Management Directory Structure Description

Directory Name	Description
jars	Contains the Java Archive (JAR) files to run Java Data Management components implemented in Java (see *jars Subdirectory* for more information).
logs	Contains log files that Data Management components write (see *logs Subdirectory* for more information).
scripts	Contains all the shell scripts for running Data Management components (see *scripts Subdirectory* for more information).
/inbox/<yyyymmdd>	Backup of input files (for restart purposes, if necessary).
/data/<firm or market>/load	● for loading into the database as <data type>_<yyyymmdd>_<batch_name>_<N>.XDP. ● Load control files.
/logs/<yyyymmdd>	Pre-processing 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.

This section covers the following topics:

- jars Subdirectory
- scripts Subdirectory
- data Subdirectory
- extract Subdirectory
- transform Subdirectory
- load Subdirectory
- inbox Subdirectory
- logs Subdirectory

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 (see *scripts Subdirectory* for more information).

scripts Subdirectory

The scripts subdirectory within the ingestion_manager directory contains the UNIX Bourne Shell scripts to run runtime components. Executing a run script runs a new instance of a component. 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). The following table defines the run scripts for starting each component and any special instructions.

Table 110. Run Scripts by Component

Script Names	Description or Special Instructions
runDP.sh <data type>	Launches an instance of the data Pre-processor (runDP.sh). For example: runDP.sh Customer To run a specific Data Pre-processor, specify a valid input component that the run 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, see Figure 55
runFDT.sh	Launches the FDT. This script 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). For example: runDL.sh Customer To run a specific data loader, specify a valid component that the run 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, see *Figure 55*.
runRebuildIndexes.sh	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.
runIMC.sh	Launches the Ingestion Manager Cleaner (IMC) utility. The utility terminates after it finishes removing old data subdirectories and their contents.
env.sh	Contains common configuration settings required to run Data Management subsystem components. The run*.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 must run runDL.sh <data type> after this script has executed successfully. For example: runUtility.sh AccountDailySecurityProfile runDL.sh AccountDailySecurityProfile

The run scripts in Table 111 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 111 describes.

Table 111. 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.
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 too small may result in component failure.

Note: Default values that are appropriate to the operating system in use , such as 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 112 describes:

- DataIngestCustom.xml (see section *Data Ingest XML Configuration File* for more information).
- DataIngest.properties (see section *Data Ingest Properties Configuration File* for more information).
- DataIngest.xml (see section *Data Ingest XML Configuration File* 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 112. 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 (see Table 113).
DataIngest.xml	XML configuration file that contains settings that normally remain as is (see Table 114).
DataIngestCustom.xml	XML configuration file that contains overridden settings from DataIngest.xml.

The following sections describe each of these configuration files:

Data Ingest Properties Configuration File

The following table describes the parameters for the `DataIngest.properties` configuration file.

Table 113. 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. Oracle recommends that you use Thin Driver.	jdbc:oracle:thin:@ofss220074.in.oracle.com:1521:Ti1O11L56
`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 Management requires for interacting with the database.	ATOMIC
`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 ATOMIC database schema. Data Management accesses the ATOMIC schema when allocating sequence IDs to ingested records.	ATOMIC
`MARKET.DBSchema`	Schema name for the ATOMIC database schema. Data Management stores market data related records in the ATOMIC schema.	ATOMIC
`BUSINESS.DBSchema`	Schema name for the ATOMIC database schema. Data Management stores market data related records in the ATOMIC schema.	ATOMIC

Data Ingest XML Configuration File

The following table 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 114. 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 Management. This property should be blank during normal operation.	
Miscellaneous		
`DefaultSourceSystem.value`	Indicates the default value to use for source system when manufacturing reference data records.	`MTS`
`BufferSize.value`	Specifies the buffer size in kilobytes for I/O byte buffers that the MDS and FDT processes create to read input files. Use care when changing this parameter due to impact on performance and memory requirements.	`1024`
`DirectBufferSize.value`	Specifies the buffer size in kilobytes for Java NIO direct byte buffers that the MDS, MDT, and FDT processes create to read input files. Use care when changing this parameter due to impact on performance and memory requirements	`1024`
`DefaultCurrency.value`	Indicates the value to use as the issuing currency when manufacturing security records from order or trade execution records.	`USD`
`UseDirectBuffers.value`	Specifies whether to make use of Java NIO's direct buffer mechanism.	`TRUE`

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`Separator.value`	Specifies the delimiter that separates fields in data file records.	`~`
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`
`DB.Connection.NumRetries`	Specifies the maximum number of times to attempt to connect to a database before failing.	`5`
BUSINESS: Specifies properties related to data loaded into the ATOMIC schema.		
`BUSINESS.ExtractDir`	Identifies the parent directory for intermediate files that Pre-processors produce that are applicable to the ATOMIC 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 ATOMIC 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`

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
MANTAS: Specifies properties related to data loaded into the ATOMIC schema.		
MANTAS.ExtractDir	Specifies the parent directory for intermediate files that Pre-processors produce that are applicable to the ATOMIC schema in the database.	../data/mantas/extract
MANTAS.TransformDir	Specifies the working directory for intermediate files that utilities produce that are applicable to the ATOMIC schema in the database.	../data/mantas/transform
MANTAS.LoadDir	Specifies the parent directory for directories that store ATOMIC schema bound data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	../data/mantas/load
Directory: Specifies properties used to define directory locations.		
Directory.Log	Specifies the parent directory for log file directories and log files that Java ingestion components create.	../logs
Directory.Inbox	Specifies the input directory where Java ingestion components find files that the Oracle 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	Specifies 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

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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
Security: Specifies properties used to produce security reference data.		
Security.AdditionalColumns	Specifies additional columns of data that ingestion components must 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
FDT: Specifies properties used to configure the FDT component.		

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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.OrderEvent	Specifies the batch size when retrieving sequence IDs for OrderEvent records (during end-of-day processing).	1000
FDT.SequenceBatchSize.Order	Specifies the batch size when retrieving sequence IDs for Order records.	10000
FDT.SequenceBatchSize.Trade	Specifies the batch size when retrieving sequence IDs for Trade records.	10000
FDT.SequenceBatchSize.Execution	Specifies the batch size when retrieving sequence IDs for Execution records.	10000
FDT.SequenceBatchSize.DerivedTrade	Specifies the batch size when retrieving sequence IDs for DerivedTrade records.	10000
FDT.MarketDataSource.Value	Specifies the source of market data. Valid values are File for file based access or RMI for access using an RMI server (not recommended for performance reasons).	File
FDT.CalculateDisplayability.Value	Specifies whether to calculate displayability states.	FALSE
FDT.ExplainableCancelCodes.Value	Specifies a comma-separated list of explainable cancellation codes.	
FDT.BufferSize.value	Allows an override to the BufferSize.value property for FDT.	
FDT.LookForFutureEventTimes.value		
FDT.UsePrevailingSale.value	Specifies whether to use the prevailing reported market sales price as an execution’s expected print price when no comparable market sales occur during the order’s marketable periods.	FALSE
Data Management 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 Management 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.		

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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).	`TRUE`
`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 (see *Rejection During the Transformation Stage*, 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`

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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 Trade Blotter file.	FALSE
FDT.EnableMIFID.value	Identifies whether MiFid related data will be provided (TRUE) or not (FALSE).	FALSE
FDT.IgnoreFutureMarketRefs.value	Identifies whether the FDT will use Reported Market Sales records that occur later in time than a given trade when calculating the market reference price for that trade (FALSE) or not (TRUE).	FALSE
FDT.MaxFutureMarketRefCompTime.value	Specifies the number of seconds from the time a trade occurs during which any reported sales records cannot have the same price and quantity as the given trade to be considered as a market reference price. -1 means that this condition will not apply, 0 means the condition applies to reported sales with the same time, 5 means the condition applies to reported sales within 5 seconds of the trade, and so on. This parameter is only used if FDT.IgnoreFutureMarketRefs.value = FALSE.	-1
The next four parameters are used to generate records in the TRADE_TRXN_CORRECTION table, which record when a correction to a field of an execution, trade, or order occurs. The fields to be checked for corrections should be specified in a comma separated list of business field names. Business field names can be found in the corresponding XML data map file in the datamaps directory.		
FDT.DeriveCorrectionFields.Trade	Specifies which fields of a trade are monitored for corrections.	UnitQuantity, PriceIssuing
FDT.DeriveCorrectionFields.Execution	Specifies which fields of an execution are monitored for corrections.	UnitQuantity, PriceIssuing
FDT.DeriveCorrectionFields.DerivedTrade	Specifies which fields of a derived trade are monitored for corrections.	YieldPercentage, YieldMethodCode
FDT.DeriveCorrectionFields.Order	Specifies which fields of an order are monitored for corrections.	LimitPriceIssuing, UnitQuantity
XDP: Specifies properties used to configure the Pre-processor (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

Table 114. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`XDP.Default.ErrorLimit`	Specifies the percentage of invalid records to allow before exiting with an error. For example, a value of 10 allows 10 percent of records to be invalid before exiting with an error. A value of 0 allows no invalid records. A value of 100 allows all invalid records.	`100`
`XDP.Default.TargetDir`	Specifies the directory in which to place the resulting output file. If this is blank (the default), output files reside in the corresponding load directory (a subdirectory of `market/load` or `firm/load` depending on the schema of the data being processed).	
`XDP.Default.SequenceBatchSize`	Specifies the batch size when retrieving sequence IDs.	`100000`
`XDP.Default.AdditionalOutput`	Specifies a directory to contain the output file in addition to the target directory.	
`XDP.Default.DoFileLookups`	Specifies whether to do reference data lookups for fields that arrive as part of an input file (TRUE) or not do them (FALSE).	`FALSE`
`XDP.Default.DiscardLookupFailures`	Specifies whether to discard records that fail a reference data lookup (TRUE) or just log a message (FALSE).	`FALSE`
`XDP.Default.ValidatorClass`	Specifies the Java class used to validate records of a given data type. Use of subclasses occurs when the general functionality of AbstractValidator is not enough for a given data type.	`AbstractValidator`
`XDP.Default.AdapterClass`	Specifies the Java class used to process records of a given data type. Use of subclasses occurs when the general functionality of BaseFileAdapter is not enough for a given data type.	`BaseFileAdapter`
`XDP.Default.NumberOfThreads`	Specifies the number of worker threads to be used when Pre-processing 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.Account.DeriveAccountToPeerGroup`	When processing Account records, specifies whether to derive an AccountToPeerGroup record if the AccountPeerGroupIdentifier field is populated.	

Table 114. 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 Pre-processor instance.	
XDL: Specifies properties used to configure the Data Loader (XDL) component.		
`XDL.Default.FullRefresh`	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, .XDT`
`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 , such as data type and referential integrity. This results in rejection of records with a warning instead of a fatal failure.	`1, 1400, 1401, 1407, 1438, 1722, 1840, 1841, 2291, 2359, 1839, 1847, 12899`
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 114. 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 114. 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, AccountChangeLogSummary, CustomerChangeLogSummary, AccountToCustomerChangeLogSummary.		
`<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`	Identifies the maximum number of names that can be added to the Watch List Service after it is initialized. If additional names must be added, the service must be re-initialized.	

Data Ingest Custom XML Configuration File

Oracle clients can modify the DataIngest.xml file to override default settings that the system provides. However, this file is subject to change in future OFSBD releases. Therefore, upon installation of a newer OFSBD 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 OFSBD version, the client can copy the older DataIngestCustom.xml file to DataIngestCustom.xml in the new installation.

data Subdirectory

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

Note: Processing date stamps should appear as YYYYMMDD for Data Management directories and subdirectories. The system provides this processing date to the set_mantas_date.sh shell script when starting the first batch for the day.

data/errors Subdirectory

The errors subdirectory within the data subdirectory stores error files that Data Management 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.

The following table identifies the error file signatures that each component can output to the errors subdirectory.

Table 115. Error File Signatures Output by Component

Component	Error File
Pre-processor	<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/backup Subdirectory

The backup subdirectory stores files that Data Management 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. The following table references the files that the system writes to the backup subdirectory, by component.

Table 116. Backed Up Files by Component

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

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 Management. The following sections describe each subdirectory.

extract Subdirectory

The `extract` subdirectory within the `firm` subdirectory contains checkpoint data and working files for each Pre-processor during Pre-processing.

Each Pre-processor 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 Pre-processes its data. When finished, each Pre-processor moves its final Pre-processed 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 Pre-processor 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 Pre-processed 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 Management subsystem Data Pre-processor components. Each Market or Firm Data Pre-processor retrieves the file it is assigned to process from the `inbox` subdirectory and then moves the file to the appropriate extract subdirectory for Pre-processing. The DIS describes the naming convention and content of each data file that an Oracle client provides.

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. The following table identifies log files for each component, based on the file name's prefix.

Table 117. Log Files Output by Component

Prefix	Component
XDP	Pre-processor
XDL	Data loader
FDT	File Data Transformer
IMC	IMC

BD Directory Structure

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

Table 118. Directory Structure Description

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

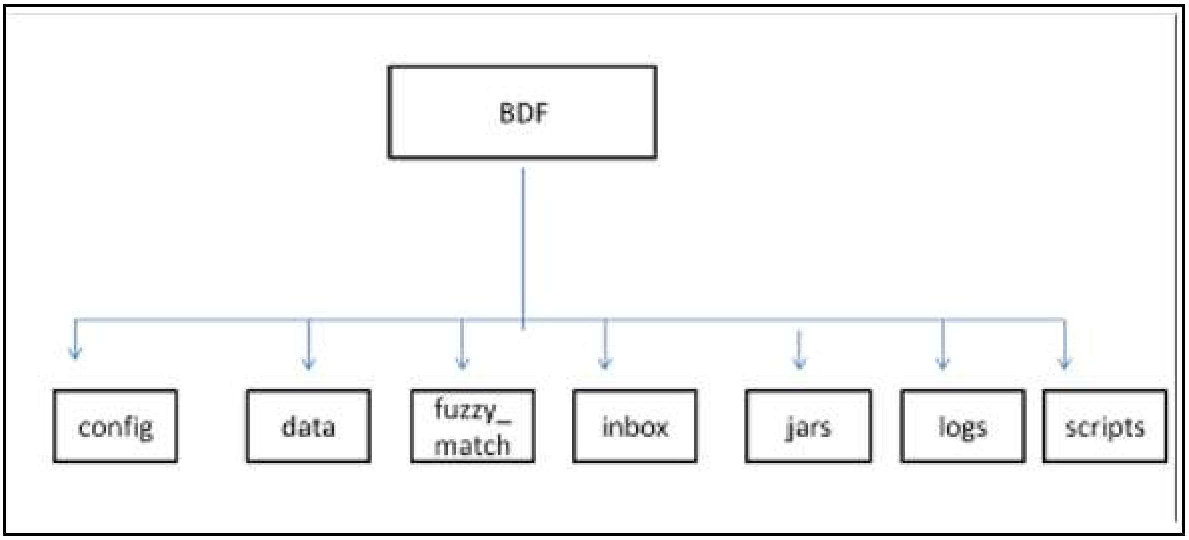


Figure 55. BD Subsystem Directory Structure

The following sections describe the BD directory structure.

Scripts

The scripts folder contains the following files:

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

For Example:

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

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

Running these files in the BD subsystem improves performance time.

Logs

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

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

For example:

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

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

For example:

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

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

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

Parameters

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

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

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

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

For example:

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

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

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

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

For example:

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

```
<OFSAAI Installed Directory>/bdf/scripts/execute.sh CorrespondentBankProfile
NumberOfThreads=4
```

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

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

Config

The config subdirectory contains configuration files.

- `<OFSAAI Installed Directory>/bdf/config/BDF.xml` contains all default product configuration parameters. It should not be modified.
- `<OFSAAI Installed Directory>/bdf/config/install/BDF.xml` contains all configuration parameters set at installation time (refer to the *Installation Guide* for more information).
- `<OFSAAI Installed Directory>/bdf/config/custom/BDF.xml` contains any product configuration parameters that have been overridden for this installation. It is initially empty. Any changes to default product configuration parameters should be put here.

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

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

For example:

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

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

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

BDF.xml Configuration Parameters

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

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
MISCELLANEOUS		
NumberOfThreads	The number of worker threads used by some BD components	4
SequenceBatchSize	The batch size when retrieving sequence IDs for new records	100000
SourceSystem	he default value for source system when one is not provided	MTS
Currency	The default value for issuing currency when one is not provided	USD
Separator	The delimiter that separates fields in data file records.	~
DB: Parameters related to database access.		
DB.Connection.Driver	The JDBC driver class name.	oracle.jdbc.OracleDriver
DB.Timeout	The number of seconds to wait before timing out on a database connection attempt.	10
DB.NumRetries	The maximum number of times to attempt to connect to a database before failing.	5
DB.MaxNumberOfDeadlocks	The maximum number of times a deadlock is encountered during a JDBC insert or update operation, before an error is generated.	10
Directory: Parameters used to define directory locations.		
Directory.Inbox	The input directory where the Oracle client will write DIS files. Date subdirectories will be created in this directory where these files will be archived	../inbox
Directory.InternalData	The directory where files generated by BD components will reside. This includes log files, error files, and any temporary processing files.	..
Log: Parameters used to configure the common logging module		
Log.Format	Identifies the log formatting string.	%d [%t] %p - %m%n
Log.UseDefaultLog	Specifies whether the system uses the default log file for a component. The default log file has the name of the component and resides in a date subdirectory of the logs directory (in YYYYMMDD format).	true
Log.SysLogHostName	The host name of syslog for messages sent to syslog.	hostname
Log.SMTPHostName	The host name of the SMTP server for messages that processing sends to an e-mail address.	hostname
Log.MaxSize	The maximum size (in MB) of a log file before the system creates a new log file.	2000MB
Log.MaxIndex	If a log file exceeds Log.MaxSize, this will be the maximum number of additional log files that are created (Component.log.1, Component.log.2, etc).	10
Log.TRACE.Enabled	Indicates that trace logging is not enabled; true indicates enabling of trace logging.	false
Log.TRACE.Location	Specifies additional locations to send TRACE log messages to, other than the default BD log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BD log location.	false

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
Log.TRACE.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.DIAGNOSTIC.Enabled	DIAGNOSTIC logging is used to log database statements and will slow down performance. Make it true if needed.	false
Log.DIAGNOSTIC.Location	Additional locations to send DIAGNOSTIC log messages to, other than the default BD log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BD log location.	
Log.DIAGNOSTIC.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.NOTICE.Enabled	Indicates enabling of notice logging; false indicates that notice logging is not enabled.	true
Log.NOTICE.Location	Specifies additional locations to send NOTICE log messages to, other than the default BD log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BD log location.	
Log.NOTICE.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.WARN.Enabled	Indicates enabling of warning logging; false indicates that warning logging is not enabled.	true
Log.WARN.Location	Specifies additional locations to send WARN log messages to, other than the default BD log file (logs/YYYYMMDD/Component.log).	
Log.WARN.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.FATAL.Enabled	Indicates enabling of Fatal logging; false indicates that fatal logging is not enabled.	true
Log.FATAL.Location	Specifies additional locations to send FATAL log messages to, other than the default BD log file (logs/YYYYMMDD/Component.log).	
Log.FATAL.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Load: Parameters used to configure common Loading data		
Load.FullRefresh	For DIS files defined as Overwrite, whether to fully replace FSDM tables with the contents of the DIS file (true) or to treat the DIS file as a delta (false)	True
Load.BatchSize	The batch size when loading data.	5000
Load.Direct	Specifies whether to use direct path loading (TRUE) or conventional path loading (FALSE).	false
Load.Unrecoverable	Specifies whether a direct path load does not use redo logs (TRUE) or uses redo logs (FALSE).	false
Load.Partitioned	Specifies whether a direct path load uses the current date partition (TRUE) or any partition (FALSE).	false
Load.SkipIndexes	Specifies whether a direct path load skips index maintenance (TRUE) or maintains indexes (FALSE). If set to TRUE, rebuilding of indexes must occur after running the DataMap XML.	false
Load.DoAnalyze	Specifies whether to run a stored procedure to analyze a database table after loading data into it.	true

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
Load.AnalyzeType	Specifies the type of analyze statistics has to perform if DoAnalyze has a value of True.	DLY_POST_ LOAD
Load.LogRecordInterval	Specifies how often to log a message saying how many records a particular thread has inserted/updated,	1000
Load.MaxErrorRate	Specifies the percentage of invalid records to allow before exiting with an error. For example, a value of 10 allows 10 percent of records to be invalid before exiting with an error. A value of 0 allows no invalid records. A value of 100 allows all invalid records.	100
Load.RecordQueueSize	Specifies the number of records the query reader thread will write to a database writer thread queue before waiting for the reader thread to catch up. Higher values will require more memory usage.	100
Load.SkipIndexesErrorCode	Specifies a database error code that occurs in the log file when skipping index maintenance.	26025
Load.IndexParallelLevel	Specifies the parallel level of an index rebuild (that is, number of concurrent threads for rebuilding an index).	1
Load.DataErrorCodes	Specifies a comma-separated list of database error codes that indicate data level errors , such as data type and referential integrity. This results in rejection of records with a warning instead of a fatal failure.	1,1400,1401,1407,1438,1722,1840,1841,2291,2359,1839,1847,12899
Load.ParallelLevel	Specifies the level of parallelization to apply when loading data from a set of source tables to a target table.	8
Load.WriteErrorFiles	Whether to check a DIS file for errors before loading as an external table (true) or not (false)	True
DIS: Parameters related to processing DIS files		
DIS.Source	The mechanism used to load DIS data. **FILE:** DIS files will be provided and will be loaded using SQL*Loader processes running on the application server. **FILE-EXT:** DIS files will be provided and will be loaded using external tables with the DIS files accessed directly by the database. **FSDW:** DIS data will be obtained from database tables in the FSDW.	FILE
DIS.ArchiveFlag	Whether DIS files will be archived to a date subdirectory (true) or not (false).	True
DIS.BufferSize	The size in KB of the byte buffer used to read in DIS file records.	100
DIS.InputFileCharset	The character set of the DIS files. Note that output data is always written in UTF8, this parameter just allows the DIS files to be in a different character set.	
DIS.Default.Check.Requirement	Whether to check for mandatory fields on DIS records (true) or not (false).	True
DIS.Default.Reject.Requirement	Whether to reject DIS records for failing a mandatory field check (true) or to log a warning and attempt to load the record (false).	True
DIS.Default.Check.Domain	Whether to check that a DIS field has a valid domain value (true) or not (false).	True

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
DIS.Default.Reject.Domain	Whether to reject DIS records that fail a domain check (true) or not (false).	True
DIS.Default.Check.Length	Whether a DIS field should be checked for a valid length (true) or not (false).	True
DIS.Default.Reject.Length	Whether to reject DIS records that fail a length check (true) or not (false)	True
DIS.Default.Check.Threshold	Whether a DIS field should be checked that it is within an acceptable threshold (i.e. greater than 0) (true) or not (false).	True
DIS.Default.Reject.Threshold	Whether to reject DIS records that fail a threshold check (true) or not (false).	True
DIS.Default.Check.Lookup	Not currently supported.	True
DIS.Default.Reject.Lookup -	Not currently supported	True
Parameters used by queries defined in the data maps:		
MinimumGeographyRisk	Defines what is considered High Risk For the Account Profile attributes related to High Risk Geography , such as Incoming High Risk Wire Count. Processing compares this parameter using a strict greater-than operation.	0
AccountInactivityInMonths	Specifies the number of months that processing aggregated to determine whether an account is inactive. If the sum of trades and transactions over this number of months is <= 3, the account is considered inactive. This setting can impact the Escalation in Inactive Accounts scenario. The default value is six months.	6
TransactionsReversalLookbackDays	This parameter controls how many days of transactions to look across. Verify whether the new data contains reversals of prior transactions.	7
LowPriceSecurityThreshold	Defines Low Priced in the base currency for the Account Profile attributes named Low-Priced Equity Range # Opening Trade Count. Processing compares the value of this parameter to the Trade table's Last Execution Price-Base.	5000
CommissionEquityPercentUpperLimit	Defines the upper limit for Commission Versus Average Daily Equity Percentage in Account Profile Calculation.	5
TurnOverRateUpperLimit	Defines the upper limit for Total Turnover Rate in Account Profile Calculation.	5

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
BankCodeListWithIA	Defines the List of Financial Institution Identifier Types, these are type of unique identifiers which are used to represent the financial institutions. This parameter also contains IA (Internal Account Identifier) to be used in datamaps and is mainly used in Correspondent Bank related datamap derivations. Below are the list of examples ● BIC: SWIFT Bank Identifier Code (BIC) ● CHU: CHIPS Participant User Identifier ● CO: Corporate Identifier ● CHP: CHIPS Participant Identifier ● FED: Federal Reserve Routing (ABA) Number ● CU: Customer Identifier ● GL: General Ledger Account ● IA: Internal Account Identifier	BIC,FED,CHP,CHU, DTC,CDL,EPN,KID, CBI,CSN,OTF,BLZ,IBAN,ABLZ,BSB,CP AP, SDIC, HEBIC, BCHH, NSC, IFSC, IDIC, PNCC, RCBIC, UKDSC, Swiss BC, Swiss SIC,IA
BankCodeList	Defines the List of Financial Institution Identifier Types, these are type of unique identifiers which are used to represent the financial institutions excluding Internal Account (IA). This parameter does not contain IA (Internal Account Identifier) to be used in datamaps and is typically used to derive financial institutions. Below are the list of examples ● BIC: SWIFT Bank Identifier Code (BIC) ● CHU: CHIPS Participant User Identifier ● CO: Corporate Identifier ● CHP: CHIPS Participant Identifier ● FED: Federal Reserve Routing (ABA) Number ● CU: Customer Identifier ● GL: General Ledger Account	BIC,FED,CHP,CHU, DTC,CDL,EPN,KID, CBI,CSN,OTF,BLZ,IBAN,ABLZ,BSB,CP AP, SDIC, HEBIC, BCHH, NSC, IFSC, IDIC, PNCC, RCBIC, UKDSC, Swiss BC, Swiss SIC
IdRiskWinLevel	Defines the Risk level to calculate Effective Risks for internal parties (Account/ Customer). For example: Account 1234 has an Effective Risk of 5, IdRiskWinLevel can be set by the client. If the party identifier effective risk is greater than the set IdRiskWinLevel, then the party identity risk wins compared to fuzzy matcher (Party Name Risk). If not, fuzzy matcher wins.	1
InternalAccountCodeList	Codes to define types of Internal Entities with client, for example: ● IA: Internal Account Identifier ● GL: General Ledger Account	IA, GL
ExternalEntityCodeList	Codes to define types of External Entities with client, for example: ● XA: External Account Identifier ● CO: Corporate Identifier ● DL: Driver License ● IBAN: International Bank Account Number	XA,CC,CO,DL,GM, GP,LE,MC,ND,NR, PP,SS,TX,AR,OT,IB AN

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
TrustedPairReviewReasonText 1	Defines the reason text1 for recommendation of cancelling the Trusted Pair, due to increase in Risk of parties involved in trusted pair.	Risk of <Party1> increased from <A> to <b>
TrustedPairReviewReasonText 2	Defines the reason text2 for recommendation of cancelling the Trusted Pair, due to increase in Risk of parties involved in trusted pair.	Risk of <Party2> increased from <C> to <D>
CorporateActionLookBackDay s	This parameter determines the how many days trades to look back from the Corporate Effective Date.	7
DealNearTermMaturityDays	Defines the maximum number of days between the End Date and Trade Date. This helps to calculate Structured Deals Initiated w/ Near-Term Exp. In Customer Profile/ Institutional Account Profile.	7
ProfitLossUpperLimit	Helps determine how much a security must move by the end of the day to be considered a win or loss. If the security moves by less than a specified percentage, processing does not count it either way. If it moves by this percentage or more, it counts as a win or a loss, depending on whether the movement was beneficial to the account that made the trade.	5
HouseholdTurnOverRateUppe rLimit	Defines the upper limit for Total Turnover Rate in Household Profile Calculation.	10000
HouseholdCommissionEquityP ercentUpperLimit	Defines the upper limit for Commission Versus Average Daily Equity Percentage in Account Profile Calculation.	10000
OptionTradeAmountRange1 OptionTradeAmountRange2 OptionTradeAmountRange3 OptionTradeAmountRange4 OptionTradeAmountRange5 OptionTradeAmountRange6	Define the lower bound of each range for the Account Profile attributes named Options Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount- Base. Each range is from the lower bound entered here to the lower bound of the next range.	
EquityTradeAmountRange1 EquityTradeAmountRange2 EquityTradeAmountRange3 EquityTradeAmountRange4 EquityTradeAmountRange5 EquityTradeAmountRange6	Define the lower bound of each range for the Account Profile attributes named Equity Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount- Base. Each range is from the lower bound entered here to the lower bound of the next range.	
LowPricedEquityTradeAmount Range1 LowPricedEquityTradeAmount Range2 LowPricedEquityTradeAmount Range3 LowPricedEquityTradeAmount Range4 LowPricedEquityTradeAmount Range5 LowPricedEquityTradeAmount Range6	Define the lower bound of each range for the Account Profile attributes named Low-Priced Equity Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount-Base. Each range is from the lower bound entered here to the lower bound of the next range.	

Table 119. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
MutualFundTradeAmountRange1 MutualFundTradeAmountRange2 MutualFundTradeAmountRange3 MutualFundTradeAmountRange4 MutualFundTradeAmountRange5 MutualFundTradeAmountRange6	Define the lower bound of each range for the Account Profile attributes named Mutual Fund Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount-Base. Each range is from the lower bound entered here to the lower bound of the next range.	
UnrelatedWhenOffsetAccountIsNull	This parameter is used to assign unrelated party code as "J" in the BackOfficeTransaction table, If OFFST_ACCT_INTRL_ID is null and UnrelatedWhenOffsetAccountIsNull is "Y", If OFFST_ACCT_INTRL_ID is null and UnrelatedWhenOffsetAccountIsNull is "N", then unrelated party code is NULL.	Y

BD Datamap Configuration File

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

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

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

- `AccountTrustFromCustomer`
- `DefaultJurisdiction`
- `UseTaxidForUnrelatedPartyCode`
- `BaseCountry`
- `ProcessForeignFlag`
- `ProcessBankToBank`
- `ProcessTransactionXRefFlag`
- `TrustedPairRiskReviewFlag`

These parameters are stored in the following file:

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

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

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

- `<Parameter name="AccountTrustFromCustomer" type="STRING" value="Y"/>`
- `<Parameter name="DefaultJurisdiction" type="STRING" value="AMEA"/>`
- `<Parameter name="UseTaxidForUnrelatedPartyCode" type="STRING" value="Y"/>`
- `<Parameter name="BaseCountry" type="STRING" value="US"/>`

- `<Parameter name="ProcessForeignFlag" type="STRING" value="N"/>`
- `<Parameter name="ProcessBankToBank" type="STRING" value="N"/>`
- `<Parameter name="ProcessTransactionXRefFlag" type="STRING" value="Y"/>`
- `<Parameter name="TrustedPairRiskReviewFlag" type="STRING" value="N"/>`

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

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

Table 120. BD Datamap Configuration Parameters

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

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

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

And can also be created in the following:

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

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 56 depicts dependency between the `process_market_summary.sh` script and the `runFDT.sh` script.

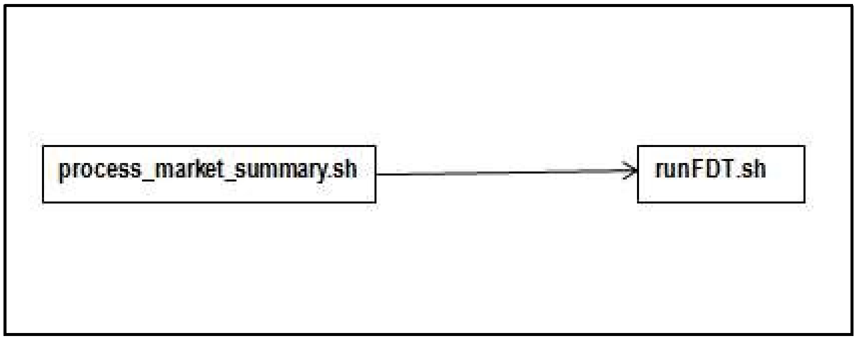


Figure 56. Dependency between *`process_market_summary.sh`* and *`runFDT.sh`*

For intra-day batch ingestion or intra-day execution of the MiFID scenarios, the process flow does not change from Figure 55. 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, see section *Market Supplementary Guidance*, *Data Interface Specification*.

APPENDIX E

Processing Derived Tables and Fields

This appendix covers the following topics:.

- Ingestion through Batches
- Derivations
- Ingestion Timeline
- Guidelines for Duplicate Record Handling
- Data Rejection During Ingestion
- Alternatives to Standard Data Management Practices

Customizing Scripts

For OFSAAI to execute the shell scripts, the customized scripts have to be placed in the ficdb layer. The customized scripts should be placed under `<Installed Path>ficdb/bin`. When the customized scripts are called from

OFSAAI, it appends the Batch Flag and Wait Flag parameters. This must be internally handled in the customized script to eliminate these additional parameters.

Note: The Batch Flag and Wait Flag are the default parameters expected by the AAI Batch. For more information on these parameters refer the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

The following paths should be set inside the scripts:

- **MANTAS_HOME**: The path where the solution is installed.

  For Example: `/scratch/ofsaaapp/FCCM804`
- **INGESTION_HOME**: The path under installed area pointing to the ingestion_manager subsystem.

  For Example: `/scratch/ofsaaapp/FCCM804/ingestion_manager`
- **DB_TOOLS_HOME**: The path under installed area pointing to database subsystem.

  For Example: /scratch/ofsaaapp/FCCM804/database/db_tools
- **BDF_HOME:** The path under the installed area pointing to the BD subsystem.

  For Example: `/scratch/ofsaaapp/FCCM804/bdf`

Note: BDF_HOME should be exported only if Ingestion has to be run through the BD subsystem.

After exporting the respective paths inside the script, the product script must be called from the customized script. For more information about how to create an OFSAA Batch and add a task for executing the custom script, please refer to the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

Sample customized script for execute.sh is given below:

```
#!/bin/sh
if [[ $# == 0 || $# > 3 ]]; then
  ##echo "Usage: run_GD_dpdl.sh YYYYMMDD"
  exit -1;
fi
export MANTAS_HOME=/scratch/ofsaadb/BD_801_BUILD2/BD_801C2WL
export BDF_HOME=$MANTAS_HOME/bdf
export DB_TOOLS_HOME=$MANTAS_HOME/database/db_tools
##export DIS_FILES=$HOME/GD_Scripts/disfile.cfg
export FILE_NAME=$1
$BDF_HOME/scripts/execute.sh $FILE_NAME
      err=$?
      if [ $err -ne 0 ]
      then
        echo " BDF Execution failed"
        exit 1
      fi
```

The above script is used to trigger BD Ingestion using execute.sh. This script expects only the file name (Eg. Account)as a parameter. Since AAI batch appends two additional default parameters (Batch Flag and Wait Flag) during batch execution, these should be handled inside the script and only the file name should be passed as a parameter. Internally this customized script calls the product script, execute.sh. Similarly, other scripts can also be customized.

Derivations

These utilities populate a single table in the data model. They should be executed after all the files in Table 7 have been loaded. A utility should not be executed until its predecessors have executed successfully.

Commands to execute:

```
<OFSAAI Installed Directory>/ingestion_manager/scripts/runUtility.sh <Utility Name>
<OFSAAI Installed Directory>/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.

Table 121. Utilities

Product	Utility Name	Table Name	Predecessor
ECTC	EnergyAndCommodityFirmDailyDerived	EC_FIRM_DAILY	
ECTC	EnergyAndCommodityMarketDailyDerived	EC_MARKET_DAILY	
ECTC	EnergyAndCommodityTradeDerived	EC_TRADE	
ECTC	EnergyFlow	ENERGY_FLOW	
BC	MutualFundFamilyAccountPosition	MUTUAL_FUND_FAM_ACCT_POSN	
BC	RegisteredRepresentativeCommissionProfile	RGSTD_REP_CMSN_SMRY	

Table 121. Utilities

Product	Utility Name	Table Name	Predecessor
BC	RegisteredRepresentativeCommissionProductMixProfile	RGSTD_REP_CMSN_PRDCT_SMRY	
ECTC	EnergyFlowDailyProfile	ENERGY_FLOW_SMRY_DAILY	Energy Flow

AccountDailySecurityProfile

The AccountDailySecurityProfile Utility is used to populate the Account Daily Security Profile table.

This Utility reads the Trade table, and processes the trade records to populate the `ACCT_SCRTY_SMRY_DAILY` table.

Execute the following commands:

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

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

Example:

```
runUtility.sh AccountDailySecurityProfile
runDL.sh AccountDailySecurityProfile
```

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 must 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 must 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 99, for more information.

Ingestion Timeline

Intra-Day Ingestion Processing

The following figure provides a high-level flow of the intra-day ingestion process of extracting, transforming, and loading data.

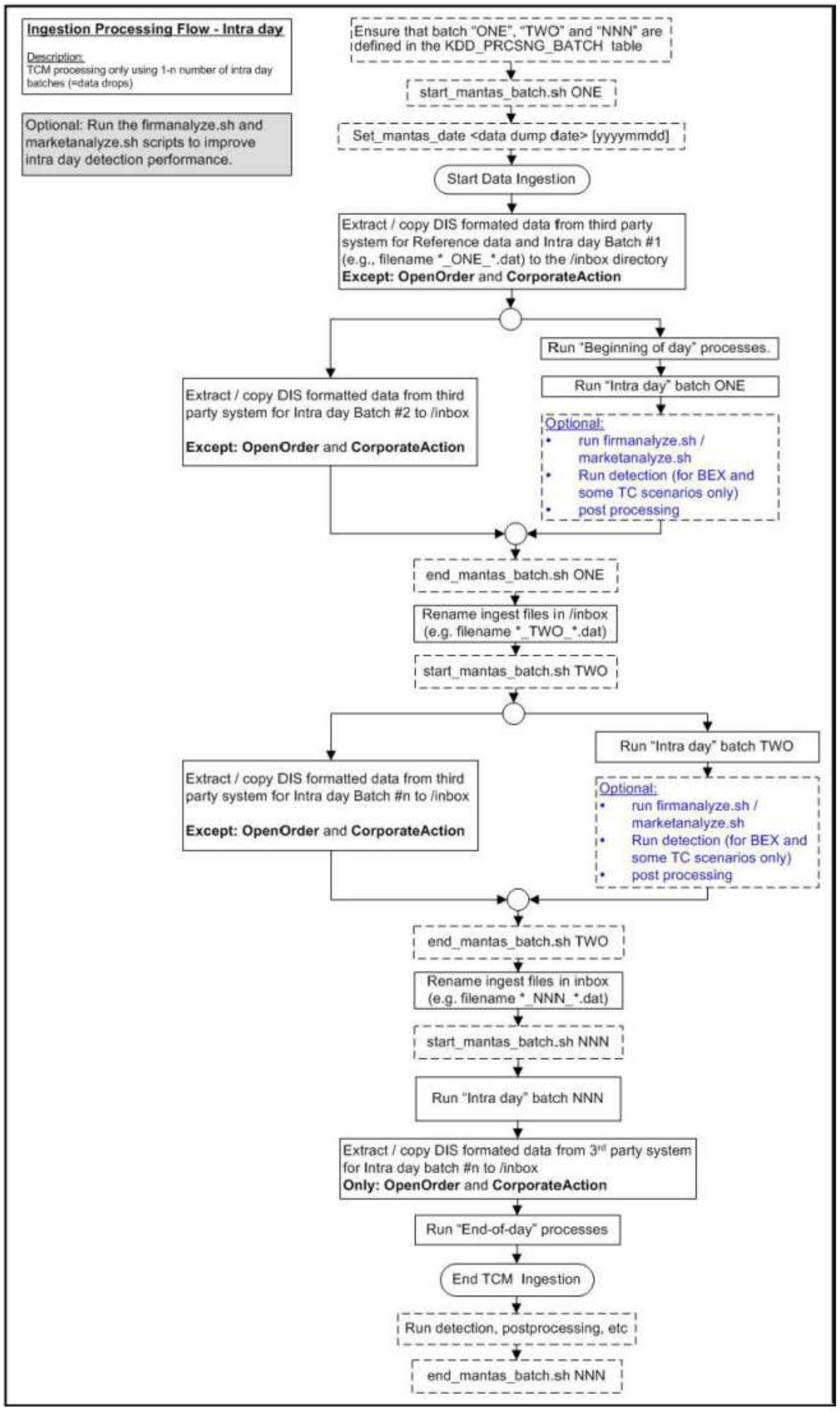

Ingestion Processing Flow - Intra day
Description:
TCM processing only using 1-n number of intra day batches (=data drops)
Optional: Run the firmanalyze.sh and marketanalyze.sh scripts to improve intra day detection performance.
Ensure that batch "ONE", "TWO" and "NNN" are defined in the KDD_PRCSNG_BATCH 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 57. Intra-Day Data Management Processing

Intra-day processing references different processing groups as Figure 57 illustrates , such as beginning-of-day processing and intra-day processing. Multiple batches run throughout the day. As in Figure 57, 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.

The following table provides an example of setting up the `KDD_PRCSNG_BATCH` table.

Table 122. 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

Guidelines for Duplicate Record Handling

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

Data Rejection During Ingestion

The Ingestion Manager can reject records at the Pre-processing, 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 Pre-processing Stage:** Describes how rejections occur during the Pre-processing stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Pre-processing Stage* 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* 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* for more information).

Rejection During the Pre-processing Stage

The first stage of ingestion is Pre-processing. At this stage, Data Management examines Oracle 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 Pre-processing include problems with data type, missing data, referential integrity, and domain values.

During normal operation, the number of rejections at the Pre-processor 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 Pre-processor 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 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(Yes) and the system writes the event to the database. An Out of Sequence event has no effect on the order or trade that it refers if processing sets the Out-of-sequence flag set for the event to Y

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 , such as setup of partitions, are incorrect for the data being ingested).

Alternatives to Standard Data Management Practices

Data Management Archiving

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

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

Periodically, an Oracle client can run the `runIMC.sh` script to perform the 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.

Fuzzy Name Matcher Utility

During BD Datamap 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.

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

The Fuzzy Name Matcher utility can be used in the following ways:

- Through Ingestion Manager as a standalone Fuzzy Name Matcher. For more information, refer to *Executing the Fuzzy Name Matcher Utility*. To configure Fuzzy Name Matcher, modify `<ingestion_manager>/fuzzy_match/mantas_cfg/install.cfg`.
- Through BD Datamaps (`NameMatchStaging.xml,RegOToBorrower.xml`) file in folder (`<OFSAAI Installed Directory>/bdf/config/datamaps`). For more information, refer *Chapter 3, Managing Data*. To configure Fuzzy Name Matcher, modify `<OFSAAI Installed Directory>/bdf/config/BDF.xml` under the Fuzzy Name Matcher section.

The following figure provides a sample configuration appearing in `<OFSAAI Installed Directory>/bdf/config/BDF.xml`

```
#---------------------------------------------------------------
#                Fuzzy Name Matcher Configuration Items
#---------------------------------------------------------------
<Parameter name="fuzzy_name.match_multi" type="BOOLEAN" value="true"/>
  <Parameter name="fuzzy_name.file.delimiter" type="STRING" value="~"/>
  <Parameter name="fuzzy_name.default.prefix" type="STRING" value="P"/>
  <Parameter name="fuzzy_name.min.intersection.first.letter.count" type="INTEGER" 
value="2"/>
  <Parameter name="fuzzy_name.max_added_names" type="INTEGER" value="10000"/>
  <Parameter name="fuzzy_name.B.match_threshold" type="REAL" value="80"/>
  <Parameter name="fuzzy_name.B.initial_match_score" type="REAL" value="75"/>
  <Parameter name="fuzzy_name.B.initial_match_p1" type="INTEGER" value="2"/>
  <Parameter name="fuzzy_name.B.initial_match_p2" type="INTEGER" value="1"/>
  <Parameter name="fuzzy_name.B.extra_token_match_score" type="REAL" value="100"/>
  <Parameter name="fuzzy_name.B.extra_token_min_match" type="INTEGER" value="2"/>
  <Parameter name="fuzzy_name.B.extra_token_pct_decrease" type="INTEGER" value="50"/>
  <Parameter name="fuzzy_name.B.first_first_match_score" type="INTEGER" value="1"/>
  <Parameter name="fuzzy_name.P.match_threshold" type="REAL" value="70"/>
  <Parameter name="fuzzy_name.P.initial_match_score" type="REAL" value="75"/>
  <Parameter name="fuzzy_name.P.initial_match_p1" type="INTEGER" value="2"/>
  <Parameter name="fuzzy_name.P.initial_match_p2" type="INTEGER" value="1"/>
  <Parameter name="fuzzy_name.P.extra_token_match_score" type="REAL" value="50"/>
  <Parameter name="fuzzy_name.P.extra_token_min_match" type="INTEGER" value="2"/>
  <Parameter name="fuzzy_name.P.extra_token_pct_decrease" type="INTEGER" value="50"/>
  <Parameter name="fuzzy_name.P.first_first_match_score" type="INTEGER" value="0"/>
```

Figure 58. Sample BDF.xml Configuration Parameters

The following table describes the utility's configuration parameters as they appear in the BDF.xml file. Note that all scores have percentage values.

Table 123. 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 123. 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 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 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`	This parameter keeps 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, type the following at the UNIX command line:

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

Refresh Temporary Tables 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.

Use of Control Data

After installing the OFSBD software, you can use control data provided to test end-to-end processing of data (that is, running data management, executing scenarios, and viewing generated alerts in the Alert Management UI). 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 338 for more information).
2. Prepare for ingestion of the control data (refer to section *Control Data Management* on page 338 for more information).
3. Install the control data (refer to section *Loading Control Data Thresholds* on page 339 for more information).
4. Run Behavior Detection on control data to generate alerts (refer to section *Running Behavior Detection on Control Data* on page 340 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 25 different dates in 13 months.
2. The current day according to control data is 20151210.
3. Unless specified, set the current date as 20151210, to generate alerts on control data, before running Behavior Detection Platform.

Note: For more information about control data on your site, contact your OFSBD Administrator.

Control Data Management

Control data uses a specific set of dates to ensure that all the OFSBD lock-stock 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 25 different dates in 13 months. The dates (YYYYMMDD format) being used by control data are:

Table 124. Dates used by Control Data

20141231	20151123
20150130	20151124
20150227	20151125
20150331	20151126

Table 124. Dates used by Control Data

20150430	20151127
20150529	20151130
20150630	20151203
20150731	20151204
20150831	20151208
20150930	20151209
20151030	20151210
20151201	20151202
20151121	

On all these dates, ingest the data and run the complete Behavior Detection batch for the respective date. Except for Behavior Detection and Post-Processing tasks, perform all other activities for the Control Data Management dates. Activities required during any Behavior Detection Framework 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*for more information).
2. Run ingestion for all the control Data Management dates. Refer to section *Intra-Day Ingestion Processing*, for more information about the ingestion process.

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

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 `<OFSAAI Installed Directory>/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 4, Behavior Detection Jobs* to get information regarding about running Behavior Detection Jobs.)

Important Notes

1. Run loaded scenarios with the system date as 20151210 with the following exceptions:
   a. For Portfolio Pumping scenario, the system date must be 20151204
   b. For Active Trading scenario, the system date must be 20151130
2. Check for system errors in the appropriate logs (refer to *Appendix A, Logging*, 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.

APPENDIX F BD Datamap Details

This appendix lists the BD datamaps used in OFSAAI and a brief explanation of the each datamap. This section contains the following sections:

- AML Brokerage Datamaps
- AML Banking Datamaps
- Broker Compliance Datamaps
- Fraud Detection Datamaps
- Insurance Datamaps
- Trade Finance Datamaps
- Processing BD Datamaps
- Firm Data Transfer Datamaps

Note: Oracle recommends all datamaps are run in the order described in the following tables.

AML Brokerage Datamaps

AML Brokerage - Pre-Watch List Datamaps

Pre-Watch List Datamaps are used to facilitate the application to populate various business areas, such as Financial Institutions, Account To Client Bank, Settlement Instructions, Front Office and Back Office Transaction.

These datamaps populate the relevant data which is used by watch list datamaps in calculating risks.

Table 125. AML Brokerage - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
50010	Customer_TotAcctUpd	NA
10010	EmployeeControlledAccount (*Optional*)	NA
10015	FrontOfficeTransactionParty_SecondaryNames	NA
10020	FinancialInstitution_ThomsonDataInstitutionInsert (*Optional*)	NA
10030	AccountToClientBank_ThomsonDataInstitutionInsert (*Optional*)	10020
10040	FinancialInstitution_AIIMSPopulation	NA
10050	AccountToClientBank_AIIMSInstitutionInsert	10040
10060	AccountToClientBank_InstitutionInsert	10050
10070	AccountToClientBank_InstitutionUpd	10060

Table 125. AML Brokerage - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10080	FinancialInstitution_FOTPSPopulation	10020 10030 10040 10050 10060 10070
10090	AccountToClientBank_FOTPSInstitutionInsert	10020 10030 10040 10050 10060 10070 10080
10100	AccountManagementStage	NA
10110	LoanProfile_LoanProfileStage	NA
10111	LoanDailyActivity_RepCurrencyUpd	NA
10114	BackOfficeTransaction_UnrelatedPartyCodeUpd	NA
10116	BackOfficeTransaction_CollateralUpd	10114
10120	BackOfficeTransaction_OriginalTransactionReversalUpd	NA
10130	BackOfficeTransaction_CancelledTransactionReversalCreditUpd	NA
10140	BackOfficeTransaction_CancelledTransactionReversalDebitUpd	NA
10150	FrontOfficeTransactionParty_InstnSeqID	10020 10030 10040 10050 10060 10070 10090
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	10150
10170	FinancialInstitution_AnticipatoryProfile	10020 10030 10040 10050 10060 10070
10180	AccountToClientBank_AnticipatoryProfile	10020 10030 10040 10050 10060 10070 10170

Table 125. AML Brokerage - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10190	AnticipatoryProfile_AccountToClientBank	10020 10030 10040 10050 10060 10070 10170 10180
50020	DailyAggregateStage	NA
50030	OffsettingAccountPairStage	NA
50040	TradeDailyTotalCountStage	NA
10200	CustomerAccountStage_FrontOfficeTransactionParty	NA
10210	FrontOfficeTransaction_UnrelatedPartyUpd	10120 10130 10140 10200
10220	FinancialInstitution_SettlementInstruction	10020 10030 10040 10050 10060 10070
10230	AccountToClientBank_SettlementInstruction	10020 10030 10040 10050 10060 10070 10220
10240	SettlementInstruction_AccountToClientBank	10020 10030 10040 10050 10060 10070 10230
10014	FrontOfficeTransaction_PassThroughFlag	NA

Note:

- **FrontOfficeTransaction_PassThroughFlag** - This data map should only be run if the P*ass Through Indicator* field is not being provided in the `Front Office Transaction DIS` file, and the client requires support to derive this datamap.
- **FrontOfficeTransactionParty_SecondaryNames** - This data map should only be run if Secondary Originator and Secondary Beneficiary party records are not being provided in in the Front Office Transaction Party DIS file, and the client requires support to derive them from the Bank-to-Bank Instructions and Originator-to-Beneficiary Instructions fields.

AML Brokerage - Watch List Datamaps

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

These datamaps assist other datamaps which are used to calculate Effective Risk and Activity Risk for various entities, such as Account, Customer, Transaction Tables, and so on.

Table 126. AML Brokerage - Watch List Datamaps

Datamap Number.	Datamap Name	Predecessors
10245	WLMProcessingLock	NA
10250	WatchListEntry_WatchListEntryCurrDayInsert	10020 10030 10040 10050 10060 10070 10245
10260	WatchListAudit_StatusUpd	10020 10030 10040 10050 10060 10070
10270	WatchList_WatchListSourceAuditInsert	10020 10030 10040 10050 10060 10070
10280	WatchList_WatchListSourceAuditUpd	10020 10030 10040 10050 10060 10070
10290	WatchList_WatchListSourceUpd	10020 10030 10040 10050 10060 10070
10300	WatchListEntry_WatchListAuditUpd	10020 10030 10040 10050 10060 10070 10260

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10310	WatchListEntryAudit_WatchListEntryUpdate	10020 10030 10040 10050 10060 10070 10300
10320	Customer_KYCRiskUpd	NA
10330	DerivedAddress_SettlementInstructionInsert	NA
10340	DerivedAddress_SettlementInstructionUpd	NA
10350	SettlementInstruction_PhysicalDlvryAddrUpd	NA
10360	DerivedAddress_FrontOfficeTransactioPartyStageInsert	NA
10370	DerivedAddress_FrontOfficeTransactioPartyStageUpd	NA
10380	FrontOfficeTransactionParty_DerivedAddress	10360 10370
10390	DerivedEntity_FrontOfficeTransactionPartyInsert	10080 10090
10400	DerivedEntity_FrontOfficeTransactionPartyUpd	10080 10090
10410	DerivedEntity_SettlementInstructionInsert	10220 10230 10240
10420	DerivedEntity_SettlementInstructionUpd	10220 10230 10240
10430	CorrespondentBank_FrontOfficeTransactionPartyStageInsert	10080 10090
10440	CorrespondentBank_FrontOfficeTransactionPartyStageUpd	10080 10090
10450	WatchListStagingTable_WatchList	10250 10260 10270 10280 10290 10300 10310
10460	WatchListStagingTable_WatchListInstnIDUpd	10250 10260 10270 10280 10290 10300 10310

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10470	PreviousWatchList_WatchList	10250 10260 10270 10280 10290 10300 10310
10480	DerivedAddress_WatchListNewCountries	10250 10260 10270 10280 10290 10300 10310
10485	WLMProcessingUnlock	10480
10490	LinkStaging_FrontOfficeTransactionParty	10360 10370 10380 10390 10400 10485
10500	LinkStaging_InstructionDerivedEntDerivedAdd	10330 10340 10350 10410 10420
10510	NameMatchStaging	10450 10460 10470 10480 10390 10400
10520	WatchListStagingTable_NameMatchStageInsert	10510
10530	DerivedEntityLink_LinkStage	10490 10500
10540	DerivedEntitytoDerivedAddress_LinkStage	10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	10490 10500

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10570	WatchListStagingTable2_WatchListStage2AcctExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10580	WatchListStagingTable2_WatchListStage2CBExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10590	WatchListStagingTable2_WatchListStage2CustExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10600	WatchListStagingTable2_WatchListStage2DAExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10610	WatchListStagingTable2_WatchListStage2EEExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10620	WatchListStagingTable2_WatchListStage	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10630	WatchListStagingTable2_AcctListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10640	WatchListStagingTable2_CBListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10650	WatchListStagingTable2_CustListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10660	WatchListStagingTable2_EEListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10670	WatchListStagingTable2_EEListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10680	WatchListStagingTable2_DAListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10690	WatchListStagingTable2_DAListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10700	WatchListStagingTable2_WatchListStage2SeqIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10710	WatchListStagingTable2_WatchListStage2IntrlIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10720	Customer_WatchListStage2ListRisk	10320 10700 10710
10730	CorrespondentBank_WatchListStage2EffectiveRisk	10320 10700 10710
10740	Customer_WatchListStage2EffectiveRisk	10320 10700 10710
10750	DerivedAddress_WatchListStage2EffectiveRisk	10320 10700 10710

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10760 10700 10710	DerivedEntity_WatchListStage2EffectiveRisk	10320 10700 10710
10770	WatchListStagingTable2_WatchListStage2SeqId	10320 10700 10710
10780	AccountListMembership_WatchListStage2Insert	10700 10710
10790	AccountListMembership_WatchListStage2Upd	10700 10710
10800	CorrespondentBankListMembership_WatchListStage2Insert	10700 10710
10810	CorrespondentBankListMembership_WatchListStage2Upd	10700 10710
10820	CustomerListMembership_WatchListStage2Insert	10700 10710
10830	CustomerListMembership_WatchListStage2Upd	10700 10710
10840	DerivedAddressListMembership_WatchListStage2Insert	10700 10710
10850	DerivedAddressListMembership_WatchListStage2Upd	10700 10710
10860	DerivedEntityListMembership_WatchListStage2Insert	10700 10710
10870	DerivedEntityListMembership_WatchListStage2Upd	10700 10710
10875	Account_EffectiveRiskFactorTxtUpd	10700 10701
10880	Account_OverallEffectiveRiskUpd	10720 10730 10740 10750 10760 10770 10780 10790 10800 10810 10820 10830 10840 10850 10860 10870
10881	Account_AccountCustRiskUpd	10880

Table 126. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10890	Account_EffRiskUpdAfterWLRiskRemoval	10720 10730 10740 10750 10760 10770 10880
10900	Account_WatchListStage2EffectiveRisk	10720 10730 10740 10750 10760 10770 10880
10910	WatchListStagingTable2_WatchListStage2IntrlId	10320 10700 10710
10920	BackOfficeTransaction_EffectiveAcctivityRiskUpd	10890 10900
10930	SettlementInstruction_EntityAcctivityRiskUpd	10890 10900
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	10890 10900

Note: If you are running any of these combinations you must run datamap 10320 and 10880.
OFSBD AML and KYC
OFSBD Fraud and KYC
OFSBD AML, Fraud, and KYC

AML Brokerage - Post-Watch List Datamaps

Post-Watch List Datamaps are used to populate or rather ingest data into various transaction tables using Front Office and Back Office Transaction files, these are executed only after the Watch List Datamaps are run.

These datamaps are used to populate data into the Cash, Wire, and Monetary Instruments tables. These are also used to update Trusted Pair and Jurisdiction information into various other entities. Table 127 describes the Post-Watch List datamaps for AML Brokerage.

Oracle clients can configure the Risk Zones and customize the Review Reason Text for the following datamaps:

- TrustedPair_StatusRRCInsert (Datamap Number 11080)
- TrustedPair_StatusRRCUpd (Datamap Number 11090)
- TrustedPairMember_AcctExtEntEffecRiskUpd (Datamap Number 11070)

Table 127. AML Brokerage - Post Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10960	AccountGroup_JurisdictionUpd	NA
10970	TransactionPartyCrossReference_BackOfficeTransaction	10360 10370 10380 10940
10980	CashTransaction_FrontOfficeTransaction	10360 10370 10380 10940
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	10360 10370 10380 10940
11000	TransactionPartyCrossReference_FrontOfficeTransaction	10360 10370 10380 10940 11060 11070 11080 11090
11010	WireTransaction_FrontOfficeTransaction	10360 10370 10380 10940
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	10360 10370 10380 10940
11030	CashTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11040	MonetaryInstrumentTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11050	WireTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020

Table 127. AML Brokerage - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
11060	TrustedPair_StatusEXPUpd	10970 10980 10990 11000 11010 11020
11070	TrustedPairMember_AcctExtEntEffecRiskUpd	10970 10980 10990 11000 11010 11020
11080	TrustedPair_StatusRRCInsert	11160
11090	TrustedPair_StatusRRCUpd	11170
11100	ApprovalActionsAudit_TrustedPair	10970 10980 10990 11000 11010 11020
11110	TrustedPairMember_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11120	BackOfficeTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11140	MonetaryInstrumentTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11150	WireTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110

AML Brokerage - Summary Datamaps

Summary Datamaps are used to calculate aggregations across various entities using the Trade, Transaction, Positions and Balances Tables.

These datamaps populate various profile tables for different entities like Account Profile, Household Profile, Correspondent Bank Profile.The aggregation is done daily, weekly or monthly depending on the business areas.

Table 128. AML Brokerage - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
50050	CustomerDailyProfile_BOT	NA
50060	CustomerDailyProfile_FOTPS	NA
50070	InstitutionalAccountDailyProfile_DEAL	NA
50080	CustomerDailyProfile_DEAL	NA
50090	InstitutionalAccountDailyProfile_INST	NA
50100	CustomerDailyProfile_INST	NA
50110	InstitutionalAccountDailyProfile_CorpAction	NA
50120	CustomerDailyProfile_CorpAction	NA
50130	InstitutionalAccountDailyProfile_Trade	NA
50140	CustomerDailyProfile_Trade	NA
11160	AccountDailyProfile-Trade	NA
11170	AccountDailyProfile-Transaction	NA
11180	AccountProfile_Trade	10940 11160 11170
11190	AccountProfile_Transaction	10940 11160 11170
11200	AccountProfile_Stage	NA
11210	AccountProfile_Position	11180 11190 11200
11220	AccountProfile_Balance	11180 11190 11200 11210
50150	InstitutionalAccountProfile	50070 50090 50110 50130
50160	CustomerProfile	50050 50060 50080 50100 50120 50140
11230	ChangeLog_AcctProfileInactivity	11180 11190 11200 11210 11220

Table 128. AML Brokerage - Summary Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
11240	AccountPeerGroupMonthlyTransactionProfile	11180 11190 11200 11210 11220
11300	AccountChangeLogSummary	The datamap should be executed once the change log processing is done.
11310	AccountToCustomerChangeLogSummary	
11320	CustomerChangeLogSummary	

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

AML Brokerage - Balances and Positions Datamaps

Balances and Positions Datamaps derive attributes that are useful in assessment of the financial status of an account, customer, or Household. These datamaps are used to populate business areas, such as account balance, account position, portfolio manager positions, and so on.

Table 129. AML Brokerage - Balances and Positions Datamaps

Datamap Number	Datamap Name	Predecessors
50170	CustomerBalance_ActiveOTCTradeCtUpd	NA

AML Banking Datamaps

AML Banking - Pre-Watch List Datamaps

Pre-Watch List Datamaps are used to facilitate the application to populate various business areas like Financial Institutions, Account To Client Bank, Settlement Instructions, Front Office and Back Office Transaction. These datamaps populate the relevant data which are used by watch list datamaps in calculating risks

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 130. AML Banking - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10010	EmployeeControlledAccount (*Optional*)	NA
10015	FrontOfficeTransactionParty_SecondaryNames	NA
10020	FinancialInstitution_ThomsonDataInstitutionInsert (*Optional*)	NA
10030	AccountToClientBank_ThomsonDataInstitutionInsert (*Optional*)	10020
10040	FinancialInstitution_AIIMSPopulation	NA
10050	AccountToClientBank_AIIMSInstitutionInsert	10040

Table 130. AML Banking - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10060	AccountToClientBank_InstitutionInsert	10050
10070	AccountToClientBank_InstitutionUpd	10060
10080	FinancialInstitution_FOTPSPopulation	10020 10030 10040 10050 10060 10070
10090	AccountToClientBank_FOTPSInstitutionInsert	10020 10030 10040 10050 10060 10070 10080
10100	AccountManagementStage	NA
10110	LoanProfile_LoanProfileStage	NA
10114	BackOfficeTransaction_UnrelatedPartyCodeUpd	NA
10116	BackOfficeTransaction_CollateralUpd	10114
10120	BackOfficeTransaction_OriginalTransactionReversalUpd	NA
10130	BackOfficeTransaction_CancelledTransactionReversalCreditUpd	NA
10140	BackOfficeTransaction_CancelledTransactionReversalDebitUpd	NA
10150	FrontOfficeTransactionParty_InstnSeqID	10020 10030 10040 10050 10060 10070 10090
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	10020 10030 10040 10050 10060 10070 10150
10200	CustomerAccountStage_FrontOfficeTransactionParty	NA
10210	FrontOfficeTransaction_UnrelatedPartyUpd	10120 10130 10140 10200
10014	FrontOfficeTransaction_PassThroughFlag	NA

Note:

- **FrontOfficeTransaction_PassThroughFlag** - This data map should only be run if the P*ass Through Indicator* field is not being provided in the `Front Office Transaction DIS` file, and the client requires support to derive this datamap.

- **FrontOfficeTransactionParty_SecondaryNames** - This data map should only be run if Secondary Originator and Secondary Beneficiary party records are not being provided in in the Front Office Transaction Party DIS file, and the client requires support to derive them from the Bank-to-Bank Instructions and Originator-to-Beneficiary Instructions fields.

AML Banking - Watch List Datamaps

Watch List Datamaps facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. These datamaps finally assist other datamaps which are used to calculate Effective Risk and Activity Risk for various entities, such as Account, Customer, Transaction, and so on.

Table 131. AML Banking - Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10245	WLMProcessingLock	NA
10250	WatchListEntry_WatchListEntryCurrDayInsert	10020 10030 10040 10050 10060 10070 10245
10260	WatchListAudit_StatusUpd	10020 10030 10040 10050 10060 10070
10270	WatchList_WatchListSourceAuditInsert	10020 10030 10040 10050 10060 10070 10260
10280	WatchList_WatchListSourceAuditUpd	10020 10030 10040 10050 10060 10070
10290	WatchList_WatchListSourceUpd	10020 10030 10040 10050 10060 10070

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10300	WatchListEntry_WatchListAuditUpd	10020 10030 10040 10050 10060 10070 10260
10310	WatchListEntryAudit_WatchListEntryUpdate	10020 10030 10040 10050 10060 10070 10300
10320	Customer_KYCRiskUpd	NA
10360	DerivedAddress_FrontOfficeTransactioPartyStageInsert	NA
10370	DerivedAddress_FrontOfficeTransactioPartyStageUpd	NA
10380	FrontOfficeTransactionParty_DerivedAddress	10360 10370
10390	DerivedEntity_FrontOfficeTransactionPartyInsert	10080 10090
10400	DerivedEntity_FrontOfficeTransactionPartyUpd	10080 10090
10410	DerivedEntity_SettlementInstructionInsert	10220 10230 10240
10420	DerivedEntity_SettlementInstructionUpd	10220 10230 10240
10430	CorrespondentBank_FrontOfficeTransactionPartyStageInsert	10080 10090
10440	CorrespondentBank_FrontOfficeTransactionPartyStageUpd	10080 10090
10450	WatchListStagingTable_WatchList	10250 10260 10270 10280 10290 10300 10310
10460	WatchListStagingTable_WatchListInstnIDUpd	10250 10260 10270 10280 10290 10300 10310

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10470	PreviousWatchList_WatchList	10250 10260 10270 10280 10290 10300 10310
10480	DerivedAddress_WatchListNewCountries	10250 10260 10270 10280 10290 10300 10310
10485	WLMProcessingUnlock	10480
10490	LinkStaging_FrontOfficeTransactionParty	10360 10370 10380 10390 10400 10485
10500	LinkStaging_InstructionDerivedEntDerivedAdd	10330 10340 10350 10410 10420
10510	NameMatchStaging	10450 10460 10470 10480 10390 10400
10520	WatchListStagingTable_NameMatchStageInsert	10510
10530	DerivedEntityLink_LinkStage	10490 10500
10540	DerivedEntitytoDerivedAddress_LinkStage	10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	10490 10500

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10570	WatchListStagingTable2_WatchListStage2AcctExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10580	WatchListStagingTable2_WatchListStage2CBExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10590	WatchListStagingTable2_WatchListStage2CustExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10600	WatchListStagingTable2_WatchListStage2DAExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10610	WatchListStagingTable2_WatchListStage2EEExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10620	WatchListStagingTable2_WatchListStage	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10630	WatchListStagingTable2_AcctListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10640	WatchListStagingTable2_CBListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10650	WatchListStagingTable2_CustListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10660	WatchListStagingTable2_EEListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10670	WatchListStagingTable2_EEListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10680	WatchListStagingTable2_DAListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10690	WatchListStagingTable2_DAListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10700	WatchListStagingTable2_WatchListStage2SeqIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10710	WatchListStagingTable2_WatchListStage2IntrlIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10720	Customer_WatchListStage2ListRisk	10320 10700 10710
10730	CorrespondentBank_WatchListStage2EffectiveRisk	10320 10700 10710
10740	Customer_WatchListStage2EffectiveRisk	10320 10700 10710
10750	DerivedAddress_WatchListStage2EffectiveRisk	10320 10700 10710

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10760	DerivedEntity_WatchListStage2EffectiveRisk	10320 10700 10710
10770	WatchListStagingTable2_WatchListStage2SeqId	10320 10700 10710
10780	AccountListMembership_WatchListStage2Insert	10700 10710
10790	AccountListMembership_WatchListStage2Upd	10700 10710
10800	CorrespondentBankListMembership_WatchListStage2Insert	10700 10710
10810	CorrespondentBankListMembership_WatchListStage2Upd	10700 10710
10820	CustomerListMembership_WatchListStage2Insert	10700 10710
10830	CustomerListMembership_WatchListStage2Upd	10700 10710
10840	DerivedAddressListMembership_WatchListStage2Insert	10700 10710
10850	DerivedAddressListMembership_WatchListStage2Upd	10700 10710
10860	DerivedEntityListMembership_WatchListStage2Insert	10700 10710
10870	DerivedEntityListMembership_WatchListStage2Upd	10700 10710
10875	Account_EffectiveRiskFactorTxtUpd	10700 10701
10880	Account_OverallEffectiveRiskUpd	10720 10730 10740 10750 10760 10770 10780 10790 10800 10810 10820 10830 10840 10850 10860 10870
10881	Account_AccountCustRiskUpd	10880

Table 131. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10890	Account_EffRiskUpdAfterWLRiskRemoval	10720 10730 10740 10750 10760 10770 10880
10900	Account_WatchListStage2EffectiveRisk	10720 10730 10740 10750 10760 10770 10880
10910	WatchListStagingTable2_WatchListStage2IntrlId	10320 10700 10710
10920	BackOfficeTransaction_EffectiveAcctivityRiskUpd	10890 10900
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	10890 10900

AML Banking - Post-Watch List Datamaps

Post-Watch List Datamaps are used to ingest data into various transaction tables using Front Office and Back Office Transaction files, these are executed only after the Watch List Datamaps are run. These datamaps are used to populate data into the Cash, Wire, and Monetary Instruments tables, and to update Trusted Pair and Jurisdiction information into various other entities.

Note: Datamaps 10970,10980,10990, 11000,11010,11020 can be run in parallel.

Table 132. AML Banking - Post-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
20010	CorrespondentBank_JurisdictionUpd	10430 10440
20020	CorrespondentBank_AcctJurisdictionReUpd	10430 10440
20030	FinancialInstitution_InstNameUpd	10430 10440
10960	AccountGroup_JurisdictionUpd	NA
10970	TransactionPartyCrossReference_BackOfficeTransaction	10360 10370 10380 10940
10980	CashTransaction_FrontOfficeTransaction	10360 10370 10380 10940

Table 132. AML Banking - Post-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	10360 10370 10380 10940
11000	TransactionPartyCrossReference_FrontOfficeTransaction	10360 10370 10380 10940
11010	WireTransaction_FrontOfficeTransaction	10360 10370 10380 10940
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	10360 10370 10380 10940
11030	CashTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11040	MonetaryInstrumentTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11050	WireTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11060	TrustedPair_StatusEXPUpd	10970 10980 10990 11000 11010 11020
11070	TrustedPairMember_AcctExtEntEffecRiskUpd	10970 10980 10990 11000 11010 11020

Table 132. AML Banking - Post-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
11080	TrustedPair_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11090	TrustedPair_StatusRRCUpd	10970 10980 10990 11000 11010 11020
11100	ApprovalActionsAudit_TrustedPair	10970 10980 10990 11000 11010 11020 11060 11080 11090
11110	TrustedPairMember_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11120	BackOfficeTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11140	MonetaryInstrumentTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11150	WireTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110

AML Banking - Summary Datamaps

Summary Datamaps are used to calculate aggregations across various entities using the Trade, Transaction, Positions and Balances tables. These datamaps populate various profile tables for different entities such as Account Profile, Household Profile, and Correspondent Bank Profile. The aggregation is done either daily, weekly or monthly depending on the business areas.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 133. AML Banking - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
11160	AccountDailyProfile-Trade	NA
11170	AccountDailyProfile-Transaction	NA
11180	AccountProfile_Trade	11160
11190	AccountProfile_Transaction	11170
11200	AccountProfile_Stage (*Optional*:Run the datamap if there is any record in Account Profile Stage.)	NA
11210	AccountProfile_Position	11180 11190
11220	AccountProfile_Balance	10940 11160 11170 11180 11190 11210
20040	CorrespondentBankProfile	11180 11190 11200 11210 11220
20050	AccountATMDailyProfile	10940
11230	ChangeLog_AcctProfileInactivity	11180 11190 11200 11210 11220
11240	AccountPeerGroupMonthlyTransactionProfile	11180 11190 11200 11210 11220
20060	CorrespondentBankPeerGroupTransactionProfile	20040
20070	AccountChannelWeeklyProfile	10940
11300	AccountChangeLogSummary	The datamap should be executed once the change log processing is done.
11310	AccountToCustomerChangeLogSummary	
11320	CustomerChangeLogSummary	

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

Broker Compliance Datamaps

Broker Compliance - Pre-Watch List Datamaps

Pre-Watch List Datamaps are used to facilitate the application to populate various business areas such as Financial Institutions, Account To Client Bank, Settlement Instructions, Front Office and Back Office Transaction. These datamaps populate the relevant data which is used by watch list datamaps in calculating risks.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Before running the following datamaps, please run the AccountDailySecurityProfile utility to populate the account daily security profile data, if the deployed scenarios demand. Oracle recommends that datamaps are run in the order described below.

Table 134. Broker Compliance - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10010	EmployeeControlledAccount (*Optional*)	NA
10020	FinancialInstitution_ThomsonDataInstitutionInsert (*Optional*)	NA
10030	AccountToClientBank_ThomsonDataInstitutionInsert (*Optional*)	10020
10040	FinancialInstitution_AIIMSPopulation	NA
10050	AccountToClientBank_AIIMSInstitutionInsert	10040
10060	AccountToClientBank_InstitutionInsert	10050
10070	AccountToClientBank_InstitutionUpd	10060
10080	FinancialInstitution_FOTPSPopulation	10020 10030 10040 10050 10060 10070
10090	AccountToClientBank_FOTPSInstitutionInsert	10020 10030 10040 10050 10060 10070 10080
10100	AccountManagementStage	NA
10114	Security_CIRRatingUpd	NA
10116	BackOfficeTransaction_CollateralUpd	10114

Table 134. Broker Compliance - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10120	BackOfficeTransaction_OriginalTransactionReversalUpd	NA
10130	BackOfficeTransaction_CancelledTransactionReversalCreditUpd	NA
10140	BackOfficeTransaction_CancelledTransactionReversalDebitUpd	NA
10150	FrontOfficeTransactionParty_InstnSeqID	10020 10030 10040 10050 10060 10070 10090
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	10150
10200	CustomerAccountStage_FrontOfficeTransactionParty	NA
10210	FrontOfficeTransaction_UnrelatedPartyUpd	10120 10130 10140 10200
10220	FinancialInstitution_SettlementInstruction	10020 10030 10040 10050 10060 10070
10230	AccountToClientBank_SettlementInstruction	10020 10030 10040 10050 10060 10070 10220
10240	SettlementInstruction_AccountToClientBank	10020 10030 10040 10050 10060 10070 10230
10014	FrontOfficeTransaction_PassThroughFlag	NA

Note:

- **FrontOfficeTransaction_PassThroughFlag** - This data map should only be run if the P*ass Through Indicator* field is not being provided in the `Front Office Transaction DIS` file, and the client requires support to derive this datamap.
- **FrontOfficeTransactionParty_SecondaryNames** - This data map should only be run if Secondary Originator and Secondary Beneficiary party records are not being provided in in the Front Office Transaction Party DIS file, and the client requires support to derive them from the Bank-to-Bank Instructions and Originator-to-Beneficiary Instructions fields.

Broker Compliance - Post-Watch List Datamaps

Oracle recommends that datamaps are run in the order described below.

Table 135. Broker Compliance - Post-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	10150
10200	CustomerAccountStage_FrontOfficeTransactionParty	NA
10955	AccountGroup_InvestmentObjectiveUpd	NA
10960	AccountGroup_JurisdictionUpd	NA

Broker Compliance - Balances and Positions Datamaps

Balances and Positions Datamaps derive attributes that are useful in assessment of the financial status of an account, customer, or Household. These datamaps are used to populate business areas such as, account balance, account position, portfolio manager positions, and so on.

Table 136. Broker Compliance - Balances and Positions Datamaps

Datamap Number	Datamap Name	Predecessors
60010	PortfolioManagerPosition	NA
60020	AccountGroupProductAllocation	NA
60030	AccountProductAllocation	NA
60145	AccountPosition_Percentof PortfolioUpd	NA
60150	AccountPositionDerived	NA
60160	AccountBalance_AcctPosnPair	60150
60170	AccountBalance_Acctposn	60150
60180	HouseholdBalance	60160 60170

Broker Compliance - Summary Datamaps

Summary Datamaps are used to calculate aggregations across various entities using the Trade, Transaction, Positions and Balances tables. These datamaps populate various profile tables for different entities, such as Account Profile, Household Profile, and Correspondent Bank Profile. The aggregation is done either daily, weekly or monthly depending on the business areas.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 137. Broker Compliance - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
60040	UncoveredOptionExposureDaily	NA
60050	InvestmentAdvisorProfile	NA

Table 137. Broker Compliance - Summary Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
60060	RegisteredRepresentativeProfile	NA
60100	ManagedAccountDailyProfile_SameDayTrade	NA
60110	ManagedAccountDailyProfile_Trade	NA
60120	ManagedAccountDailyProfile_BOT	NA
11160	AccountDailyProfile-Trade	NA
11170	AccountDailyProfile-Transaction	10940 10950
11180	AccountProfile_Trade	11160
11190	AccountProfile_Transaction	11170 11180
11200	AccountProfile_Stage (*Optional*:Run the datamap if there is any record in Account Profile Stage.)	11190
11210	AccountProfile_Position	11170 11180 60150
11220	AccountProfile_Balance	11180 11120 60160 60170
60130	HouseholdProfile	11180 11190 11200 11210 11220
60070	RegOToBorrower (*Optional)*	NA
60080	InterestedPartyToEmployee (*Optional)*	NA
60140	ManagedAccountProfile	60100 60110 60120
11300	AccountChangeLogSummary	The datamap should be executed once the change log processing is done.
11310	AccountToCustomerChangeLogSummary	
11320	CustomerChangeLogSummary	

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

Fraud Detection Datamaps

Fraud Detection - Pre-Watch List Datamaps

Pre-Watch List Datamaps are used to facilitate the application to populate various business areas such as, Financial Institutions, Account To Client Bank, Settlement Instructions, Front Office and Back Office Transaction. These datamaps populate the relevant data which would be used in watch list datamaps in calculating risks.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 138. Fraud Detection - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10010	EmployeeControlledAccount (*Optional)*	NA
10015	FrontOfficeTransactionParty_SecondaryNames	NA
10020	FinancialInstitution_ThomsonDataInstitutionInsert (*Optional*)	NA
10030	AccountToClientBank_ThomsonDataInstitutionInsert (*Optional*)	10020
10040	FinancialInstitution_AIIMSPopulation	NA
10050	AccountToClientBank_AIIMSInstitutionInsert	10040
10060	AccountToClientBank_InstitutionInsert	10050
10070	AccountToClientBank_InstitutionUpd	10060
10080	FinancialInstitution_FOTPSPopulation	10020 10030 10040 10050 10060 10070
10090	AccountToClientBank_FOTPSInstitutionInsert	10020 10030 10040 10050 10060 10070 10080
10100	AccountManagementStage	NA
10114	BackOfficeTransaction_UnrelatedPartyCodeUpd	NA
10116	BackOfficeTransaction_CollateralUpd	10114
10120	BackOfficeTransaction_OriginalTransactionReversalUpd	NA
10130	BackOfficeTransaction_CancelledTransactionReversalCreditUpd	NA
10140	BackOfficeTransaction_CancelledTransactionReversalDebitUpd	NA

Table 138. Fraud Detection - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10150	FrontOfficeTransactionParty_InstnSeqID	10020 10030 10040 10050 10060 10070
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	10150
10170	FinancialInstitution_AnticipatoryProfile	10020 10030 10040 10050 10060 10070
10180	AccountToClientBank_AnticipatoryProfile	10020 10030 10040 10050 10060 10070 10170
10190	AnticipatoryProfile_AccountToClientBank	10170 10180
10200	CustomerAccountStage_FrontOfficeTransactionParty	NA
10210	FrontOfficeTransaction_UnrelatedPartyUpd	10120 10130 10140 10200
10220	FinancialInstitution_SettlementInstruction	10020 10030 10040 10050 10060 10070
10230	AccountToClientBank_SettlementInstruction	10020 10030 10040 10050 10060 10070 10220
10240	SettlementInstruction_AccountToClientBank	10020 10030 10040 10050 10060 10070 10230
10014	FrontOfficeTransaction_PassThroughFlag	NA

Note:

- **FrontOfficeTransaction_PassThroughFlag** - This data map should only be run if the P*ass Through Indicator* field is not being provided in the `Front Office Transaction DIS` file, and the client requires support to derive this datamap.
- **FrontOfficeTransactionParty_SecondaryNames** - This data map should only be run if Secondary Originator and Secondary Beneficiary party records are not being provided in in the Front Office Transaction Party DIS file, and the client requires support to derive them from the Bank-to-Bank Instructions and Originator-to-Beneficiary Instructions fields.

Fraud Detection - Watch List Datamaps

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

These datamaps finally assist other datamaps which are used to calculate Effective Risk and Activity Risk for various entities, such as Account, Customer, Transaction, and so on.

Table 139. Fraud Detection - Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10245	WLMProcessingLock	NA
10250	WatchListEntry_WatchListEntryCurrDayInsert	10020 10030 10040 10050 10060 10070 10245
10260	WatchListAudit_StatusUpd	10020 10030 10040 10050 10060 10070
10270	WatchList_WatchListSourceAuditInsert	10020 10030 10040 10050 10060 10070 10260
10280	WatchList_WatchListSourceAuditUpd	10020 10030 10040 10050 10060 10070
10290	WatchList_WatchListSourceUpd	10020 10030 10040 10050 10060 10070

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10300	WatchListEntry_WatchListAuditUpd	10020 10030 10040 10050 10060 10070 10260
10310	WatchListEntryAudit_WatchListEntryUpdate	10020 10030 10040 10050 10060 10070 10300
10320	Customer_KYCRiskUpd	NA
10330	DerivedAddress_SettlementInstructionInsert	NA
10340	DerivedAddress_SettlementInstructionUpd	NA
10350	SettlementInstruction_PhysicalDlvryAddrUpd	NA
10360	DerivedAddress_FrontOfficeTransactioPartyStageInsert	NA
10370	DerivedAddress_FrontOfficeTransactioPartyStageUpd	NA
10380	FrontOfficeTransactionParty_DerivedAddress	NA
10390	DerivedEntity_FrontOfficeTransactionPartyInsert	10080 10090
10400	DerivedEntity_FrontOfficeTransactionPartyUpd	10080 10090
10410	DerivedEntity_SettlementInstructionInsert	10220 10230 10240
10420	DerivedEntity_SettlementInstructionUpd	10220 10230 10240
10430	CorrespondentBank_FrontOfficeTransactionPartyStageInsert	10080 10090
10440	CorrespondentBank_FrontOfficeTransactionPartyStageUpd	10080 10090
10450	WatchListStagingTable_WatchList	10250 10260 10270 10280 10290 10300 10310

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10460	WatchListStagingTable_WatchListInstnIDUpd	10250 10260 10270 10280 10290 10300 10310
10470	PreviousWatchList_WatchList	10250 10260 10270 10280 10290 10300 10310
10480	DerivedAddress_WatchListNewCountries	10250 10260 10270 10280 10290 10300 10310
10485	WLMProcessingUnlock	10480
10490	LinkStaging_FrontOfficeTransactionParty	10360 10370 10380 10390 10400 10485
10500	LinkStaging_InstructionDerivedEntDerivedAdd	10330 10340 10350 10410 10420
10510	NameMatchStaging	10450 10460 10470 10480 10390 10400
10520	WatchListStagingTable_NameMatchStageInsert	10510
10530	DerivedEntityLink_LinkStage	10490 10500
10540	DerivedEntitytoDerivedAddress_LinkStage	10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	10490 10500

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10570	WatchListStagingTable2_WatchListStage2AcctExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10580	WatchListStagingTable2_WatchListStage2CBExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10590	WatchListStagingTable2_WatchListStage2CustExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10600	WatchListStagingTable2_WatchListStage2DAExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10610	WatchListStagingTable2_WatchListStage2EEExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10620	WatchListStagingTable2_WatchListStage	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10630	WatchListStagingTable2_AcctListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10640	WatchListStagingTable2_CBListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10650	WatchListStagingTable2_CustListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10660	WatchListStagingTable2_EEListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10670	WatchListStagingTable2_EEListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10680	WatchListStagingTable2_DAListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10690	WatchListStagingTable2_DAListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10700	WatchListStagingTable2_WatchListStage2SeqIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10710	WatchListStagingTable2_WatchListStage2IntrlIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10720	Customer_WatchListStage2ListRisk	10320 10700 10710
10730	CorrespondentBank_WatchListStage2EffectiveRisk	10320 10700 10710
10740	Customer_WatchListStage2EffectiveRisk	10320 10700 10710
10750	DerivedAddress_WatchListStage2EffectiveRisk	10320 10700 10710

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10760	DerivedEntity_WatchListStage2EffectiveRisk	10320 10700 10710
10770	WatchListStagingTable2_WatchListStage2SeqId	10320 10700 10710
10780	AccountListMembership_WatchListStage2Insert	10700 10710
10790	AccountListMembership_WatchListStage2Upd	10700 10710
10800	CorrespondentBankListMembership_WatchListStage2Insert	10700 10710
10810	CorrespondentBankListMembership_WatchListStage2Upd	10700 10710
10820	CustomerListMembership_WatchListStage2Insert	10700 10710
10830	CustomerListMembership_WatchListStage2Upd	10700 10710
10840	DerivedAddressListMembership_WatchListStage2Insert	10700 10710
10850	DerivedAddressListMembership_WatchListStage2Upd	10700 10710
10860	DerivedEntityListMembership_WatchListStage2Insert	10700 10710
10870	DerivedEntityListMembership_WatchListStage2Upd	10700 10710
10875	Account_EffectiveRiskFactorTxtUpd	10700 10710
10880	Account_OverallEffectiveRiskUpd	10720 10730 10740 10750 10760 10770 10780 10790 10800 10810 10820 10830 10840 10850 10860 10870
10881	Account_AccountCustRiskUpd	10880

Table 139. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10890	Account_EffRiskUpdAfterWLRiskRemoval	10720 10730 10740 10750 10760 10770 10880
10900	Account_WatchListStage2EffectiveRisk	10720 10730 10740 10750 10760 10770 10880
10910	WatchListStagingTable2_WatchListStage2IntrlId	10320 10700 10710
10920	BackOfficeTransaction_EffectiveAcctivityRiskUpd	10890 10900
10930	SettlementInstruction_EntityAcctivityRiskUpd	10890 10900
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	10890 10900

Fraud Detection - Post-Watch List Datamaps

Post-Watch List Datamaps are used to populate or rather ingest data into various transaction tables using Front Office and Back Office Transaction files, these are executed only after the Watch List Datamaps are run.

These datamaps are used to populate data into Cash, Wire, Monetary Instruments tables, and to update Trusted Pair and Jurisdiction information into various other entities. These datamaps (10970,10980,10990, 11000,11010,11020) can be run in parallel.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 140. Fraud Detection - Post-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10960	AccountGroup_JurisdictionUpd	NA
10970	TransactionPartyCrossReference_BackOfficeTransaction	10360 10370 10380 10940
10980	CashTransaction_FrontOfficeTransaction	10360 10370 10380 10940

Table 140. Fraud Detection - Post-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	10360 10370 10380 10940
11000	TransactionPartyCrossReference_FrontOfficeTransaction	10360 10370 10380 10940 11060 11070 11080 11090
11010	WireTransaction_FrontOfficeTransaction	10360 10370 10380 10940
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	10360 10370 10380 10940
11030	CashTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11040	MonetaryInstrumentTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11050	WireTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020
11060	TrustedPair_StatusEXPUpd	10970 10980 10990 11000 11010 11020
11070	TrustedPairMember_AcctExtEntEffecRiskUpd	10970 10980 10990 11000 11010 11020

Table 140. Fraud Detection - Post-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
11080	TrustedPair_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11090	TrustedPair_StatusRRCUpd	10970 10980 10990 11000 11010 11020
11100	ApprovalActionsAudit_TrustedPair	10970 10980 10990 11000 11010 11020
11110	TrustedPairMember_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11120	BackOfficeTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11140	MonetaryInstrumentTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11150	WireTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110

Fraud Detection - Summary Datamaps Detection

Summary Datamaps are used to calculate aggregations across various entities using Trade, Transaction, Positions and Balances tables. These datamaps populate various profile tables for different entities, such as Account Profile, Household Profile, and Correspondent Bank Profile. The aggregation is done either daily, weekly or monthly depending on the business areas.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 141. Fraud Detection - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
11160	AccountDailyProfile-Trade	NA
11170	AccountDailyProfile-Transaction	10940
11180	AccountProfile_Trade	11160
11190	AccountProfile_Transaction	11170
11200	AccountProfile_Stage (*Optional:* Run the datamap if there is any record in Account Profile Stage)	11180 11190
11210	AccountProfile_Position	11180 11190
11220	AccountProfile_Balance	11180 11190 11210
11230	ChangeLog_AcctProfileInactivity	11180 11190 11200 11210 11220
11240	AccountPeerGroupMonthlyTransactionProfile	11180 11190 11200 11210 11220
11300	AccountChangeLogSummary	The datamap should be executed once the change log processing is done.
11310	AccountToCustomerChangeLogSummary	
11320	CustomerChangeLogSummary	

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

Insurance Datamaps

Insurance - Pre-Watch List Datamaps

Pre-Watch List Datamaps are used to facilitate the application to populate various business areas such as, Financial Institutions, Account To Client Bank, Settlement Instructions, Front Office and Back Office Transaction. These datamaps populate the relevant data which would again be used in watch list datamaps in calculating risks.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 142. Insurance - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10010	EmployeeControlledAccount (*Optional*)	NA
10020	FinancialInstitution_ThomsonDataInstitutionInsert (*Optional*)	NA
10030	AccountToClientBank_ThomsonDataInstitutionInsert (*Optional*)	10020
10040	FinancialInstitution_AIIMSPopulation	NA
10050	AccountToClientBank_AIIMSInstitutionInsert	10040
10060	AccountToClientBank_InstitutionInsert	10050
10070	AccountToClientBank_InstitutionUpd	10060
10080	FinancialInstitution_FOTPSPopulation	10020 10030 10040 10050 10060 10070
10090	AccountToClientBank_FOTPSInstitutionInsert	10020 10030 10040 10050 10060 10070 10080
10100	AccountManagementStage	NA
10114	BackOfficeTransaction_UnrelatedPartyCodeUpd	NA
10116	BackOfficeTransaction_CollateralUpd	10114
10150	FrontOfficeTransactionParty_InstnSeqID	10020 10030 10040 10050 10060 10070
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	10150
10170	FinancialInstitution_AnticipatoryProfile	10020 10030 10040 10050 10060 10070

Table 142. Insurance - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10180	AccountToClientBank_AnticipatoryProfile	10020 10030 10040 10050 10060 10070 10170
10190	AnticipatoryProfile_AccountToClientBank	10020 10030 10040 10050 10060 10070 10180
10220	FinancialInstitution_SettlementInstruction	10020 10030 10040 10050 10060 10070
10230	AccountToClientBank_SettlementInstruction	10020 10030 10040 10050 10060 10070 10220
10240	SettlementInstruction_AccountToClientBank	10020 10030 10040 10050 10060 10070 10230
40010	FinancialInstitution_InsuranceTransaction	10020 10030 10040 10050 10060 10070

Table 142. Insurance - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
40020	AccountToClientBank_InsuranceTransaction	10020 10030 10040 10050 10060 10070 40010
40030	InsuranceTransaction_AccountToClientBank	10020 10030 10040 10050 10060 10070 40020

Insurance - Watch List Datamaps

Watch List Datamaps facilitate the application of customer-supplied measures of risk to corresponding entities, transactions, and instructions. These datamaps assist other datamaps which are used to calculate Effective Risk and Activity Risk for various entities, such as, Account, Customer, Transaction tables, and so on.

Table 143. Insurance - Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10245	WLMProcessingLock	NA
10250	WatchListEntry_WatchListEntryCurrDayInsert	10020 10030 10040 10050 10060 10070 10245
10260	WatchListAudit_StatusUpd	10020 10030 10040 10050 10060 10070
10270	WatchList_WatchListSourceAuditInsert	10020 10030 10040 10050 10060 10070
10280	WatchList_WatchListSourceAuditUpd	10020 10030 10040 10050 10060 10070

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10290	WatchList_WatchListSourceUpd	10020 10030 10040 10050 10060 10070
10300	WatchListEntry_WatchListAuditUpd	10020 10030 10040 10050 10060 10070
10310	WatchListEntryAudit_WatchListEntryUpdate	10020 10030 10040 10050 10060 10070
10320	Customer_KYCRiskUpd	NA
10360	DerivedAddress_FrontOfficeTransactioPartyStageInsert	NA
10370	DerivedAddress_FrontOfficeTransactioPartyStageUpd	NA
10380	FrontOfficeTransactionParty_DerivedAddress	NA
40040	DerivedAddress_InsuranceTransactionInsert	NA
40050	DerivedAddress_InsuranceTransactionUpd	NA
40060	InsuranceTransaction_InstitutionAddrUpd	NA
40070	DerivedEntity_InsuranceTransactionInsert	40010 40020 40030
40080	DerivedEntity_InsuranceTransactionUpd	40010 40020 40030
10390	DerivedEntity_FrontOfficeTransactionPartyInsert	10080 10090
10400	DerivedEntity_FrontOfficeTransactionPartyUpd	10080 10090
10410	DerivedEntity_SettlementInstructionInsert	10220 10230 10240
10420	DerivedEntity_SettlementInstructionUpd	10220 10230 10240
10430	CorrespondentBank_FrontOfficeTransactionPartyStageInsert	10080 10090
10440	CorrespondentBank_FrontOfficeTransactionPartyStageUpd	10080 10090

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10450	WatchListStagingTable_WatchList	10250 10260 10270 10280 10290 10300 10310
10460	WatchListStagingTable_WatchListInstnIDUpd	10250 10260 10270 10280 10290 10300 10310
10470	PreviousWatchList_WatchList	10250 10260 10270 10280 10290 10300 10310
10480	DerivedAddress_WatchListNewCountries	10250 10260 10270 10280 10290 10300 10310
10485	WLMProcessingUnlock	10480
10490	LinkStaging_FrontOfficeTransactionParty	10360 10370 10380 10390 10400
40090	LinkStaging_InsTrxnDerivedEntDerivedAdd	40040 40050 40060 40070 40080
10500	LinkStaging_InstructionDerivedEntDerivedAdd	10330 10340 10350 10410 10420
10510	NameMatchStaging	10450 10460 10470 10480 10390 10400
10520	WatchListStagingTable_NameMatchStageInsert	10510

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10530	DerivedEntityLink_LinkStage	40090 10490 10500
10540	DerivedEntitytoDerivedAddress_LinkStage	40090 10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	40090 10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	40090 10490 10500
10570	WatchListStagingTable2_WatchListStage2AcctExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10580	WatchListStagingTable2_WatchListStage2CBExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10590	WatchListStagingTable2_WatchListStage2CustExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10600	WatchListStagingTable2_WatchListStage2DAExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10610	WatchListStagingTable2_WatchListStage2EEExistence	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10620	WatchListStagingTable2_WatchListStage	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10630	WatchListStagingTable2_AcctListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10640	WatchListStagingTable2_CBListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10650	WatchListStagingTable2_CustListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10660	WatchListStagingTable2_EEListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10670	WatchListStagingTable2_EEListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10680	WatchListStagingTable2_DAListMembershipUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10690	WatchListStagingTable2_DAListMembershipStatusUpd	10450 10460 10470 10480 10390 10400 10510 10520 10410 10420 10430 10440
10700	WatchListStagingTable2_WatchListStage2SeqIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690
10710	WatchListStagingTable2_WatchListStage2IntrlIdUpd	10570 10580 10590 10600 10610 10620 10630 10640 10650 10660 10670 10680 10690

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10720	Customer_WatchListStage2ListRisk	10320 10700 10710
10730	CorrespondentBank_WatchListStage2EffectiveRisk	10320 10700 10710
10740	Customer_WatchListStage2EffectiveRisk	10320 10700 10710
10750	DerivedAddress_WatchListStage2EffectiveRisk	10320 10700 10710
10760	DerivedEntity_WatchListStage2EffectiveRisk	10320 10700 10710
10770	WatchListStagingTable2_WatchListStage2SeqId	10320 10700 10710
10780	AccountListMembership_WatchListStage2Insert	10700 10710
10790	AccountListMembership_WatchListStage2Upd	10700 10710
10800	CorrespondentBankListMembership_WatchListStage2Insert	10700 10710
10810	CorrespondentBankListMembership_WatchListStage2Upd	10700 10710
10820	CustomerListMembership_WatchListStage2Insert	10700 10710
10830	CustomerListMembership_WatchListStage2Upd	10700 10710
10840	DerivedAddressListMembership_WatchListStage2Insert	10700 10710
10850	DerivedAddressListMembership_WatchListStage2Upd	10700 10710
10860	DerivedEntityListMembership_WatchListStage2Insert	10700 10710
10870	DerivedEntityListMembership_WatchListStage2Upd	10700 10710
10875	Account_EffectiveRiskFactorTxtUpd	10700 10710

Table 143. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10880	Account_OverallEffectiveRiskUpd	10720 10730 10740 10750 10760 10770 10780 10790 10800 10810 10820 10830 10840 10850 10860 10870
10881	Account_AccountCustRiskUpd	10880
10890	Account_EffRiskUpdAfterWLRiskRemoval	10720 10730 10740 10750 10760 10770 10880
10900	Account_WatchListStage2EffectiveRisk	10720 10730 10740 10750 10760 10770 10880
10910	WatchListStagingTable2_WatchListStage2IntrlId	10320 10700 10710
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	10890 10900
40100	InsuranceTransaction_EntityAcctivityRiskUpd	10890 10900

Insurance - Post-Watch List Datamaps

Post-Watch List Datamaps are used to populate or ingest data into various transaction tables using Front Office and Back Office Transaction files, these are executed only after the Watch List Datamaps are run. These datamaps are used to populate data into Cash, Wire, Monetary Instruments tables, and to update Trusted Pair and Jurisdiction information into various other entities.

Table 144. Insurance - Post-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
11060	TrustedPair_StatusEXPUpd	10970 10980 10990 11000 11010 11020
11070	TrustedPairMember_AcctExtEntEffecRiskUpd	10970 10980 10990 11000 11010 11020
11080	TrustedPair_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11090	TrustedPair_StatusRRCUpd	10970 10980 10990 11000 11010 11020 11060 11070 11080 11090
11100	ApprovalActionsAudit_TrustedPair	10970 10980 10990 11000 11010 11020
11110	TrustedPairMember_StatusRRCInsert	10970 10980 10990 11000 11010 11020
11120	BackOfficeTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110

Table 144. Insurance - Post-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
11130	InsuranceTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11140	MonetaryInstrumentTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110
11150	WireTransaction_TrustedFlagsUpd	11060 11070 11080 11090 11100 11110

Insurance - Summary Datamaps

Summary Datamaps are used to calculate aggregations across various entities using Trade, Transaction, Positions and Balances tables. These datamaps populate various profile tables for different entities such as Account Profile, Household Profile, Correspondent Bank Profile, the aggregation is done either daily, weekly or monthly depending on the business areas. The following table describes the Summary datamaps for Insurance.

Note: To execute the datamap `WatchListStagingTable_WatchListInstnIDUpd` against 1.5 million records, the tempspace should be set to 400GB or above.

Table 145. Insurance - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
40110	InsurancePolicyDailyProfile_InsTrxnInsPolicyBal	NA
40120	InsurancePolicyProfile_InsurancePolicyDailyProfile	40110
11300	AccountChangeLogSummary	The datamap should be executed once the change log processing is done.
11310	AccountToCustomerChangeLogSummary	
11320	CustomerChangeLogSummary	

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

Trade Finance Datamaps

Trade Finance - Pre-Watch List Datamaps

Pre-Watch List Datamaps are used to facilitate the application to populate various business areas such as, Financial Institutions, Account To Client Bank, Settlement Instructions, Front Office and Back Office Transaction. These datamaps populate the relevant data which would again be used in watch list datamaps in calculating risks.

Optional Datamaps are used to perform processing to support other datamaps in multiple functional areas. These datamaps may or may not be completely relevant to a particular solution set. Execute the datamap if a scenario in your implementation requires this information.

Table 146. Trade Finance - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
60200	TradeFinanceContractEvent.xml	NA
60210	TradeFinanceContractEventAcknowledgementStage.xml	NA
60220	TradeFinanceContractAmendmentStatusStage.xml	NA
60230	TradeFinanceContractEvent_AcknowledgeUpd.xml	60200 60210
60240	TradeFinanceContractEvent_AmendmentUpd.xml	60200 60220
60250	TradeFinanceContract.xml	60200
60260	TradeFinancetoAccount.xml	NA
60270	TradeFinanceDocument.xml	NA
60280	TradeFinanceDraft.xml	NA
60290	TradeFinanceGoodorService.xml	NA
60300	TradeFinanceParty.xml	NA
60310	TradeFinanceParty_TradeFinancePartyStage.xml	60300
60320	TradeFinanceContract_PartyUpd.xml	60300 60240 60230 60220 60210 60200
60330	TradeFinanceContract_DocUpd.xml	60270 60240 60230 60220 60210 60200
60340	TradeFinanceContract_GoodsUpd.xml	60290 60240 60230 60220 60210 60200
60350	DerivedAddress_TradeFinancePartyInsert.xml	60300 60310

Table 146. Trade Finance - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
60360	DerivedAddress_TradeFinancePartyUpd.xml	60350 60300 60310
60370	TradeFinancePartyTF_DerivedAddressUpd.xml	60360 60350 60310 60300
60380	TradeFinancePartyDC_DerivedAddressUpd.xml	60370 60360 60350 60310 60300
60390	FinancialInstitution_TradeFinanceParty.xml	60300 60310
60400	DerivedEntity_TradeFinancePartyInsert.xml	60300 60310
60410	DerivedEntity_TradeFinancePartyUpd.xml	60300 60310 60400
60420	TradeFinancePartyTF_DerivedEntityUpd.xml	60410 60400 60310 60300
60430	TradeFinancePartyDC_DerivedEntityUpd.xml	60410 60400 60310 60300
60440	TradeFinancePartyTF_EntityActivityRiskUpd.xml	60420 60410 60400 60310 60300
60450	TradeFinancePartyDC_EntityActivityRiskUpd.xml	60420 60410 60400 60310 60300
60460	CustomerImportLicense.xml	NA
60470	CustomerImportLicensetoGoods.xml	NA
60480	TradeFinanceBrokerage.xml	NA
60490	ExternalInsurancePolicy.xml	NA
60500	ExternalPartyStage.xml	NA
60510	ExternalParty.xml	60500
60520	DerivedAddress_ExternalPartyStageInsert.xml	60510 60500
60530	DerivedAddress_ExternalPartyStageUpd.xml	60520 60510 60500

Table 146. Trade Finance - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
60540	ExternalParty_DerivedAddress.xml	60530 60520 60510 60500
60550	DerivedEntity_ExtrlOrgInsert.xml	60540 60530 60520 60510 60500
60560	TradeFinanceBrokerageDistributionStage.xml	NA
60570	TradeFinanceBrokerageDistribution.xml	60550
60580	FinancialInstitution_BrokerageDistribution.xml	60560 60550 60390 60300 60310
60590	BrokerageDistribution_FinancialInstnUpd.xml	60570 60560 60550 60390 60300 60310
60600	DerivedAddress_TradeFinanceBrokerageDistributionStageInsert.xml	60580 60570 60560 60550 60390 60300 60310
60610	DerivedAddress_TradeFinanceBrokerageDistributionStageUpd.xml	60590 60580 60570 60560 60550 60390 60300 60310
60620	BrokerageDistribution_DerivedAddress.xml	60600 60590 60580 60570 60560 60550 60390 60300 60310
60630	DocumentaryCollectionContractEvent.xml	NA
60640	DocCollectionContractAcknowlegementStage.xml	NA
60650	DocumentaryCollectionContractAccePTAceStage.xml	NA

Table 146. Trade Finance - Pre-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
60660	DocumentaryCollectionContractEvent_AcknowledgeUpd.xml	60620 60630
60670	DocumentaryCollectionContractEvent_AcceptanceUpd.xml	60620 60640
60680	DocumentaryCollectionDiscrepancyDetail.xml	NA
60690	DocumentaryCollectionDiscrepancyDetail_DiscrpDtUpd.xml	60620 60670
60700	DocumentaryCollectionInvoice.xml	NA
60710	DocumentaryCollectionMulti-tenorDetail.xml	NA
60720	DocumentaryCollectionShipmentDetail.xml	NA
60730	DocumentaryCollectionContract.xml	60630 60640 60650 60660 60670 60680 60690

Trade Finance- Post-Watch List Datamaps

Post-Watch List Datamaps are used to populate or ingest data into various transaction tables using Front Office and Back Office Transaction files, these are executed only after the Watch List Datamaps are run. These datamaps are used to populate data into Cash, Wire, Monetary Instruments tables, and to update Trusted Pair and Jurisdiction information into various other entities

Table 147. Trade Finance - Post-Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
60730	DocumentaryCollectionContract_LiquidationUpd.xml	60720
60740	TradeFinancePartyTF_EntityActivityRiskUpd.xml	NA
60750	TradeFinancePartyDC_EntityActivityRiskUpd.xml	NA
60760	ExternalParty_ExternalEntitySeqUpd.xml	60200 60210
60770	ExternalParty_EntityRiskInsert.xml	60200 60220

Processing BD Datamaps

The following table provides a list of datamaps and description for each datamap. These datamaps are listed in order.

Table 148. BD Datamaps

Datamap Number	Datamap Name	Description
10010	EmployeeControlledAccount	This datamap creates entry for Employee personal accounts and Employee Related account using same tax ID
60010	PortfolioManagerPosition	The datamap is used to populate the portfolio manager positions.It reads tables (Account and Account Position), populated while executing Pre-processors and creates records to populate the PORTFOLIO_MGR_POSN table.
60020	AccountGroupProductAllocation	The datamap captures the actual proportionate distribution of holdings for an account group aggregated by reporting classifications.
60030	AccountProductAllocation	The datamap captures the actual proportionate distribution of holdings for an account aggregated by product classifications.
60040	UncoveredOptionExposureDaily	This datamap derives the value from the uncvrd_optns_smry_dly table and insert/updates the records in UNCVRD_OPTNS_EXPOSURE_DLY table.
60050	InvestmentAdvisorProfile	This datamap updates the Investment Manager Summary Month table from the daily activity
60060	RegisteredRepresentativeProfile	This datamap updates the Registered Representative Summary Month table with daily activity
60070	RegOToBorrower	This datamap use the fuzzy match logic to match the Regulation O list against the Borrower.
60080	InterestedPartyToEmployee	This datamap use fuzzy matcher to match Interested Parties in Account Scheduled Event table against Employee name.
50010	Customer_TotAcctUpd	This datamap calculates the total number of accounts for an institutional customer.
10015	FrontOfficeTransactionParty_SecondaryNames	This datamap kicks off the Pass Thru process. It generates second orgininator and beneficiary records for Front Office Transaction. It also sets the pass thru flag based on the a set of expressions.
10020	FinancialInstitution_ThomsonDataInstitutionInsert	This datamap builds the many-to-one relationship in INSTN_MASTER that is the relationships between bics and feds with INSTN_SEQ_ID. The INSTN_MASTER table gets populated from BANK_REFERENCE_STAGE table.
10030	AccountToClientBank_ThomsonDataInstitutionInsert	This datamap builds the many-to-one relationship in ACCT_ID_INSTN_ID_MAP that is the relationships between bics and feds with INSTN_SEQ_ID. The ACCT_ID_INSTN_ID_MAP table gets populated from BANK_REFERENCE_STAGE table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10040	FinancialInstitution_AIIMSPopulation	This datamap inserts new records in Financial Institution table from the ACCT_INSTN_MAP_STAGE table, the datamap creates unique identifiers for banks based on the third party vendors.
10050	AccountToClientBank_AIIMSInstitutionInsert	This datamap creates unique identifiers for banks based BIC records on the third party vendors. 1) Retrieve Institution information from ACCT_INSTN_MAP_STAGE in comparison of INSTN_MASTER and loads it into ACCT_ID_INSTN_ID_MAP.
10060	AccountToClientBank_InstitutionInsert	This datamap creates unique identifiers for banks based on the third party vendors. 1) Retrieve Institution information from ACCT_INSTN_MAP_STAGE and load it into ACCT_ID_INSTN_ID_MAP.
10070	AccountToClientBank_InstitutionUpd	This datamap updates unique identifiers for banks based on the third party vendors. 1) Retrieve Institution information from ACCT_INSTN_MAP_STAGE and update it into ACCT_ID_INSTN_ID_MAP.
10080	FinancialInstitution_FOTPSPopulation	This datamap inserts new records in Financial Institution table for the institutions found in front office transaction party table for both party ID type code as IA and BIC, INSTN_SEQ_ID are OFSAAI generated.
10090	AccountToClientBank_FOTPSInstitutionInsert	This datamap marks all institutions with an OFSAAI generated INTSN_SEQ_ID in FOTPS. 1) Prior to this datamap execution the predecessor datamaps finds the new institutions from the transaction data and loads them in the INSTITUTION_MASTER. 2) This data map finds the new institutions from the transaction data for IA and BIC party ID type and loads them in the ACCT_ID_INSTN_ID_MAP table using OFSAAI generated INTSN_SEQ_ID from INSTITUTION_MASTER.
10100	AccountManagementStage	This datamap identifies the relationship between accounts and the employees who have a management role on that account. Management roles include positions such as Financial Advisor, Banker, and Registered Representative.
10110	LoanProfile_LoanProfileStage	This datamap is used to populate Loan Summary from LOAN_SMRY_MNTH_STAGE table. 1) Select set of information/columns from LOAN_SMRY_MNTH_STAGE table, if the record is new insert the details in LOAN_SMRY_MNTH else update the existing record.
10112	ServiceTeam_SprvsncdUpd	This datamap updates service team table with the Employee Maximum Supervision Code.
10113	InvestmentAdvisor_MangdAcctUpd	This datamap updates ManagedAccountNetworth and ActiveSubAccountCount column in InvestmentAdvisor table.
10114	Security_CIRRatingUpd	This datamap derives the column CIRRating and updates back to Security table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10116	BackOfficeTransaction_CollateralUpd	This datamap updates Collateral Percentage , Collateral Value for that transaction.
10120	BackOfficeTransaction_OriginalTransactionReversalUpd	This datamap handles reverserals for Back Office Transactions. 1) Select the set of information from today's BackOfficeTransaction to update records with columns CXL_PAIR_TRXN_INTRL_ID in BackOfficeTransaction table. 2) Updates the "cancellation pair" column in the original back office transaction table as per the "Internal ID" of the reversing or adjusting record.
10130	BackOfficeTransaction_CancelledTransactionReversalCreditUpd	This datamap updates Cancelled Transaction details for CREDIT record of Back Office Transactions. 1) Finds original-reversal back-office transaction pairs, links them via their respective transaction identifiers. 2) For original transactions: update Canceled Pairing Transaction Identifier by reversal transaction ID;3) For reversal transactions: update the transaction's Debit Credit Code, Unit Quantity, Transaction Amount, Canceled Pairing Transaction Identifier by original transaction's field values, and Mantas Transaction Adjustment Code by 'REV'.
10140	BackOfficeTransaction_CancelledTransactionReversalDebitUpd	This datamap updates Cancelled Transaction details for DEBIT record of Back Office Transactions. 1) Finds original-reversal back-office transaction pairs, links them via their respective transaction identifiers. 2) For original transactions: update Canceled Pairing Transaction Identifier by reversal transaction ID; 3) For reversal transactions: update the transaction's Debit Credit Code, Unit Quantity, Transaction Amount, Canceled Pairing Transaction Identifier by original transaction's field values, and Mantas Transaction Adjustment Code by 'REV'.
10150	FrontOfficeTransactionParty_InstnSeqID	This datamap marks all the records of FO_TRXN_PARTY_STAGE table with institutions by OFSAAI generated INTSN_SEQ_ID.
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	This datamap marks all the records of FO_TRXN_PARTY_STAGE table with institutions by OFSAAI generated INTSN_SEQ_ID. 1) To update HOLDG_INSTN_SEQ_ID and HOLDG_ADDR_CNTRY_CD based on DATA_DUMP_DT and country code (BASE_COUNTRY).
10170	FinancialInstitution_AnticipatoryProfile	This datamap inserts new records in Financial Institution table for the institutions found in Anticipatory Profile table, INSTN_SEQ_ID are OFSAAI generated. This datamap should be executed before AccountToClientBank_AnticipatoryProfile datamap as generated INSTN_SEQ_ID will be used to populate Anticipatory Profile table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10180	AccountToClientBank_AnticipatoryProfile	This datamap marks all institutions with an OFSAAI generated INTSN_SEQ_ID in FOTPS. 1) Prior to this datamap execution the predecessor datamaps finds the new NTCPTRY_PRFL from the transaction data and loads them in the INSTITUTION_MASTER. 2) This data map finds the new institutions from the NTCPTRY_PRFL data and loads them in the ACCT_ID_INSTN_ID_MAP table using OFSAAI generated INTSN_SEQ_ID from INSTITUTION_MASTER.
10190	AnticipatoryProfile_AccountToClientBank	This datamap marks all institutions with an OFSAAI generated INTSN_SEQ_ID in the Anticipatory Profile tables. It should be executed after FinancialInstitution_AnticipatoryProfile and AccountToClientBank_AnticipatoryProfile datamaps are executed.
50020	DailyAggregateStage	This datamap 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.
50030	OffsettingAccountPairStage	This datamap is used to populate OFFSETING_ACCT_PAIRS table by self-joining the table DAILY_AGG_STAGE to generate offsetting trade account pairs.The accounts have the lower ACCT_INTRL_ID while the offsetting accounts have the higher ACCT_INTRL_ID.
50040	TradeDailyTotalCountStage	This datamap aggregates the total trades done by that account for the current processing day.
10200	CustomerAccountStage_FrontOfficeTransactionParty	This datamap populates the Customer Account Stage table with the Cust-Acct pairs which appears in FOTPS with Party type as IA.
10210	FrontOfficeTransaction_UnrelatedPartyUpd	This datamap updates the FOT table for records where UNRLTD_PARTY_FL is 'Y' with a value as 'N', by determining the pairs of parties (internal) in the role of Orig & Benef having either common Tax ID/Common Customer/Common HH.
10220	FinancialInstitution_SettlementInstruction	This datamap inserts new records in Financial Institution records for the institutions found in INSTRUCTION that have not been previously identified, INSTN_SEQ_ID are OFSAAI generated. This datamap should be executed before AccountToClientBank_SettlementInstruction datamap.
10230	AccountToClientBank_SettlementInstruction	This datamap marks all institutions with an OFSAAI generated INTSN_SEQ_ID in FOTPS. 1) Prior to this datamap execution the predecessor datamaps finds the new INSTRUCTION from the transaction data and loads them in the INSTITUTION_MASTER. 2) This data map finds the new institutions from the INSTRUCTION data and loads them in the ACCT_ID_INSTN_ID_MAP table using OFSAAI generated INTSN_SEQ_ID from INSTITUTION_MASTER.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10240	SettlementInstruction_AccountToClientBank	This datamap updates Destination Institution and Physical Delivery Institution in INSTRUCTION table using the values from ACCT_ID_INSTN_ID_MAP table.
40010	FinancialInstitution_InsuranceTransaction	This datamap inserts new records in Financial Institution table for the institutions found in Insurance Transactions, INSTN_SEQ_ID are OFSAAI generated. This datamap should be executed before AccountToClientBank_InsuranceTransaction datamap as generated INSTN_SEQ_ID will be used to populate Anticipatory Profile table.
40020	AccountToClientBank_InsuranceTransaction	This datamap marks all institutions with an OFSAAI generated INTSN_SEQ_ID in FOTPS. 1) Prior to this datamap execution the predecessor datamaps finds the new institutions from the transaction data and loads them in the INSTITUTION_MASTER. 2) This data map finds the new institutions from the INSURANCE_TRXN data and loads them in the ACCT_ID_INSTN_ID_MAP table using OFSAAI generated INTSN_SEQ_ID from INSTITUTION_MASTER.
40030	InsuranceTransaction_AccountToClientBank	This datamap marks all institutions with an OFSAAI generated Institution Identifier in Insurance Transaction records. 1) Prior to this datamap execution Financial Institution and Account To Client Bank records are inserted. 2) Henceforth this datamap uses the Account To Client Bank table and updates Institution Identifier in Insurance table.
10245	WLMProcessingLock	This datamap applies lock to restrict UI accessibility for Watch list Management.
10250	WatchListEntry_WatchListEntryCurrDayInsert	This datamap checks for records in watch list from source files for the current day, if there is no records, create the current day watch list records from the previous day.
10260	WatchListAudit_StatusUpd	This datamap take care of watchlist table for the modifications of the WL based on the new user interface WL utility.
10270	WatchList_WatchListSourceAuditInsert	This datamap takes into account the modifications of the watchlist based on the new user interface WL utility. 1) Get all the records that are active from audit table. Order by created time. 2) Take the latest change for each LIST_SRC_CD Watch List and insert records in WATCH_LIST_SOURCE table.
10280	WatchList_WatchListSourceAuditUpd	This datamap takes into account the modifications of the watchlist based on the new user interface WL utility. 1) Get all the records that are active from audit table. Order by created time. 2) Take the latest change for each LIST_SRC_CD Watch List and update records in WATCH_LIST_SOURCE table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10290	WatchList_WatchListSourceUpd	This datamap takes into account the modifications of the watchlist based on the new user interface WL utility. 1) Get all the records that are active from audit table. Order by created time. 2) Take the latest change for each LIST_SRC_CD Watch List and update records in WATCH_LIST_SOURCE table.
10300	WatchListEntry_WatchListAuditUpd	This datamap takes care of watch list entry table for the modifications of the WL based on the new user interface WL utility.
10310	WatchListEntryAudit_WatchListEntryUpdate	This datamap take care of watchlist entry audit table for the modifications of the WL based on the new user interface WL utility.
10320	Customer_KYCRiskUpd	This datamap calculates risk, If the risk was List driven, then this can ignore that record. If it was BUS/GEO driven and there is KYC risk. Apply KYC Risk in Customer table.
60090	CorrespondentBankToPeerGroup	This datamap populates the CLIENT_BANK_PEER_GRP table by associating peer group identifiers in the ACCT_PEER_GRP table with institution identifiers in the ACCT_ID_INSTN_ID_MAP table.
10330	DerivedAddress_SettlementInstructionInsert	This datamap inserts new addresses in the Derived Address table. It derives the addresses from the INSTRUCTION table.
10340	DerivedAddress_SettlementInstructionUpd	This datamap derives the addresses from the INSTRUCTION table. It updates addresses in the Derived Address table, if already existing.
10350	SettlementInstruction_PhysicalDlvryAddrUpd	This datamap updates Mantas Physical Delivery Address Identifier in INSTRUCTION table.
10360	DerivedAddress_FrontOfficeTransactioPartyStageInsert	This datamap selects the distinct set of addresses from today's front-office transactions and if non-existent, inserts new address records into Derived Address.
10370	DerivedAddress_FrontOfficeTransactioPartyStageUpd	This datamap selects the distinct set of addresses from today's front-office transactions and if existent, updates new address records into Derived Address.
10380	FrontOfficeTransactionParty_DerivedAddress	This datamap maintains the addresses in the DerivedAddress table. It derives the addresses from the FrontOfficeTransactionParty table.
40040	DerivedAddress_InsuranceTransactionInsert	This datamap derives the addresses from the INSURANCE table, and inserts the addresses in to the Derived Address table.
40050	DerivedAddress_InsuranceTransactionUpd	This datamap derives the addresses from the INSURANCE table. If the address already exists in Derived Address table, it will update the addresses in to the Derived Address table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
40060	InsuranceTransaction_InstitutionAddrUpd	This datamap updates Mantas Institution Address Identifier in the Insurance Transaction table. 1) A new record is created in Derived Address table prior to this datamap execution. 2) Update the same Derived Address Sequence ID in INSURANCE_TRXN for CP_ADDR_MSTR_SEQ_ID column.
40070	DerivedEntity_InsuranceTransactionInsert	This datamap maintains the External Entity table. It derives the entities from the INSURANCE table on current processing date.
40080	DerivedEntity_InsuranceTransactionUpd	This datamap maintains the External Entity table. It derives the entities from the INSURANCE table on current processing date.
10390	DerivedEntity_FrontOfficeTransactionPartyInsert	This datamap maintains the External Entity table. It derives the entities from the Front Office and Front Office Party transaction table.
10400	DerivedEntity_FrontOfficeTransactionPartyUpd	This datamap maintains the External Entity table. It derives the entities from the Front Office and Front Office Party transaction table.
10410	DerivedEntity_SettlementInstructionInsert	This datamap maintains the External Entity table. It derives the entities from the Instruction table on current processing date.
10420	DerivedEntity_SettlementInstructionUpd	This datamap maintains the External Entity table. It derives the entities from the INSTRUCTION table. 1) Select the distinct set of names, accounts, institutions from today's Instructions and updates matching records in the External Entity table.
10430	CorrespondentBank_FrontOfficeTransactionPartyStageInsert	This datamap populates the client bank table for current day transactions where there is an institution involved.
10440	CorrespondentBank_FrontOfficeTransactionPartyStageUpd	This datamap maintains the Correspondent Bank table. It derives the records from the FOTPS table. If there is an existing correspond bank record available, this datamap updates the LAST_ACTVY_DT for that record.
10450	WatchListStagingTable_WatchList	This datamap 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.
10460	WatchListStagingTable_WatchListInstnIDUpd	This datamap only processes watch list entries that are External Accounts, Financial Institutions, and Internal Accounts. 1) It updates the Watch List Stage table with the corresponding Institution Sequence ID of the institution or account.
10470	PreviousWatchList_WatchList	This datamap save off current day's watch list records into PREV_WATCH_LIST
10480	DerivedAddress_WatchListNewCountries	This datamap inserts new countries from WL in the derived addresses table.
10485	WLMProcessingUnlock	This datamap releases the lock for Watch list Management.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10490	LinkStaging_FrontOfficeTransactionParty	This datamap loads the Link Stage with any entity associations from FOTPS, depending on the combination of Link Type Code defined.
40090	LinkStaging_InsTrxnDerivedEntDerivedAdd	This datamap loads the Link Stage with any entity associations from INSURANCE.
10500	LinkStaging_InstructionDerivedEntDerived Add	This datamap loads the Link Stage with any entity associations from instruction. Define the entity association based on existence of entity and address associations in data.
10510	NameMatchStaging	This datamap use fuzzy match to match Candidate Name against the List Name and inserts records in Name Match Stage table.
10520	WatchListStagingTable_NameMatchStageI nsert	This datamap is a wrapper for the fuzzy matching mappings and scripts. 1) For each processing day, this datamap joins fuzzy names to their matched watch list records to create additional watch list records for subsequent application to transactional tables.
10530	DerivedEntityLink_LinkStage	This datamap selects the external entity links from today's Link Stage table and insert records in External Entity Link table in associations to various link tables.
10540	DerivedEntitytoDerivedAddress_LinkStage	This datamap writes link-stage associations to various link tables in External Entity Address Table.
10550	DerivedEntitytoInternalAccount_LinkStage	This datamap writes link-stage associations to various link tables in External Entity Account Table.
10560	DerivedAddresstoInternalAccount_LinkSta ge	This datamap writes link-stage associations to various link tables in Derived Account Address Table.
10570	WatchListStagingTable2_WatchListStage2 AcctExistence	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with ACCT entity. 2) For IA (ACCT table) watch list entries, the error status is assigned if the entity does not exist in the entity table because these entity records are expected to exist.
10580	WatchListStagingTable2_WatchListStage2 CBExistence	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with CLIENT_BANK entity. 2) Evaluates the existence of the CLIENT_BANK entity and assigns a 'Warning"" status to the record if the entity does not exist in the entity table because these entity records are expected to exist.
10590	WatchListStagingTable2_WatchListStage2 CustExistence	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with CUST entity. 2) For CU (CUST table) watch list entries, the error status is assigned if the entity does not exist in the entity table because these entity records are expected to exist.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10600	WatchListStagingTable2_WatchListStage2DAExistence	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with DERIVED_ADDRESS entity. 2) Evaluates the existence of the DERIVED_ADDRESS record and assigns status to the record accordingly.
10610	WatchListStagingTable2_WatchListStage2EEExistence	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with EXTERNAL_ENTITY entity. 2) Evaluates the existence of the EXTERNAL_ENTITY record and assigns a 'Warning""" status to the record if the entity does not exist in the entity table because these entity records are expected to exist.
10620	WatchListStagingTable2_WatchListStage	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Check for watch list stage CUST_INTRL_ID flag if it is 'Y' means that this name is fuzzy matched. 2) Insert the watch list entry into the second processing table that is Watch list stage 2 table for both the fuzzy matched as well as exact name records.
10630	WatchListStagingTable2_AcctListMembershipUpd	The datamap checks for entry membership in the corresponding entity list membership table.
10640	WatchListStagingTable2_CBListMembershipUpd	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with CB_LIST_MEMBERSHIP entity. 2) Evaluates the existence of the CB_LIST_MEMBERSHIP record and assigns a 'Warning"" status to the record if the entity does not exist in the entity table because these entity records are expected to exist.
10650	WatchListStagingTable2_CustListMembershipUpd	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with CUST_LIST_MEMBERSHIP entity. 2) Evaluates the existence of the CUST_LIST_MEMBERSHIP record and assigns a 'Warning"" status to the record if the entity does not exist in the entity table because these entity records are expected to exist.
10660	WatchListStagingTable2_EEListMembershipUpd	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with EXTERNAL_NTITY_LIST_MEMBERSHIP entity. 2) Evaluates the existence of the EXTERNAL_NTITY_LIST_MEMBERSHIP record and assigns a 'Warning"" status to the record if the entity does not exist in the entity table because these entity records are expected to exist.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10670	WatchListStagingTable2_EEListMembershipStatusUpd	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) It validates the list membership status of External Entities whose Last Activity Date is earlier than the current date. 2) Update the status of the watch list entry based the existence or non-existence of a corresponding list membership record.
10680	WatchListStagingTable2_DAListMembershipUpd	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) Processes all watch list entries that have a possible match with DERIVED_ADDR_LIST_MEMBERSHIP entity. 2) Evaluates the existence of the DERIVED_ADDR_LIST_MEMBERSHIP record and assigns a 'Warning"" status to the record if the entity does not exist in the entity table because these entity records are expected to exist.
10690	WatchListStagingTable2_DAListMembershipStatusUpd	This datamap validates each watch list entry and inserts into the processing table WATCH_LIST_STAGE2. 1) It validates the list membership status of DERIVED_ADDRESS whose Last Activity Date is earlier than the current date. 2) Update the status of the watch list entry based the existence or non-existence of a corresponding list membership record.
10700	WatchListStagingTable2_WatchListStage2SeqIdUpd	This datamap updates the list risk of each valid watch list entity based on the entity Sequence ID. The datamap sets various flags and derives the highest List Risk value for each entity on the watch list.
10710	WatchListStagingTable2_WatchListStage2IntrlIdUpd	This datamap updates the list risk of each valid watch list entity based on the entity Internal ID. The datamap sets various flags and derives the highest List Risk value for each entity on the watch list.
10720	Customer_WatchListStage2ListRisk	This datamap calculates the customer's effective risk and set the risk factor if the risk is not found for the current day in watch list stage table. After calculating the risk updates the CUST table. Use nulls for the List Risk and the List Source Code.
10730	CorrespondentBank_WatchListStage2EffectiveRisk	This datamap calculates the Client Bank Effective Risk and applies the Effective Risk and the List Risk to the CLIENT_BANKrecord.
10740	Customer_WatchListStage2EffectiveRisk	This datamap calculates the Effective Risk of Customer and applies the Effective Risk and the List Risk to the CUST record.
10750	DerivedAddress_WatchListStage2EffectiveRisk	This datamap calculates the Effective Risk of all derived address entities and applies the Effective Risk and the List Risk to the DERIVED_ADDRESS record.
10760	DerivedEntity_WatchListStage2EffectiveRisk	This datamap calculates the Effective Risk of all external entities and applies the Effective Risk and the List Risk to the EXTERNAL_ENTITY record.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10770	WatchListStagingTable2_WatchListStage2SeqId	This datamap calculates the Effective Risk of all entities and applies the Effective Risk and the List Risk to the entity record where sequence ID is not null.
10780	AccountListMembership_WatchListStage2Insert	This datamap inserts List Membership records for entities into ACCT_LIST_MEMBERSHIP table that are new to a list.
10790	AccountListMembership_WatchListStage2Upd	This datamap updates the existing retired ACCT_LIST_MEMBERSHIP records by setting List Removal Date to the current processing date.
10800	CorrespondentBankListMembership_WatchListStage2Insert	This datamap inserts List Membership records for entities that are new to a list into CB_LIST_MEMBERSHIP table.
10810	CorrespondentBankListMembership_WatchListStage2Upd	This datamap updates the existing retired CB_LIST_MEMBERSHIP records by setting List Removal Date to the current processing date.
10820	CustomerListMembership_WatchListStage2Insert	This datamap inserts List Membership records for entities that are new to a list into CUST_LIST_MEMBERSHIP table.
10830	CustomerListMembership_WatchListStage2Upd	This datamap updates the existing retired CUST_LIST_MEMBERSHIP records by setting List Removal Date to the current processing date.
10840	DerivedAddressListMembership_WatchListStage2Insert	This datamap maintains the Derived Address List membership table based on the current WL processing results.
10850	DerivedAddressListMembership_WatchListStage2Upd	This datamap maintains the Derived Address List membership tables based on the current WL processing results by setting List Removal Date to the current processing date.
10860	DerivedEntityListMembership_WatchListStage2Insert	This datamap inserts List Membership records for entities that are new to a list into EXTERNAL_NTITY_LIST_MEMBERSHIP table.
10870	DerivedEntityListMembership_WatchListStage2Upd	This datamap maintains the External Entity membership tables based on the current WL processing results by setting List Removal Date to the current processing date.
10880	Account_OverallEffectiveRiskUpd	This datamap updates the risk on the ACCT based on KYC, Primary customer, as well as other external risks.
10881	Account_AccountCustRiskUpd	This data map updates the risk on the ACCT based on KYC, Primary customer, as well as other external risks.
10890	Account_EffRiskUpdAfterWLRiskRemoval	This datamap Updates the account Effective Risk to the maximum of the business risk, geographic risk, and customer risk. The account Effective Risk was already set to the higher of the customer-supplied business and geography risk. List risk is ignored here, as this mapping is where we're removing list risk.
10900	Account_WatchListStage2EffectiveRisk	This datamap calculates all risk related values like Effective Risk of Acct and applies the Effective Risk, List Risk to the ACCT record.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10910	WatchListStagingTable2_WatchListStage2IntrlId	This datamap calculates the Effective Risk of all entities and applies the Effective Risk and the List Risk to the entity record based on NTITY_INTRL_ID.
10920	BackOfficeTransaction_EffectiveAcctivityRiskUpd	This datamap updates the risk related values to all parties involved in Back Office Transaction 1) Select risk values from BACK_OFFICE_TRXN, ACCT, Offset Account in the sub query. 2) Derive the effective and activity risks from the transaction. 3) Update BACK_OFFICE_TRXN table using BO_TRXN_SEQ_ID in the main query.
10930	SettlementInstruction_EntityAcctivityRiskUpd	This datamap updates Entity Risk and Activity Risk in INSTRUCTION table
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	This datamap populates the Effective Risk and Activity Risk related values to all the parties in FO_TRXN_PARTY_RISK_STAGE table.
10955	AccountGroup_InvestmentObjectiveUpd	This datamap updates Investment Objective column in Account Group table.
40100	InsuranceTransaction_EntityAcctivityRiskUpd	This datamap updates the risk related values to all parties in Insurance Transaction. 1) Select different risk related values from various tables like watchlist, external entity and derived address etc. 2) Updates Entity Risk and Activity Risk in INSURANCE_TRXN table.
20010	CorrespondentBank_JurisdictionUpd	This datamap updates the JRSDCN_CD and BUS_DMN_LIST_TX for an existing client bank record where either the JRSDCN_CD or the BUS_DMN_LIST_TX is null.
20020	CorrespondentBank_AcctJurisdictionReUpd	This datamap updates the jurisdiction for CLIENT_BANK (Correspondent Bank).
20030	FinancialInstitution_InstNameUpd	This datamap updates INSTN_NM for an existing INSTN_MASTER record.
10960	AccountGroup_JurisdictionUpd	This datamap updates the primary account in a HH with the jurisdiction & business domain present in Account table for it.
10970	TransactionPartyCrossReference_BackOfficeTransaction	This datamap is used to build the record for Transaction Party Cross Reference table from today’s Back Office Transactions. 1) Select the set of information from today's Back Office Transactions and insert records in Transaction Party Cross Reference table. 2) Parameter ProcessTransactionXRefFlag = 'N' or 'Y' accordingly.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
10980	CashTransaction_FrontOfficeTransaction	This datamap is used to build the record for Cash Transaction Table from today's Front Office Transaction and Front Office Transaction Party. 1) Select the set of Cash Transaction categories information from today's Front Office Transaction and Front Office Transaction Party to Insert records In Cash Transaction Table. 2) Some fields are not null-able. The NVL function is used in the SQL to plug the default values in place of a null. Also, various "NB" fields are set to zero whenever they are null in the expression prior to the inserting them into the target table.
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	This datamap select the set of information from today's Front Office Transaction and Front Office Transaction Party to Insert records In Monetary Instrument Transaction Table.
11000	TransactionPartyCrossReference_FrontOfficeTransaction	This datamap is used to build the record for Transaction Party Cross Reference table from today's Front Office Transaction and Front Office Transaction Party. 1) Select the set of information from today's Front Office Transaction and Front Office Transaction Party to Insert records In Transaction Party Cross Reference Table. 2) Some fields are not null-able. The NVL function is used in the SQL to plug the default values in place of a null. Also, various "NB" fields are set to zero whenever they are null in the expression prior to the inserting them into the target table. 3) Parameter ProcessTransactionXRefFlag = 'N' or 'Y' accordingly.
11010	WireTransaction_FrontOfficeTransaction	This datamap is used to build the record for Wire Transaction Table from today's Front Office Transaction and Front Office Transaction Party. 1) Select the set of Wire Transaction categories information from today's Front Office Transaction and Front Office Transaction Party to Insert records In Wire Transaction Table. 2) Some fields are not null-able. The NVL function is used in the SQL to plug the default values in place of a null. Also, various "NB" fields are set to zero whenever they are null in the expression prior to the inserting them into the target table. 3) Parameter ProcessBankToBank = 'N' or 'Y' accordingly.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
11020	WireTransactionInstitutionLeg_FrontOffice Transaction	This datamap is used to build the record for Wire Transaction Institution Leg Table from today's Front Office Transaction and Front Office Transaction Party. 1) Select the set of Wire Transaction categories and it should have more than 1 leg information from today's Front Office Transaction and Front Office Transaction Party to Insert records In Wire Transaction Institution Leg Table. 2) Some fields are not null-able. The NVL function is used in the SQL to plug the default values in place of a null. Also, various "NB" fields are set to zero whenever they are null in the expression prior to the inserting them into the target table. 3) Parameter ProcessBankToBank = 'N' or 'Y' accordingly.
11030	CashTransaction_FrontOfficeTransactionRevAdj	This datamap adjusts the reversals for Cash Transaction table. 1) Select the set of information from today's Front Office Transaction to update records with columns CXL_PAIR_TRXN_INTRL_ID, REBKD_TRXN_INTRL_ID in Cash Transaction table.
11040	MonetaryInstrumentTransaction_FrontOfficeTransactionRevAdj	This datamap adjusts the reversals for front office transaction tables in Monetary Instrument Transaction table
11050	WireTransaction_FrontOfficeTransactionRevAdj	This datamap adjusts the reversals for Wire Transaction table. 1) Select the set of information from today's Front Office Transaction to update records with columns CXL_PAIR_TRXN_INTRL_ID, REBKD_TRXN_INTRL_ID in Wire Transaction table.
11060	TrustedPair_StatusEXPUpd	This datamap selects Trusted Pair Records From Kdd_Trusted_Pair Table Which Are To Be Expired, set the Status Code to 'EXP' in Kdd_Trusted_Pair table.
11070	TrustedPairMember_AcctExtEntEffecRiskUpd	This datamap selects The Trusted Pair Records From Kdd_Trusted_Pair Table Which Are Active, and get the trusted Pair parties from kdd_trusted_pair_mbr table with their effective risk and new effective risks from the base tables (i.e. ACCT and EXTERNAL_ENTITY tables) and updates kdd_trusted_pair_mbr table for columns ACCT_EFCTV_RISK_NB, EXTRL_NTITY_EFCTV_RISK_NB for parties whose risk got changed.
11080	TrustedPair_StatusRRCInsert	This datamap 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).

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
11090	TrustedPair_StatusRRCUpd	This datamap gets the trusted Pair parties from kdd_trusted_pair_mbr table with their effective risk and new effective risks from the base tables (i.e. ACCT and EXTERNAL_ENTITY tables).Update kdd_trusted_pair table with two columns REVIEW_DT, REVIEW_REASON_TX for existing RRC record.
11100	ApprovalActionsAudit_TrustedPair	This datamap inserts auditing records in KDD_APRVL_ACTVY_AUDIT table. 1) Inserts the EXP record of kdd_trusted_pair table in the KDD_APRVL_ACTVY_AUDIT table 2) Inserts RRC record either which is inserted or updated in KDD_TRUSTED_PAIR with sysdate as review date
11110	TrustedPairMember_StatusRRCInsert	This datamap 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).
11120	BackOfficeTransaction_TrustedFlagsUpd	This datamap flags the Back Office Transactions as Trusted or Not Trusted based on entry in the kdd_trusted_pair and kdd_trusted_pair_mbr tables. It only looks at today's transactions. 1) Select the set of information from today's Back Office Transactions, Trusted Pair and Trusted Pair Member Details to update records with columns TRSTD_TRXN_FL, ACCT_OFFSET_ACCT_TRSTD_FL in Back Office Transactions table.
11130	InsuranceTransaction_TrustedFlagsUpd	This datamap flags today's Insurance Transaction as Trusted or Not Trusted based on entry in the kdd_trusted_pair and kdd_trusted_pair_mbr tables. It only looks at today's transactions. 1) Select the set of information from today's Insurance Transaction and Trusted Pair Member Details to update records with columns TRSTD_TRXN_FL, NSRN_PLCY_ID_CNTRPTY_ID_FL in Insurance Transaction table.
11140	MonetaryInstrumentTransaction_TrustedFlagsUpd	This datamap flags the Monetary Instruction 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.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
11150	WireTransaction_TrustedFlagsUpd	This datamap flags the Wire Transactions as Trusted or Not Trusted based on entry in the kdd_trusted_pair and kdd_trusted_pair_mbr tables. It only looks at today's transactions. 1) Select the set of information from today's Wire Transactions, Trusted Pair and Trusted Pair Member Details to update records with columns TRSTD_TRXN_FL, ORIG_BENEF_TRSTD_FL, ORIG_SCND_BENEF_TRSTD_FL, SCND_ORIG_BENEF_TRSTD_FL,SCND_ORIG_SCND_BENEF_TRSTD_FL in Wire Transaction table.
50050	CustomerDailyProfile_BOT	This datamap aggregates Back Office Transaction data by Customer and Date and updates into CUST_SMRY_DAILY table.
50060	CustomerDailyProfile_FOTPS	This datamap aggregates Front Office Transaction data by Customer and Date and updates into CUST_SMRY_DAILY table.
50070	InstitutionalAccountDailyProfile_DEAL	This datamap updates INSTL_ACCT_SMRY_DAILY table from Deal, grouping by account and data dump date.
50080	CustomerDailyProfile_DEAL	This datamap updates CUST_SMRY_DAILY table from Structured Deal, grouping by customer and data dump date.
50090	InstitutionalAccountDailyProfile_INST	This datamap updates INSTL_ACCT_SMRY_DAILY table from Instruction, grouping by account and data dump date.
50100	CustomerDailyProfile_INST	This datamap updates CUST_SMRY_DAILY table from Instruction data, grouping by Customer and data dump date.
50110	InstitutionalAccountDailyProfile_CorpAction	This datamap aggregates institutional trading activity, grouping by Account ID and data dump date.
50120	CustomerDailyProfile_CorpAction	This datamap aggregates Corporate Action trading activity, grouping by Customer ID.
50130	InstitutionalAccountDailyProfile_Trade	This datamap updates INSTL_ACCT_SMRY_DAILY table from Trade, grouping by account and data dump date.
50140	CustomerDailyProfile_Trade	This datamap updates CUST_SMRY_DAILY table from Trade data, grouping by customer and data dump date.
60100	ManagedAccountDailyProfile_SameDayTrade	This datamap is used for the daily aggregation of the block allocation day trades data. This populates the managed account daily summary.
60110	ManagedAccountDailyProfile_Trade	This datamap is used for the daily aggregation of the block allocation trades data. This populates the managed account daily summary .
60120	ManagedAccountDailyProfile_BOT	This datamap populates MANGD_ACCT_SMRY_DAILY table using Back Office Transaction.
11160	AccountDailyProfile-Trade	This datamap performs daily aggregation of trades from trade table , Profit Loss from Account Realized Profit Loss table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
11170	AccountDailyProfile-Transaction	This datamap populates the table ACCT_TRXN_SMRY_DAILY using both Front office and Back Office transaction for that account on current processing date.
11180	AccountProfile_Trade	This datamap populates the table ACCT_SMRY_MNTH using ACCT_TRADE_SMRY_DAILY table for that account starting from Month Start date till current processing date.
11190	AccountProfile_Transaction	This datamap populates the table ACCT_SMRY_MNTH using ACCT_TRXN_SMRY_DAILY table for that account starting from Month Start date till current processing date.
11200	AccountProfile_Stage	This datamap populates the table ACCT_SMRY_MNTH using ACCT_PRFL_STAGE table for that account starting from Month Start date till current processing date.
11210	AccountProfile_Position	This datamap populates the table ACCT_SMRY_MNTH using ACCT_POSN table for that account starting from Month Start date till current processing date. Updates values by calculating aggregate values for AGGR_SHRT_PUT_EXPSR_AM, AGGR_SHRT_CALL_EXPSR_AM, SHRT_PUT_EXPSR_RATIO and SHRT_CALL_EXPSR_RATIO for each account internal ID present in ACCT_SMRY_MNTH.
11220	AccountProfile_Balance	This datamap populates the ACCT_SMRY_MNTH table using ACCT_BAL_POSN_SMRY. If there is already record in Account summary Month for Account and Month Start Date, then it will update the record. Else it will do insert, remaining columns defaulted to 0.
60130	HouseholdProfile	This datamap aggregates monthly account summaries into their respective households. All monthly records must be processed each day since account households are subject to change daily.
50150	InstitutionalAccountProfile	This datamap performs Insert or Update of Institutional Account Summary Month Table from its corresponding Daily table. Aggregate daily activity with counts and amounts for the current month. If already record exists for the account in the current month, the datamap will update the record, else insert a new record.
50160	CustomerProfile	This Datamap loads into CUST_SMRY_MNTH from CUST_SMRY_DAILY table. Check for the customer record exists for t he month, if record not available Insert records in CUST_SMRY_MNTH table
60140	ManagedAccountProfile	This datamap updates the Managed Account Summary Month Table from its corresponding Managed Account Daily Summary table.
60145	AccountPosition_PercentofPortfolioUpd	This datamap updates Percent of Portfolio column in Account Position table.

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
20040	CorrespondentBankProfile	This datamap performs daily re-aggregation of the Correspondent Bank Summary Month table out of the account summary month table.
20050	AccountATMDailyProfile	This datamap calculates the total Transaction Amount for Account ATM Daily Profile Select information from Front Office Transaction, Account and Account ATM Daily Profile and insert or update (if record exist) into ACCT_ATM_SMRY_DAILY
11230	ChangeLog_AcctProfileInactivity	This datamap 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).
11240	AccountPeerGroupMonthlyTransactionProfile	This datamap calculates average values and insert into Account Peer Group Monthly Transaction Profile. Select and calculate average values for withdrawal amount and count from ACCT_SMRY_MNTH table Insert the above values into ACCT_PEER_TRXN_SMRY_MNTH.
20060	CorrespondentBankPeerGroupTransactionProfile	This datamaps populate CorrespondentBankPeerGroupTransactionProfile from Client Bank Summary Month. 1) Select set of information from CLIENT_BANK_SMRY_MNTH, CLIENT_BANK_PEER_GRP 2) Data is populated in the target table after aggregating the required columns.
20070	AccountChannelWeeklyProfile	This datamap populates the table ACCT_CHANL_SMRY_WKLY using FO_TRXN, BACK_OFFICE_TRXN table for that account starting from Weekly Start date till current processing date.
40110	InsurancePolicyDailyProfile_InsTrxnInsPolicyBal	This datamap performs inserts or updates of Insurance Policy Summary Daily Table from the Insurance Transaction table on the current processing day.
40120	InsurancePolicyProfile_InsurancePolicyDailyProfile	This datamap performs updates of Insurance Policy Summary Month Table using the values from Insurance Policy Daily Profile table. 1) Records are inserted into Insurance Policy Daily Profile table prior to this datamap execution. 2) This datamap inserts new records or Updates matched records in Insurance Policy Profile table using the values from Insurance Policy Daily Profile table.
50170	CustomerBalance_ActiveOTCTradeCtUpd	This datamap counts the records in the Deal table which has an end date greater than or equal to the current date by customer and update the ACTV_OTC_TRD_CT column in customer balance table.
60150	AccountPositionDerived	This datamap processes account option position pair data and updates the corresponding account position records. Updates are made to attributes relating to uncovered option contracts

Table 148. BD Datamaps (Continued)

Datamap Number	Datamap Name	Description
60160	AccountBalance_AcctPosnPair	This datamap processes account option position pair data and updates the corresponding account balance records. Updates are made to option market value long attributes.
60170	AccountBalance_Acctposn	This datamap aggregates current-day security positions by product category and account for update of the account balance record. Rejoins for single update to avoid deadlocks.
60180	HouseholdBalance	This datamap aggregates daily records of account balances data and inserts into household balances table based household group id.
11300	AccountChangeLogSummary	This datamap inserts new records to the ACCT_CHG_LOG_SMRY table. The datamap should be executed once the change log processing is done.
11310	AccountToCustomerChangeLogSummary	This datamap inserts new records to the CUST_ACCT_CHG_LOG_SMRY. The datamap should be executed once the change log processing is done.
11320	CustomerChangeLogSummary	This datamap inserts new records to the CUST_CHG_LOG_SMRY table. The datamap should be executed once the change log processing is done.

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

Firm Data Transfer Datamaps

The following table lists the Firm Data Transfer (FDT) Datamaps and the order they must be run in. .

Table 149: FDT Datamaps

Datamap Number	Datamap Name	Predecessors
70010	Scrty_TradeExecutionStageInsert	NA
70020	Scrty_OrderStageInsert	NA
70030	MktCntr_OrderStageInsert	NA
70040	OrderStage_DQupdate	NA
70050	TradeExecutionEventStage_DQupdate	NA
70060	OrderStage_FDTupdate	70040
70070	OrderStage_RmngQtupdate	70040 70060
70080	OrderSummary	70040 70060 70070

Table 149: FDT Datamaps

Datamap Number	Datamap Name	Predecessors
70090	OrderSummary_OpenOrdrInsrt	70040 70060 70070 70080
70100	OrderSummary_QtyUpdate	70040 70060 70070 70080 70090
70110	OrderStage_OpenOderUpd	70040 70060 70070 70080 70090 70100
70120	OrderSummary_Update	70040 70060 70070 70080 70090 70100 70110
70130	OrderStage_OrdrSeqUpd	70040 70060 70070 70080 70090 70100 70110 70120
70140	OrderEvent_OrderStage	70040 70060 70070 70080 70090 70100 70110 70120 70130

Table 149: FDT Datamaps

Datamap Number	Datamap Name	Predecessors
70150	Execution_NewEvents	70010 70050
70160	Execution_CancelAndReplace	70010 70050
70170	Execution_CancelEvents	70010 70050 70150 70160
70180	Execution_CorrectionEvents	70010 70050 70150 70160 70170
70190	Trade_NewEvents	70010 70050
70200	Trade_CancelAndReplace	70010 70050
70210	Trade_CorrectionEvents	70010 70050 70190 70200
70220	Trade_CancelEvents	70010 70050 70190 70200 70210
70230	Trade_DerivedTrade	70010 70050 70190 70200 70210 70220
70240	Trade_OrigSeqIDUpd	70010 70050 70190 70200 70210 70220 70230

Table 149: FDT Datamaps

Datamap Number	Datamap Name	Predecessors
70250	Trade_ParentSeqIDUpd	70010 70050 70190 70200 70210 70220 70230 70240
70260	Trade_RplcngSeqIDUpd	70010 70050 70190 70200 70210 70220 70230 70240 70250
70270	TradeExecutionEvent_Trade	70010 70050
70280	TradeExecutionEvent_Execution	70010 70050
70290	TradeExecutionEvent_CancelReplaceTrade	70010 70050
70300	TradeExecutionEvent_FirmRefTrade	70010 70050 70270 70280 70290
70310	TradeExecutionEvent_MktRefTrade	70010 70050 70270 70280 70290

Table 149: FDT Datamaps

Datamap Number	Datamap Name	Predecessors
70320	Trade_RefData	70010 70050 70270 70280 70290
70330	Execution_Update	70010 70050 70150 70160 70170 70180

The following table provides a list of datamaps and description for each datamap. These datamaps are listed in order

Table 150: FDT Datamap Description

Datamap Number	Datamap Name	Description
70010	Scrty_TradeExecutionStageInsert	This datamap populates the SCRTY table using ingested trade records present at TRADE_EXECUTION_EVENT_STAGE for that scurity, if security is not present already in the SCRTY table
70020	Scrty_OrderStageInsert	This datamap populates the SCRTY table using ingested order records present at ORDR_STAGE for that security, if the security is not present already in the SCRTY table.
70030	MktCntr_OrderStageInsert	This datamap populates the MARKET_CENTER table using ingested order records present at ORDR_STAGE for that market centre, if themarket centre is not present already in the MARKET_CENTER table.
70040	OrderStage_DQupdate	This datamap updates the ORDR_STAGE table to mark invalid records.
70050	TradeExecutionEventStage_DQupdate	This datamap updates the TRDE_EXECUTION_EVENT_STAGE table to mark invalid trade events.
70060	OrderStage_FDTupdate	This datamap calculates and update information for each order event indentifying corresponding trade and quote information.
70070	OrderStage_RmngQtupdate	This datamap calculates and updates remaining units for each order event.
70080	OrderSummary	This datamap aggregates order events properties to identify the property for order, and populates the ORDR table.
70090	OrderSummary_OpenOrdrInsrt	This datamap populates the ORDR table based on the records in the OPEN_ORDR_STAGE table if required.

Table 150: FDT Datamap Description

Datamap Number	Datamap Name	Description
70100	OrderSummary_QtyUpdate	The datamap calculates the various quantity units and updates the ORDR table using those values.
70110	OrderStage_OpenOderUpd	This datamap populates the ORDR_STAGE table with order events not provided by customer but evident from the information provided by customer.
70120	OrderSummary_Update	This datamap updates the ORDR table for various events and trades occurred for order.
70130	OrderStage_OrdrSeqUpd	This datamap updates the ORDR_STAGE table using the corresponding order_seq_id from the ORDR table
70140	OrderEvent_OrderStage	This datamap populates the ORDR_EVENT table with records processed and calculated at the ORDR_STAGE table.
70150	Execution_NewEvents	This datamap populates the EXECUTION table identifying NEW events in the TRADE_EXECUTION_EVENT_STAGE table.
70160	Execution_CancelAndReplace	This datamap populates the EXECUTION table identifying CANCEL AND REPLACE events in the TRADE_EXECUTION_EVENT_STAGE table.
70170	Execution_CancelEvents	This datamap updates the EXECUTION table identifying CANCEL events in the TRADE_EXECUTION_EVENT_STAGE table.
70180	Execution_CorrectionEvents	This datamap updates the EXECUTION table identifying CORRECTION events in the TRADE_EXECUTION_EVENT_STAGE table.
70190	Trade_NewEvents	This datamap populates the TRADE table identifying NEW events in the TRADE_EXECUTION_EVENT_STAGE table.
70200	Trade_CancelAndReplace	This datamap populates the TRADE table identifying CANCEL AND REPLACE events in the TRADE_EXECUTION_EVENT_STAGE table.
70210	Trade_CorrectionEvents	This datamap updates the TRADE table identifying CORRECTION events in the TRADE_EXECUTION_EVENT_STAGE table.
70220	Trade_CancelEvents	This datamap updates the TRADE table identifying CANCEL events in the TRADE_EXECUTION_EVENT_STAGE table.
70230	Trade_DerivedTrade	This datamap populates TRADE tables identifying DERIVED TRADES in the TRADE_EXECUTION_EVENT_STAGE table.
70240	Trade_OrigSeqIDUpd	This datamap updates the original sequence identifier for non replaced trades.
70250	Trade_ParentSeqIDUpd	This datamap updates the parent sequence identifiers for the TRADE table.
70260	Trade_RplcngSeqIDUpd	This datamap updates the replacing sequence identifiers for the TRADE table.

Table 150: FDT Datamap Description

Datamap Number	Datamap Name	Description
70270	TradeExecutionEvent_Trade	This datamap populates the TRADE_EXECUTION_EVENT table with non order based trade records from the TRADE_EXECUTION_EVENT_STAGE.
70280	TradeExecutionEvent_Execution	This datamap populates the TRADE_EXECUTION_EVENT table with executed order records from the TRADE_EXECUTION_EVENT_STAGE table.
70290	TradeExecutionEvent_CancelReplaceTrade	This datamap populates the TRADE_EXECUTION_EVENT table with CANCEL AND REPLACE event executed order records from the TRADE_EXECUTION_EVENT_STAGE table.
70300	TradeExecutionEvent_FirmRefTrade	This datamap updates firm reference information in the TRADE_EXECUTION_EVENT table using the EXECUTION and TRADE tables.
70310	TradeExecutionEvent_MktRefTrade	This datamap updates market reference information in the TRADE_EXECUTION_EVENT table using the REPORTED SALE and TRADE tables.
70320	Trade_RefData	This datamap updates market and firm reference data in the TRADE table using the TRADE_EXECUTION_EVENT table.
70330	Execution_Update	This datamap updates the EXECUTION table in using various events which occur for the trade in the TRADE_EXECUTION_EVENT_STAGE table.

APPENDIX G Datamaps Matrix

This appendix provides a single window view of datamaps required for each solution set.

'X' denotes mandatory datamaps for each solution set.

'NA' denotes not applicable datamaps for the same solution set.

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
10010	EmployeeControlledAccount	X	X	X	X	X
60010	PortfolioManagerPosition	NA	NA	NA	NA	X
60020	AccountGroupProductAllocation	NA	NA	NA	NA	X
60030	AccountProductAllocation	NA	NA	NA	NA	X
60040	UncoveredOptionExposureDaily	NA	NA	NA	NA	X
60050	InvestmentAdvisorProfile	NA	NA	NA	NA	X
60060	RegisteredRepresentativeProfile	NA	NA	NA	NA	X
60070	RegOToBorrower	NA	NA	NA	NA	X
60080	InterestedPartyToEmployee	NA	NA	NA	NA	X
50010	Customer_TotAcctUpd	NA	NA	NA	X	NA
10015	FrontOfficeTransactionParty_SecondaryNames	X	X	NA	X	NA
10020	FinancialInstitution_ThomsonDataInstitutionInsert	X	X	X	X	X
10030	AccountToClientBank_ThomsonDataInstitutionInsert	X	X	X	X	X
10040	FinancialInstitution_AIIMSPopulation	X	X	X	X	X
10050	AccountToClientBank_AIIMSInstitutionInsert	X	X	X	X	X
10060	AccountToClientBank_InstitutionInsert	X	X	X	X	X
10070	AccountToClientBank_InstitutionUpd	X	X	X	X	X
10080	FinancialInstitution_FOTPSPopulation	X	X	X	X	X
10090	AccountToClientBank_FOTPSInstitutionInsert	X	X	X	X	X
10100	AccountManagementStage	X	X	X	X	X
10110	LoanProfile_LoanProfileStage	X	NA	NA	X	NA
10112	ServiceTeam_SprvsncdUpd	NA	NA	NA	NA	NA
10113	InvestmentAdvisor_MangdAcctUpd	NA	NA	NA	NA	NA
10114	Security_CIRRatingUpd	X	X	X	X	X
10116	BackOfficeTransaction_CollateralUpd	X	X	X	X	X

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
10120	BackOfficeTransaction_OriginalTransactionReversalUpd	X	X	NA	X	X
10130	BackOfficeTransaction_CancelledTransactionReversalCreditUpd	X	X	NA	X	X
10140	BackOfficeTransaction_CancelledTransactionReversalDebitUpd	X	X	NA	X	X
10150	FrontOfficeTransactionParty_InstnSeqID	X	X	X	X	X
10160	FrontOfficeTransactionParty_HoldingInstnSeqID	X	X	X	X	X
10170	FinancialInstitution_AnticipatoryProfile	NA	X	X	X	NA
10180	AccountToClientBank_AnticipatoryProfile	NA	X	X	X	NA
10190	AnticipatoryProfile_AccountToClientBank	NA	X	X	X	NA
50020	DailyAggregateStage	NA	NA	NA	X	NA
50030	OffsettingAccountPairStage	NA	NA	NA	X	NA
50040	TradeDailyTotalCountStage	NA	NA	NA	X	NA
10200	CustomerAccountStage_FrontOfficeTransactionParty	X	X	NA	X	X
10210	FrontOfficeTransaction_UnrelatedPartyUpd	X	X	NA	X	X
10220	FinancialInstitution_SettlementInstruction	NA	X	X	X	X
10230	AccountToClientBank_SettlementInstruction	NA	X	X	X	X
10240	SettlementInstruction_AccountToClientBank	NA	X	X	X	X
40010	FinancialInstitution_InsuranceTransaction	NA	NA	X	NA	NA
40020	AccountToClientBank_InsuranceTransaction	NA	NA	X	NA	NA
40030	InsuranceTransaction_AccountToClientBank	NA	NA	X	NA	NA
10245	WLMProcessingLock	X	X	X	X	NA
10250	WatchListEntry_WatchListEntryCurrDayInsert	X	X	X	X	X
10260	WatchListAudit_StatusUpd	X	X	X	X	X
10270	WatchList_WatchListSourceAuditInsert	X	X	X	X	X
10280	WatchList_WatchListSourceAuditUpd	X	X	X	X	X
10290	WatchList_WatchListSourceUpd	X	X	X	X	X
10300	WatchListEntry_WatchListAuditUpd	X	X	X	X	X

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
10310	WatchListEntryAudit_WatchListEntryUpdate	X	X	X	X	X
10320	Customer_KYCRiskUpd	X	X	X	X	X
60090	CorrespondentBankToPeerGroup	NA	NA	NA	NA	X
10330	DerivedAddress_SettlementInstructionInsert	NA	X	NA	X	NA
10340	DerivedAddress_SettlementInstructionUpd	NA	X	NA	X	NA
10350	SettlementInstruction_PhysicalDlvryAddrUpd	NA	X	NA	X	NA
10360	DerivedAddress_FrontOfficeTransactioPartyStageInsert	X	X	X	X	NA
10370	DerivedAddress_FrontOfficeTransactioPartyStageUpd	X	X	X	X	NA
10380	FrontOfficeTransactionParty_DerivedAddress	X	X	X	X	NA
40040	DerivedAddress_InsuranceTransactionInsert	NA	NA	X	NA	NA
40050	DerivedAddress_InsuranceTransactionUpd	NA	NA	X	NA	NA
40060	InsuranceTransaction_InstitutionAddrUpd	NA	NA	X	NA	NA
40070	DerivedEntity_InsuranceTransactionInsert	NA	NA	X	NA	NA
40080	DerivedEntity_InsuranceTransactionUpd	NA	NA	X	NA	NA
10390	DerivedEntity_FrontOfficeTransactionPartyInsert	X	X	X	X	X
10400	DerivedEntity_FrontOfficeTransactionPartyUpd	X	X	X	X	X
10410	DerivedEntity_SettlementInstructionInsert	X	X	X	X	X
10420	DerivedEntity_SettlementInstructionUpd	X	X	X	X	X
10430	CorrespondentBank_FrontOfficeTransactionPartyStageInsert	X	X	X	X	X
10440	CorrespondentBank_FrontOfficeTransactionPartyStageUpd	X	X	X	X	X
10450	WatchListStagingTable_WatchList	X	X	X	X	X
10460	WatchListStagingTable_WatchListInstnIDUpd	X	X	X	X	X
10470	PreviousWatchList_WatchList	X	X	X	X	X
10480	DerivedAddress_WatchListNewCountries	X	X	X	X	X

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
10485	WLMProcessingUnlock	X	X	X	X	NA
10490	LinkStaging_FrontOfficeTransactionParty	X	X	X	X	NA
40090	LinkStaging_InsTrxnDerivedEntDerivedAdd	NA	NA	X	NA	NA
10500	LinkStaging_InstructionDerivedEntDerivedAdd	X	X	X	X	NA
10510	NameMatchStaging	X	X	X	X	X
10520	WatchListStagingTable_NameMatchStageInsert	X	X	X	X	X
10530	DerivedEntityLink_LinkStage	X	X	X	X	NA
10540	DerivedEntitytoDerivedAddress_LinkStage	X	X	X	X	NA
10550	DerivedEntitytoInternalAccount_LinkStage	X	X	X	X	NA
10560	DerivedAddresstoInternalAccount_LinkStage	X	X	X	X	NA
10570	WatchListStagingTable2_WatchListStage2AcctExistence	X	X	X	X	X
10580	WatchListStagingTable2_WatchListStage2CBExistence	X	X	X	X	X
10590	WatchListStagingTable2_WatchListStage2CustExistence	X	X	X	X	X
10600	WatchListStagingTable2_WatchListStage2DAExistence	X	X	X	X	X
10610	WatchListStagingTable2_WatchListStage2EEExistence	X	X	X	X	X
10620	WatchListStagingTable2_WatchListStage	X	X	X	X	X
10630	WatchListStagingTable2_AcctListMembershipUpd	X	X	X	X	X
10640	WatchListStagingTable2_CBListMembershipUpd	X	X	X	X	X
10650	WatchListStagingTable2_CustListMembershipUpd	X	X	X	X	X
10660	WatchListStagingTable2_EEListMembershipUpd	X	X	X	X	X
10670	WatchListStagingTable2_EEListMembershipStatusUpd	X	X	X	X	X
10680	WatchListStagingTable2_DAListMembershipUpd	X	X	X	X	X
10690	WatchListStagingTable2_DAListMembershipStatusUpd	X	X	X	X	X
10700	WatchListStagingTable2_WatchListStage2SeqIdUpd	X	X	X	X	X

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
10710	WatchListStagingTable2_WatchListStage2IntrlIdUpd	X	X	X	X	X
10720	Customer_WatchListStage2ListRisk	X	X	X	X	X
10730	CorrespondentBank_WatchListStage2EffectiveRisk	X	X	X	X	X
10740	Customer_WatchListStage2EffectiveRisk	X	X	X	X	X
10750	DerivedAddress_WatchListStage2EffectiveRisk	X	X	X	X	X
10760	DerivedEntity_WatchListStage2EffectiveRisk	X	X	X	X	X
10770	WatchListStagingTable2_WatchListStage2SeqId	X	X	X	X	X
10780	AccountListMembership_WatchListStage2Insert	X	X	X	X	X
10790	AccountListMembership_WatchListStage2Upd	X	X	X	X	X
10800	CorrespondentBankListMembership_WatchListStage2Insert	X	X	X	X	X
10810	CorrespondentBankListMembership_WatchListStage2Upd	X	X	X	X	X
10820	CustomerListMembership_WatchListStage2Insert	X	X	X	X	X
10830	CustomerListMembership_WatchListStage2Upd	X	X	X	X	X
10840	DerivedAddressListMembership_WatchListStage2Insert	X	X	X	X	X
10850	DerivedAddressListMembership_WatchListStage2Upd	X	X	X	X	X
10860	DerivedEntityListMembership_WatchListStage2Insert	X	X	X	X	X
10870	DerivedEntityListMembership_WatchListStage2Upd	X	X	X	X	X
10875	Account_EffectiveRiskFactorTxtUpd	X	X	X	X	NA
10880	Account_OverallEffectiveRiskUpd	X	X	X	X	X
10881	Account_AccountCustRiskUpd	X	X	X	X	X
10890	Account_EffRiskUpdAfterWLRiskRemoval	X	X	X	X	X
10900	Account_WatchListStage2EffectiveRisk	X	X	X	X	X
10910	WatchListStagingTable2_WatchListStage2IntrlId	X	X	X	X	X
10920	BackOfficeTransaction_EffectiveAcctivityRiskUpd	X	X	NA	X	NA

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
10930	SettlementInstruction_EntityAcctivityRiskUpd	NA	X	NA	X	NA
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	X	X	X	X	X
10955	AccountGroup_InvestmentObjectiveUpd	NA	NA	NA	NA	X
40100	InsuranceTransaction_EntityAcctivityRiskUpd	NA	NA	X	NA	NA
20010	CorrespondentBank_JurisdictionUpd	X	NA	NA	NA	NA
20020	CorrespondentBank_AcctJurisdictionReUpd	X	NA	NA	NA	NA
20030	FinancialInstitution_InstNameUpd	X	NA	NA	NA	NA
10960	AccountGroup_JurisdictionUpd	X	X	NA	X	X
10970	TransactionPartyCrossReference_BackOfficeTransaction	X	X	NA	X	NA
10980	CashTransaction_FrontOfficeTransaction	X	X	NA	X	NA
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	X	X	NA	X	NA
11000	TransactionPartyCrossReference_FrontOfficeTransaction	X	X	NA	X	NA
11010	WireTransaction_FrontOfficeTransaction	X	X	NA	X	NA
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	X	X	NA	X	NA
11030	CashTransaction_FrontOfficeTransactionRevAdj	X	X	NA	X	NA
11040	MonetaryInstrumentTransaction_FrontOfficeTransactionRevAdj	X	X	NA	X	NA
11050	WireTransaction_FrontOfficeTransactionRevAdj	X	X	NA	X	NA
11060	TrustedPair_StatusEXPUpd	X	X	X	X	NA
11070	TrustedPairMember_AcctExtEntEffecRiskUpd	X	X	X	X	NA
11080	TrustedPair_StatusRRCInsert	X	X	X	X	NA
11090	TrustedPair_StatusRRCUpd	X	X	X	X	NA
11100	ApprovalActionsAudit_TrustedPair	X	X	X	X	NA
11110	TrustedPairMember_StatusRRCInsert	X	X	X	X	NA
11120	BackOfficeTransaction_TrustedFlagsUpd	X	X	X	X	NA
11130	InsuranceTransaction_TrustedFlagsUpd	NA	NA	X	NA	NA
11140	MonetaryInstrumentTransaction_TrustedFlagsUpd	X	X	X	X	NA

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
11150	WireTransaction_TrustedFlagsUpd	X	X	X	X	NA
50050	CustomerDailyProfile_BOT	NA	NA	NA	X	NA
50060	CustomerDailyProfile_FOTPS	NA	NA	NA	X	NA
50070	InstitutionalAccountDailyProfile_DEAL	NA	NA	NA	X	NA
50080	CustomerDailyProfile_DEAL	NA	NA	NA	X	NA
50090	InstitutionalAccountDailyProfile_INST	NA	NA	NA	X	NA
50100	CustomerDailyProfile_INST	NA	NA	NA	X	NA
50110	InstitutionalAccountDailyProfile_CorpAction	NA	NA	NA	X	NA
50120	CustomerDailyProfile_CorpAction	NA	NA	NA	X	NA
50130	InstitutionalAccountDailyProfile_Trade	NA	NA	NA	X	NA
50140	CustomerDailyProfile_Trade	NA	NA	NA	X	NA
60100	ManagedAccountDailyProfile_SameDayTrade	NA	NA	NA	NA	X
60110	ManagedAccountDailyProfile_Trade	NA	NA	NA	NA	X
60120	ManagedAccountDailyProfile_BOT	NA	NA	NA	NA	X
11160	AccountDailyProfile-Trade	X	X	NA	X	X
11170	AccountDailyProfile-Transaction	X	X	NA	X	X
11180	AccountProfile_Trade	X	X	NA	X	X
11190	AccountProfile_Transaction	X	X	NA	X	X
11200	AccountProfile_Stage	X	X	NA	X	X
11210	AccountProfile_Position	X	X	NA	X	X
11220	AccountProfile_Balance	X	X	NA	X	X
60130	HouseholdProfile	NA	NA	NA	NA	X
50150	InstitutionalAccountProfile	NA	NA	NA	X	NA
50160	CustomerProfile	NA	NA	NA	X	NA
60140	ManagedAccountProfile	NA	NA	NA	NA	X
60145	AccountPosition_PercentofPortfolioUpd	NA	NA	NA	NA	X
20040	CorrespondentBankProfile	X	NA	NA	NA	NA
20050	AccountATMDailyProfile	X	NA	NA	NA	NA
11230	ChangeLog_AcctProfileInactivity	X	X	NA	X	NA
11240	AccountPeerGroupMonthlyTransactionProfile	X	X	NA	X	NA
20060	CorrespondentBankPeerGroupTransactionProfile	X	NA	NA	NA	NA
20070	AccountChannelWeeklyProfile	X	NA	NA	NA	NA
40110	InsurancePolicyDailyProfile_InsTrxnInsPolicyBal	NA	NA	X	NA	NA
40120	InsurancePolicyProfile_InsurancePolicyDailyProfile	NA	NA	X	NA	NA

Table 151. BD Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
50170	CustomerBalance_ActiveOTCTradeCtUpd	NA	NA	NA	X	NA
60150	AccountPositionDerived	NA	NA	NA	NA	X
60160	AccountBalance_AcctPosnPair	NA	NA	NA	NA	X
60170	AccountBalance_Acctposn	NA	NA	NA	NA	X
60180	HouseholdBalance	NA	NA	NA	NA	X
60190	AccountManagementStage	*	*	*	*	*
11300	AccountChangeLogSummary	X	X	X	X	X
11310	AccountToCustomerChangeLogSummary	X	X	X	X	X
11320	CustomerChangeLogSummary	X	X	X	X	X

Note: The AccountChangeLogSummary, AccountToCustomerChangeLogSummary, and CustomerChangeLogSummary datamaps must be run with `execute.sh` from 8.0.2 onwards.

Note: BackOfficeTransaction must be loaded after the AccountManagementStage utility has been executed.

APPENDIX H Configuring Administration Tools

This appendix provides instructions on how to configure the Administration Tools feature.

Follow these steps for Administration Tools configuration:

If the administration tool is deployed on a separate web application server, then perform these steps:

1. Log in as an OFSECM Administrator User. The Home page displays.
2. Click **Manage Configuration** from the LHS menu.
3. Select the **Manage Common Parameters**.
4. In the Parameter Category drop-down, select **Used for Design**.
5. In the Parameter Name drop-down, select **Admin Tools**.
6. Set the Attribute 2 Value as follows: <PROTOCOL>://<AdminTools_WEB_SERVER_NAME> :<PORT>

- <PROTOCOL> is web page access PROTOCOL (http or https).
- <AdminTools_WEB_SERVER_NAME> is the FQDN of the web application server hosting Administrative Tools.
- <PORT> is the web application server port hosting Admin Tools.

APPENDIX I

Mapping Compliance Regulatory Reports Actions

Alert and Case Management allows users to take regulatory report actions as a part of resolution of the alerts or cases. Regulatory Reports has different templates for each jurisdiction hence the actions associated to it also differs.

This chapter provides step-by-step instructions for mapping and unmapping the actions associated to a regulatory report template, from OFSECM.

Unmapping RRS Actions from Case Management

RRS actions are unmapped from the following tables.

- `KDD_ACTION`
- `KDD_CASETYPE_ACTION_MAP`
- `KDD_ROLE_ACTION_MAP`
- `KDD_STATUS_ACTION_MAP`

To unmap the RRS actions from these tables, follow these steps:

1. Make a back up of the following tables.

- `KDD_ACTION`
- `KDD_CASETYPE_ACTION_MAP`
- `KDD_ROLE_ACTION_MAP`
- `KDD_STATUS_ACTION_MAP`

2. Execute the following delete statements in the Case Management Schema in the order mentioned below. In the sample action, please update the action names as per the template provided in each of the regulatory report templates.

   - Delete from `KDD_STATUS_ACTION_MAP` where `ACTION_CD` in ('<sample action1>','<sample action2>').
   - Delete from `KDD_ROLE_ACTION_MAP` where `ACTION_CD` in ('<sample action1>','<sample action2>').
   - Delete from `KDD_CASETYPE_ACTION_MAP` where `ACTION_CD` in ('<sample action1>','<sample action2>').
   - Delete from `KDD _ACTION` where `ACTION_CD` in ('<sample action1>','<sample action2>').
   - Commit the transactions.

Note: The complete set of actions under a template must be unmapped.

Use the following tables to decide the actions to be unmapped.

Table 152. Actions

ACTION NAME	ACTION _CD
SAR Actions	
Failed Recommend SAR	CA-99
Approve SAR	CA25
Reject SAR	CA26
SAR Filed	CA47
Escalation Review Completed - File SAR	CA79S
SAR Filed	CA98A
Generate US SAR	CA99A
SAR Filed	CA98S
Generate Corrected SAR	CA224AC
Generate Supplemental SAR	CA226ASU
RR-SAR Approved	CA237
RR-SAR Request Approval	CA238
RR-SAR Closed	CA239
RR-SAR E-file Generated	CA240
RR-SAR Filed	CA241
RR-SAR Rejected	CA242
RR-SAR Reopened	CA243
Generate Corrected SAR Unsuccessful	CA262
Generate US SAR Unsuccessful	CA264
Generate Supplemental SAR Unsuccessful	CA265
No SAR Filed - Close	CA306A
No SAR Filed - Close	CA307S
MY STR Actions	
Generate MY STR	CA232A
MY STR Filed	CA222S
MY STR Filed	CA222A
No MY STR Filed - Close	CA309S
No MY STR Filed - Close	CA308A
Generate MY STR Unsuccessful	CA257
RR-MY STR Rejected	CA256
RR-MY STR Filed	CA255
RR-MY STR Closed	CA253
RR-MY STR Request Approval	CA252
RR-MY STR Approved	CA251
SG STR Actions	
Generate SG STR	CA234A

Table 152. Actions (Continued)

ACTION NAME	ACTION _CD
SG STR Filed	CA228S
SG STR Filed	CA228A
No SG STR Filed - Close	CA311S
No SG STR Filed - Close	CA310A
RR-SG STR Rejected	CA263
Generate SG STR Unsuccessful	CA261
RR-SG STR Closed	CA260
RR-SG STR Request Approval	CA259
RR-SG STR Approved	CA258
NG STR Actions	
Generate NG STR	CA236A
NG STR Filed	CA230S
NG STR Filed	CA230A
Generate NG STR Unsuccessful	CA254
RR-NG STR Reopened	CA250
RR-NG STR Rejected	CA249
RR-NG STR Filed	CA248
RR-NG STR E-file Generated	CA247
RR-NG STR Closed	CA246
RR-NG STR Request Approval	CA245
RR-NG STR Approved	CA244
PAK STR Actions	
PAK STR Filed	CA288S
No PAK STR Filed - Close	CA287S
PAK STR Filed	CA288A
No PAK STR Filed - Close	CA287A
RR-PAKSTR Rejected	CA283
RR-PAKSTR Filed	CA282
RR-PAKSTR Closed	CA281
RR-PAKSTR Request Approval	CA280
RR-PAKSTR Approved	CA279
Generate Supplemental PAK STR	CA278
Generate Corrected PAK STR	CA277
Generate PAK STR	CA276
Generate Supplemental PAK STR Unsuccessful	CA286
Generate Corrected PAK STR Unsuccessful	CA285
Generate PAK STR Unsuccessful	CA284
NZ STR Actions	

Table 152. Actions (Continued)

ACTION NAME	ACTION _CD
Generate NZ STR Unsuccessful	CA305
Generate NZ STR	CA304
RR-NZSTR Reopened	CA303
RR-NZSTR Acknowledged	CA302
RR-NZSTR Submitted	CA301
RR-NZSTR Rejected	CA300
RR-NZSTR Filed	CA298
RR-NZSTR Closed	CA297
RR-NZSTR Request Approval	CA296
RR-NZSTR Approved	CA295
No NZ STR Filed - Close	CA293S
No NZ STR Filed - Close	CA293A
NZ STR Filed	CA291S
NZ STR Filed	CA291A

Unmapping RRS Actions from Alert Management

RRS actions must be unmapped from the following tables.

- `KDD_ACTIVITY_TYPE_CD`
- `KDD_ROLE_ACTIVITY_TYPE`
- `KDD_SCNRO_CLASS_ACTVY_TYPE`
- `KDD_ACTVY_TYPE_REVIEW_STATUS`
- `KDD_ACTVY_TYPE_RSTRN`

To unmap the RRS actions, follow these steps:

1. Make a back up of the following tables:

- `KDD_ACTIVITY_TYPE_CD`
- `KDD_ROLE_ACTIVITY_TYPE`
- `KDD_SCNRO_CLASS_ACTVY_TYPE`
- `KDD_ACTVY_TYPE_REVIEW_STATUS`
- `KDD_ACTVY_TYPE_RSTRN`

2. Execute the following delete statements in the Case Management Schema in the order mentioned below:

   a.Delete from `KDD_ACTVY_TYPE_RSTRN` where `ACTVY_TYPE_CD` in ('<sample action1>','<sample action2>')

   b.Delete from `KDD_ACTVY_TYPE_REVIEW_STATUS` where `ACTVY_TYPE_CD` in ('<sample action1>','<sample action2>')

   c.Delete from `KDD_SCNRO_CLASS_ACTVY_TYPE` where `ACTVY_TYPE_CD` in ('<sample action1>','<sample action2>')

d.Delete from KDD_ROLE_ACTIVITY_TYPE where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')

e.Delete from KDD_ACTIVITY_TYPE_CD where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')

f.Commit the transactions.

Use the following tables to decide the actions to be unmapped.

Table 153. Actions

Activity Type Code	Activity Short Name
SAR Actions	
MTS395	US SAR Filed in Case Mgmt
MTS400	Generate US SAR
MTS400F	Generate US SAR Unsuccessful
MTS401	Generate Corrected SAR
MTS401F	Generate Corrected SAR Unsuccessful
MTS700	RR-SAR Approved
MTS701	RR-SAR Request Approval
MTS702	RR-SAR Closed
MTS703	RR-SAR Efile Generated
MTS704	RR-SAR Filed
MTS705	RR-SAR Rejected
MTS706	RR-SAR Reopened
MTS750	Generate Supplemental SAR
MTS750F	Generate Supplement SAR Unsuccessful
SG STR Actions	
MTS402	Generate SG STR
MTS402F	Generate SG STR Unsuccessful
MTS714	RR-SG STR Approved
MTS715	RR-SG STR Request Approval
MTS716	RR-SG STR Closed
MTS717	RR-SG STR Efile Generated
MTS718	RR-SG STR Filed
MTS719	RR-SG STR Rejected
MTS720	RR-SG STR Reopened
MY STR Actions	
MTS403	Generate MY STR
MTS403F	Generate MY STR Unsuccessful
MTS721	RR-MY STR Approved
MTS722	RR-MY STR Request Approval
MTS723	RR-MY STR Closed
MTS724	RR-MY STR Efile Generated

Table 153. Actions

Activity Type Code	Activity Short Name
SAR Actions	
MTS725	RR-MY STR Filed
MTS726	RR-MY STR Rejected
MTS727	RR-MY STR Reopened
NG STR Actions	
MTS707	RR-NG STR Approved
MTS708	RR-NG STR Request Approval
MTS709	RR-NG STR Closed
MTS710	RR-NG STR Efile Generated
MTS711	RR-NG STR Filed
MTS712	RR-NG STR Rejected
MTS713	RR-NG STR Reopened
MTS751	Generate NG STR
MTS752F	Generate NG STR Unsuccessful
PAK STR Actions	
MTS761	Generate Pakistan STR
MTS761F	Generate Pakistan STR Unsuccessful
MTS762	Generate Corrected Pakistan STR
MTS762F	Generate Corrected PAK STR Unsuccessful
MTS763	Generate Supplemental Pakistan STR
MTS763F	Supplemental PAK STR Unsuccessful
MTS765	RR-PAKSTR Approved
MTS766	RR-PAKSTR Request Approval
MTS767	RR-PAKSTR Closed
MTS768	RR-PAKSTR Filed
MTS769	RR-PAKSTR Rejected
NZ STR Actions	
MTS770	Generate NZ STR
MTS770F	Generate NZ STR Unsuccessful
MTS771	RR-NZSTR Request Approval
MTS772	RR-NZSTR Approved
MTS773	RR-NZSTR Closed
MTS774	RR-NZSTR Filed
MTS775	RR-NZSTR Rejected
MTS776	RR-NZSTR Submitted
MTS777	RR-NZSTR Acknowledged
MTS778	RR-NZSTR Reopened

APPENDIX J Alerts from IPE and External System - Run/Process/Tasks

This appendix provides details about the RUN, process, and the tasks for external alerts.

This appendix covers the following topics:

- RUN Information
- Process Information
- Run to Process to Task Mapping
- Task Information

RUN Information

The following table provides RUN information.

Table 154. RUN Information

Sl No.	Run Code	Run Name
1	BD_TC_TRADE_ACC_SMALL_SELLS	BD Trade Account Small Sells
2	BD_TC_EXEC_TRADE_MKT_VOL	BD Execution Large Trade Volume
3	BD_GENERATE_ALERTS_FROM_IPE	BD Generate Alerts from IPE
4	BD_EXTRL_ALERT_GENERATION	BD External Alert Generation

Process Information

The following table provides Process information.

Table 155. Process Information

Sl No.	Process Code	Process Name
1	BD_TC_TRADE_ACC_SMALL_SELLS	BD Trade Account Small Sells
2	BD_TC_EXEC_TRADE_MKT_VOL	BD Execution Large Trade Volume
3	BD_GENERATE_ALERTS_FROM_IPE	BD Generate Alerts from IPE
4	BD_EXTRL_ALERT_GENERATION	BD External Alert Generation

Run to Process to Task Mapping

The following table provides information about Run to Process to Task Mapping.

Table 156. Run to Process to Task Mapping

SL No.	Run Code	Process Code	Task Code
1	BD_TC_TRADE_ACC_SMALL_SELLS	BD_TC_TRADE_ACC_SMALL_SELLS	● Trade ● BD_POPULATE_LAST_RUN_BATCH
2	BD_TC_EXEC_TRADE_MKT_VOL	BD_TC_EXEC_TRADE_MKT_VOL	● Execution ● BD_POPULATE_LAST_RUN_BATCH

Table 156. Run to Process to Task Mapping

3	BD_GENERATE_ALERTS_FROM_IPE	BD_GENERATE_ALERTS_FROM_IPE	● BD_SET_BATCH_DATE_FOR_IPE ● BD_START_BATCH_FOR_IPE ● BD_POPULATE_STAG_FRM_IPE ● BD_POPULATE_RUN_TABLE ● BD_POPULATE_BREAK_INFO_FRM_COMMON_PROCESSING ● BD_UPDATE_RUN_COUNTER ● BD_POPULATE_BREAK_MATCH_INFO_FRM_COMMON_PROCESSING ● BD_POPULATE_BREAK_BNDNG_INFO_FRM_COMMON_PROCESSING ● BD_GEN_SING_MATCH_ALERTS ● BD_POPULATE_FIN_INFO_OF_ALERT ● BD_UPDATE_REVIEW_INFO ● BD_HISTORICAL_DATA_COPY ● BD_ALERT_ASSIGNMENT ● BD_END_BATCH_FOR_IPE
4	BD_EXTRL_ALERT_GENERATION	BD_EXTRL_ALERT_GENERATION	● BD_SET_BATCH_DATE_FOR_IPE ● BD_START_BATCH_FOR_IPE ● BD_POPULATE_BREAK_INFO_FRM_COMMON_PROCESSING ● BD_POPULATE_BREAK_MATCH_INFO_FRM_COMMON_PROCESSING ● BD_POPULATE_ALERT_INFO_FRM_COMMON_PROCESSING ● BD_POPULATE_ALERT_SCENARIO_MAPPING_FRM_COMMON_PROCESSING ● BD_POPULATE_ALERT_AUDIT ● BD_UPDATE_ALERT_BREAK_INFO ● BD_POPULATE_FIN_INFO_OF_ALERT ● BD_ALERT_ASSIGNMENT ● BD_HISTORICAL_DATA_COPY ● BD_END_BATCH_FOR_IPE

Task Information

The following table provides Task Information:

Table 157. Task Information

Sl No.	Task Code	Task Name	Tables Affected
1	Trade	IPE Activity - Trade	RTI_ASSMNT_RESULT, RTI_ASSMNT_EVAL_RESULT
2	Execution	IPE Activity - Execution	RTI_ASSMNT_RESULT, RTI_ASSMNT_EVAL_RESULT
3	BD_POPULATE_LAST_RUN_BATCH	Populate Batch Execution List Table	KDD_EXTRNL_BATCH_LAST_RUN
4	BD_SET_BATCH_DATE_FOR_IPE	Set the Batch Date	KDD_CAL
5	BD_START_BATCH_FOR_IPE	Start the Batch	KDD_PRCSNG_BATCH_CONTROL
6	BD_POPULATE_STAG_FRM_IPE	Populate Common Processing Table	● KDD_EXTRL_MTCH ● KDD_EXTRL_BREAK_MTCHS ● KDD_EXTRL_MTCH_ACTUAL_VAL
7	BD_POPULATE_RUN_TABLE	Populate Run Table	KDD_RUN
8	BD_POPULATE_BREAK_INFO_FRM_COMMON_PROCESSING	Populate Break Information Table	KDD_BREAK
9	BD_UPDATE_RUN_COUNTER	Update Run Counter Sequence	KDD_COUNTER
10	BD_POPULATE_BREAK_MATCH_INFO_FRM_COMMON _PROCESSING	Populate into Break Match Information TableKDD_BREAK _MTCHS Table	KDD_BREAK_MTCHS
11	BD_POPULATE_BREAK_BNDNG_INFO_FRM_COMMON_PROCESSING	Populate Break Binding	KDD_BREAK_BINDING
12	BD_GEN_SING_MATCH_ALERTS	Generates Alerts in Review Table	● KDD_REVIEW KDD_REVIEW_SCNRO ● KDD_ACTIVITY
13	BD_POPULATE_FIN_INFO_OF_ALERT	Populate Financial Data Information Table	KDD_REVIEW_FINANCIAL
14	BD_UPDATE_REVIEW_INFO	Update Review with Scenario and Highlight Name	KDD_REVIEW
15	BD_ALERT_ASSIGNMENT	Alert Assignment process	KDD_REVIEW
16	BD_HISTORICAL_DATA_COPY	Historical Data Copy Process	Archive Tables
17	BD_END_BATCH_FOR_IPE	End the Batch	KDD_PRCSNG_BATCH_CONTROL

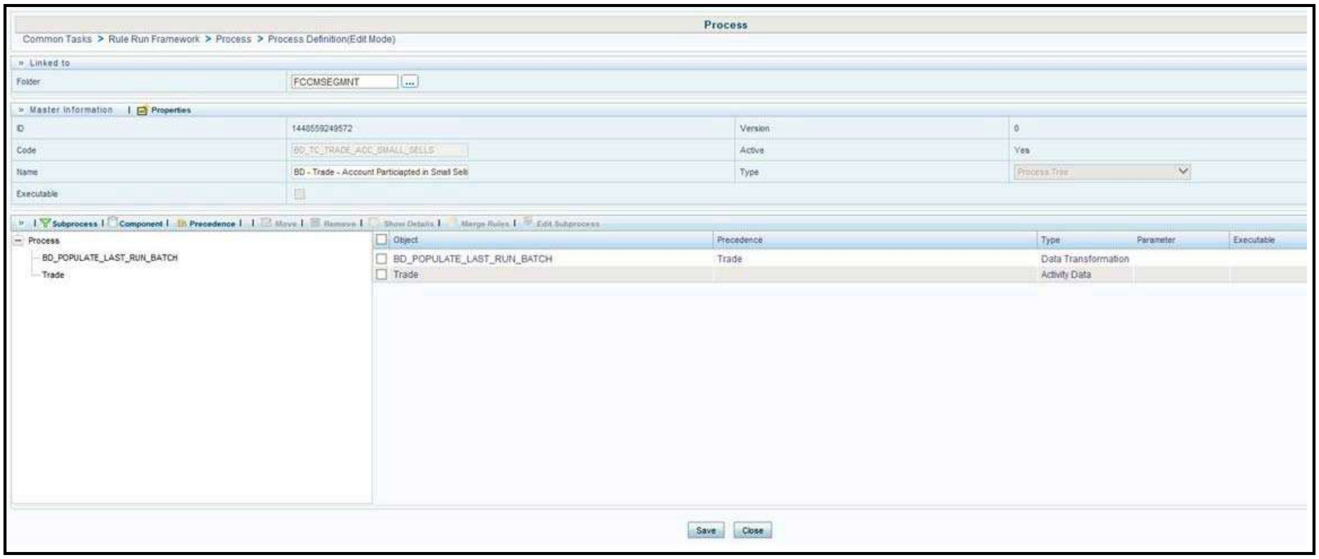
.

Table 158. IPE Assessment Execution Results to BD System

Sl No.	Task Name	Task Description
1	BD_SET_BATCH_DATE_FOR_IPE	Set the Batch Date
2	BD_START_BATCH_FOR_IPE	Start the Batch
3*	BD_POPULATE_MTCH_THSLD_ACT_VAL	Populate Common Processing Table
4*	BD_POPULATE_MATCH_INFO_FRM_IPE	Populate Common Processing Table
5	BD_POPULATE_BREAK_INFO_FRM_COMMON_PROCESSING	Populate Break Information
6	BD_POPULATE_BREAK_MATCH_INFO_FRM_COMMON_PROCESSING	Populate Break Match Information
7	BD_POPULATE_ALERT_INFO_FRM_COMMON_PROCESSING	Populate Alert Information
8	BD_POPULATE_ALERT_SCENARIO_MAPPING_FRM_COMMON_PROCESSING	Populate Alert Scenario Mapping
9	BD_UPDATE_ALERT_BREAK_INFO	Update Break with the Review ID's
10	BD_POPULATE_FIN_INFO_OF_ALERT	Populate Financial Data Information Table
11	BD_ALERT_ASSIGNMENT	Alert Assignment process
12	BD_HISTORICAL_DATA_COPY	Historical Data Copy Process
13	BD_END_BATCH_FOR_IPE	End the Batch

Execution

Once an Assessment is defined, the assessment must be executed as a task in batch mode using the Run Rule Framework. As a prerequisite for further processing of assessment results into alerts, the BD_POPULATE_LAST_RUN_BATCH task should be executed after the IPE assessment task.



Process

Common Tasks > Rule Run Framework > Process > Process Definition(Edit Mode)

» Linked to

Folder	FCCMSEGMNT

» Master Information | Properties

ID	1448559249572	Version	0
Code	BD_TC_TRADE_ACC_SMALL_SELLS	Active	Yes
Name	BD - Trade - Account Particiapted in Small Sells	Type	Process Tree
Executable	□		

» | Subprocess | Component | Precedence | | Move | Remove | Show Details | Merge Rules | Edit Subprocess
Process
BD_POPULATE_LAST_RUN_BATCH
Trade

□ Object	Precedence	Type	Parameter	Executable
□ BD_POPULATE_LAST_RUN_BATCH	Trade	Data Transformation		
□ Trade		Activity Data		

Figure 59. Execution

An IPE assessment is configured using the Run Rule Framework, and the trade task is created for executing IPE assessment in batch mode. The BD_POPULATE_LAST_RUN_BATCH task must be populated in the same process as the IPE assessment task, with the IPE assessment task having precedence.

For more information on how to create and execute the tasks sequentially in Run Rule Framework, see Oracle Financial Services Analytical Applications Infrastructure User Guide.

Index

ORACLE®