Administration Guide

Oracle Financial Services:

Anti-Money Laundering | Fraud | Trading Compliance | Broker Compliance | Energy and Commodity Trading Compliance | Enterprise Case Management | FATCA Management

Release 6.2.1
September 2013

ORACLE

Administration Guide

Oracle Financial Services:

Anti-Money Laundering | Fraud | Trading Compliance | Broker Compliance | Energy and Commodity Trading Compliance | Enterprise Case Management | FATCA Management

Release 6.2.1
September 2013

Document Control Number: 9MN12-62110002
Document Number: AG-13-FCCM-0002-6.2.1-01

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

Document Number: AG-13-FCCM-0002-6.2.1-01
First Edition (September 2013)

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

Contents

List of Figures

List of Tables

About this Guide

This guide explains the concept behind the Oracle Financial Services Behavior Detection Framework (OFSBDF), 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 OFSBDF, include the following:

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

Users who have access to any of the Financial Crime and Compliance Management (FCCM) modules will get unrestricted access to all the administration utilities that are required to administer the module.

Scope of this Guide

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

OFSBDF provides the foundation for all its products. Advanced data mining algorithms and sophisticated pattern recognition technologies power OFSBDF. It provides an open and scalable infrastructure that supports rich, end-to-end functionality across all Oracle Financial Services solution sets. OFSBDF'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)
- Enterprise Case Management (ECM)

Your implementation may not be utilizing all of these products.

How this Guide is Organized

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

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

- Appendix B, *OFSBDF Software Updates*, describes the application of OFSBDF software updates (hotfix) and their impact on customization.
- Appendix C, *BDF Datamap Details*, lists the Datamap XML and their use in OFSBDF.
- Appendix D, *Datamaps Matrix*, lists which datamaps are required for each solution set.
- Appendix E, *Configuring Admin Tools*, describes how to configure the Admin Tools feature.
- Appendix F, *Mapping Regulatory Reports Actions* provides information about integration of OFSRRS.
- Appendix G, *Ingestion Using FSDF Datamaps* lists the Oracle Financial Services Data Foundation (OFSDF) datamaps used in OFSAAI.
- 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 documents:

- *Scenario Manager User Guide*
- *Administration Tools User Guide*
- *Services Guide*
- *Data Interface Specification (DIS)*
- *Configuration Guide*
- *Oracle Financial Services Analytical Applications Infrastructure User Manual*
- *Oracle Financial Services Analytical Applications Infrastructure User Guide Release 7.3*
- *Installation Guide - Stage 1*
- *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration Release 7.3*
- *Installation Guide - Stage 3*

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

Conventions Used in this Guide

Table 1 lists the conventions used in this guide.

Table 1. Conventions Used in this Guide

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

CHAPTER 1 About OFSBDF

This chapter provides a brief overview of OFSBDF in terms of its architecture and operations.

This chapter focuses on the following topics:

- Architecture
- Operations
- Utilities

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.

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. For the purpose of this administration document there is a tier which represents Enterprise Case Management components and subsystems. And a tier that represents the grouping of the behavior detection products of AML, Fraud, TC, BC and ECTC and their associated alert management interface. A set of components further divides each tier.

Components are units of a tier that can be installed separately onto a different server. Table 2 outlines the tiers and components. In some cases, however, individual deployments can add subsystems to meet a client's custom requirements.

Table 2. Components and Subsystems

Component	Subsystem	Directory Name	Contents
Tier-OFSBDF			
Data Ingestion		ingestion_manager	Java components, scripts, and stored procedures
Data Ingestion	Financial Services Data Model	database	Database utilities and database creation scripts
Data Ingestion	BDF Datamaps	bdf	Datamap XML and configuration parameters.
Behavior Detection	Behavior Detection	behavior_detection	(Subsystem)
Behavior Detection	Behavior Detection Framework	bdf	Datamap XML and configuration parameters.
Behavior Detection	Detection Algorithms	algorithms	C++ behavior detection algorithms
Behavior Detection	Scenario Manager	toolkit	Job and scenario editors
Alert Management	Alert Management Web	solution\am	JSPs used in Alert Management
Alert Management	Alert Management UI	ftpshare\< alert infodom>\erwin\forms	Xmls for rendering the UI
Alert Management	Web Services	services	Web services for watch list scanning and for the alert management supervisor (used when posting alerts to Behavior Detection)
Alert Management	Correlation		
Alert Management	Administration Tools	admin_tools	Web-enabled Administration Tools
Alert Management	Trade Blotter		
Alert Management	Manage Security Restriction		
Alert Management	Manage Controlling Customer		
Alert Management	Watch List Management		
Tier-OFSECM			
Case Management	Case Management UI	ftpshare\<case infodom>\erwin\forms	Xmls for rendering the UI
Case Management	Case Management Web	solution\cm	JSPs used in Case Management

The following sections describe the tiers and their components.

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 has the prowess to filter data that you are viewing and using for analysis.

OFSBDF

OFSBDF contains the following components:

- **Data Ingestion:** 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 the OFSBDF 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.

Data Ingestion

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 Ingestion 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 OFSBDF client's job scheduling system invokes BDF datamap XML to derive and aggregate data. The Data Ingestion component also uses the Fuzzy Name Matcher Utility to compare names found in source data with names in the Watch List.

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

Behavior Detection

OFSBDF 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.
OFSBDF 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 displays to you while an alert is processing.

Alert Management

An alert represents a unit of work that is the result of the detection of potentially suspicious behavior by Oracle Scenarios. OFSBDF 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

OFSECM

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 OFSBDF that have been promoted to a case (if your firm has implemented Oracle Financial Services Behavior Detection). When used in conjunction with OFSBDF, based on your roles and permissions, you can link or unlink additional alerts to a case during the investigation.

Deployment View

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

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.

OFSBDF components can operate when deployed on a single computer or when distributed across multiple computers. In addition to being horizontally scalable, OFSBDF 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 an Behavior Detection architecture deployment.. Behavior Detection uses inbuilt SMS for its authentication and authorization. The SMS has a set of database tables which store information about user authentication.

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

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

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 OFSBDF directory server uses.

Operations

As the OFSBDF administrator, you coordinate the overall operations of OFSBDF: Data Ingestion, Behavior Detection, and Post-Processing (Figure 1).

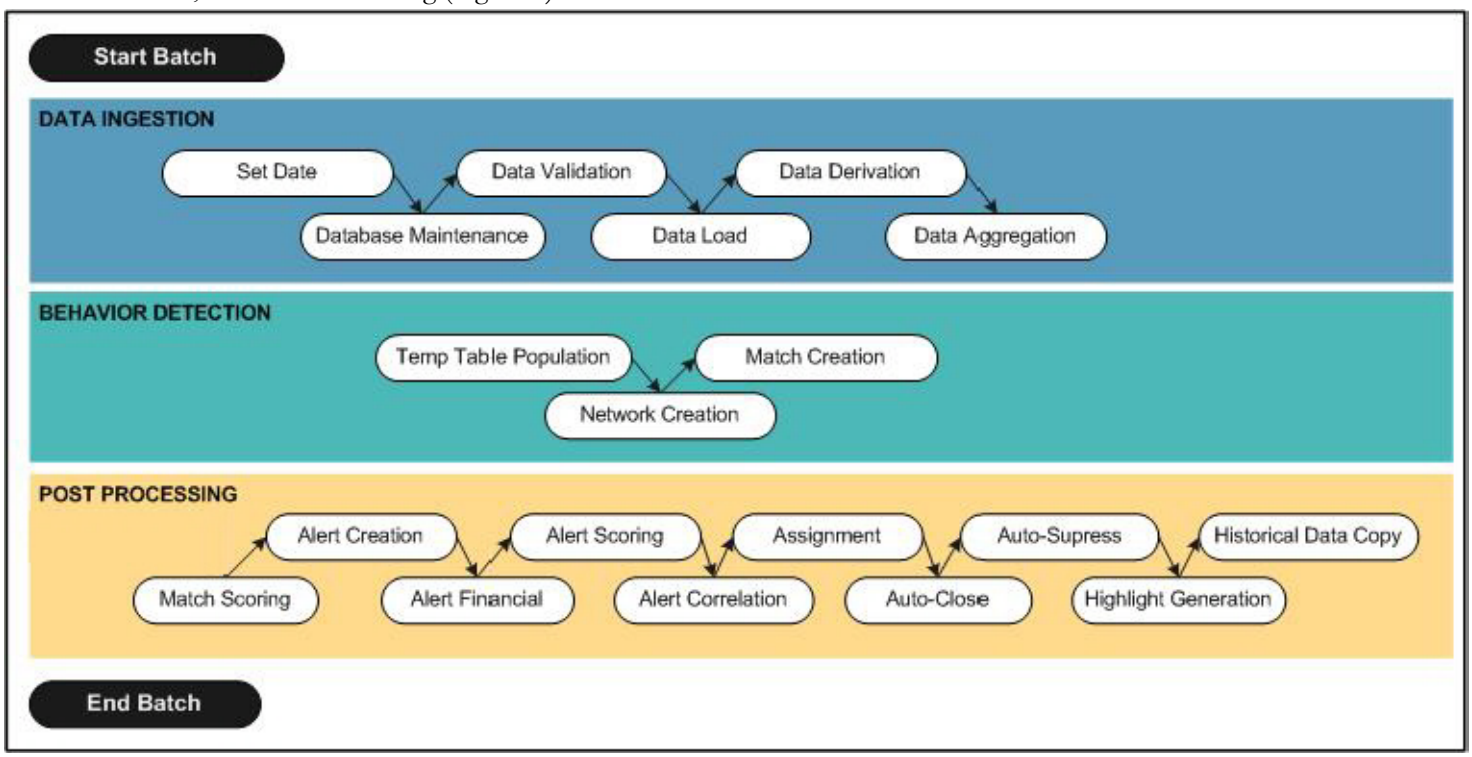

Figure 1. OFSBDF Architecture—Behavior Detection Framework Processing

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

As Figure 1 illustrates, each cycle of OFSBDF process begins with Data Ingestion, Behavior Detection, and Post-Processing, which prepares the detection results for presentation for the users.

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

- Start Batch
- Data Ingestion

- Behavior Detection
- Post-Processing
- End Batch

Start Batch

Using the Batch Control Utility, you can manage the beginning of the OFSBDF batch process (Refer to Chapter 7, *Batch Processing Utilities* on page 201 for more information).

Data Ingestion

The OFSBDF Ingestion Manager controls the data ingestion process. The *Oracle Financial Services Behavior 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 ingestion involves receiving source data from an external data source in one of these forms. The Ingestion Manager validates this data against the DIS, applies required derivations and aggregations, and populates the OFSBDF database with the results (Refer to Chapter 3, *Data Ingestion,* on page 33 for more information).

Behavior Detection

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

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

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

Post-Processing

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

- **Augmentation**: Collects additional information related to the matched behavior and focus for pattern detection, which enables proper display or analysis of the generated matches.
- **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.
- **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 (Refer to section *Ending a Batch Process* on page 237 for more information). The Alert & Case Management subsystem then controls alert and case management processes. Refer to the *Alert Management User Guide*, Release 6.2.1, for more information.

Utilities

OFSBDF 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 Ingestion to alert post-processing) and enables access to the currently running batch.
- **Calendar Manager Utility**: Updates calendars in the system based on predefined 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 E-mail Notification**: Enables you to configure users of the Alert Management subsystem to receive e-mail 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.

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.

CHAPTER 2 Security Configuration

This chapter provides instructions for setting up and configuring the Security Management System (SMS) to support OFSAAI user authentication and authorization. It also contains instructions for setting up user accounts in the OFSAAI database to access the Scenario Manager.

This chapter focuses on the following topics:

- About OFSECM User Authentication
- About User Setup
- About Configuring Access Control Metadata
- Mapping Users To Access Control Metadata
- About Scenario Manager Login Accounts
- About Changing Passwords for System Accounts
- About Configuring File Type Extensions
- About Configuring File Size
- About Configuring Status To User Role Table

About OFSECM User Authentication

The primary way to access information is through a Web browser that accesses the Alert Management, Case Management, and Administration Tools. The Scenario Manager authenticates use of the OFSAAI database only.

Web server authentication is also available for Oracle clients who want to utilize their own External Authentication Management (EAM) tool.

Accessing OFSECM

A user gains access to OFSECM based on the following:

- Authentication of a unique user ID and password that enables access to Alert Management, Case Management, and Administration Tools.

For accessing Alert Management:

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

For accessing Case Management:

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

For accessing Watch List Management:

- Set of policies that associate functional roles with access to specific system functions in OFSECM.
- Access to one or more jurisdictions.
- Access to one or more business domains.

For accessing Administration Tool:

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

About User Setup

To set up a user and provide the user access to OFSFCCM, perform the following steps:

1. Create a user: Refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3* for setting up a user. You can create as many users as there are roles

   For example, ECM Administrator/ECM Supervisor/ECM Analyst, WLM Supervisor, KYC Investigator/KYC Relationship Manager, and so on. One user can also be used against multiple roles.

   If multiple roles are allocated to a single user, then the availability of actions will depend 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.
2. Once the user is created, perform mapping of User Groups and Information Domains. Refer to section *Mapping User Group(s) to Domain(s),* on page 11 for more information.
3. Once the user is created and User Groups are mapped with Information Domains, then map the user to the User Group. This in turn maps the user to the role. With this the user will have access to the privileges as per the role.

Note: You must assign at least one Alert Management or Case Management role and one Administrator role per user.

Refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual* for further information.

User Group and User Roles

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

For creating a new user group and mapping it to en existing role, refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3*:

Note: Different solutions have different predefined/preoccupied precedence of User Groups. Therefore, if ECM Admin/System Admin is creating a new User Group make sure while providing precedence value to not use the following precedence:

Table 3. Solution with Predefined 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

While creating a new User Group, we can give precedence as 5001

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

Mapping User Group(s) to Domain(s)

To map User Group(s) to Domain(s), follow these steps:

1. Map all the Alert Management User Groups to Alert Management Information Domain (Infodom).
2. Map all the Case Management User Groups to Alert Management Information Domain (Infodom)and Case Management Information Domain (Infodom).
3. Map all the Know Your Customer User Groups to Alert Management Information Domain (Infodom), Case Management Information Domain (Infodom) and Know Your Customer Information Domain (Infodom).

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

Actions to Role mappings are done through Database tables. Sample action to role mappings are included in the application. Refer to the following sections of the *Configuration Guide*, for changing the mapping of roles to actions.

- 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.

The following table describes the predefined User Roles and corresponding User Groups present in OFSFCCM.

Table 4. Alert Management Roles and User Groups

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

The following table describes the Case Management Roles and corresponding User Groups present in OFSFCCM.

Table 5. Case Management Roles and User Groups

Role	Group Name	User group Code
Case Analyst1	Case Analyst1 User Group	CMANALYST1UG
Case Analyst2	Case Analyst2 User Group	CMANALYST2UG
Case Supervisor	Case Supervisor User Group	CMSUPERVISORUG
Case Executive	Case Executive User Group	CMEXECUTIVEUG
Case Internal Auditor	Case Internal Auditor User Group	CMINAUDITORUG
Case External Auditor	Case External Auditor User Group	CMEXAUDITORUG
Case Viewer	Case Viewer User Group	CMVIEWERUG
Case Initiator	Case Initiator User Group	CMINITIATRUG
Case Administrator	Case Administrator User Group	CMMANADMNUG
KYC Relationship Manager	KYC Relationship Manager User Group	CMKYCRMUG
KYC Investigator	KYC Investigator User Group	CMKYCINVSTGTRUG
KYC Administrator	KYC Administrator User Group	KYCADMNGRP

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

Table 6. Watch List Roles and User Groups

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

Note: If you wish 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.

Mapping a User to a Single User Group

If a user is to have only one role then that user can be mapped to a single User Group associated with that User Role. Refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 7.3* to know more about User to User Group mapping.

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

If a user needs to have more than one role within OFSECM (that is, within both Alert Management and Case Management), then the user needs to be mapped to the different User Groups associated with the corresponding role. When the user logs into OFSECM, user access permissions would be the union of access and permissions across all roles.

Mapping a user to multiple User Groups across Alert Management and Case Management and other applications

If a user needs to have different roles in both Alert and Case Management and roles for other platform supported applications, then that user has to be mapped to different user groups.

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 will be 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 will still show (I) in the Organization list to show that the organization is a child of the line organization. But the fact that it is unselected 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. 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.
3. 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.
4. Clicl **Save.**

Mapping a Function to a Role

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

AMACCESS

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

CMACCESS

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

RSGNTALL

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

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

Defining the User Access Properties and Relationships

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

- **Business Domain(s):** Property that enables an OFSECM client to model client data along operational business lines and practices.
- **Jurisdiction(s):** Property that enables an OFSECM client to model client data across such attributes as geographic location or type or category of a business entity.
- **Organization(s):** Department or organization to which an individual user belongs.
- **Role(s):** Permissions or authorizations assigned to a user in the system (such as, Behavior Detection Framework OFSECM administrator or Auditor).
- **Scenario Group(s):** Group of scenarios in OFSBDF that identify a set of scenario permissions and to which a user has access rights.
- **Case Type/Subtype(s):** Case type/subtypes combinations to which, a user has access rights.

The following figure illustrates the OFSECM user authorization model.

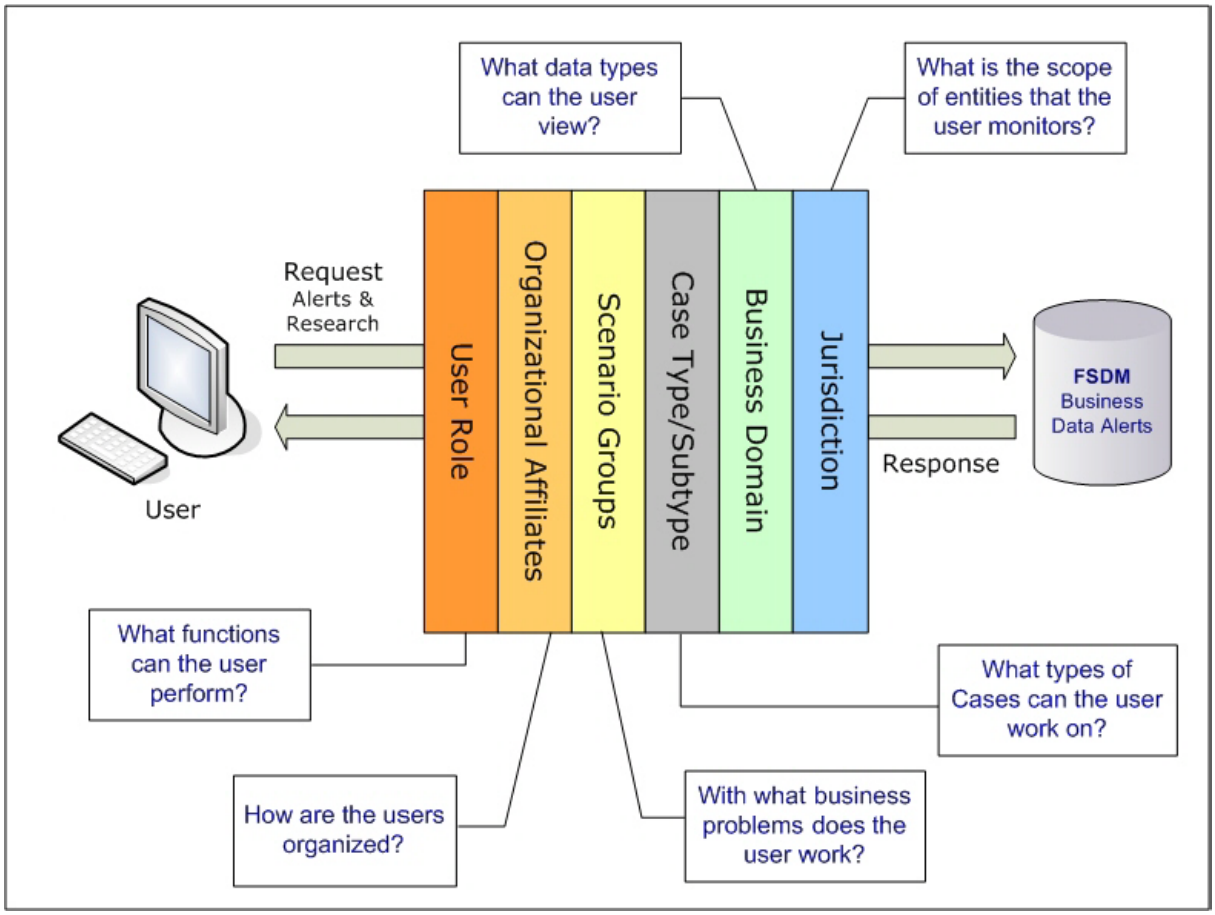

Figure 2. OFSFCCM User Authorization Model

The following provides the relationships between the data points that Figure 5 illustrates.

Table 7. Relationships between Data Points

Data Point	Relationship
Organization	Root of an OFSECM 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

Obtaining Information Before Configuring Access Control

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

Note: Email ID is mandatory for users who would need to take Email action. The user ID should configured with valid email IDs while configuring the same through the User Maintenance UI.

About Configuring Access Control Metadata

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

- **Jurisdiction:** Scope of activity monitoring for example, Geographical Jurisdiction or Legal entity (Refer to *Creating Jurisdiction in the Database* on page 17 for more information).
- **Business Domain:** Operational line of business (Refer to *Creating Business Domain* on page 18, for more information).
- **Scenario Group:** Grouping of scenarios to control user access to scenarios.
- **Role:** Permissions or authorizations assigned to a user.
- **Organization:** User group to which a user belongs.

Some data types such as Scenario Group, Role, Business Domain, Case Type, and Case Subtype which compose the user security configuration are predefined with sample values which are available through the installer. Clients can change or add new values for these data types (with the exception of User Role) based on specific requirements. The following section explains how to add or modify these data types.

Creating Jurisdiction in the Database

OFSECM uses Jurisdictions to limit user access to data in the database. Records from the OFSECM client that the Ingestion Manager loads must be identified with a jurisdiction, users of the system must be associated with one or more jurisdictions. In the Alert and Case Management system, users can view only data or alerts or case associated with jurisdictions to which they have access. You can use a jurisdiction to divide data in the database; for example:

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

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

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

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

Creating Jurisdiction in the Database through Scripts

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

1. Add the appropriate record to the `KDD_JRSDCN` database table, which Table 8 describes.

Table 8. `KDD_JRSDCN` Table Attributes

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

Table 8. KDD_JRSDCN Table Attributes (Continued)

Column Name	Description
JRSDCN_DSPLY_NM	Display name of the jurisdiction (for example, North or South).
JRSDCN_DESC_TX	Description of the jurisdiction (for example, Northern US or Southern US).

2. Add records to the table by using a SQL script similar to the sample script in Figure 6.

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

Figure 3. Sample SQL Script for Loading KDD_JRSDCN

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

Creating Jurisdiction in the Database through Excel Upload

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

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

The Excel Upload screen can be accessed by logging in as an OFSECM Administrator user by clicking the **Administration** Menu and select **Excel Upload**.

Creating Business Domain

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

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

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

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

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

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

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

- Creating Business Domain in the Database through scripts
- Creating Business Domain in the Database through Excel Upload

Creating Business Domain in the Database through scripts

To create a business domain, follow the steps:

1. Add the appropriate user record to the KDD_BUS_DMN database table, which Table 9 describes.

Table 9. KDD_BUS_DMN Table Attributes

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

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

2. Add more records to the table by using a SQL script similar to the sample script in Figure 8.

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

Figure 4. Loading the KDD_BUS_DMN Table

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

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

Creating Business Domain in the Database through Excel Upload

Refer to *Creating Jurisdiction in the Database* on page 17 to perform the Excel Upload for Business Domain. The excel template to be used is `KDD_BUS_DMN.xls`.

Creating Scenario Group

There are two approaches to creating a Scenario Group in the database:

- Creating Scenario Group in the Database through scripts
- Creating Scenario Group in the Database through Excel Upload

Creating Scenario Group in the Database through scripts

To create a Scenario Group, follow these steps:

1. Add the appropriate user record to the `KDD_SCNRO_GRP` database table, which Table 10 describes.

Table 10. 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 Figure 5.

```
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;
```

Figure 5. Loading the KDD_SCNRO_GRP Table

Creating Scenario Group in the Database through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 17, to perform the Excel Upload for Scenario Group. The excel template to be used is `KDD_SCNRO_GRP.xls`.

Creating Scenario Group Membership

There are two approaches to creating a Scenario Group Membership in the database:

- Creating Scenario Group Membership in the Database through scripts
- Creating Scenario Group Membership in the Database through Excel Upload

Creating Scenario Group Membership in the Database through scripts

To create a Scenario Group Membership, follow these steps:

1. Add the appropriate user record to the KDD_SCNRO_GRP_MEMBERSHIP database table, which Table 11 describes.

Table 11. 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 by using a SQL script similar to the sample script in Figure 6.

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

Figure 6. Loading the KDD_SCNRO_GRP_MEMBERSHIP Table

Creating Scenario Group Membership in the Database through Excel Upload

Refer to *Creating Jurisdiction in the Database*, on page 17, to perform the Excel Upload for Scenario Group Membership. The excel template to be used is KDD_SCNRO_GRP_MEMBERSHIP.xls.

Creating a Case Type/Subtype

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

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

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

- KDD_TYPE_CLASS_MAP- Each record in the Case Type and Class Map table represents the set of valid combinations of case type/subtype, subclass1 and subclass2 values which can be used to group the cases for proper investigation.

Creating CaseType/SubType in Investigation Schema

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

- Adding Entries through Excel Upload

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database,* on page 17 for the steps to perform the Excel Upload of Case Subtype.

The excel template to be used is KDD_CASE_TYPE_SUBTYPE.xls

Case type / subtype going to be created must pertain to its classification code 'AML', 'FR', 'KYC'.

Creating Case Subclass1 in Investigation Schema

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database,* on page 17 for the steps to perform the Excel Upload of Case Subclass1.

The excel template to be used is KDD_CASE_SUBCLASS1.xls

Creating Case Subclass2 in Investigation Schema

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database,* on page 17 for the steps to perform the Excel Upload of Case Subclass2.

The excel template to be used is KDD_CASE_SUBCLASS2.xls

Creating Case Type and Class Map in Investigation Schema

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

Adding Entries through Excel Upload

Refer to *Creating Jurisdiction in the Database,* on page 17 for the steps to perform the Excel Upload of Case Type and Class Map.

The excel template to be used is KDD_TYPE_CLASS_MAP.xls

Note: All template excel files for excel upload will be available in ftpshare/STAGE/Excelupload/AMCMLookupFiles

Creating Organizations in the Database

There is one approach to create an Organization in the database:

- Creating Organization in the Database through Excel Upload

Creating Organization in the Database through Excel Upload

Refer to *Creating Jurisdiction in the Database* on page 17 to perform the Excel Upload of organization.

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

Mapping Users To Access Control Metadata

An Administrator can map each user to Access Control Metadata and Security attributes which will control the user's access permissions. The Security Attribute Administration can be accessed from the Administration menu and then selecting **User Administration** (Figure 8).

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

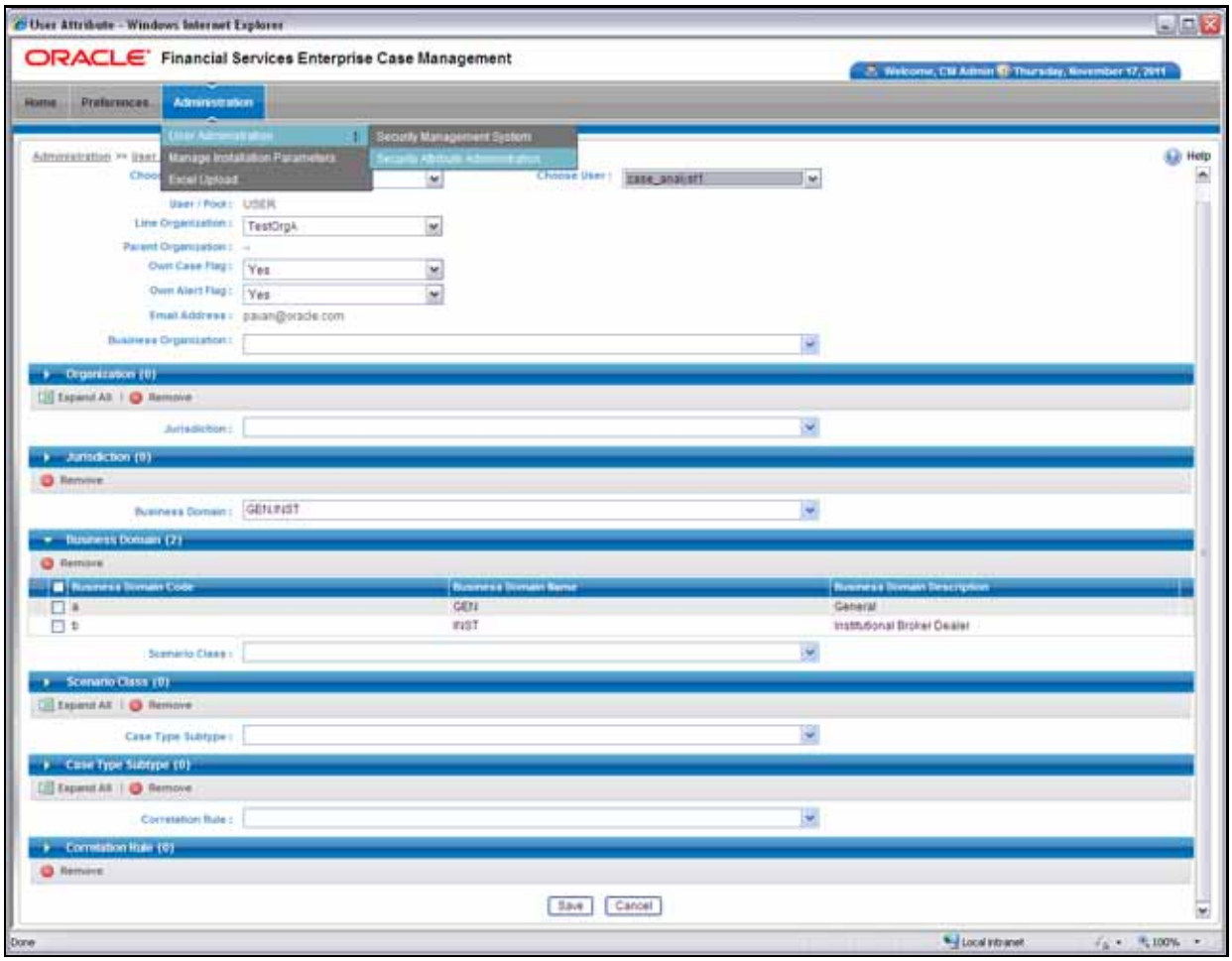

Figure 7. Security Attribute Administration

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

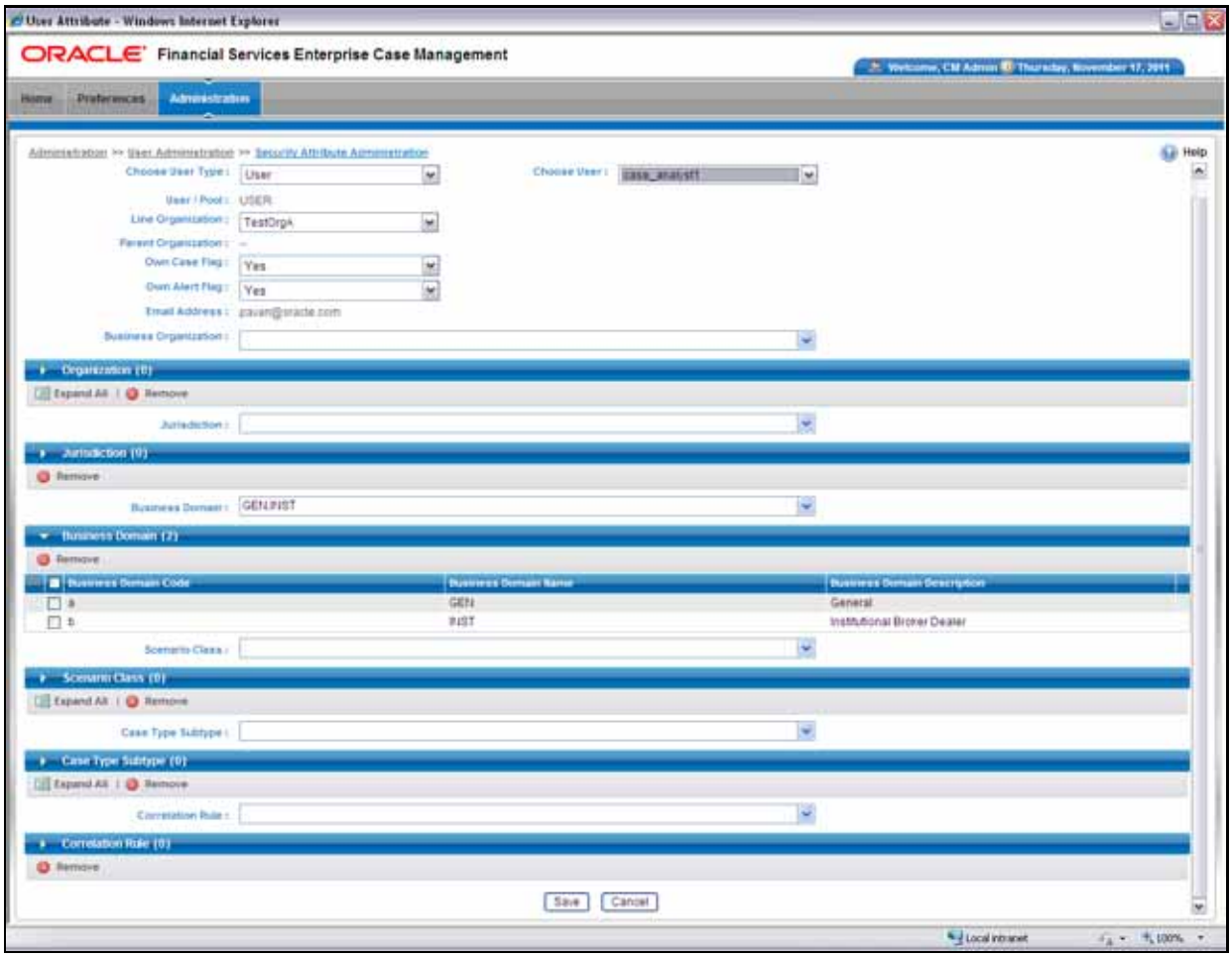

Figure 8. Components of Security Attribute

Note: In order to update the user profiles before proceeding with mapping any security attributes, select the value **User** from the **Choose User Type** drop-down list. When chosen, all the updates made to all the user profiles through User Maintenance UI would be imported from `CSSMS_USER_PROFILE` table of OFSSAAI configuration schema to `KDD_REVIEW_OWNER` table of mantas schema.
If you delete a user through 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. Only then the deleted user will be updated in the `KDD_REVIEW_OWNER` table against the column `actv_flg` as *N*, that is the user is inactive.

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

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

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

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

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

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

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

Jurisdiction
Mapping of one or more jurisdictions to a user or organization, gives the privilege of accessing cases, alerts, watch lists, and watch list members that belong to the mapped jurisdiction.

Business Domain
Mapping of one or more business domains to a user or organization gives privilege of accessing cases, alerts, watch lists, and watch list members that belong to the mapped business domains.

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

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

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

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

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

---

1. The user should be mapped to any one of the following user groups:

- Case Supervisor
- Case Analyst1
- Case Analyst2

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

Note: You must map the scenario group and case type to all users even if they are not case or alert management users.

About Scenario Manager Login Accounts

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

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

Creating Scenario Manager Login Accounts

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

1. Create a database login account and password (Refer to section *To Create the Database Login Account* on page 26, for more information).
2. Set up an account and functional roles in the Scenario Manager. Before performing any tasks in the Scenario Manager, you must set up a user login account that establishes access and roles in the Scenario Manager. Perform these setups by adding records to the database (Refer to section *To Set Up an Account and Functional Roles,* on page 27, for more information).
3. Grant the database roles that the functional roles require. You can grant the role of data miner, or MNR to an *Behavior Detection Framework Scenario Manager* user (Refer to section *To Grant a Database Role*, on page 27, for more information).

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

To Create the Database Login Account

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

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

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

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

To Set Up an Account and Functional Roles

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

1. Access the KDD_USER table.

   The following table defines the attributes for the KDD_USER table.

Table 12. KDD_USER Table Attributes

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

2. Enter the following information into the table using an SQL script:
   a. User database login ID in the USER_ID column. (The Scenario Manager and Oracle database login user IDs must be identical.)
   b. User name in the USER_NM column.
   c. Default user role in the USER_ROLE_CD column.

      To use the Scenario Manager, the user needs the MNR (data miner) database role. The MNR database role is responsible for adjusting the pattern logic of existing scenarios and employs data mining techniques to create new patterns and scenarios.
   d. Flag of Y(es) or N(o) in the ACTV_FL column to specify whether the user is active.

   A sample SQL insert statement is:

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

To Grant a Database Role

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

1. Access the KDD_USER_ROLE table.

   The following table defines the attributes in the KDD_USER_ROLE table.

Table 13. KDD_USER_ROLE Table Attributes

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

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

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

A sample SQL insert statement is:

```
INSERT INTO KDD_USER_ROLE values ('KDD_MNR', 'MNR');
```

About Changing Passwords for System Accounts

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

The following table summarizes the different system account passwords used by Oracle Financial Services Behavior Detection Framework, the subsystems that use those passwords, and instructions on how to change the passwords.

Table 14. System Account Passwords

System Account	Subsystem	Instructions
Data Ingest User (INGEST_USER)	Data Ingestion	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password in Oracle Financial Services Behavior Detection Framework to the new password.
Algorithm User (KDD_ALG)	Behavior Detection Services	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password in Oracle Financial Services Behavior Detection Framework to the new password.
data miner User (KDD_MNR)	Alert & Case Management Data Ingestion	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password in Oracle Financial Services Behavior Detection Framework to the new password.
Web Application User (KDD_WEB)	Alert & Case Management Services	1. Change the password in the database server for this user. 2. Use the Password Manager Utility to change the password in OFSBDF to the new password.
Behavior Detection Framework	Bdf	1. Execute <OFSBDF Installed Directory>/bdf/scripts/changePasswords.sh to generate an encrypted version of the password. 2. Find the <OFSBDF Installed Directory>/bdf/config/custom/bdf.xml with the encrypted password. Refer to the *Installation Guide-Stage 1, Release 6.2.1* for more information. **Note:** Please note that for BDF does not use Password Management utility.
Reports User (KDD_REPORT)	OBIEE Reports	Open the <ORACLEBI_HOME>/server/Repository and expand the Physical Layer. Open the Connection Pool and change the Password parameter to set a new value of the KDD_REPORT schema password. **Note**: OBIEE is an optional application.

About Configuring File Type Extensions

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

About Configuring File Size

By default the size supported by attachment is 1 MB. If you want to attach files greater than 1 MB size using the Save & Attach button, follow these steps:

Configure the size that is allowed to be attached through the document attachment action in the CONFIGURATION table of the OFSSAAI configuration schema. You can configure this in the PARAMVALUE column where the PARAMNAME is DOCUMENT_MAX_SIZE.

About Configuring Status To User Role Table

Within Watch List Management, each watch list and watch list entry (referred to as a "Watch List Member" on the Watch List Management UI) is assigned a status. In addition to the rules defined earlier in this chapter for accessing Watch List Management, OFSECM uses this status to limit user access to watch lists and watch list entries within the Watch List Management. For example. a WLM Supervisor user role can view "Active" watch lists and watch list entries only if the user role "WLM Supervisor" is mapped to status "Active". These mappings reside in the Status To User Role table and are applicable only to the Watch List Management. Each mapping of status to user role applies to both watch lists and watch list entries.

Mapping Status to Role in the Database through Scripts

You can create a Status to User Role mapping in the database by following these steps:

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

Table 15. KDD_STATUS_ROLE Table Attributes

Business Field	Column Name	Date Type	Definition	Null
Status Code	STATUS_CD	CHAR(3)	Status that can be accessed by the user role on this record.	Yes
User Role	USER_ROLE_CD	CHAR(50)	User role that is being assigned access to this status.	Yes

2. Add records to the table by using a SQL script similar to the sample script in Figure 9.

```
insert into kdd_status_role (status_cd,user_role_cd) values ('ACT','WLSUPVISR')
insert into kdd_status_role (status_cd,user_role_cd) values ('REJ','WLSUPVISR')
```

Figure 9. Sample SQL Script for Loading KDD_STATUS_ROLE

Note: The KDD_STATUS_ROLE table is pre populated after system initialization with the following records:

Table 16. KDD_STATUS_ROLE

STATUS_CD	USER_ROLE_CD
ACT	AMEXAUDITR
ACT	AMEXCUTIVE
ACT	AMINAUDITR
ACT	WLSUPVISR
DAC	WLSUPVISR

Configuring Alert and Case Management

The following sections describe how to disable and enable Oracle Financial Services Alert Management and Enterprise Case Management. By default, both workflows are enabled.

- Enabling and Disabling Alert Management
- Enabling and Disabling Case Management

Enabling and Disabling Alert Management

This parameter allows the system to identify whether or not Alert Management Actions and Fields are to be displayed based on the deployment installation. The values to be provided for this parameter are Yes(Y) or No (N).

By default, the parameter is set to Y.

To modify this parameter, follow these steps:

1. Login as an OFSECM Admin User with valid username and password. You are navigated to the Home page.
2. Click **FCCM** and then click the **Administration** Menu and select the **Manage Parameters**.
3. Click **Common Parameters**.
4. Select **Deployment Based** in the Parameter category.
5. Select **Alert Management** from the Parameter Name drop-down list.
6. Edit the parameter.

Enabling and Disabling Case Management

This parameter allows the system to identify whether or not Case Management Actions and Fields are to be displayed based on the deployment installation. The values to be provided for this parameter are Yes(Y) or No(N).

By default the parameter is set to *Y*. To modify this parameter, follow these steps:

1. Login as an OFSECM Admin User with valid username and password. You are navigated to the Home page.
2. Click **FCCM** and then click the **Administration** Menu and select the **Manage Parameters**.

3. Click **Common Parameters**.
4. Select **Deployment Based** in the Parameter category.
5. Select **Case Management** from the Parameter Name drop-down list.
6. Edit the parameter.

CHAPTER 3 Data Ingestion

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

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

Apart from file adapter interface to accept the data, FCCM also supports Financial Services Data Foundation (FSDF), single data storage for multiple functional areas and applications having 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 OFSBDF. OFSBDF invokes BDF datamaps and loads Business, Market and Reference data required for alert processing. Refer to Appendix G, *Ingestion Using FSDF Datamaps,* on page 357 for more information).

About Data Ingestion

The Ingestion Manager receives, transforms, and loads Market, Business, and Reference data that alert detection processing requires. The Ingestion Manager receives data from the file adapter interface, and both Business and Reference data through the file adapter interface. The Data Ingestion subsystem transforms Market, Business, and Reference data to create derived attributes that the detection algorithms require (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 BDF datamaps to derive and aggregate data. The Data Ingestion component also uses the Fuzzy Name Matcher Utility to compare names found in source data with names in the Watch List.

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

Process Flow

The Data Ingestion subsystem components receive and process data in a series of workflow steps that include extract or preprocess, transform, load, and post-load transformations. Figure 10 shows how the Data Ingestion subsystem components participate in each workflow step to extract, transform, and load data. The workflow processes include the following:

- In the **Extract or Preprocess Workflow**, preprocessor components receive the raw market data, business (firm) data, and reference data from their external interfaces. The components then perform data validation and prepare the data for further processing.
- In the **Transform Workflow**, transformer components receive the preprocessed market and business data. The components then create derived attributes that support the downstream alert processing.
- In the **Load and Transform Workflow**, loader components receive preprocessed Reference data and transformed market and business data. The components then load this data into the database. In Post Data Load, data transformations occur through BDF Datamap: derivations and aggregations, risk assignment, and watch list processing (refer to Chapter 4, *BDF Datamaps*, on page 93 for more information).

The following sections describe this process flow in more detail.

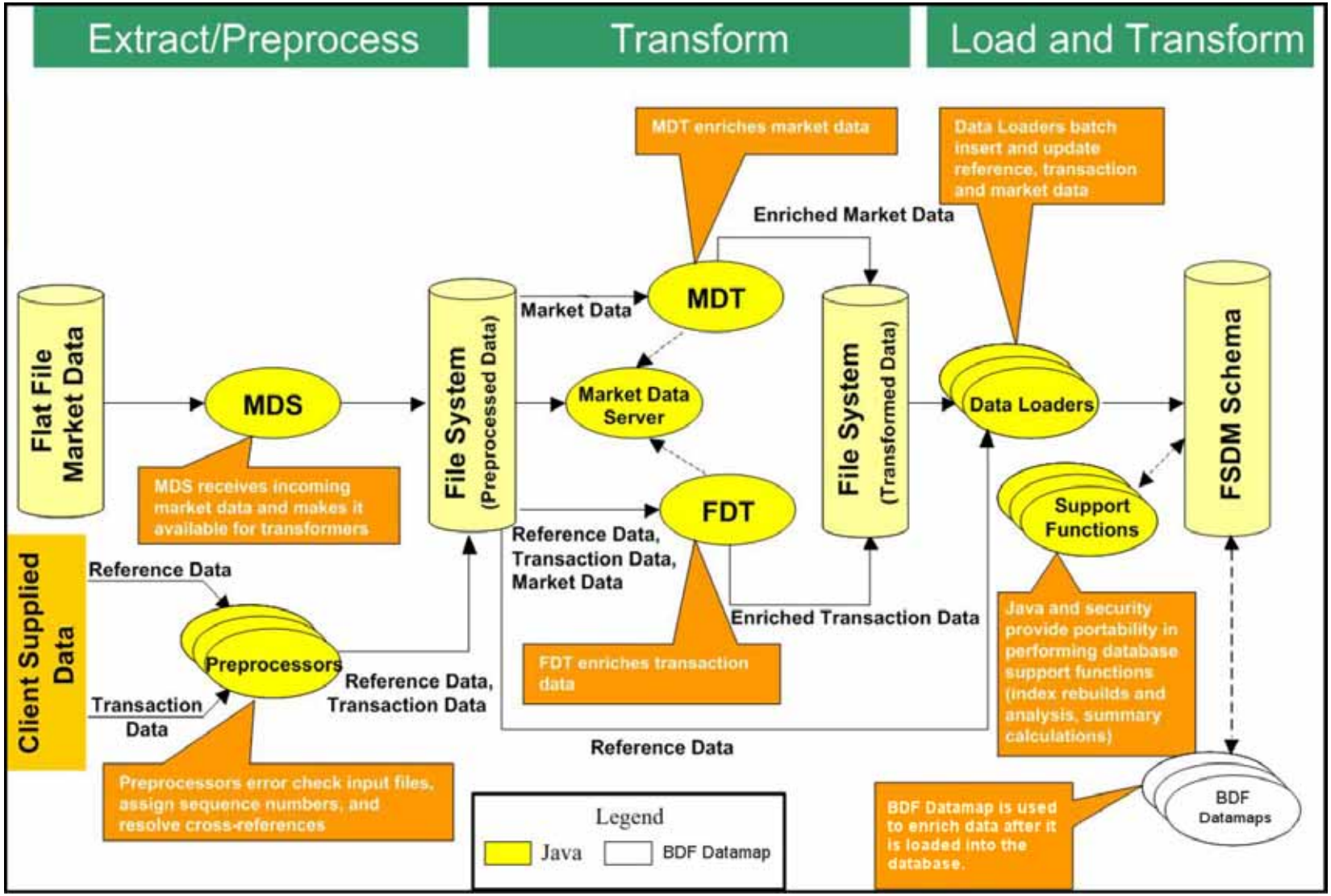

Figure 10. Data Ingestion Subsystem Components Workflow

Data Ingestion Process Summary

Figure 11 provides a high-level view of the Data Ingestion process for OFSBDF Trading Compliance Solution (TC), Anti-Money Laundering (AML), Broker Compliance Solution (BC), Fraud (FR) and Insurance.

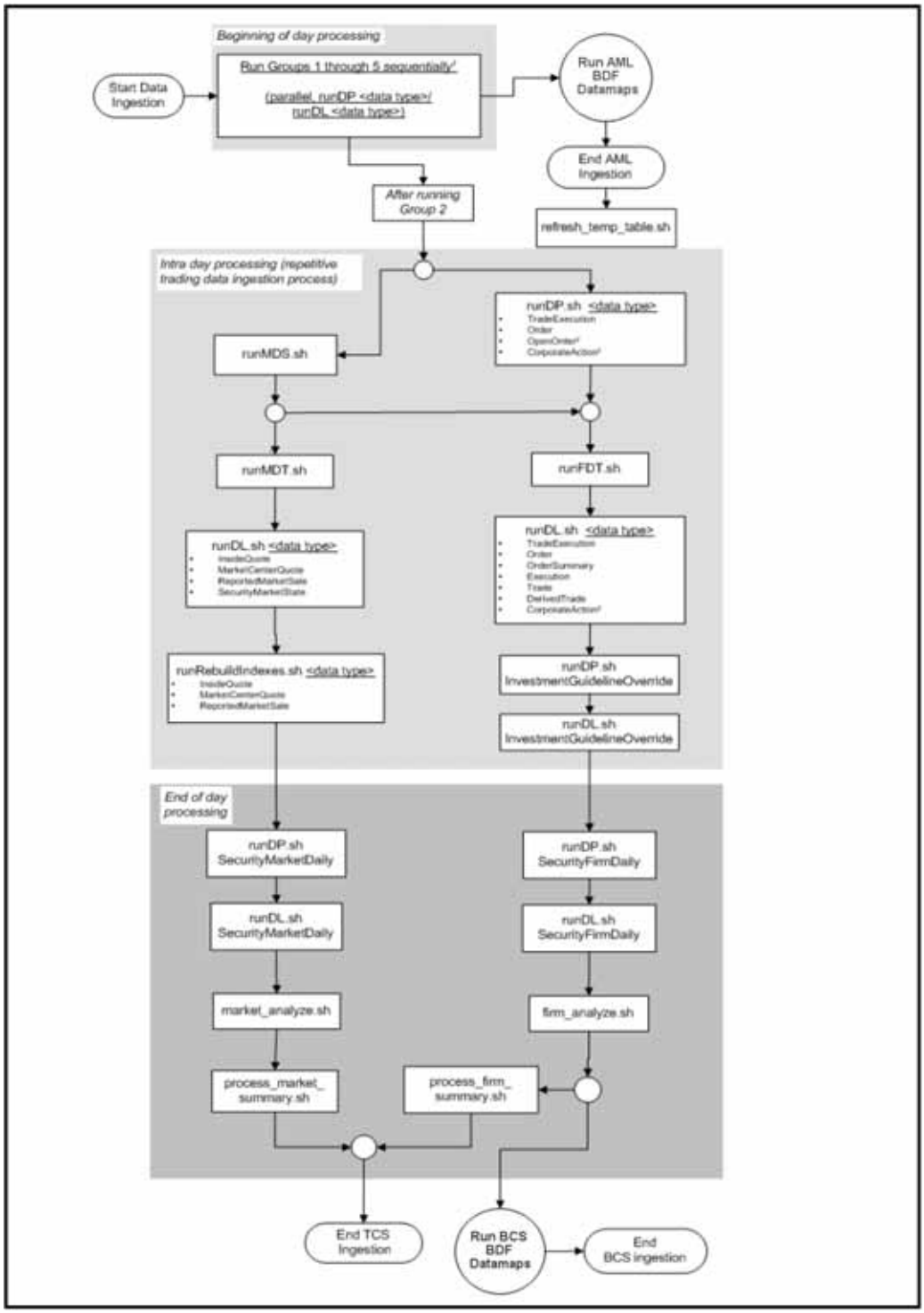

Figure 11. Data Ingestion Process

Alternate Process Flow for MiFID Clients

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

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

Figure 12 depicts dependency between the `process_market_summary.sh` script and the `runFDT.sh` script.

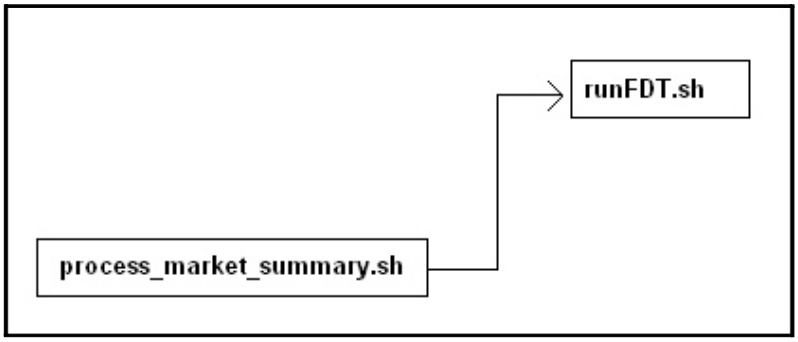

Figure 12. Dependency between *process_market_summary.sh* and *runFDT.sh*

For intra-day batch ingestion or intra-day execution of the MiFID scenarios, the process flow does not change from Figure 11. Since the current day's 4-wk ADTV is not available until the end of the day, the previous day's 4-wk ADTV is used to determine order size.

For additional information on configuring the percentage values used to define a MiFID-eligible order as Small, Medium, or Large, refer to the *Market Supplemental Guidance* section in the *Data Interface Specification (DIS)*, Release 6.2.1.

Data Ingestion Flow Processes

The following sections take the high-level view of Figure 11 and divide the Data Ingestion flow into distinct processes:

- Data Ingestion Directory Structure
- Beginning Preprocessing and Loading
- Preprocessing Trading Compliance Solution Data
- Preprocessing Alternative to the MDS
- Processing Data through FDT and MDT
- Running Trading Compliance Solution Data Loaders
- Rebuilding and Analyzing Statistics
- Populating Market and Business Data Tables

Data Ingestion Directory Structure

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

Processing datestamps should appear as YYYYMMDD for Data Ingestion 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.

For detailed information about the Data Ingestion directory structure, refer to section *Data Ingestion Directory Structure* on page 52 for more information.

Beginning Preprocessing and Loading

In Figure 11, section A, preprocessing begins. The system executes preprocessors using the `runDP.sh` script. The following sample command shows invoking of a preprocessor:

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runDP.sh Account
```

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

The following table lists the data files by group.

Table 17. Data Files by Group

Group	Data Files	
1	AccountCustomerRole AccountPhone AccountEmailAddress AccountRealizedProfitAndLoss CollateralValueCurrency CollateralValueProduct CTRTransaction CommissionProduct ComplaintRegistration ComplaintTypeRating Country EmployeeToInsurancePolicy EnergyAndCommodityInstrument FrontOfficeTransaction InsuranceProduct InsurancePolicy InsurancePolicyBalance InsuranceSeller InsuranceSellerToLicense	InsuranceTransaction Issuer InsurancePolicyToCustomer InsurancePolicyFeature InvestmentGuideline InvestmentGuidelineToAccount LetterofIntent Loan LoanDailyActivity MarketCenter MarketIndex MarketIndexDaily Organization RegisteredRepresentativeComplaint ServiceTeam ServiceTeamMember SystemLogonType WatchList OnlineAccount
2	AccountGroup MatchedEntity SecurityFirmDaily SecurityMarketDaily TrustedPair Security	AccountToPeerGroup FirmAccountPositionPair MarketIndexMemberSecurity NaturalGasFlow PeerGroup

Table 17. Data Files by Group (Continued)

Group	Data Files	
3	Account	AccountSupplementalAttribute
	Customer	CustomerSupplementalAttribute
	Employee	WatchListEntry
	FrontOfficeTransactionParty	RestrictionList
	MarketTradingSession	AutomatedQuote
	OrganizationRelationship	LoanProduct
	AccountGroupAddress	MailHandlingInstructionActivity
	AccountGroupInvestmentObjective	OrganizationToMortgageType
	AccountGroupIOSMember	ReferenceTableDetail
	AccountGroupMemberExperience	ServiceVendor
	BankerToOfficer	EnergyAndCommodityTrade
	GeneralUsageList	LoanOriginationProduct
	LoanOriginationAction	SecuritiesLicense

Table 17. Data Files by Group (Continued)

Group	Data Files	
4	AccessEvents	EmployeePhone
	AccountAddress	EmployeeToAccount
	AccountAssetAllocation	EmployeeToOrganization
	AccountBalance	EmployeeTradingRestriction
	AccountCollateral	EmployeeExamHistory
	AccountFeature	EmployeeSupervisionList
	AccountFees	EmployeeDisciplinaryAction
	AccountGroupMember	EmployeeFirmTransferHistory
	AccountInvestmentObjective	EnergyAndCommodityFirmDaily
	AccountPosition	EnergyAndCommodityMarketDaily
	AccountPositionPair	EnergyAndCommodityMarketCenter
	AccountProfileStage	EnergyAndCommodityMarketTradingSession
	AccountQualificationAgreement	EnergyAndCommodityReportedMarketSale
	AccountToCorrespondent	EnergyAndCommodityInstrumentPosition
	AccountToCustomer	EnergyAndCommodityLocation
	AccountToOrganization	EnergyFlowMode
	AccountRepresentativePosition	ExternalInvestmentAccountPosition
	AccountProfitAndLoss	FirmAccountPosition
	AccountScheduledEvent	FrontOfficeTransactionRemittanceDocument
	AccountPositionProfitAndLoss	LoanOrigination
	AccountIdentifierChangeHistory	MailHandlingInstruction
	AnticipatoryProfile	MarketNewsEvent
	ControllingCustomer	ManagedAccount
	CustomerAddress	MutualFundBreakpoint
	CustomerBalance	MutualFundFamilyConfiguration
	CustomerCountry	MutualFundFamilyLetterOfIntent
	CustomerEmailAddress	OnlineAccountToAccount
	CustomerPhone	PlanOfSolicitation
	CustomerToCustomerRelationship	RelatedFrontOfficeTransactionInformation
	CustomerIdentificationDocument	SecurityInvestmentRating
	CustomerToProductsOffered	SecuritySelectListEntry
	CustomerToMarketsServed	SecurityTradingRestriction
	EmployeeToSecuritiesLicense	SecurityGroupMember
	EmploymentHistory	StructuredDeal
	EmployeeAddress	SystemLogon
	EmployeeEmailAddress	UncoveredOptionAccountPosition
5	AccountRestriction	LoanOriginationNote
	BackOfficeTransaction	LoanOriginationToService
	ChangeLog	OptionsViolation
	InvestmentAdvisor	RegisteredRepresentativeAccountCommission
	SettlementInstruction	RegisteredRepresentativeAccountCommissionPriorYear
	SystemLogonToSystemLogonType	
	Borrower	RegisteredRepresentativeCommissionMonthlyProfile
	LoanOriginationCondition	RegisteredRepresentativeCommissionProduct
	LoanOriginationConditionType	SystemLogonToOrganization
	LoanOriginationDocumentPrintLog	
	LoanOriginationFeeDetail	

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.

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

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

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

Process Flow

The ingestion process flow is as follows:

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

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

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

Oracle clients must only supply file types required by the solution sets on their implementation.

2. Ingestion Manager executes preprocessors simultaneously (within hardware capacities). The preprocessors use XML configuration files in the `/config/datamaps` directory to verify that the format of the incoming Oracle client data is correct and validate its content; specifically:
   - Error-checking of input data
   - Assigning sequence IDs to records
   - Resolving cross-references to reference data
   - Checking for missing records
   - Flagging data for insertion or update

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

Table 18. Preprocessing Output Directories

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

Figure 13 summarizes preprocessing input and output directories.

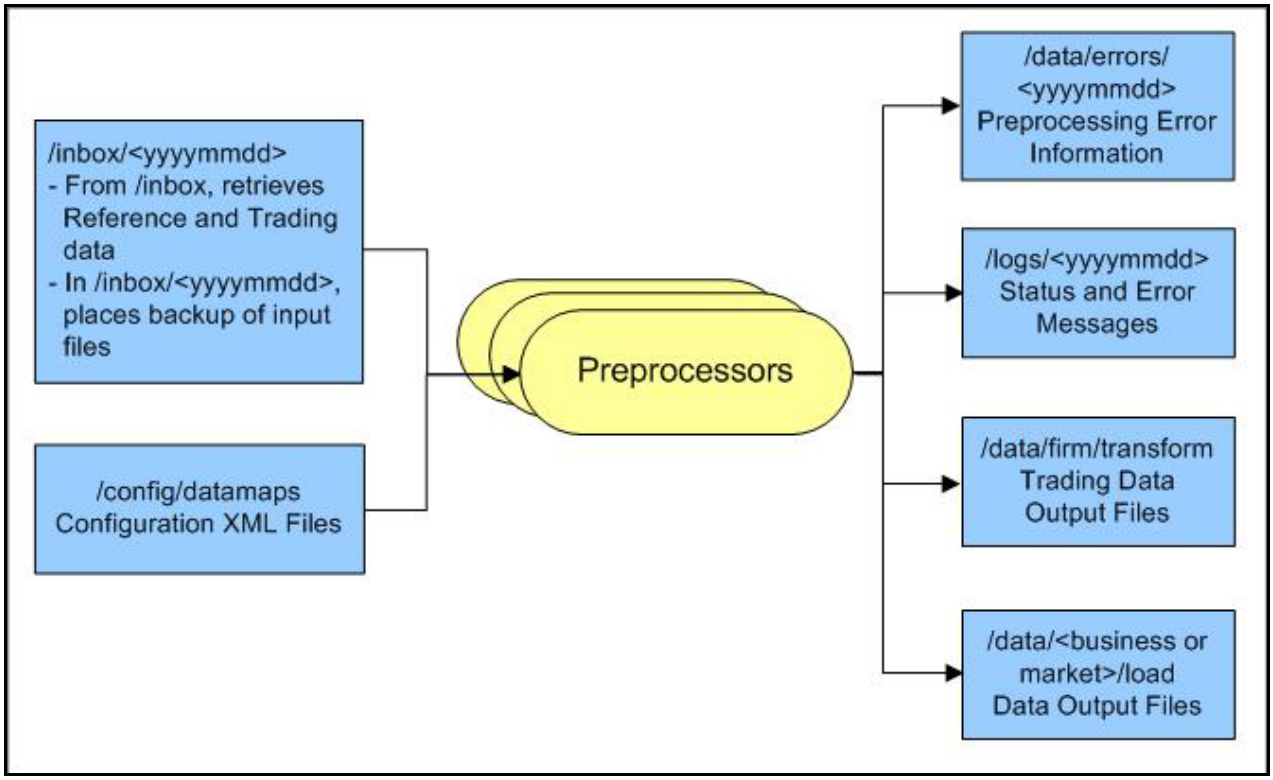

Figure 13. Preprocessing Input and Output Directories

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

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

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

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

The following sample command illustrates execution of data loaders:

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runDL.sh Account
```

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

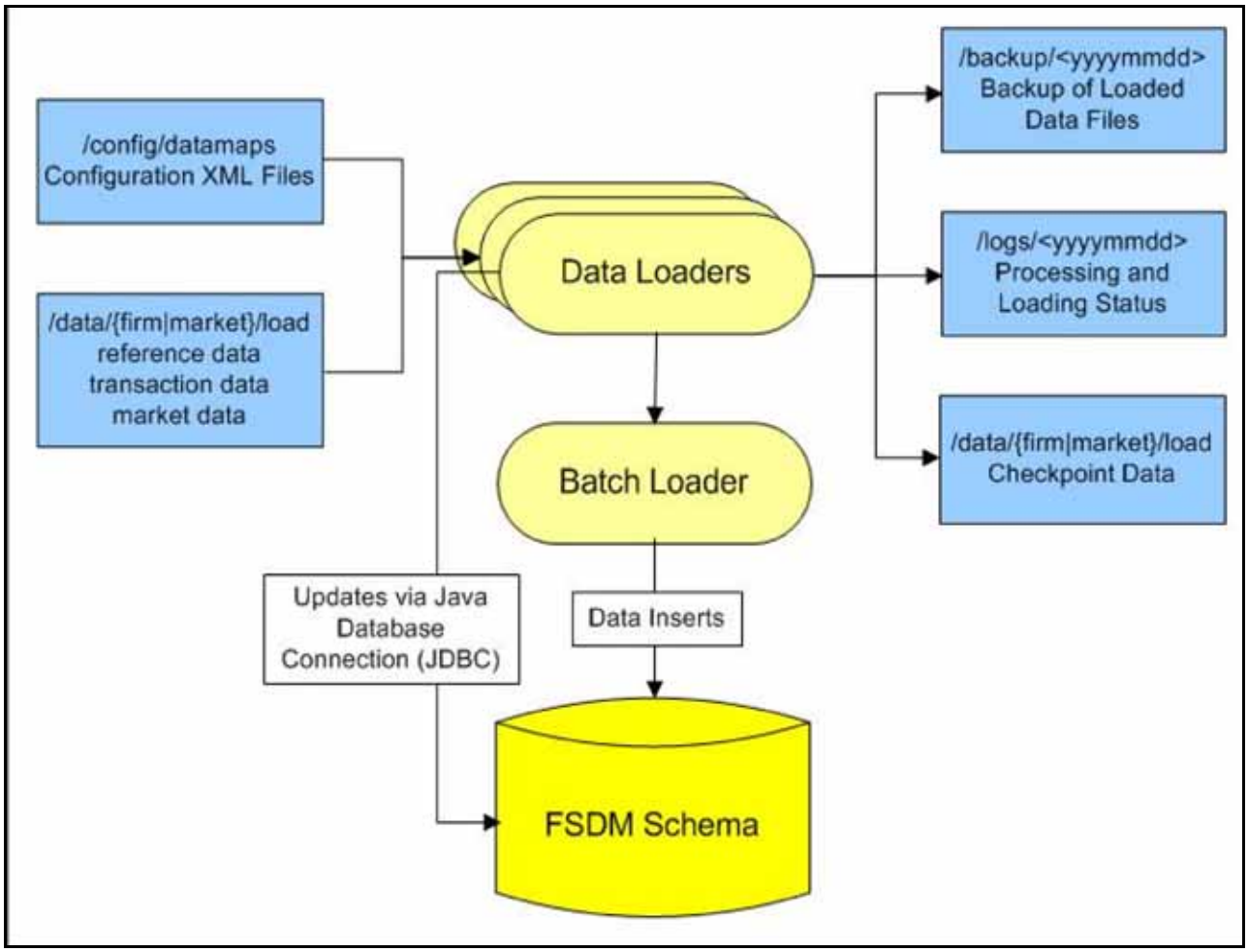

Figure 14. TC Data Loading Process

Guidelines for Duplicate Record Handling

The Ingestion Manager considers records as duplicates if the primary business key for multiple records are the same. The Ingestion Manager manages these records by performing either an insert or update of the database with the contents of the first duplicate record. The system inserts the record if a record is not currently in the database with

the same business key. The record updates the existing database record if one exists with the same business key. The Ingestion Manager handles additional input records with the same business key by performing database updates. Therefore, the final version of the record reflects the values that the last duplicate record contains.

Preprocessing Trading Compliance Solution Data

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

1. When Ingestion Manager satisfies dependencies from Group2 and preprocesses or loads the data in Group3, it executes the `runMDS.sh` script to process market data. This script invokes the Market Data server, which does the following:

- Supports input of market data in flat files.
- Assigns sequence numbers to market data records.
- Stores market data so that Firm Data Transformer (FDT) and Market Data Transformer (MDT) can retrieve it efficiently.

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

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runMDS.sh
```

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

2. After Ingestion Manager preprocesses and loads the data in Group 2, it executes the `runDP.sh` script to process TC trading data. This script invokes:

- Checking input trading data for errors
- Assigning sequence IDs to records
- Resolving cross-references to market reference data
- Checking for missing fields or attributes

When Ingestion Manager executes `runMDS.sh`, it places output files in the directories in Table 19.

Table 19. *runMDS.sh* and *runDP.sh* Output Directories

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

The following figure illustrates input and output directories for preprocessing market and trading data.

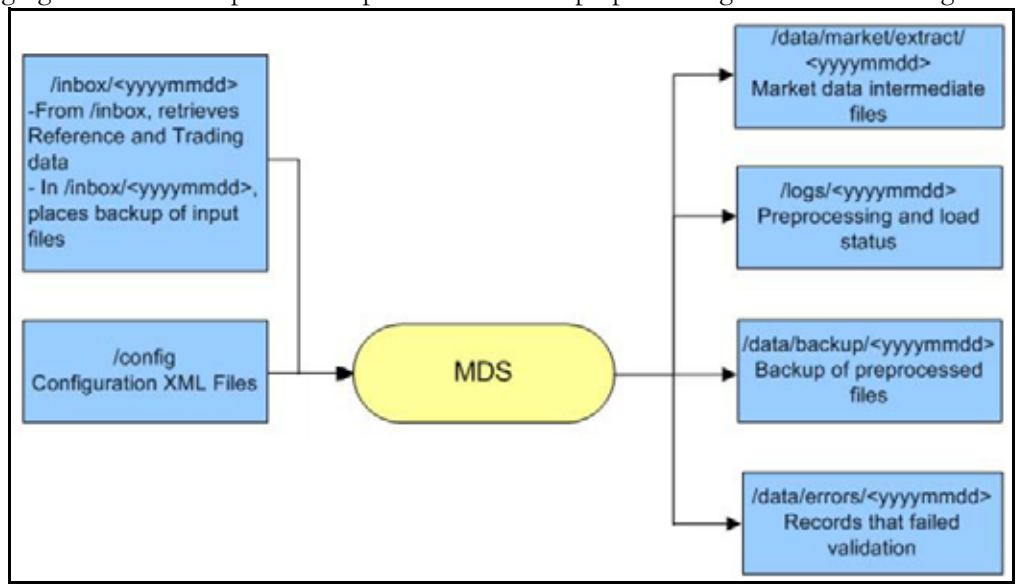

Figure 15. runMDS.sh Input and Output Directories

Preprocessing Alternative to the MDS

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

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runDP.sh InsideQuote
<OFSBDF Installed Directory>/ingestion_manager/scripts/runDP.sh MarketCenterQuote
<OFSBDF Installed Directory>/ingestion_manager/scripts/runDP.sh ReportedMarketSale
```

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

Processing Data through FDT and MDT

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

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

FDT Processing

During execution of the `runFDT.sh` script, Ingestion Manager processes trade-related data, orders and executions, and trades through the Firm Data Transformer, or FDT (Figure 16). The FDT does the following:

- Enriches data.
- Produces summary records for orders and trades.
- Calculates derived values to support detection needs.

- Derives state chains (that is, order life cycle states, marketability states, and displayability states).
- Provides data for loading into FSDM schema.

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

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

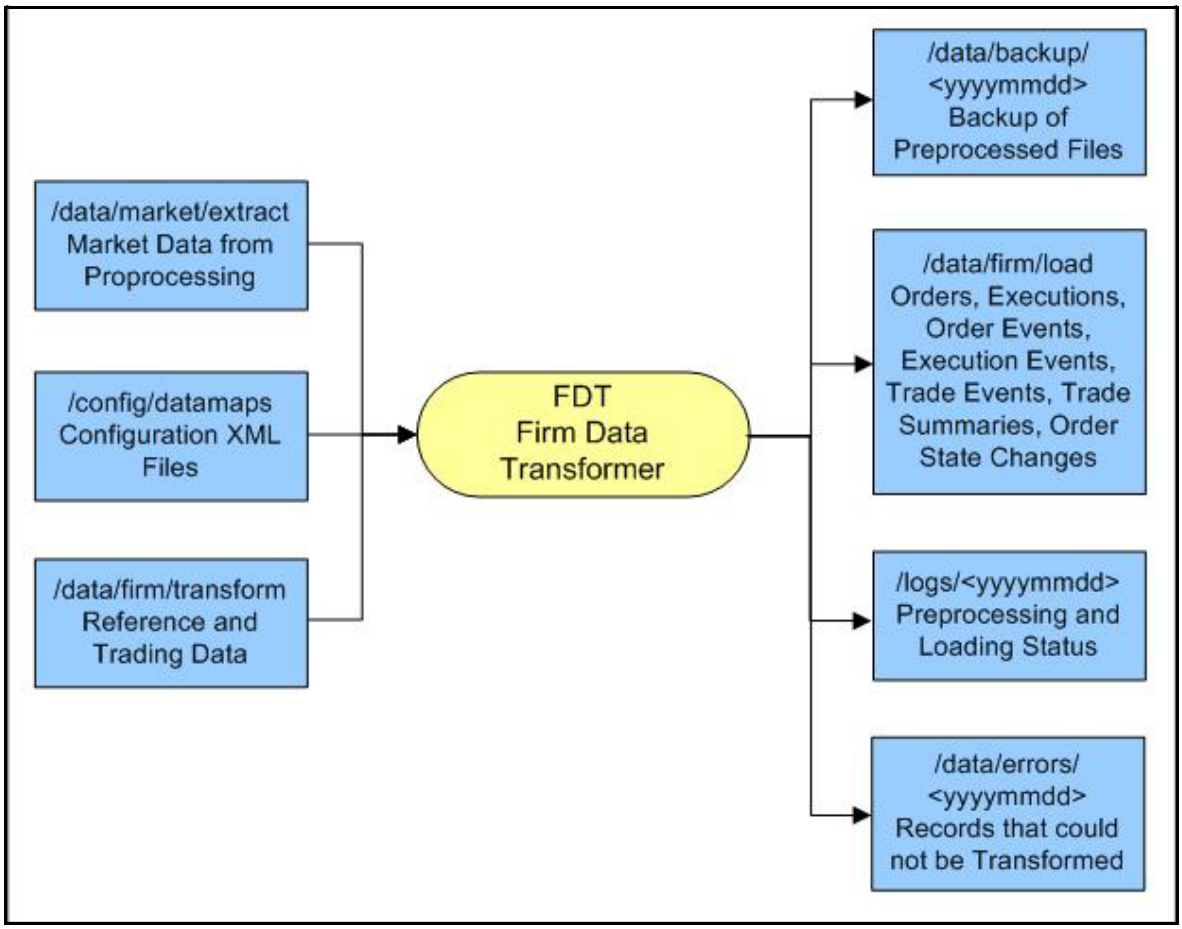

Figure 16. Firm Data Transformer (FDT) Processing

The FDT:

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

When Ingestion Manager executes `runFDT.sh`, it places output files in the directories in Table 20.

Table 20. `runFDT.sh` Output Directories

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

MDT Processing

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

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

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

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runMDT.sh
```

The following figure illustrates MDT data processing through `runMDT.sh`.

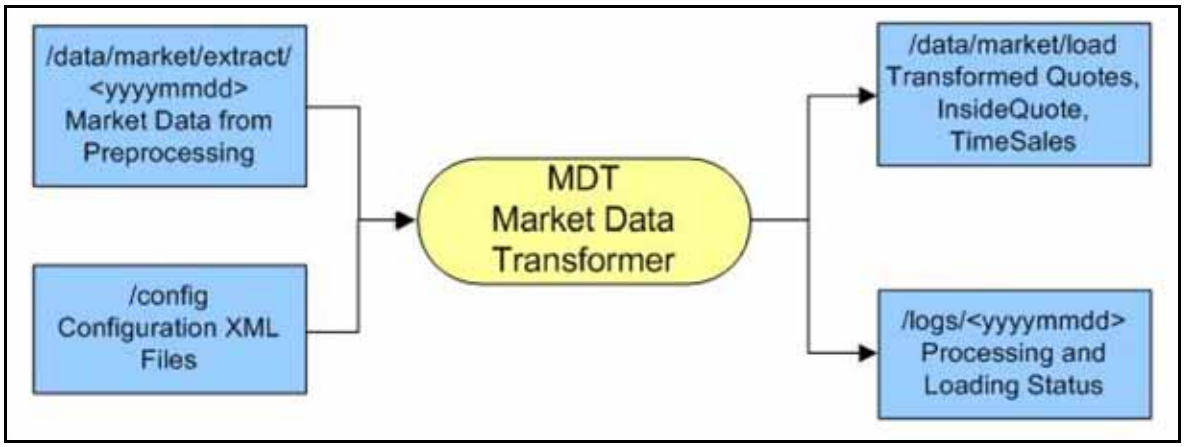

Figure 17. Market Data Transformer (MDT) Processing

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

Table 21. *runMDT.sh* **Output Directories**

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

Running Trading Compliance Solution Data Loaders

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

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

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

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

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

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runDL.sh Order
```

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

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

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

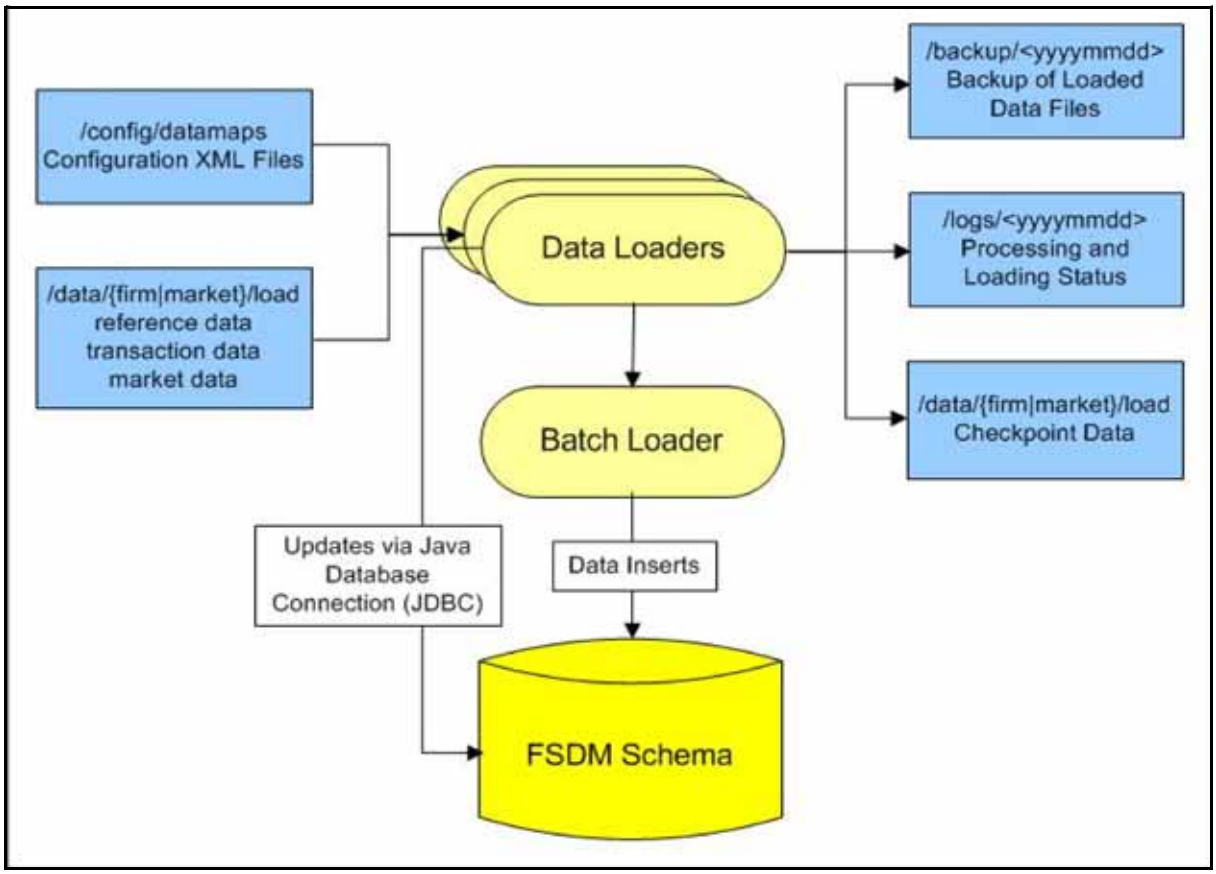

Figure 18. TCS Data Loading Process

Rebuilding and Analyzing Statistics

When TCS market data loading is complete, Ingestion Manager does the following (Figure 11 on page 36):

1. Rebuilds database indexes by executing the `runRebuildIndexes.sh` script for the loaded market data files (refer to section *Rebuilding Indexes* on page 49 for more information).
2. Analyzes data table characteristics (refer to section *Analyzing Statistics* on page 50 for more information).

The following sections describe these procedures.

Rebuilding Indexes

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

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

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runRebuildIndexes.sh
InsideQuote
```

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

For example:

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/firm_analyze.sh
```

Analyzing Statistics

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

Populating Market and Business Data Tables

To build and update trade and market summary data in the database, Ingestion Manager runs the `process_firm_summary.sh` and `process_market_summary.sh` scripts, as in Figure 11 on page 36.

The following examples illustrate execution of the scripts:

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/process_firm_summary.sh
<OFSBDF Installed Directory>/ingestion_manager/scripts/process_market_summary.sh
```

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

Intra-Day Ingestion Processing

The following figure provides a high-level flow of the intra-day ingestion process of extracting, transforming, and loading data.

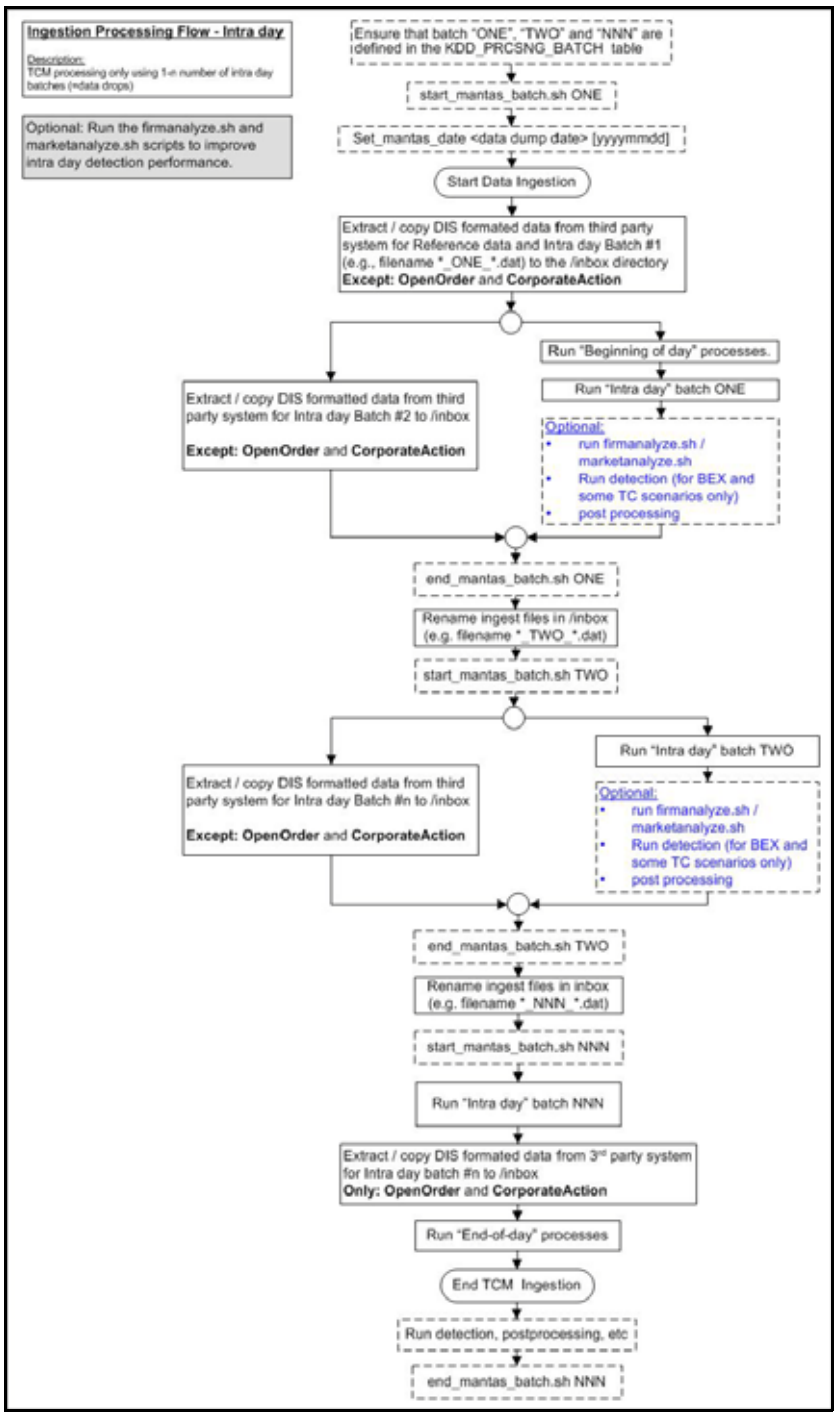

Figure 19. Intra-Day Data Ingestion Processing

Intra-day processing references different processing groups as Figure 19 illustrates (for example, beginning-of-day processing and intra-day processing as Figure 11 illustrates). Multiple batches run throughout the day. As in Figure 19, 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 22. 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

Alternatives to Standard Data Ingestion Practices

Table 23 describes alternatives to the following Data Ingestion processing options and provides advantages and disadvantages to each option:

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

Data Ingestion Directory Structure

The Data Ingestion subsystem components and data are organized in subdirectories below the `ingestion_manager` root level. Figure 20 illustrates the subdirectories that the `ingestion_manager` directory contains. Additionally, Table 24 provides details about each subdirectory.

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

Table 23. Data Ingestion Options and Alternatives

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

Table 23. Data Ingestion Options and Alternatives (Continued)

Typical Process	Alternative Process
Single Source Process that considers all data to reside in one set of source systems for the same time. Self-containment of data sustains referential integrity across data types. Data processing occurs in a single ingestion cycle. Advantages: ● Simplifies configuration. ● Makes processing easier to monitor than that for multiple batch processing. Disadvantages: ● Eliminates support of timely delivery of data from multiple geographic regions (for example, Asia Pacific or Europe). ● Requires more processing power for a smaller batch window. This window shrinks in multi-regional deployment, as batch does not start until data from all regions is available.	Multiple Sources Multiple processes that follow the same rules for self-containment of data that single source processing follows. For example, transactions in one source are unable to reference an account in another source. Processing of each batch of data occurs in sequential batches. Behavior Detection does not allow overlap of batches. Also, you cannot combine processing of "*multiple sources*" with intra-day batch processing. Advantages: ● Spreads processing of multiple geographic regions across an entire day (which contributes to fewer hardware requirements). ● Makes alerts from a particular region available in a more timely manner. Disadvantages: ● Makes processing more complicated to configure and monitor. ● Makes completion of processing of later batches in a timely manner more difficult when a delay of an earlier batch occurs.
Truncate Reference Data Process for overwriting non-transactional data (for example, account information, customer lists, security symbols, etc.) on a daily basis. Truncating the data tables before loading them provides a faster mechanism for data loading. The Oracle client must provide a complete set of reference data if using this approach. Advantages: ● Supports an Oracle client that does not track updating of daily records well. ● Provides increased performance when a large percentage of records change on a daily basis. Disadvantages: ● Makes tables unavailable from the time the system truncates the data until it loads the data. ● Causes decreased performance if only a small percentage of records change on a daily basis. ● Makes multi-batch processing unsuitable, although the truncating process can be used for periodic or beginning-of-day table refresh.	Update Reference Data Mechanism that provides Behavior Detection with a set of records for updating in place in a table. A look-up of every record in the input determines whether Behavior Detection needs to update or insert data. Advantages: ● Requires a minimal amount of updates for better performance. ● Supports multiple batch processing and high data availability. Disadvantages: ● Causes decreased performance if a large percentage of records require updating. ● Requires an Oracle client to determine what records have changed from day to day.

Table 23. Data Ingestion Options and Alternatives (Continued)

Typical Process	Alternative Process
Single Instance Mechanism by which a single instance of Data Ingestion software processes an input data stream. This configuration resides on one computer system. Advantages: ● Makes processing easier to configure and monitor. ● Simplifies the Oracle client data extraction process. Disadvantages: ● Limits performance as a result of the limitations of a single computer system, which results in a longer batch processing time.	Multiple Instances Mechanism by which configuration of multiple instances occurs on multiple computer systems. Each configuration processes a distinct data stream. Configuration spreads Data Ingestion instances across multiple computers. Advantages: ● Enables an Oracle client to scale for higher processing volume by using multiple computers. This increases bandwidth with the use of multiple I/O channels. ● Provides a lower cost structure with several smaller computers than with a single, large-capacity computer. Disadvantages: ● Makes configuration and monitoring of the ingestion process more difficult. ● Requires an Oracle client to split data prior to delivery for Data Ingestion.

The following sections describe the Data Ingestion directory structure.

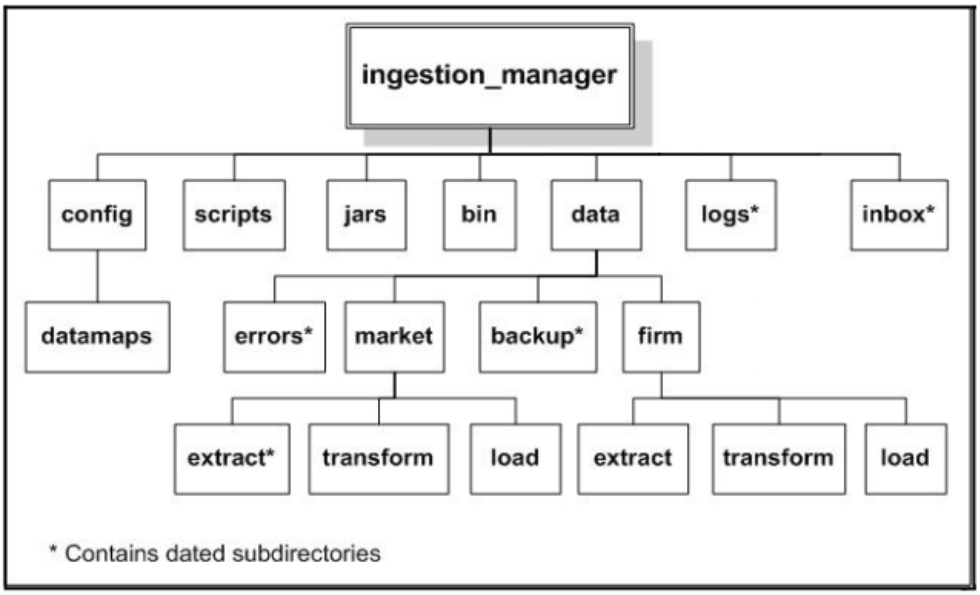

Figure 20. Data Ingestion Subsystem Directory Structure

Directory Structure Descriptions

The following table lists important subdirectories that compose the `<OFSBDF Installed Directory>/ingestion_manager` directory structure.

Table 24. Data Ingestion Directory Structure Description

Directory Name	Description
`bin`	Contains the programs that interface with the Market data feed to capture Market data and to stream that data to the MDS (refer to *bin Subdirectory* on page 56 for more information).
`config`	Contains files used to configure the Data Ingestion components (refer to *config Subdirectory* on page 58 for more information).
`data/backup`	Contains backup files for the various Data Ingestion components (refer to *data/backup Subdirectory* on page 75 for more information).
`data/errors`	Contains error files for various Data Ingestion components (refer to *data/errors Subdirectory* on page 73 for more information).
`data/firm`	Contains Oracle client data files that Data Ingestion components write (refer to *data/firm Subdirectory* on page 76 for more information).
`data/market`	Contains market data files that Data Ingestion components write (refer to *data/market Subdirectory* on page 74 for more information).
`inbox`	Contains data files that the Oracle client provides (refer to *inbox Subdirectory* on page 76 for more information).
`jars`	Contains the Java Archive (JAR) files to run Java Data Ingestion components implemented in Java (refer to *jars Subdirectory* on page 56 for more information).
`logs`	Contains log files that Data Ingestion components write (refer to *logs Subdirectory* on page 76 for more information).
`scripts`	Contains all the shell scripts for running Data Ingestion components (refer to *scripts Subdirectory* on page 56 for more information).

bin Subdirectory

The `bin` subdirectory within the `ingestion_manager` directory contains the programs that interface with the Market data feed to capture market and business client data and to stream that data to the Market Data server. A run script in the `scripts` subdirectory launches each program (refer to *scripts Subdirectory* on page 56 for more information).

jars Subdirectory

The `jars` subdirectory within the `ingestion_manager` directory contains Java programs that Ingestion Manager uses. A run script in the `scripts` subdirectory launches each program (refer to *scripts Subdirectory* on page 56 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 25. Run Scripts by Component

Script Names	Description or Special Instructions
`runMDS.sh`	Launches an instance of MDS (`runMDS.sh`). The preprocessor terminates after it finishes preprocessing the data that is currently in its memory.
`runDP.sh <data type>`	Launches an instance of the data preprocessor (`runDP.sh`). For example: runDP.sh Customer To run a specific Data Preprocessor, 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, refer to Figure 11 on page 36.
`runMDT.sh`	Launches the MDT (`runMDT.sh`). After receiving a soft-kill, the MDT terminates only when it has finished transforming all securities. You can stop the MDT immediately by using the UNIX **kill** command to stop the process ID for the Java process that is a child of the `runMDT.sh` script. **Note:** An MDT that you configure to run without polling (that is, batch mode) stops automatically when no more data remains for processing.
`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, refer to Figure 11 on page 36.
`runRebuildIndexes.sh <data type>`	Launches a process to rebuild the indexes of the given component. Processing requires this script only during use of a live market feed. A valid <component> value is one of InsideQuote, ReportedMarketSale, or MarketCenterQuote.
`process_firm_summary.sh`	Calls a database procedure to build summary statistics about the Oracle client (firm) data.
`process_market_summary.sh`	Calls a database procedure to build summary statistics about the Market data.
`market_analyze.sh`	Calls a database procedure to create internal database statistics for Market tables.
`firm_analyze.sh`	Calls a database procedure to create internal database statistics for Oracle client (firm) tables.
`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 Ingestion subsystem components. The `run*.sh` scripts use this script.

Table 25. Run Scripts by Component (Continued)

Script Names	Description or Special Instructions
`truncate_table.sh <schema.tablename>`	Truncates a specified table in the database. Processing runs this script prior to loading reference data when an Oracle client wants to perform a full refresh of the data.
`runUtility.sh <datatype>`	Launches a Java based utility to derive the contents of a given database table. You need to run `runDL.sh <data type>` after this script has executed successfully. For example: runUtility.sh AccountDailySecurityProfile runDL.sh AccountDailySecurityProfile

The run scripts in Table 25 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 26 describes.

Table 26. 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.

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

- For 64-bit operating systems, the maximum value should not be greater than 3500 MB.
- For 32-bit operating systems, the maximum value should not be greater than 1800 MB.

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

config Subdirectory

The `config` subdirectory within the `data_ingest` directory contains the application configuration files, as Table 27 describes:

- DataIngestCustom.xml (refer to section *Data Ingest Custom XML Configuration File* on page 59 for more information).
- DataIngest.properties (refer to section *Data Ingest Properties Configuration File* on page 59 for more information).
- DataIngest.xml (refer to section *Data Ingest XML Configuration File* on page 61 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 27. Application Configuration Files

File Name	Description
DataIngest.properties	Property file that contains settings that are configured at installation. These settings are of the most interest to an Oracle client regarding modification (refer to Table 28).
DataIngest.xml	XML configuration file that contains settings that normally remain as is (refer to Table 29).
DataIngestCustom.xml	XML configuration file that contains overridden settings from DataIngest.xml.

The following sections describe each of these configuration files.

Data Ingest Custom XML Configuration File

The client can modify the DataIngest.xml file to override default settings that the system provides. However, this file is subject to change in future OFSBDF releases. Therefore, upon installation of a newer OFSBDF 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 OFSBDF version, the client can copy the older DataIngestCustom.xml file to DataIngestCustom.xml in the new installation.

Data Ingest Properties Configuration File

The following table describes the parameters for the DataIngest.properties configuration file.

Table 28. 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 Ingestion requires for interacting with the database.	INGEST_USER
DB.Connection.Password	Password that Java Ingestion components use when connecting with the database. This is set by the Password Manager Utility.	

Table 28. DataIngest.properties File Configuration Parameters (Continued)

Property Name	Description	Example
DB.Type	The type of database being used.	Oracle
MANTAS.DBSchema	Schema name for the Mantas database schema. Typically, an Oracle client uses the default name of “MANTAS.” Data Ingestion accesses the MANTAS schema when allocating sequence IDs to ingested records.	MANTAS
MARKET.DBSchema	Schema name for the MARKET database schema. Typically, an Oracle client uses the default name of “MARKET.” Data Ingestion stores market data related records in the MARKET schema.	MARKET
BUSINESS.DBSchema	Schema name for the BUSINESS database schema. Typically, an Oracle client uses the default name of “BUSINESS.” Data Ingestion stores market data related records in the BUSINESS schema.	BUSINESS

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 29. DataIngest.xml File Configuration Parameters

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

Table 29. 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
MARKET: Specifies properties related to data loaded into the MARKET schema.		
MARKET.ExtractDir	Specifies the parent directory for directories where the MDS component stores intermediate market data files.	../data/market/extract
MARKET.TransformDir	Specifies the directory where the MDT component stores intermediate market data files.	../data/market/transform
MARKET.LoadDir	Identifies the parent directory for directories that store market data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	../data/market/load
BUSINESS: Specifies properties related to data loaded into the BUSINESS schema.		

Table 29. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
BUSINESS.ExtractDir	Identifies the parent directory for intermediate files that preprocessors produce that are applicable to the BUSINESS schema in the database.	../data/firm/extract
BUSINESS.TransformDir	Specifies the working directory for the FDT component which transforms BUSINESS trade-related data.	../data/firm/transform
BUSINESS.LoadDir	Indicates the parent directory for directories that store BUSINESS schema bound data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	../data/firm/load
MANTAS: Specifies properties related to data loaded into the MANTAS schema.		
MANTAS.ExtractDir	Specifies the parent directory for intermediate files that preprocessors produce that are applicable to the MANTAS schema in the database.	../data/mantas/extract
MANTAS.TransformDir	Specifies the working directory for intermediate files that utilities produce that are applicable to the MANTAS schema in the database.	../data/mantas/transform
MANTAS.LoadDir	Specifies the parent directory for directories that store MANTAS schema bound data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	../data/mantas/load
Directory: Specifies properties used to define directory locations.		
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

Table 29. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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`
`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 need to populate when manufacturing security records.	`SCRTY_SHRT_NM, SCRTY_ISIN_ID, PROD_CTGRY_CD, PROD_TYPE_CD, PROD_SUB_TYPE_CD`
Symbol: Specifies properties used for looking up security reference data by security short name.		
`Symbol.DbTableName`	Specifies the name of the database table to use when looking up security records by security short name.	`SCRTY`
`Symbol.KeyColumn`	Specifies the column name to use when looking up security records by security short name.	`SCRTY_SHRT_NM`
`Symbol.ValueColumn`	Specifies the column to use for retrieving the Behavior Detection assigned identifier for a security.	`SCRTY_INTRL_ID`
`Symbol.Category`	Specifies the category of data for the security table. The category is a key for mapping to the database schema in which the security table resides.	`BUSINESS`
SecurityISIN: Specifies properties used for looking up security ISINs.		

Table 29. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
SecurityISIN.DbTableName	Specifies the name of the table to use when looking up a security using the Behavior Detection assigned security identifier.	SCRTY
SecurityISIN.KeyColumn	Specifies the column name to use when looking up security records by Behavior Detection assigned security identifier.	SCRTY_INTRL_ID
SecurityISIN.ValueColumn	Specifies the column to retrieve when looking up a security using the Behavior Detection assigned security identifier.	SCRTY_ISIN_ID
SecurityISIN.Category	Specifies the category of data in which the security table resides. The category is a key for mapping to the database schema in which the security table resides.	BUSINESS
MDS: Specifies properties used to configure the MDS component.		
MDS.NumberOfThreads.value	Specifies the number of worker threads that the MDS uses when processing data.	4
MDS.BufferSize.value	Allows an override to the BufferSize.value property for MDS.	1024
MDS.DataFeed.RICExchangeCodes	Specifies a set of mappings of RIC exchange codes to Behavior Detection exchange codes.	N-XNYS,B-XBOS,C-XCIS, P-XPSE,T-XTHM,PH-XPHL, A-XASE,MW-XCHI
MDS.DataFeed.FeedExchangeCodes	Specifies a set of mappings of feed exchange codes to Behavior Detection exchange codes.	1-XASE,2-XNYS,3-XBOS, 4-XCIS,5-XPSE,6-XPHL, 7-XTHM,8-XCHI,43-XNAS
MDS.TimeInterval.value	Specifies the frequency in minutes with which MDS writes output data files when processing data from a live market feed.	10
MDS.CacheSize.value	Specifies the data cache byte size that the MDS uses.	1000000
MDS.RvSession.Timeout	Specifies the communication timeout value in seconds for the MDS when retrieving market summary information from a live market feed.	60
MDS.MarketHours.marketTimeZone	Specifies the time zone that the live market feed uses when reporting timestamps.	GMT
MDS.MarketHours.localTimeZone	Specifies the time zone that the local system uses.	EST
MDS.DailySummary.SubscriptionWait	Specifies the number of milliseconds to wait between subscription requests to the live market feed.	100
MDS.DailySummary.LastSubscriptionWait	Specifies the number of seconds to wait for a response for all subscription requests before rejecting subscriptions that have not received a response.	60

Table 29. DataIngest.xml File Configuration Parameters (Continued)

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

Table 29. DataIngest.xml File Configuration Parameters (Continued)

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

Table 29. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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 (refer to section *Rejection During the Transformation Stage*, on page 79, for a discussion of lost events) should be treated as errors (TRUE) or as lost events to be read in on the next run of FDT (false).	TRUE
FDT.OpenOrderFileExpected.value	Identifies whether an OpenOrder file will be provided by the client during an end of day batch (TRUE) or whether it will not be provided (FALSE).	TRUE
FDT.NonExecutionTradePurposeCodes.value	Specifies a comma-separated list of trade purpose codes. For Trade Execution records that refer to an Order and have one of these codes, the FDT will create a Trade record rather than an Execution record.	CLNT, BTBX
FDT.DeriveTradeBlotter.value	Specifies whether or not the FDT will create a TradeBlotter file.	FALSE

Table 29. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
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 Preprocessor (XDP) component.		
XDP.Default.ArchiveFlag	Specifies whether to archive data files. The system copies input files to the backup directory (TRUE) or deletes input files (FALSE).	TRUE
XDP.Default.ErrorLimit	Specifies the percentage of invalid records to allow before exiting with an error. For example, a value of 10 allows 10 percent of records to be invalid before exiting with an error. A value of 0 allows no invalid records. A value of 100 allows all invalid records.	100

Table 29. DataIngest.xml File Configuration Parameters (Continued)

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

Property Name	Description	Example
XDP.EmployeeTradingRestriction.DescendOrgTree	If an Employee Trading Restriction record contains an Organization Identifier, then it specifies: • Whether to create Employee Trading Restriction records for all employees in the organization and all the related child organizations defined in the Organization Relationship file (TRUE) or • Whether to create records only for employees in the specified organization (False).	FALSE
XDP.<Data Type>.<Property>	Overrides the given property for the given Preprocessor instance.	
XDL: Specifies properties used to configure the Data Loader (XDL) component.		
XDL.Default.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 (for example, 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 29. 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 29. 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 need to be added, the service needs to be re-initialized.	

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 preprocessing, 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 Ingestion subsystem components perform. The following sections describe the contents of each subdirectory and the components that read or write to each subdirectory.

data/errors Subdirectory

The `errors` subdirectory within the `data` subdirectory stores error files that Data Ingestion subsystem components create or move upon detection of errors during file processing. The system places error files in subdirectories within

the errors subdirectory. These error file subdirectories are name-based on the processing date for the files that they contain. The date has the format YYYYMMDD, where YYYY is the four-digit year, MM is the two-digit month, and DD is the two-digit day. The files in the errors subdirectory have the same name as the file in which the error was detected. However, the component that identified the errors appends its extension to the end of the file.

The following table identifies the error file signatures that each component can output to the errors subdirectory.

Table 30. Error File Signatures Output by Component

Component	Error File
Preprocessor	<data type>_*.XDP.err
Data Loader	<data type>_*.XDL.err
FDT	Order_*.FDT.err TradeExecution_*.FDT.err
MDS	InsideQuote_*.MDS.err MarketCenterQuote_*.MDS.err ReportedMarketSale_*.MDS.err

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

data/market Subdirectory

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

extract Subdirectory

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

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

transform Subdirectory

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

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

Table 31. Files Output by Market Data Transformer

Component	Output Data Files

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

load Subdirectory

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

The following table identifies the files that each data loader reads and loads into the database.

Table 32. Files that Market Data Loaders Read and Process

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

data/backup Subdirectory

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

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

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

Table 33. Backed Up Files by Component

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

data/firm Subdirectory

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

extract Subdirectory

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

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

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

transform Subdirectory

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

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

load Subdirectory

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

inbox Subdirectory

The `inbox` subdirectory within the `ingestion_manager` directory is an electronic mailbox or queue in which the Oracle client writes its data files for subsequent processing by Data Ingestion subsystem Data Preprocessor components. Each Market or Firm Data Preprocessor retrieves the file it is assigned to process from the `inbox` subdirectory and then moves the file to the appropriate extract subdirectory for preprocessing. The DIS describes the naming convention and content of each data 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 34. Log Files Output by Component

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

Startup and Shutdown

This section discusses Data Ingestion subsystem startup in both normal and recovery modes. You can start and stop all components manually with their respective run and stop scripts, with the exception of some components when configured to run in *batch* mode. Given the complexity of Data Ingestion processing, Oracle recommends that an Oracle client use a scheduling or monitoring tool to execute the run scripts (and stop scripts, if needed) automatically, monitor each component's progress, and alert support staff if problems arise. Scheduling or monitoring tools typically invoke a job control script that executes a Data Ingestion subsystem run and stop scripts. In addition, using the distributed processing configuration startup approach varies (refer to section *Distributed Processing Configuration* on page 78 for more information).

Backup Server Configuration

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

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

Recovery

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

Distributed Processing Configuration

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

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

Data Rejection During Ingestion

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

- **Rejection During Preprocessing Stage:** describes how rejections occur during the Preprocessing stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Preprocessing Stage* on page 78 for more information).
- **Rejection During Transformation Stage:** describes how rejections occur during the Transformation stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Transformation Stage* on page 79 for more information).
- **Rejection During Loading Stage:** describes how rejections occur during the Loading stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Loading Stage* on page 80 for more information).

Rejection During the Preprocessing Stage

The first stage of ingestion is Preprocessing. At this stage, Data Ingestion 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 Preprocessing include problems with data type, missing data, referential integrity, and domain values.

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

Data Type

Every field in a record that processing submits to the Ingestion Manager must meet the data type and length requirements that the DIS specifies. Otherwise, the process rejects the entire record. For example, fields with a *Date Type* must appear in the format YYYYMMDD. Thus, the date April 30, 2005 has a format of 20050430 and,

therefore, is unacceptable. In addition, a field cannot contain more characters or digits than specified. Thus, if an Order Identifier in an Order record contains more than the maximum allowed length of 40 characters, rejection of the entire record occurs.

Missing Data

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

Referential Integrity

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

Domain Values

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

Rejection During the Transformation Stage

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

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

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

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

- Lost Events
- Out of Sequence Events

The following sections describe these issues.

Lost Events

If the system receives an order event other than a New or Cancel or Replace in a set of files before receiving the corresponding New or Cancel or Replace, it writes the order event to a lost file. The system examines events in the lost file during processing of subsequent sets of files to determine whether the system received the corresponding New or Cancel or Replace event. If so, processing of this event is normal. If an event resides in the lost file when execution of open order processing occurs (that is, execution of `runDP.sh OPEN_ORDER`), processing rejects the

event. The same applies to trade execution events. In addition, if a New trade execution event references an order but the system did not receive the order, the New event also resides in the lost file subject to the same rules.

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

Out-of-Sequence Events

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

For example, File Set 1 contains the following events:

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

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

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

For example, File Set 1 contains the following events:

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

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

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

An out-of-sequence event has no effect on the order or trade that it references. Processing sets the out-of-sequence flag for the event to Y(es) and the system writes the event to the database. 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 (for example, setup of partitions are incorrect for the data being ingested).

Data Ingestion Archiving

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

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

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

Archiving Database Information

The database archiving process is explained in Figure 21.

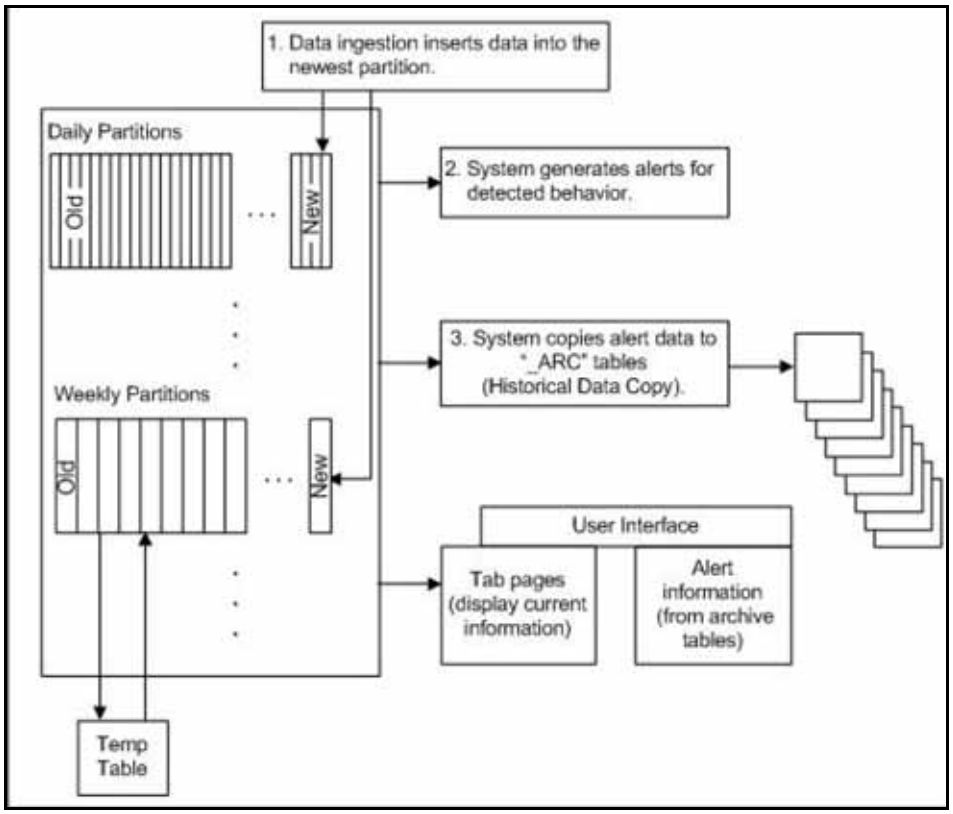

Figure 21. Database Archiving Process

The Data Ingestion subsystem uses the following procedure:

1. Processing places data in the newest partition of the partitioned tables.
2. Scenarios examine the data in the partitioned tables; the system then generates alerts for detected behaviors.

3. Historical Data Copy processing copies the information that generated alerts reference to the *_ARC* archive tables. The Platform UI displays alert information from the archive tables and information from the non-archived tables. This ensures that the alert information is in the same state as when the system generated the alert, while the most recent information is available to the user.

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

Miscellaneous Utilities

These utilities populate a single table in the data model. They should be executed after all the files in Table 17 have been loaded. A utility should not be executed until its predecessors have executed successfully.

Commands to execute:

```
<OFSBDF Installed Directory>/ingestion_manager/scripts/runUtility.sh <Utility Name>
```

```
<OFSBDF 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 35. Utilities

Utility Name	Table Name	Predecessor
EnergyAndCommodityFirmDailyDerived	EC_FIRM_DAILY	
EnergyAndCommodityMarketDailyDerived	EC_MARKET_DAILY	
EnergyAndCommodityTradeDerived	EC_TRADE	
EnergyFlow	ENERGY_FLOW	
MutualFundFamilyAccountPosition	MUTUAL_FUND_FAM_ACCT_POSN	
RegisteredRepresentativeCommissionProfile	RGSTD_REP_CMSN_SMRY	
RegisteredRepresentativeCommissionProductMixProfile	RGSTD_REP_CMSN_PRDCT_SMRY	
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.

Commands to Execute:

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

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

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

Example:

```
runUtility.sh AccountDailySecurityProfile
```

```
runDL.sh AccountDailySecurityProfile
```

Change Log Processing

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

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

Table 36. Change Log Parameters

Parameter	Description
`class`	The Java class name to perform change log processing. For the first type of processing defined above, `ChangeLog` should be used. For the second type, `SetChangeLog` should be used.
`seqBatchSize`	Specifies the batch size when retrieving sequence ids for creating ChangeLog records.
`enabled`	Specifies whether ChangeLog processing should be done for the given data type.

Table 36. Change Log Parameters (Continued)

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

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

ChangeLog records are produced when the preprocessor is run for the given data type (that is, `runDP.sh AccountAddress`). A ChangeLog file is produced with a name prefix containing the data type (that is, `ChangeLog_AccountAddress`). In order to load these ChangeLog files, the ChangeLog data loader is run (that is, `runDL.sh ChangeLog`). This data loader is run after all preprocessors that produce ChangeLog records are complete and any ChangeLog DIS files are preprocessed (i.e. `runDP.sh ChangeLog`).

Change Log Summary Processing

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

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

All of these scripts must be run after the ChangeLog data loader is run (refer to *Change Log Processing*, on page 83, for more information). The `AccountChangeLogSummary` loads the `ACCT_CHG_LOG_SMRY` table, the `CustomerChangeLogSummary` loads the `CUST_CHG_LOG_SMRY` table, and the `AccounttoCustomerChangeLogSummary` loads the `CUST_ACCT_CHG_LOG_SMRY` table.

Trade Blotter

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

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

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

```
runScoreTradeBlotter.sh
```

Refer to *Score Trade Blotter*, on page 193, for more information.

Trusted Pair

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

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

Note: OFSBDF 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.

Fuzzy Name Matcher Utility

During BDF 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

In addition to log configuration parameters, configuration information that the Fuzzy Name Matcher Utility requires for processingis available in the following locations:

- `<ingestion_manager>/fuzzy_match/mantas_cfg/install.cfg.`
- `<OFSBDF Installed Directory>/bdf/config/bdf.xml`

The Fuzzy Name Matcher Utility can be run manually using `fuzzy_match.sh` script. For more information, refer to section *Executing the Fuzzy Name Matcher Utility*, on page 88.

In order to process fuzzy name matching functionality through BDF Datamaps (`NameMatchStaging.xml,RegOToBorrower.xml`), these configuration parameters exists in bdf.xml (`<OFSBDF Installed Directory>/bdf/config/datamaps`).

The following figure provides sample configuration parameters.

```
#-------------------------------------------------------------
#                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 22. 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 37. 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 37. 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 Table Commands

Prior to running post-processing, you must execute database scripts after ingestion and prior to running AML scenarios. These scripts refresh the required temporary tables for selected AML scenario detection.

Use of Control Data

After installing the OFSBDF software, you can use control data provided to test end-to-end processing of data (that is, running data ingestion, 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 89 for more information).
2. Prepare for ingestion of the control data (refer to section *Control Data Ingestion* on page 89 for more information).
3. Install the control data (refer to section *Loading Control Data Thresholds* on page 90 for more information).
4. Run Behavior Detection on control data to generate alerts (refer to section *Running Behavior Detection on Control Data* on page 90 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 20091210.
3. Unless specified, set the current date as 20091210, to generate alerts on control data, before running Behavior Detection Framework.

Note: For more information about control data on your site, contact your OFSBDF Administrator.

Control Data Ingestion

Control data uses a specific set of dates to ensure that all the OFSBDF 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 38. Dates used by Control Data

20081231	**20091123**
20090130	**20091124**
20090227	**20091125**
20090331	**20091126**
20090430	**20091127**
20090529	**20091130**

Table 38. Dates used by Control Data

20090630	**20091203**
20090731	**20091204**
20090831	**20091208**
20090930	**20091209**
20091030	**20091210**
20091201	**20091202**
20091121	

On all these dates, run the complete Behavior Detection batch and ingest the data for the respective date. Except for Behavior Detection and Post-Processing tasks, perform all other activities for the Control Data ingestion dates. Activities required during any 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* on page 76 for more information).
2. Run ingestion for all the control data ingestion dates. Refer to section *Process Flow*, on page 34, for more information about the ingestion process.

Note: You need to adjust the partitions of the database tables as per the new dates, if you intend to process Control Data after the database upgrade to OFSBDF 6.2.1.

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 `<OFSBDF 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 5, *Behavior Detection Jobs*, on page 169 to get information regarding about running Behavior Detection Jobs.)

Important Notes

1. Run loaded scenarios with the system date as 20091210 with the following exceptions:
   a. For Portfolio Pumping scenario, the system date must be 20091204
   b. For Active Trading scenario, the system date must be 20091130
2. Check for system errors in the appropriate logs (refer to Appendix A, *Logging*, on page 309, for more information).
3. Run post-processing procedures.
4. Close the batch to enable display of alerts in the Behavior Detection UI.
5. Log in to the Behavior Detection UI with the correct user credentials.
6. Verify that you can view alerts in the UI.

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

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

Resetting the Environment

Once the testing is complete, you can reset the environment to its clean state by purging the test alerts (refer to section *Alert and Case Purge Utility*, on page 221, for more on purging the alerts) and deleting the control data threshold sets. The `delete_thresholds.sh` shell script helps you to delete control data thresholds from the environment.

The `delete_thresholds.sh` script:

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

CHAPTER 4 BDF Datamaps

This chapter describes the derivation and aggregation of data through datamaps in the BDF Datamap subsystem. After the OFSBDF Ingestion Manager loads the base tables (refer to Chapter 3, *Data Ingestion*, on page 33), the process of deriving and aggregating data begins. The Oracle client's job scheduling system invokes the BDF Datamap component that performs this data manipulation. It is the Oracle client's responsibility, in consultation with Oracle technical staff, to configure the job scheduling system for successful completion of this processing.

This chapter focuses on the following topics:

- About BDF Datamaps
- Structure of BDF Datamaps
- AML Brokerage Datamaps
- AML Banking Datamaps
- Broker Compliance Datamaps
- Fraud Detection Datamap
- Insurance Datamaps

Note: If you are migrating from Informatica to BDF Datamaps, please refer to *Administrative Utilities*, on page 269 for more information.

About BDF Datamaps

The BDF Datamap component is a simple ETL tool, responsible for taking data from one or more source database tables, transforming and enhancing it, and then loading it into a target database table.

The BDF Datamap component does the following:

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

Structure of BDF Datamaps

The BDF Datamap component is organized as subdirectories below the `<OFSBDF Installed Directory>/bdf`. Table 39 provides details about each subdirectory. .

Table 39. Directory Structure Description

Directory Name	Description	Use
`Scripts`	Contains shell scripts for running BDF components.	1. Change BDF Datamap component passwords. 2. View Environment settings. 3. Execute an individual datamap or schedule a batch job.
`Logs`	Contains log files that BDF components write.	View status and error messages once the BDF datamap or batch job has run.
`Config`	Contains files used to configure the BDF components.	Change the default configuration parameter in the Properties file.
`Jars`	Contains the Java Archive (JAR) files to run Java BDF components implemented in Java.	View all the java class files which are in use .
`data/errors`	Contains error files for the BDF Datamap component.	View records rejected during datamap processing.
`data/temp`	Contains Oracle temporary data files that BDF components write.	View temporary files before loading the data into the target file.
`Inbox`	Contains data files that the Oracle client provides.	This is for future purpose to process flat files as input for BDF Datamap processing.
`fuzzy_match`	Contains c++ library files used for fuzzy match algorithm	View supporting library files for the fuzzy match algorithm processing (for example, Watch list Name Match process)

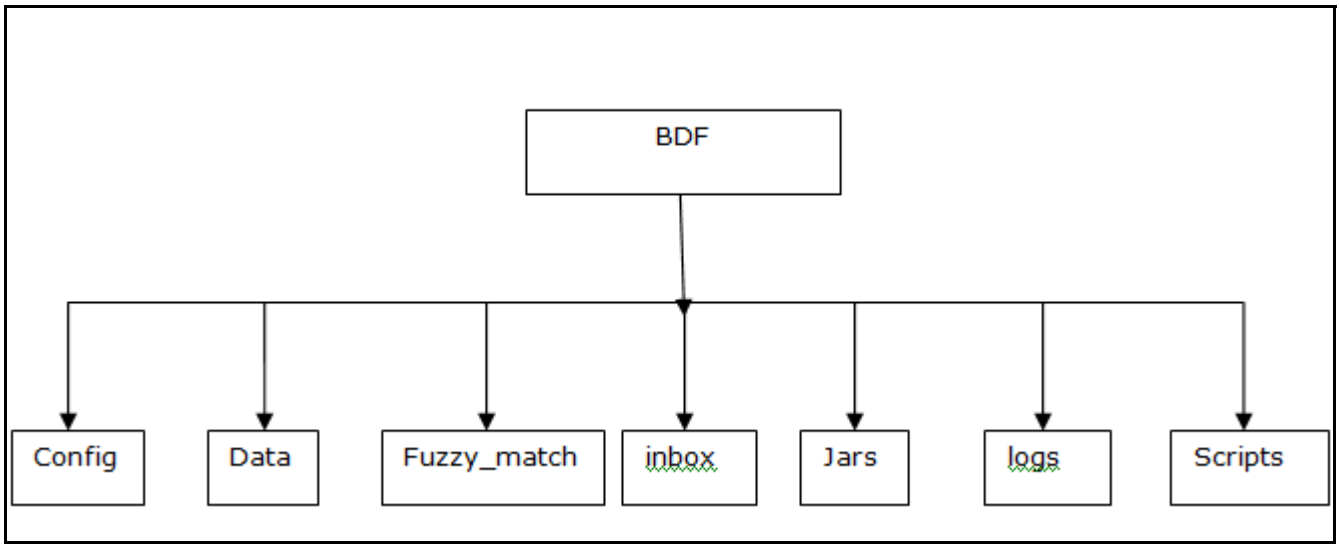

Figure 23. BDF Subsystem Directory Structure

The following sections describe the BDF directory structure.

Scripts

The scripts folder contains the following files:

- **changePassword.sh** - Used to change the password which is being used for processing Datamap. Refer to the *Installation Guide- Stage 1* for more information.
- **env.sh** - Contains the environment setting.
- **Execute.sh** - Used to execute any datamap which is under `<OFSBDF Installed Directory>/bdf/config/datamaps`.

  For Example:

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

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

Note: *Component* in this document means a batch process which is part of the BDF Datamap subsystem.

For the most part, these components will refer to XML data maps. For example, the AccountProfile_Balance component refers to the AccountProfile_Balance.xml data map.

Logs

The Log file has information about the warnings, errors, and status of the component.

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:

```
<OFSBDF Installed Directory>/bdf/logs/<processing date>/<component>.log
```

For example:

```
<OFSBDF 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.

For example:

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

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

Config

The config subdirectory within the BDF directory contains the datamap properties configuration files:

bdf.xml configuration parameters under `<OFSBDF Installed Directory>/bdf/config`, mentions the property name and default value. For more information, refer to section *BDF Datamap Properties Configuration File*, on page 96.

The BDF.xml install configuration parameter file under `<OFSBDF Installed Directory>/bdf/install/config`, mentioned the database connection details (database user IDs and encrypted passwords) and the parameters mentioned in Install Properties file.

The BDF.xml custom configuration parameter file under <OFSBDF Installed Directory>/bdf/config/custom , used to override the default value mentioned in `/bdf/config/BDF.xml`.

The `config/datamaps` subdirectory 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.

BDF Datamap Properties Configuration File

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

Table 40. bdf.xml File Configuration Parameters

Property Name	Description	Example
MISCELLANEOUS		
`NumberOfThreads`	Specifies the number of worker threads that the BDF uses when processing data.	`4`
`SequenceBatchSize`	Specifies the batch size when retrieving sequence IDs for BDF table records.	`100000`
`SourceSystem`	Indicates the default value to use for source system when processing reference data records.	`MTS`
`Currency`	Indicates the value to use as the issuing currency when processing records in BDF.	`USD`
`Separator`	Specifies the delimiter that separates fields in data file records.	`~`
DB: Specifies properties related to database access.		
`DB.Connection.Driver`	Indicates the JDBC driver class name.	`oracle.jdbc.OracleDriver`
`DB.Timeout`	Identifies the number of seconds to wait before timing out on a database connection attempt.	`10`
`DB.NumRetries`	Specifies the maximum number of times to attempt to connect to a database before failing.	`5`
`DB.MaxNumberOfDeadlocks`	Specifies the maximum number of times a deadlock in encountered during a JDBC insert or update operation, before an error is generated.	`10`
Directory: Specifies properties used to define directory locations.		
`Directory.Inbox`	Specifies the input directory where BDF 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.InternalData`	Specifies the directory where files generated by BDF components will reside. This includes log files, error files, and any temporary processing files.	`..`
Log: Specifies properties used to configure the common logging module		
`Log.Format`	Identifies the log formatting string.	`%d [%t] %p - %m%n`

Table 40. bdf.xml File Configuration Parameters

Property Name	Description	Example
Log.UseDefaultLog	Specifies whether the system uses the default log file for a component. The default log file has the name of the component and resides in a date subdirectory of the logs directory (in YYYYMMDD format).	true
Log.SysLogHostName	Indicates the host name of syslog for messages sent to syslog.	hostname
Log.SMTPHostName	Indicates the host name of the SMTP server for messages that processing sends to an e-mail address.	hostname
Log.MaxSize	Determines the maximum size (in MB) of a log file before the system creates a new log file.	2000MB
Log.MaxIndex	If a log file exceeds Log.MaxSize, this will be the maximum number of additional log files that are created (Component.log.1, Component.log.2, etc).	10
Log.TRACE.Enabled	Indicates that trace logging is not enabled; true indicates enabling of trace logging.	false
Log.TRACE.Location	Specifies additional locations to send TRACE log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BDF log location.	false
Log.TRACE.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.DIAGNOSTIC.Enabled	DIAGNOSTIC logging is used to log database statements and will slow down performance. Make it true if needed.	false
Log.DIAGNOSTIC.Location	Specifies additional locations to send DIAGNOSTIC log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BDF log location.	
Log.DIAGNOSTIC.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.NOTICE.Enabled	Indicates enabling of notice logging; false indicates that notice logging is not enabled.	true
Log.NOTICE.Location	Specifies additional locations to send NOTICE log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BDF log location.	
Log.NOTICE.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.WARN.Enabled	Indicates enabling of warning logging; false indicates that warning logging is not enabled.	true
Log.WARN.Location	Specifies additional locations to send WARN log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log).	
Log.WARN.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.FATAL.Enabled	Indicates enabling of Fatal logging; false indicates that fatal logging is not enabled.	true

Table 40. bdf.xml File Configuration Parameters

Property Name	Description	Example
Log.FATAL.Location	Specifies additional locations to send FATAL log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log).	
Log.FATAL.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Load: Specifies Properties used to configure common Loading data		
Load.FullRefresh	This parameter controls the use of full refresh or delta mode. When this parameter is true, the system uses full refresh mode; when it is false, the system uses delta mode.	false
Load.BatchSize	Specifies the batch size when loading data.	5000
Load.Direct	Specifies whether to use direct path loading (TRUE) or conventional path loading (FALSE).	false
Load.Unrecoverable	Specifies whether a direct path load does not use redo logs (TRUE) or uses redo logs (FALSE).	false
Load.Partitioned	Specifies whether a direct path load uses the current date partition (TRUE) or any partition (FALSE).	false
Load.SkipIndexes	Specifies whether a direct path load skips index maintenance (TRUE) or maintains indexes (FALSE). If set to TRUE, rebuilding of indexes must occur after running the DataMap XML.	false
Load.DoAnalyze	Specifies whether to run a stored procedure to analyze a database table after loading data into it.	true
Load.AnalyzeType	Specifies the type of analyze statistics has to perform if DoAnalyze has a value of True.	DLY_POST_LOAD
Load.LogRecordInterval	Specifies how often to log a message saying how many records a particular thread has inserted/updated,	1000
Load.MaxErrorRate	Specifies the percentage of invalid records to allow before exiting with an error. For example, a value of 10 allows 10 percent of records to be invalid before exiting with an error. A value of 0 allows no invalid records. A value of 100 allows all invalid records.	100
Load.RecordQueueSize	Specifies the number of records the query reader thread will write to a database writer thread queue before waiting for the reader thread to catch up. Higher values will require more memory usage.	100
Load.SkipIndexesErrorCode	Specifies a database error code that occurs in the log file when skipping index maintenance.	26025
Load.IndexParallelLevel	Specifies the parallel level of an index rebuild (that is, number of concurrent threads for rebuilding an index).	1
Load.DataErrorCodes	Specifies a comma-separated list of database error codes that indicate data level errors (for example, 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

Table 40. bdf.xml File Configuration Parameters

Property Name	Description	Example
Load.ParallelLevel	Specifies the level of parallelization to apply when loading data from a set of source tables to a target table.	8
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 (for example, 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
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,I BAN,ABLZ,BSB,CP AP, SDIC, HEBIC, BCHH, NSC, IFSC, IDIC, PNCC, RCBIC, UKDSC, Swiss BC, Swiss SIC,IA

Table 40. bdf.xml File Configuration Parameters

Property Name	Description	Example
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 include IA (Internal Account Identifier) and is majorly 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,I BAN,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
TrustedPairReviewReasonText1	Defines the reason text1 for recommendation of cancelling the Trusted Pair, due to increase in Risk of parties involved in trusted pair.	Risk of <Party1> increased from <A> to <b>
TrustedPairReviewReasonText2	Defines the reason text2 for recommendation of cancelling the Trusted Pair, due to increase in Risk of parties involved in trusted pair.	Risk of <Party2> increased from <C> to <D>
CorporateActionLookBackDays	This parameter determines the how many days trades to look back from the Corporate Effective Date.	7

Table 40. bdf.xml File Configuration Parameters

Property Name	Description	Example
DealNearTermMaturityDays	Defines the maximum number of days between the End Date and Trade Date. This helps to calculate Structured Deals Initiated w/ Near-Term Exp. In Customer Profile/ Institutional Account Profile.	7
ProfitLossUpperLimit	Helps determine how much a security must move by the end of the day to be considered a win or loss. If the security moves by less than a specified percentage, processing does not count it either way. If it moves by this percentage or more, it counts as a win or a loss, depending on whether the movement was beneficial to the account that made the trade.	5
HouseholdTurnOverRateUpperLimit	Defines the upper limit for Total Turnover Rate in Household Profile Calculation.	10000
HouseholdCommissionEquityPercentUpperLimit	Defines the upper limit for Commission Versus Average Daily Equity Percentage in Account Profile Calculation.	10000
OptionTradeAmountRange1 OptionTradeAmountRange2 OptionTradeAmountRange3 OptionTradeAmountRange4 OptionTradeAmountRange5 OptionTradeAmountRange6	Define the lower bound of each range for the Account Profile attributes named Options Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount- Base. Each range is from the lower bound entered here to the lower bound of the next range.	
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.	
LowPricedEquityTradeAmountRange1 LowPricedEquityTradeAmountRange2 LowPricedEquityTradeAmountRange3 LowPricedEquityTradeAmountRange4 LowPricedEquityTradeAmountRange5 LowPricedEquityTradeAmountRange6	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 40. bdf.xml File Configuration Parameters

Property 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

If any datamap errors out, or if debugging requires SQL then set the *Trace and Diagnostic* levels as `true` before running the datamap again.

Open `<OFSBDF Installed Directory>/bdf/config/BDF.XML` and set the following log levels as *Enable* for debugging purpose. Change the following values:

From:

```
<Parameter name="Log.TRACE.Enabled" type ="BOOLEAN" value="false"/>
```

To:

```
<Parameter name="Log.TRACE.Enabled" type ="BOOLEAN" value="true"/>
```

From:

```
<Parameter name="Log.DIAGNOSTIC.Enabled" type ="BOOLEAN" value="false"/>
```

To:

```
<Parameter name="Log.DIAGNOSTIC.Location" type ="STRING" value="true"/>
```

Oracle recommends keeping these levels of logging as `false` to avoid any performance issue.

For parameters pertaining to FSDF data loading, please refer to Appendix G, *Ingestion Using FSDF Datamaps*, on page 357.

Note: AML specific files which have been created for FSDF (see Appendix G, *Ingestion Using FSDF Datamaps*, on page 357), can be run using source as Flat files. The following parameter has to be changed for this:

<Parameter Name =”DIS.Source” type=”STRING” value=”FILE”/>

To run Change Log Process, the following files have to run as part of runDP and runDL:

CustomerPhone, AccountToCustomer, CustomerAddress, and AccountAddress

BDF Datamap Configuration File

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

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

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

- AccountTrustFromCustomer
- DefaultJurisdiction
- UseTaxidForUnrelatedPartyCode
- BaseCountry
- ProcessForeignFlag
- ProcessBankToBank
- ProcessTransactionXRefFlag
- TrustedPairRiskReviewFlag

These parameters are stored in the following file:

`<OFSBDF Installed Directory>/bdf/install/config/ bdf.xml`

These parameters are defined in Install Properties. Refer to the *Installation Guide- Stage 1* for more information.

The following table describes the parameters defined in Install Properties:

Table 41. BDF Datamap Configuration Parameter

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

Table 41. BDF Datamap Configuration Parameter

Property Name	Description	Example
DB.Schema.CONFIG	Name of the configuration schema owner.	REVELEUS
DB.Schema.CASE	Name of the case schema owner.	CMREVMAN
DB.Alg.Connection.User	Database user for running Behavior Detection post-processing jobs.	KDD_ALG
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 the following:

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

And can also be created in the following:

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

Parameters

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

```
<OFSBDF 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 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:

```
    <OFSBDF Installed Directory>/bdf/scripts/execute.sh <component> [parameter
name=value]*
```

For example:

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

BDF Datamap Types

The Oracle client's solution determines the required BDF datamaps, or a subset thereof:

- AML Brokerage (refer to *AML Brokerage Datamaps* on page 106, for more information).
- AML Banking (refer to *AML Banking Datamaps* on page 122, for more information).
- Broker Compliance (refer to *Broker Compliance Datamaps* on page 136, for more information).
- Fraud Detection (refer to *Fraud Detection Datamap* on page 140, for more information).
- Insurance (refer to *Insurance Datamaps* on page 154, for more information).

Caution: If you are running multiple solutions, you must perform table comparisons to avoid running duplicate datamaps. Refer to Appendix D, *Datamaps Matrix*, on page 341 for more information.

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

Table 42. 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. Refer to each section for a complete list of required categories.

Datamap Processing

This chapter 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 ingestion, 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 (for example, Summary datamaps dependent on watchlist datamaps).

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

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

`FrontOfficeTransactionParty_InstnSeqID`

`FrontOfficeTransactionParty_HoldingInstnSeqID`

In case, there is any performance issue with the datamap `FrontOfficeTransactionParty_HoldingInstnSeqID`, datamap position can be rearranged in the batch script.

Since there is possiblity that previous process (`FrontOfficeTransactionParty_InstnSeqID`) is still running, So 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 AccountProfile_Trade datamap until and unless 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.

Note: Oracle recommends all datamaps are run in the order described below.

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.

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 43. 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
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

Table 43. AML Brokerage - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 43. AML Brokerage - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10230	AccountToClientBank_SettlementInstruction	10020 10030 10040 10050 10060 10070 10220
10240	SettlementInstruction_AccountToClientBank	10020 10030 10040 10050 10060 10070 10230

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 44. 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

Table 44. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
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
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

Table 44. AML Brokerage - 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 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

Table 44. AML Brokerage - Watch List Datamaps (Continued)

Datamap Number.	Datamap Name	Predecessors
10540	DerivedEntitytoDerivedAddress_LinkStage	10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	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 44. AML Brokerage - 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 44. AML Brokerage - 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 44. AML Brokerage - 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 44. AML Brokerage - 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 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

Table 44. AML Brokerage - 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
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 need to run datamap 10320 and 10880.

- OFSBDF AML and KYC
- OFSBDF Fraud and KYC
- OFSBDF 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 Cash, Wire, Monetary Instruments tables, also these are used to update Trusted pair information, Jurisdiction into various other entities. Table 45 describes the Post-Watch List datamaps for AML Brokerage.

Customers 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)

Configuring Risk Zones

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.

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 `<OFSBDF 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.

Table 45. AML Brokerage - Post Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10960	AccountGroup_JurisdictionUpd	NA
10970	TransactionPartyCrossReference_BackOfficeTransaction	10360 10370 10380 10940 10950
10980	CashTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
11000	TransactionPartyCrossReference_FrontOfficeTransaction	10360 10370 10380 10940 10950 11060 11070 11080 11090
11010	WireTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	10360 10370 10380 10940 10950

Table 45. AML Brokerage - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 45. AML Brokerage - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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 either daily, weekly or monthly depending on the business areas.

Table 46. 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 10950 11160 11170
11190	AccountProfile_Transaction	10940 10950 11160 11170
11200	AccountProfile_Stage	NA

Table 46. AML Brokerage - Summary Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
11240	AccountPeerGroupMonthlyTransactionProfile	11180 11190 11200 11210 11220

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 47. AML Brokerage - Balances and Positions Datamaps

Datamap Number	Datamap Name	Predecessors
50170	CustomerBalance_ActiveOTCTradeCtUpd	NA

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

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 48. 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
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

Table 48. AML Banking - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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

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 49. 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

Table 49. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 49. AML Banking - 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 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

Table 49. AML Banking - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
10540	DerivedEntitytoDerivedAddress_LinkStage	10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	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 49. AML Banking - 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 49. AML Banking - 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 49. AML Banking - 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 49. AML Banking - 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 10701

Table 49. AML Banking - 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
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 Cash, Wire, Monetary Instruments tables, also these are used to update Trusted pair information, Jurisdiction into various other entities.

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

Table 50. 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 10950
10980	CashTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
11000	TransactionPartyCrossReference_FrontOfficeTransaction	10360 10370 10380 10940 10950
11010	WireTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	10360 10370 10380 10940 10950
11030	CashTransaction_FrontOfficeTransactionRevAdj	10970 10980 10990 11000 11010 11020

Table 50. AML Banking - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 50. AML Banking - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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 51. 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 10950 11160 11170 11180 11190 11210

Table 51. AML Banking - Summary Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
20040	CorrespondentBankProfile	11180 11190 11200 11210 11220
20050	AccountATMDailyProfile	10940 10950
11230	ChangeLog_AcctProfileInactivity	11180 11190 11200 11210 11220
11240	AccountPeerGroupMonthlyTransactionProfile	11180 11190 11200 11210 11220
20060	CorrespondentBankPeerGroupTransactionProfile	20040
20070	AccountChannelWeeklyProfile	10940 10950

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 - Balances and Positions Datamaps
- Broker Compliance - Summary 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 52. 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
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

Table 52. Broker Compliance - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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

Broker Compliance - Post-Watch List Datamaps

Oracle recommends that datamaps are run in the order described below.

Table 53. 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 54. 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

Table 54. Broker Compliance - Balances and Positions Datamaps

Datamap Number	Datamap Name	Predecessors
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 55. Broker Compliance - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
60040	UncoveredOptionExposureDaily	NA
60050	InvestmentAdvisorProfile	NA
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

Table 55. Broker Compliance - Summary Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
60130	HouseholdProfile	11180 11190 11200 11210 11220
60070	RegOToBorrower (*Optional)*	NA
60080	InterestedPartyToEmployee (*Optional)*	NA
60140	ManagedAccountProfile	60100 60110 60120

Fraud Detection Datamap

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

Note: Oracle recommends that datamaps are run in the order described below.

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 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 56. 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

Table 56. Fraud Detection - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 56. Fraud Detection - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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

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 57. 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

Table 57. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 57. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 57. Fraud Detection - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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 57. Fraud Detection - 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 57. Fraud Detection - 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 57. Fraud Detection - 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 57. Fraud Detection - 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 57. Fraud Detection - 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
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 information, Jurisdiction 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 58. Fraud Detection - Post Watch List Datamaps

Datamap Number	Datamap Name	Predecessors
10960	AccountGroup_JurisdictionUpd	NA
10970	TransactionPartyCrossReference_BackOfficeTransaction	10360 10370 10380 10940 10950
10980	CashTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
10990	MonetaryInstrumentTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
11000	TransactionPartyCrossReference_FrontOfficeTransaction	10360 10370 10380 10940 10950 11060 11070 11080 11090
11010	WireTransaction_FrontOfficeTransaction	10360 10370 10380 10940 10950
11020	WireTransactionInstitutionLeg_FrontOfficeTransaction	10360 10370 10380 10940 10950

Table 58. Fraud Detection - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 58. Fraud Detection - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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 59. Fraud Detection - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
11160	AccountDailyProfile-Trade	NA
11170	AccountDailyProfile-Transaction	10940 10950
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

Table 59. Fraud Detection - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
11220	AccountProfile_Balance	11180 11190 11210
11230	ChangeLog_AcctProfileInactivity	11180 11190 11200 11210 11220
11240	AccountPeerGroupMonthlyTransactionProfile	11180 11190 11200 11210 11220

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

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 60. 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

Table 60. Insurance - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 60. Insurance - Pre-Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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 finally 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 61. 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
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

Table 61. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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

Table 61. Insurance - Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
10530	DerivedEntityLink_LinkStage	40090 10490 10500
10540	DerivedEntitytoDerivedAddress_LinkStage	40090 10490 10500
10550	DerivedEntitytoInternalAccount_LinkStage	40090 10490 10500
10560	DerivedAddresstoInternalAccount_LinkStage	40090 10490 10500

Table 61. Insurance - 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 61. Insurance - 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 61. Insurance - 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 61. Insurance - 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 61. Insurance - 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

Table 61. Insurance - 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
10940	FrontOfficeTransactionPartyRiskStage_EntityActivityRiskInsert	10890 10900
40100	InsuranceTransaction_EntityAcctivityRiskUpd	10890 10900

Insurance - 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, also these are used to update Trusted pair information, Jurisdiction into various other entities.

Table 62. 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

Table 62. Insurance - Post Watch List Datamaps (Continued)

Datamap Number	Datamap Name	Predecessors
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
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 63. Insurance - Summary Datamaps

Datamap Number	Datamap Name	Predecessors
40110	InsurancePolicyDailyProfile_InsTrxnInsPolicyBal	NA
40120	InsurancePolicyProfile_InsurancePolicyDailyProfile	40110

CHAPTER 5 Behavior Detection Jobs

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

- About the OFSBDF Job Protocol
- Performing dispatcher Tasks
- Performing Job Tasks
- Clearing Out the System Logs
- Recovering Jobs from a System Crash

About the OFSBDF Job Protocol

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

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

As an OFSBDF administrator, you invoke the `dispatcher` and `mantas` processes by running the shell scripts in Table 64

Table 64. Shell scripts.

Process	Description
`start_mantas.sh`	Starts all OFSBDF 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 OFSBDF Job Protocol, the processes use a variety of metadata that the OFSBDF database provides. Some of this metadata specifies the jobs and their parameters that are associated with the regular operations of an OFSBDF installation. Some of this metadata captures the status of job execution and is useful for monitoring the progress of an OFSBDF operational cycle.

The following sections describe how the processes and metadata interact in the OFSBDF Job Protocol.

Understanding the OFSBDF Job Protocol

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

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

Table 65. KDD_JOB_TEMPLATE with Sample Job Template Group

JOB_ID	TEMPLATE_GROUP_ID
37	1
41	1

Understanding the dispatcher Process

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

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

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

Refer to *Starting the dispatcher* on page 172 and *Stopping the dispatcher* on page 173 for more information.

Understanding the mantas Process

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

The mantas process also logs job progress and final status.

Applying a Dataset Override

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

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

The following constraints apply to overriding a dataset:

- The columns returned by the dataset override must be identical to those returned by the product dataset. Therefore, the dataset override does not support returning different columns for a pattern customization to use.

- The dataset override can use fewer thresholds than the product dataset, but cannot have more thresholds than the product dataset. Only thresholds applied in the dataset from the scenario are applied.

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

To Configure the Dataset Override Feature

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

To configure a dataset override, follow the 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:

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

---

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

   `dataset<DATASET_ID>.txt`

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

Performing dispatcher Tasks

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

This section describes the following:

- Setting Environment Variables
- Starting the dispatcher
- Stopping the dispatcher
- Monitoring the dispatcher

Setting Environment Variables

Environment variables are set up during the OFSBDF 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 <OFSBDF Installed Directory>/behavior_detection/algorithms/MTS/share

Table 66. OFSBDF Environment Variables in system.env File

Variable	Description
KDD_HOME	Install path of the OFSBDF 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 67. 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 68. Operating System Environment Variables in system.env File

Variable	Description
PATH	Augmented to include <OFSBDF 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 <OFSBDF 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 OFSBDF, only one dispatcher service per database per installation should run at one time.

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

To Start the dispatcher

To start the dispatcher, follow the steps:

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

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

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

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

   Optional parameters include the following:

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

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

Stopping the dispatcher

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

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

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

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

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

To Stop the dispatcher

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

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

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

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

   This script shuts down the dispatcher.

Monitoring the dispatcher

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

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

To Monitor the dispatcher

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

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

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

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

Performing Job Tasks

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

The sections below cover the following topics:

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

Understanding the Job Status Codes

The following status codes are applicable to job processing and the `dispatcher`. The OFSBDF administrator sets these codes through an OFSBDF Job Editor:

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

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

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

Starting Jobs

The OFSBDF administrator starts jobs by running the start_mantas.sh script.

To Start a New Job

To start a new job in OFSBDF, follow the steps:

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

   OFSBDF 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 OFSBDF Job Protocol* on page 170 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 OFSBDF metadata tables. Each job consists of one or more runs.
4. The mantas process handles the job runs.

After a job runs successfully in OFSBDF, 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 OFSBDF database must run the jobs without dispatcher processes. If the client does use dispatchers on each machine, each dispatcher may run each job, which causes duplicate detection results.

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

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

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 Job without the dispatcher

To start a new job in Behavior Detection Framework without the **`dispatcher`**, follow the steps:

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

   OFSBDF 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 OFSBDF, 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 OFSBDF 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

To restart a job, follow the steps:

Note: If the `dispatcher` has stopped, restart it.

---

1. Type `restart_mantas.sh <template group id>` at the system prompt.
2. Press **Enter**.

   When the `dispatcher` picks up a job from the job queue that has a code of RES, it automatically restarts the job (Refer to section *Starting Jobs* on page 175 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 (for example, `restart_mantas.sh <template group id> <DD-MON-YYYY>`).

Restarting Jobs without the dispatcher

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

To Restart a Job without the dispatcher

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

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

Stopping Jobs

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

To Stop a Job

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

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

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

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

   The `mantas` process ID number appears in the first display line in the second column from the left (7800). The job ID number appears in the second display line in the last column (-j 123).

2. Find the job and `mantas` process ID that you want to stop.
3. Type **`kill <mantas process ID>`** at the system prompt and press **Enter**.

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

Monitoring and Diagnosing Jobs

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

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

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

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

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

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

To Monitor a Job

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

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

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

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

Clearing Out the System Logs

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

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

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

Clearing the Dispatch Log

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

1. Shut down the dispatcher by following the procedure for Stopping the dispatcher (Refer to section *Stopping the dispatcher* on page 173 for more information).
2. Type cd <$KDD_PRODUCT_HOME>/logs at the system prompt, where <$KDD_PRODUCT_HOME> is your product server installation directory.
3. Type rm dispatch.log to clear the dispatcher log.
4. Type **start_chkdisp.sh <sleep time>** and press **Enter** to restart the dispatcher.

Clearing the Job Logs

To clear the job logs, follow the steps:

1. Stop the dispatcher by following the procedure for Stopping the dispatcher (Refer to section *Stopping the dispatcher* on page 173 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 <OFSBDF 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-MON-YYYY format. The default value is the current date. Running the recover_mantas.sh script with this parameter ensures the script recovers only the jobs started that day. The dispatcher must be running to pick up the restarted jobs. This results in either a successful completion (status_cd = FIN) or failure (status_cd = ERR).

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

CHAPTER 6 Post-Processing Tasks

This chapter defines the following post-processing administrative tasks:

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

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* on page 183 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* on page 183 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* on page 185 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* on page 185 for more information).
- **Alert Assignment:** Determines the user or group of users responsible for handling each alert or case (Refer to *Alert Assignment* on page 186 for more information).

- **Auto-Close (optional):** Closes alerts that are of a lower priority to the business (Refer to *Auto-Close* on page 187 for more information).
- **Automatic Alert Suppression (optional):** Suppresses alerts that share specific scenario and focal entity attributes for a particular time frame (Refer to *Automatic Alert Suppression* on page 191 for more information).
- **Highlight Generation:** Generates highlights for alerts that appear in the alert list in the Alert Management subsystem and stores them in the database (Refer to *Highlight Generation* on page 192 for more information).
- **Augment Trade Blotter:** Provides the ability to differentiate between various types of trades using text-based codes. It also provides the ability to flag trades that require additional analysis before an analyst can mark trade as *Reviewed* or *Reviewed with Follow up*. (Refer to *Augment Trade Blotter* on page 192 for more information).
- **Score Trade Blotter:** Determines the maximum score of alerts generated in the same batch cycle associated with a trade; also determines the alert/trade mappings (Refer to *Score Trade Blotter* on page 193 for more information).
- **Historical Data Copy:** Identifies the records against which the current batch's scenario runs generated alerts and copies them to archive tables (Refer to *Historical Data Copy* on page 193 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* on page 194 for more information).
- **Case Assignment**: Determines the user or group of users responsible for handling each case. (Refer to *Case Assignment* on page 186 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

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*, Release 6.2.1, for more information.

Running the Match Scoring Job

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

To Run the Match Scoring Job

To run the Match Scoring job, follow the steps:

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

```
start_mantas.sh 501
```

All new matches in the system are scored.

Alert Creation

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

The system uses two types of Alert Creator jobs:

- Multi-match Alert Creator generates alerts for matches that share a common focus, are from scenarios in the same scenario 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

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

Alert Scoring

OFSBDF 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 OFSBDF provides to run the Alert Scoring algorithm.

Alert Assignment

OFSBDF 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 OFSBDF subsystem.

To run an Alert Assignment job, follow these steps:

1. Run the execute.sh script as follows:

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

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

Case Assignment

OFSBDF 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 OFSBDF subsystem.

To run an Case Assignment job, follow these steps:

1. Run the execute.sh script as follows:

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

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

Auto-Close

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

Defining the Auto-Close Alert Algorithm

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

- Operations are logical expressions that can be used to close alerts (for example, alert score > 50, age > 30). A set of operations based on the same attribute (for example, score) form an operation set.
- The OPRTN_SET_ID column is a grouping of mutually exclusive operations. Each operation specifies the next step that is applied to alerts that satisfy the operation. This next step is either to close the alert or execute the Next operation Set (NEXT_OPRTN_SET_ID column), or branch to further evaluate the alerts.
- The XPRSN_ORDER_ID column sets up an order of precedence by which the system attempts to satisfy the operations. 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| (for example, NW|OP).
- The value in the VALUE_TX column provides the right-hand side of the operation being evaluated.
- If the current operation is satisfied, and it is not the final operation in the operation set (indicated by a NULL value in the NEXT_OPRTN_SET_ID column), the process jumps to the NEXT_OPRTN_SET_ID. If the NEXT_OPRTN_SET_ID is NULL, and the operation is true, the system closes the alert.
- The DMN_CD column is the OFSBDF 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 OFSBDF 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 69. 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 (for example, >, <, or =). Operation sets include one or more operations chained together by the NEXT_OPRTN_SET column.

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

   Each operation's precedence must be unique within the KDD_AUTO_CLOSE_ALERT table. 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 OFSBDF 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 70. 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 71 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 71. 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. OFSBDF 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

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 OFSBDF provides to run the Auto-Close algorithm.

Automatic Alert Suppression

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

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

Defining the Suppress Alert Algorithm

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

Refer to the *Oracle Financial Services 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. OFSBDF 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

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

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

```
run_highlights.ksh
```

This script is part of the Database Tools that resides in the `<OFSBDF 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 `<OFSBDF 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 `<OFSBDF Installed Directory>/database/db_tools/logs/highlights.log` file.

Augment Trade Blotter

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

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

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

To augment trade blotter data, run the following command:

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

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

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

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

Score Trade Blotter

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

```
runScoreTradeBlotter.sh
```

Note: This script is part of the Ingestion Manager subsystem and resides in the <OFSBDF 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 (for example, scoring and assignment), but before the system ends the batch with the following shell script:

```
end_mantas_batch.sh
```

Note: This script is part of the Database Tools that reside in the <OFSBDF 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file. This property should identify the *ingest user*, a user in the database with write access to tables in both the Market and Business schemas.

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

Table 72. 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 `<OFSBDF Installed Directory>/database/db_tools/logs/hdc.log` file.

Alert Correlation

OFSBDF 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. OFSBDF 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 OFSBDF provides to run the Alert Correlation algorithm.

Understanding Alert Correlation Configuration

As mentioned above, Alert Correlation performs two major tasks—correlating alerts to business entities and correlating alerts to alerts. The second step is optional, and is governed by the `correlate_alerts_to_alerts` job parameter delivered with the template job associated to group 508. If this parameter's value is set to *true* then this

step will be performed, and if this value is set to *false* then it will not be performed. The only exception to this is if the *correlate alerts to alerts* feature is not licensed. A license for this feature can be obtained from your engagement representative, and details on enabling the license file can be found in the *Installation Guide -Stage 1*.

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

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

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

Business Entity Paths

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

Table 73. KDD_BUS_NTITY_PATH (Metadata Table)

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

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

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

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

Table 74. 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. The following is an example of the rule logic defined in an Alert Correlation Rule XML file, followed by detailed descriptions of the elements contained in the XML file:

```
<CorrelationRule id="123" name="Possible Identity Theft">
  <MinAlertCount>2</MinAlertCount>
  <PrecedenceThreshold>5</PrecedenceThreshold>
  <AlertAttrOperations>
       <![CDATA[ (BOTH.JRSDCN_CD IN ("AMEA","IND")) OR (FROM.SCORE_CT =
       TO.SCORE_CT) ]]>
  </AlertAttrOperations>
  <Lookback number="1" unit="D"/>
  <Scenarios>
     <Scenario id="234"/>
     <Scenario id="345"/>
  </Scenarios>
  <ExistingCorrelationParams>
     <ExtendFlag>TRUE</ExtendFlag>
     <NonExtendableCaseStatuses>
        <CaseStatus>CCL</CaseStatus>
        <CaseStatus>NVST</CaseStatus>
     </NonExtendableCaseStatuses>
```

```
  </ExistingCorrelationParams>
  <Actions>
     <Scoring strategy="MAX" incStrategy="ALERT_COUNT"/>
    <CasePromotion>
      <FocusTypePref>CU,AC</FocusTypePref>
      <AlertCorrAttrOperations>
        <![CDATA[(CORR.BUS_NTITY_ID = 5) AND
        (CORR.PRECEDENCE_NB <= 6)]]>
      </AlertCorrAttrOperations>
      <ExistingCasePromoteLossRcvryData>TRUE
      </ExistingCasePromoteLossRcvryData>
      <Case type="AML" subtype="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 string literal (for example, a *from alert* and *to alert* can be correlated if they both have `JRSDCN_CD = "AMEA"`, represented by `BOTH.JRSDCN IN ("AMEA", "IND"))` above, or an alert attribute compared to the same attribute (for example, a *from alert* and *to alert* can be correlated if `FROM.SCORE_CT = TO.SCORE_CT`). The set of supported comparison operators are: `=`, `!=`, `<`, `>`, `<=`, `>=`, `IN`, and `NOT IN`. Note that because the `SCNRO_ID` attribute of both alerts and correlations can potentially have multiple values, only the `IN` and `NOT IN` operators should be used in expressions involving `SCNRO_ID`. The rest of the operators can only support a single value operands. Also, there should be no space in the scenario id list specified. For example, `BOTH.SCNRO_ID IN (115600002,114690101)` is correct and `BOTH.SCNRO_ID IN (115600002, 114690101)` is incorrect.

  Multiple operations can be strung together by logical `AND` and `OR` operators and operation precedence can be defined with parentheses. Note that the text of an *AlertAttrOperation* must be wrapped in a `CDATA` tag as above to account for any special XML characters contained in the expression (for example, > or <).
- **Lookback** (*optional*): The *number* attribute indicates the number of seconds/minutes/hours/days to look back from the current date/time to create a time window in which to consider alerts for correlation. This is a create timestamp of the alert. The *unit* attribute identifies the unit of the lookback number. Possible values are S, M, H, D, and CM for seconds, minutes, hours, days, and current month, respectively. All of these require a valid number value except for CM, which essentially just makes the lookback the 1st of the current month (for example, if the current date is October 14, we will lookback to October 1 if the CM unit is selected). The create timestamp of the alert is used to determine whether or not an alert falls within the lookback period.

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

- **Scenarios** (*optional*): Identifies the Scenario(s) an alert should have been generated from in order to be considered for a correlation by this rule. If not specified, system will consider all the scenarios.
- **ExistingCorrelationParams** (*required*): Defines the conditions for extending existing correlations. When a new correlation is discovered, it is possible that it is a superset (with only the triggering alert not already included in the existing correlation) of a correlation that has previously been identified. `ExtendFlag` defines whether this correlation rule can result in extending an existing correlation. If this is set to FALSE (do not extend) then a new correlation is created when this condition is identified. If it is set to TRUE then the existing correlation is added to unless it has already been promoted to a case that has a status identified in the `CaseStatus` tags of `NonExtendableCaseStatuses`.
- **Actions** (*optional*): Once correlations are discovered, multiple types of actions can be taken on the correlation. These actions and their associated parameters are defined in between the Actions tags. The current set of possible actions include scoring the correlation and promoting the correlation to a case.
- **Scoring** (*optional*): The *strategy* attribute defines whether the correlation score should be derived from the max of the associated alert scores (`MAX_SCORE`) or the average of the associated alert scores (`AVERAGE_SCORE`). The *incStrategy* attribute provides the option of defining a fixed score to be added to the correlation score. The possible values can be `ALERT_COUNT` (each additional alert above *MinAlertCount* adds to the score), `SCENARIO_COUNT` (each distinct scenario (starting with the second scenario) involved in the correlation adds to the score), or `NONE` (the score is not incremented above what has already been calculated).

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

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

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

Activating or Deactivating Correlation Rules

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

Sample Alert Correlation Rules

OFSBDF 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.

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

CHAPTER 7 Batch Processing Utilities

OFSBDF provides utilities that enable you to set up and modify a selection of batch-related database processes. The chapter focuses on the following topics:

- About Administrative Utilities
- About Annual Activities
- Alert and Case Purge Utility
- Batch Control Utility
- Calendar Manager Utility
- Data Retention Manager
- Database Statistics Management
- Flag Duplicate Alerts Utility
- Notification
- Push E-mail Notification
- Refreshing Temporary Tables
- Truncate Manager
- ETL Process for Threshold Analyzer Utility
- Process to Deactivate Expired Alert Suppression Rules

About Administrative Utilities

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

- **Alert and Case Purge Utility**: Provides the capability to remove alerts and cases (along with their matches and activities) generated erroneously or which have exceeded the retention policies of the organization.
- **Batch Control Utility**: Manages the start and termination of a batch process (from Data Ingestion to alert post-processing) and enables access to the currently running batch.
- **Calendar Manager Utility**: Updates calendars in the OFSBDF 6.2.1 system based on predefined 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 E-mail Notification**: Enables you to configure users of the Alert Management subsystem to receive e-mail 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.

Figure 24 illustrates the frequency with which you use these batch-related database utilities when managing activities: daily, weekly, monthly, annually, or as needed.

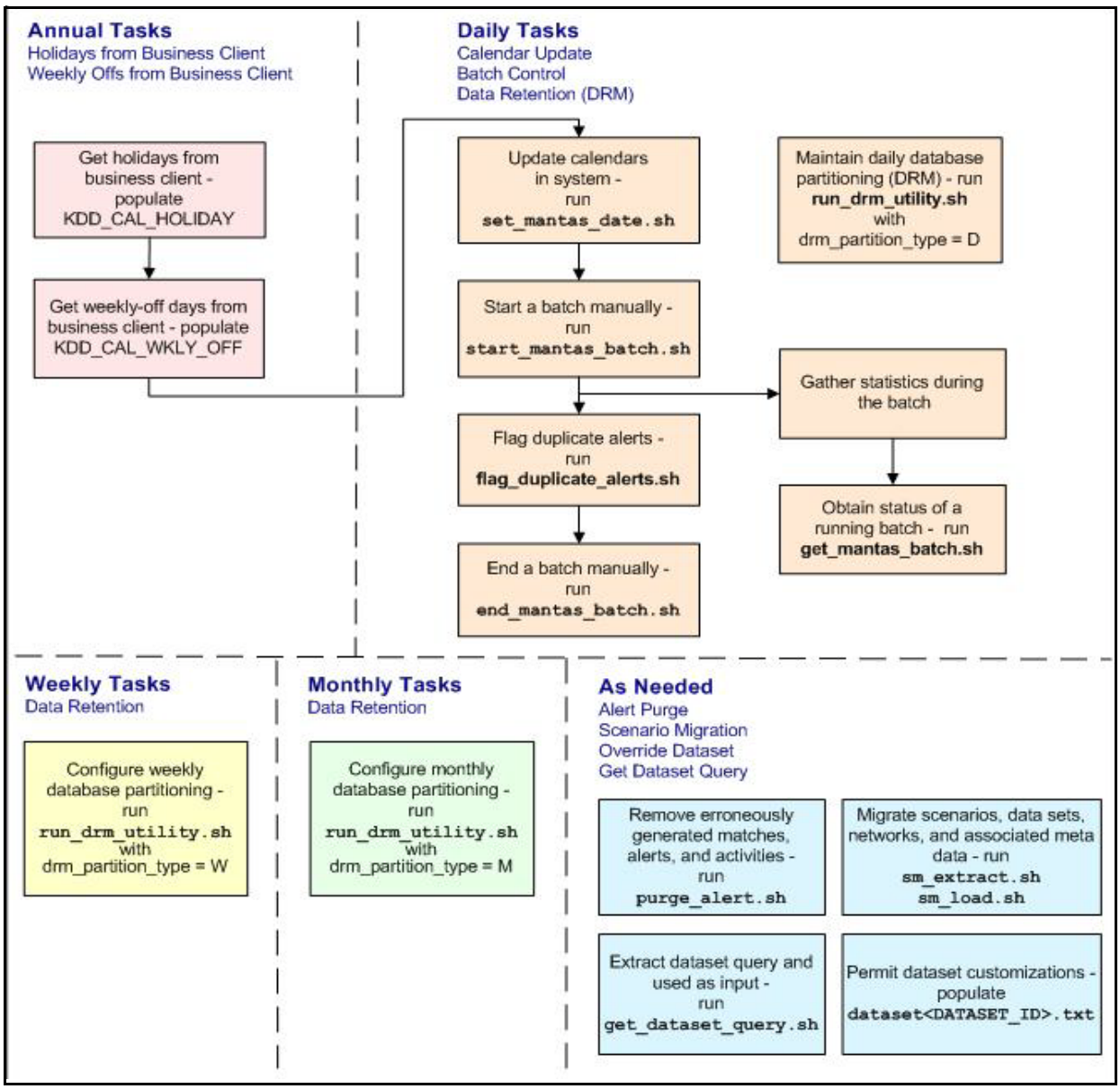

Figure 24. Managing Database Activities with Utilities

As Figure 24 illustrates, 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 need to configure data partitioning on a daily, weekly, or monthly basis.

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

Common Resources for Administrative Utilities

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

- install.cfg File
- categories.cfg File

`install.cfg` File

Configuration information resides in the `<OFSBDF 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 need to connect to the database. In this file, you can modify values of specific utility parameters, change the locations of output files, and specify database details for extraction and data loading.

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

Figure 25 (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 all schema names (that is, `MANTAS`, `BUSINESS`, and `MARKET`) are in uppercase.

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

database.driverName=oracle.jdbc.OracleDriver
utils.database.urlName=jdbc:oracle:oci:@Ti1O11L56
```

(Continued on next page)

(*Continued from previous page*)

```
utils.database.username=ECM62_B03_DB_UTIL_USER
utils.database.password=EqSHa1VHOws5vr3dbnv0nu5LK34OylgHkX2d+be0k/A=schema.mantas.owner=EC
M62_B03_MANTAS
utils.miner.user=ECM62_B03_KDD_MNR
utils.miner.password=1e38I+sp3X5E/PpdrizS9bJxT6essrW1IkR9ZwVLQKU=
schema.business.owner=ECM62_B03_BUSINESS
schema.market.owner=ECM62_B03_MARKET
utils.data.directory=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/data
ingest.user=ECM62_B03_INGEST_USER
ingest.password=sQf1TK2a+7RciP83EPs0d1Q9isu5ZCc5We+05CHOKcQ=
schema.kdd.owner=ECM62_B03_KDD
casemng.schema.owner=ECM62_B03_CASE
################ 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=200



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

(*Continued on next page*)

(Continued from previous page)

```
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/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/mantas_cfg/dep.cf
g

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=ECM62_B03_CASE

# Specify the jdbc driver details for connecting to the source database
extract.conn.driver=${database.driverName}
extract.conn.url=jdbc:oracle:oci:@Ti1O11L56

#Source System Id
extract.system.id=Ti1O11L56

# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
extract.schema.case=ECM62_B03_CASE
extract.schema.business=${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)

```
extract.dirname=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/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/ECM6.2/ECM6.2_B03/BDP_B03/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=ECM62_B03_CASE

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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/data


# Specify whether threshold can be updated
```

(Continued on next page)

(Continued from previous page)

```
load.threshold.update=Y

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

################ ALERT AND case purge CONFIGURATION ##########################
# Set the Alert And Case 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=ECM62_B03_PURGE

# Specify purge db user password.
purge.database.password=3cTpfhbVo9laz7wvrwjPfWeIR8drVK+lUejVtjnVkBM=

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

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

(Continued from previous page)

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

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

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

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

# From address for sent mails. This is ignored in Exchange mode. If omitted in SMTP mode,
the mail account associated
# with the Unix/Linux account is used.
email.from=

# SMTP settings
```

(Continued on next page)

(Continued from previous page)

```
email.smtp.host=internal-mail-router.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
email.exchange.useNTLM=false
email.exchange.draftsfoldername=drafts
email.exchange.useHTML=true

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

(Continued on next page)

(Continued from previous page)

```
email.style.disclaimer=font-style: italic;

######### 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 ##########################
#
# 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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/mantas_cf
g/analysis_aml.xml
```

(Continued on next page)

(Continued from previous page)

```
# Output file and file format control
dat.analysis.output=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/data/ana
lysis.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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/d
b_tools/mantas_cfg/db_variables.cfg
Oracle.DuplicateRow=1
Oracle.ObjectExists=955,2260,2275,1430,1442,1451,957,1408,2261
Oracle.ObjectDoesNotExist=942,1418,1434,2441,904,4043,1927,2443

dbscript.execution.users=(system|business|mantas|market|miner|ingest|report|kdd|algorithms
|case|config|fatca|ctr|kyc)

patchutil.database.urlName=@PDB_URL@
patchutil.execution.users=(pdb|system)
pdb.schema.owner=
pdb.schema.password=hu7KcS5R+wyao05BKPCpBw==

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

############## Config Migration Utility Configuration ################
```

(Continued on next page)

(Continued from previous page)

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

# 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
```

(Continued on next page)

(Continued from previous page)

```
# 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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/mantas_cfg/

# Specify where a message should get logged for a category for which there is
# no location property listed above.
# This is also the logging location of the default MANTAS category unless
# otherwise specified above.
# Note that if this property is not specified, logging will go to the console.
log.default.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/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.
# Logging will go to the console if mantaslog was selected and this property is
# not given a value.
log.mantaslog.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/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=
```

(Continued on next page)

(Continued from previous page)

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

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

Figure 25. Sample `install.cfg` File

`categories.cfg` File

In the `<OFSBDF 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 26 provides a sample `categories.cfg` file.

```
## 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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_to
ogs/purge.log
log.category.BATCH_CONTROL.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db
s/logs/batch_control.log
log.category.CALENDAR_MANAGER.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database
ools/logs/calendar_manager.log
log.category.DATA_RETENTION_MANAGER.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/da
e/db_tools/logs/DRM_Utility.log
log.category.TRUNCATE_MANAGER.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database
ools/logs/truncate_manager.log
log.category.COMMON_UTILITIES.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database
ools/logs/common_utilities.log
log.category.EXTRACT.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools
/extract.log
log.category.LOAD.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/lo
ad.log
```

(Continued on next page)

(Continued from previous page)

```
log.category.REFRESH_TEMP_TABLE.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/datab
ase/db_tools/logs/refresh_temp_table.log

log.category.RUN_STORED_PROCEDURE.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/dat
abase/db_tools/logs/run_stored_procedure.log

log.category.GET_DATASET_QUERY.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/databa
se/db_tools/logs/get_dataset_query.log

log.category.HDC.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/lo
gs/hdc.log

log.category.PUSH_EMAIL.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_t
ools/logs/push_email.log

log.category.HIGHLIGHT_GENERATOR.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/data
base/db_tools/logs/highlight_generator.log

log.category.REPORT.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools
/logs/report.log

log.category.DATA_ANALYSIS_TOOL.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/datab
ase/db_tools/logs/data_analysis_tool.log


log.category.DB_BUILDER.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_t
ools/logs/db_builder.log

log.category.DB_BUILDER_SQL.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/
db_tools/logs/db_builder.log,/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools
/logs/db_builder_sql.log


log.category.CORRRULEMIGRATIONUTIL_EXTRACT.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BD
P_B03/database/db_tools/logs/extract.log

log.category.CORRRULEMIGRATIONUTIL_LOAD.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B
03/database/db_tools/logs/load.log


log.category.CONFIGURATIONMIGRATIONUTIL_EXTRACT.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B
03/BDP_B03/database/db_tools/logs/extract.log

log.category.CONFIGURATIONMIGRATIONUTIL_LOAD.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/
BDP_B03/database/db_tools/logs/load.log


log.category.ARCHIVE_PDF.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/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.

# 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.
```

(Continued on next page)

(Continued from previous page)

```
# 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/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/pdf_archive.log



log.category.CORRRULEMIGRATIONUTIL_LOAD.notice.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/load.log
log.category.CORRRULEMIGRATIONUTIL_LOAD.fatal.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/load.log


log.category.CORRRULEMIGRATIONUTIL_EXTRACT.notice.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/extract.log
log.category.CORRRULEMIGRATIONUTIL_EXTRACT.fatal.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/extract.log


log.category.CONFIGURATIONMIGRATIONUTIL_LOAD.notice.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/load.log
log.category.CONFIGURATIONMIGRATIONUTIL_LOAD.fatal.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/load.log


log.category.CONFIGURATIONMIGRATIONUTIL_EXTRACT.notice.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/extract.log
log.category.CONFIGURATIONMIGRATIONUTIL_EXTRACT.fatal.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/extract.log


log.category.PURGE_UTIL.notice.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.warning.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.fatal.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/databas
e/db_tools/logs/purge.log
log.category.PURGE_UTIL.trace.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/databas
e/db_tools/logs/purge.log
```

Figure 26. Sample Logging Information in the `categories.cfg` File

Configuring Console Output

Figure 27 displays a section of the sample `categories.cfg` file from Figure 26. .Note the log routing information in bold text

```
log.category.PURGE_UTIL.notice.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.warning.location=console,
/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database/db_tools/logs/purge.log
log.category.PURGE_UTIL.fatal.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database
ools/logs/purge.log
log.category.PURGE_UTIL.trace.location=/scratch/ofsaaapp/ECM6.2/ECM6.2_B03/BDP_B03/database
ools/logs/purge.log
```

Figure 27. 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 27, Alert And Case Purge and Calendar Manager display relevant utility information in addition to logging it. If an entry in the `categories.cfg` file does not already include this information, you must add it manually, including the comma.

About Annual Activities

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

This section covers the following topics:

- Loading Holidays
- Loading Non-business Days

Loading Holidays

Typically, on an annual basis, you populate holidays for the upcoming calendar year into the Behavior Detection `KDD_CAL_HOLIDAY` database table. This ensures that the table contains holidays for at least the next year. Figure 28 provides an example of a SQL script for loading the table.

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


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


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


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


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


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


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


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


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


COMMIT;
```

Figure 28. Sample `KDD_CAL_HOLIDAY` Table Loading Script

The following table describes the contents of the KDD_CAL_HOLIDAY table.

Table 75. KDD_CAL_HOLIDAY Table Contents

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

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

Loading Non-business Days

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

The following provides an example of a 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 29. 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 76. KDD_CAL_WKLY_OFF Table Contents

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

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

Alert and Case Purge Utility

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

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 or Behavior Detection Scenario ID or 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 77. 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 (for example, `install.cfg` and `categories.cfg`), in which you can configure properties and logging attributes.
`logs/`	Keeps the `<OFSBDF 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 `<OFSBDF 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg` files. For more information about logging in these configuration files, Refer to *Common Resources for Administrative Utilities* on page 203 and Appendix A, *Logging*, on page 309 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 `<OFSBDF 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

---

Run the Alert And Case Purge Utility:

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

Using the Alert And Case Purge Utility

The Alert And Case Purge Utility is not part of an automated batch process that an application such as Maestro or Unicenter AutoSys controls. You run this manual process only when necessary (Refer to Figure 24, on page 202). 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

The `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that the Alert And Case Purge Utility and other utilities require for processing (Refer to Figure 25 on page 214). The following sample section from the `install.cfg` file provides configuration information specific to this utility.

```
################ ALERT PURGE CONFIGURATION ##########################
# Set the Alert Purge input variables here.
# (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=ECM62_B05_PURGE

# Specify purge db user password.
purge.database.password=nUobgigjMGXaAq0fxWJCHYRzD6iAVYfDMvi5R00snTE=

# Specify whether alerts has to be purged or not.
purge_alert_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
```

(Continued on next page)

```
(Continued from previous page)
# Specify report directory for report on process performed.
purge_report_directory=

# Specify product version
purge_product_version=6.2


#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/ofsaaapp/ECM6.2/ECM6.2_B05/BDP62_B05/database/db_tools/data
```

Figure 30. Configuration Information

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

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

Table 78. 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.

Table 78. Alert And Case Purge Utility Parameters (Continued)

Parameter	Description
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 rule(s) 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 OFSFCCM Product Version installed by the client.

The <OFSBDF 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>2011-05-25</StartDate>
        <EndDate>2011-05-25</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>2011-06-22</StartDate>
        <EndDate>2011-06-22</EndDate>
      </LastActionDate>
    </Rule>
    </Case>
</xs:RuleSet>
```

The following table describes the Purge Rules Configuration Parameters.

Table 79. 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

Table 79. Alert And Case Purge Utility Parameters (Continued)

Parameter	Description
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
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(for Days), M(for Months) or Y(for 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 79. 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(for Days), M(for Months) or Y(for 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 <OFSBDF 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 <OFSBDF 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_purge_util.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_purge_util.sh -purge alert=N

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

  ./run_purge_util.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_purge_util.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_purge_util.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

Upon execution of the run_alert_purge.sh script, the Alert And Case Purge Utility generates a listing of actions, matches, and alerts or cases that it needs to purge (as per the rules specified at the command line, or the default rule set configured in install.cfg), and records them in the <OFSBDF Installed Directory>/database/db_tools/logs/purge.log file. (The utility presumes that you have determined the input parameters to specify what matches, alerts, and actions to purge. The utility does not check against the data to verify what it should purge.)

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

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

The utility captures purging results and any errors in the `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 `<OFSBDF 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 three 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 hasclosed 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.

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 `<OFSBDF Installed Directory>/database/db_tools/lib` and UNIX script files that reside in `<OFSBDF 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` (Refer to *install.cfg File* on page 203 for more information).

The following sections describe the Batch Control Utility:

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

Note: To calculate the age in business days versus calendar days, verify that the `age.alerts.useBusinessDays` setting in the `<OFSBDF 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 Ingestion and Behavior Detection.

Deployment of a system can be with a single batch or with multiple batches. You can use multiple batches to permit intra-day processing to generate results several times per day, or to separate processing based on servicing multiple time zones.

Behavior Detection provides two types of batches:

- **End-of-day**: Represent processing at the completion of a business day for a set of data. Some processes are only appropriate for end-of-day batches. For example, daily activity summary derivations and calculating alert ages are activities that occur only in end-of-day batches. Multiple end-of-day batches per day can run if the Behavior Detection installation supports multiple time zones (for example, New York and Singapore).
- **Intra-day**: Used when loading data between end-of-day batches to obtain more frequent detection results. For example, running a batch of trading-compliance scenarios at 10:00 A.M. can identify behaviors relevant to the opening of the market without waiting for the end of the day to be able to act.

Directory Structure

Table 80 provides the directory structure for the Batch Control Utility, in `<OFSBDF Installed Directory>/database/db_tools/`:

Table 80. 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.

Table 80. Batch Control Utility Directory Structure

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

Logs

As the Batch Control Utility manages batch processing, it generates a date-stamped log in the `<OFSBDF 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg`. For more information about logging in these configuration files, Refer to *Common Resources for Administrative Utilities* on page 203, and Appendix A, *Logging*, on page 309, for more information.

Using the Batch Control Utility

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

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

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

Configuring the Batch Control Utility

The `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Batch Control and other utilities require for processing (Refer to Figure 25 on page

214). The following sample section from the install.cfg file provides configuration information specific to this utility, including the single parameter that batch control requires.

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

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

Figure 31. 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 88 on page 242, 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

OFSBDF 6.2.1 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 following table provides the contents of the KDD_PRCSNG_BATCH table.

Table 81. KDD_PRCSNG_BATCH Table Contents

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

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

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

Single Batch

In this example, the KDD_PRCSNG_BATCH table contains a single batch per day. This is typical of deployment of a single geography for which a solution set does not require detection more than once daily. The KDD_PRCSNG_BATCH table may look similar to the example in Table 82.

Table 82. 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 83 describes.

Table 83. Sample `KDD_PRCSNG_BATCH` Table with Intra-day Processing

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

In this configuration, run the Calendar Manager Utility only during the MORN batch. Refer to *Calendar Manager Utility* on page 239, 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 *Data Retention Manager* on page 243, for more information.

Multiple Countries

A single deployment supports detection against data from New York, London, and Hong Kong. In this case, three batches are all end-of-day batches, as Table 84 describes.

Table 84. 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 `<OFSBDF 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 <OFSBDF 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.

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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/categories.cfg` file. Refer to *Configuring Console Output* on page 218, 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 84).
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 85. `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 86. `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 `<OFSBDF 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 32. 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 32. 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 81 on page 234).

To End a Batch Manually

To stop a batch process manually, follow the steps:

1. Verify that the Behavior Detection database is operational.

   tnsping <database instance name>

2. Verify that the <OFSBDF 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 <OFSBDF 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 <OFSBDF 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 85 on page 236).

2. Records information about the batch in the KDD_PRCSNG_BATCH_ HIST table (Refer to Table 86 on page 236). This table contains a history of all batches that appear by start date and end date. Figure 33 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 33. 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 <OFSBDF Installed Directory>/database/db_tools/logs/batch_control.log file, stating that the end batch process has begun.

Identifying a Running Batch Process

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

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

To Obtain a Batch Name

To obtain a batch name, follow the steps:

1. Access the directory where the shell script resides:

   ```
   cd <OFSBDF 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 218, for more information).

Process for Obtaining a Batch Name

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

1. The utility retrieves the name of the currently running batch from the KDD_PRCSNG_BATCH_CONTROL table (Refer to Table 85 on page 236).
2. The utility returns the batch name to standard output.

Calendar Manager Utility

After loading holidays into the KDD_CAL_HOLIDAY table and weekly off-days into the KDD_CAL_WKLY_OFF table, you can use the Calendar Manager Utility to update and manage OFSBDF system calendars. You use the utility's Java and shell scripts to connect to the database and perform processing. The <OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg configuration file contains modifiable inputs that you use to run the utility (Refer to *install.cfg File*, on page 203, 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 <OFSBDF Installed Directory>/database/db_tools/.

Table 87. Calendar Manager Utility Directory Structure

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

Logs

As the utility updates the calendars in the OFSBDF system, it generates a log that it enters in the <OFSBDF 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 <OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg and categories.cfg. For more information about logging in these configuration files, Refer to *Common Resources for Administrative Utilities* on page 203, and Appendix A, *Logging*, on page 309, for more information.

Calendar Information

The Calendar Manager Utility obtains all holidays and weekly off-days for loading into the OFSBDF calendars by retrieving information from the KDD_CAL_HOLIDAY and KDD_CAL_WKLY_OFF tables (Refer to Table 75 and Table 76). 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 OFSBDF 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 `<OFSBDF 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 25). The following sample section from the `install.cfg` file provides configuration information specific to this utility, including default numerical values in the utility's two required parameters.

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

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

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

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

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

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

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

Executing the Calendar Manager Utility

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

To Start the Utility Manually

To start the Calendar Manager Utility, follow the steps:

1. Verify that the Behavior Detection database is operational:

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

2. Verify that the `<OFSBDF 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 <OFSBDF 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 (for example, enter November 30, 2006 as *`20061130`*). The utility then verifies that the entered date is valid and appears in the correct format.

   If you do not enter a date or enter it incorrectly, the utility terminates and logs a message that describes the error. The error message displays on the console only if you have output to the console enabled in the `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/categories.cfg` configuration file. Refer to *Configuring Console Output*, on page 218, for more information.

Updating the `KDD_CAL` Table

As previously discussed, the Calendar Manager Utility retrieves information that it needs for updating OFSBDF 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 88. KDD_CAL Table Contents

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

Table 88. KDD_CAL Table Contents (Continued)

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

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

Data Retention Manager

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

The Data Retention Manager enables you to manage Oracle database partitions and indexes on a daily, weekly, and/or monthly basis (Refer to Figure 24 on page 202). 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 89. Data Retention Manager Directory Structure

Directory	Contents
`bin/`	Executable files, including the `run_drm_utility.sh` shell script.
`lib/`	Required class files in `.jar` format.
`mantas_cfg/`	Configuration files (for example, `install.cfg` and `categories.cfg`), in which you can configure properties and logging attributes.
`logs/`	File `<OFSBDF 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 `<OFSBDF 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg`. For more information about logging in these configuration files, Refer to *Common Resources for Administrative Utilities*, on page 203, and Appendix A, *Logging*, on page 309, for more information.

Processing Flow

Figure 34 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.

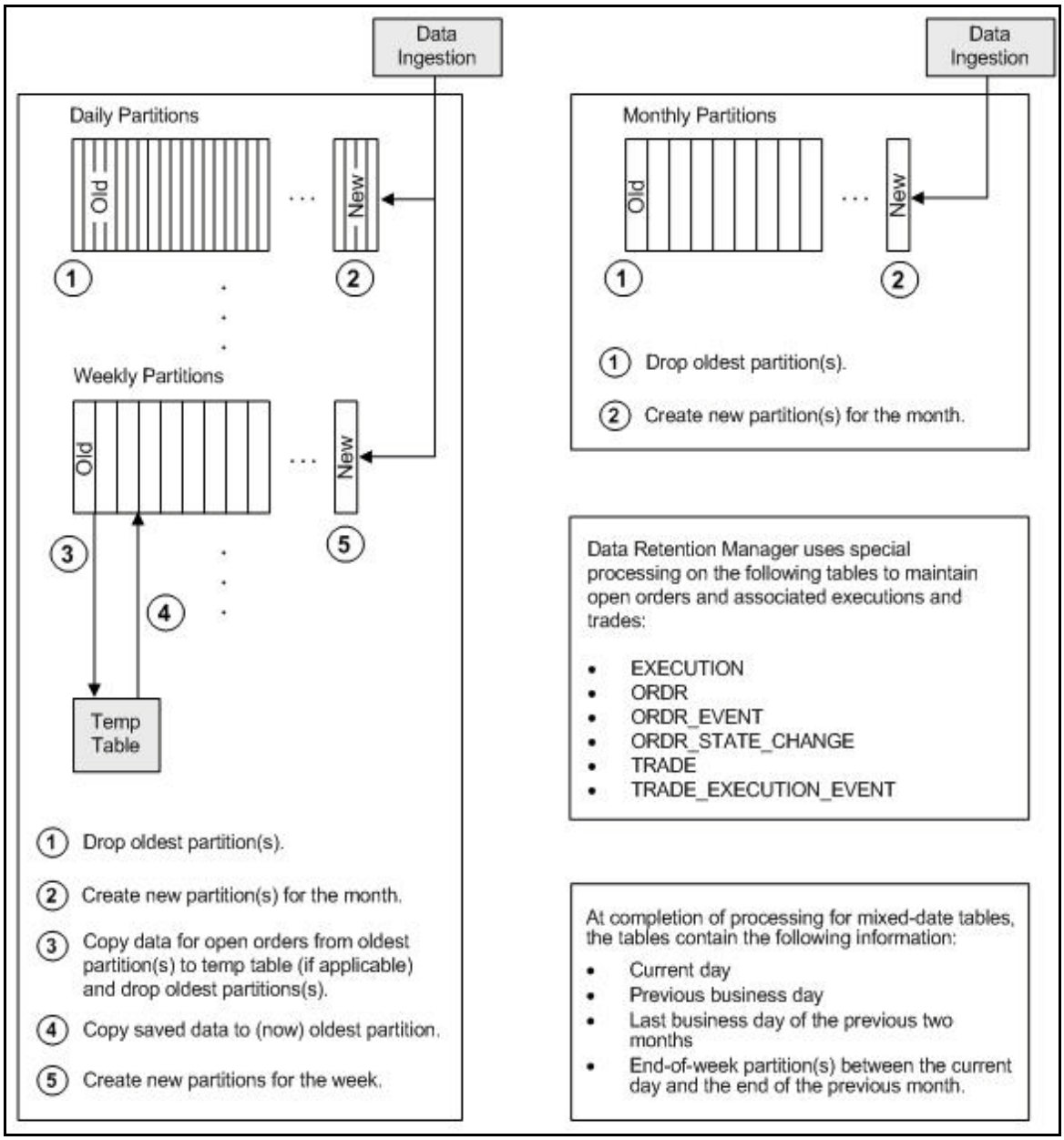

Figure 34. Database Partitioning Process

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

The `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Data Retention Manager and other utilities require for processing (Refer to Figure 25 on page 214 for a sample `install.cfg` file).

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

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

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

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

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 90. Data Retention Manager Processing Parameters

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

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

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

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

Executing the Data Retention Manager

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

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

Note: Be sure to set the system date with the Calendar Manager Utility prior to running the Data Retention Manager (Refer to *Calendar Manager Utility* on page 239, for more information).

Running the Data Retention Manager

To run Data Retention Manager manually, follow these steps:

1. Verify that the Behavior Detection database is operational:

   `tnsping <database instance name>`

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

4. Start the batch shell script with the parameters in Table 90:

   `run_drm_utility.sh <drm_operation> <drm_partition_type> <drm_owner> <drm_object_name> <drm_weekly_proc_fl>`

Script Examples:

The following are examples of running the script:

- To run the utility for all daily tables in the BUSINESS schema, execute the script:

  `run_drm_utility.sh P D BUSINESS A N`

- To run the utility to drop a monthly partition of the `BUSINESS` table `ACCT_SMRY_MNTH`, execute the script as follows (using the same parameters as in the previous example):

  `run_drm_utility.sh DM M BUSINESS ACCT_SMRY_MNTH N`

Creating Partitions

When creating partition names, use the formats in the following table.

Table 91. 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 those tables that have daily, weekly, and monthly partitions, respectively.

Partition maintenance:

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

Daily Partitioning Alternative

The Data Retention Manager also enables you to build five daily partitions only a weekly basis rather than daily. You do this by executing the `run_drm_utility.sh` shell script and setting the `drm_weekly_proc_flg` parameter to Y (Refer to Table 90 on page 247).

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

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

Note: You must set the `WEEKLY_ADD_FL` parameter in the `KDD_DR_MAINT_OPRTN` table to `Y` so that the procedure works correctly. For more information about this parameter, Refer to Table 92 on page 251, for more information.

Partition Structures

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

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

  However, you can use the Calendar Manager Utility to configure a business calendar as desired. For more information about this utility, Refer to *Calendar Manager Utility* on page 239, 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.

Alternative Monthly Partition Maintenance

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

To Add a Monthly Database Partition

To add a monthly partition, run the utility's shell script as follows (Refer to Table 90 for parameters):

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

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

To Drop a Monthly Database Partition

To drop a monthly partition, run the utility's shell script as follows (Refer to Table 90 for parameters):

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

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

Maintaining Indexes

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

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

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

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

Utility Work Tables

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

- KDD_DR_MAINT_OPRTN Table
- KDD_DR_JOB Table
- KDD_DR_RUN Table

KDD_DR_MAINT_OPRTN Table

The KDD_DR_MAINT_OPRTN table contains the processing information that manages Data Retention Manager activities. The following table describes the contents.

Table 92. 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 92. 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 describes the contents.

Table 93. 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 describes the contents.

Table 94. 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 88 on page 242 for contents of the KDD_CAL table.

Database Statistics Management

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

Logs

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

Using Database Statistics Management

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

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

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

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

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

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

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

Note: For the EST_STATS and EST_PART_STATS parameters, the default sample size that the analyze procedure uses is 5% of the table under analysis. To change the sample percentage, update the SAMPLE_PT column of the desired record in the KDD_ANALYZE_PARAM table.

- IMP_STATS: Imports statistics that were previously calculated. When running an ALL analysis, the system exports statistics for the tables for later use.

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 <OFSBDF Installed Directory>/ database/db_tools/mantas_cfg/install.cfg file. The install.cfg file also contains the following properties:

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

The system derives schema names from these properties.

For the Case Management Schema, there is no separate script for managing Oracle database statistics. But for improved query performance, we have to manage the Oracle database statistics periodically. Following are the sample commands.

To analyze table wise use, use the following commands:

```
ANALYZE table <Table name> compute statistics;
```

Example: `ANALYZE table KDD_CASES compute statistics;`

We can also perform whole schema analyze periodically.

Flag Duplicate Alerts Utility

The Flag Duplicate Alerts Utility enables you to run a script daily after the generation of alerts. This script identifies the pairs of alerts that are possible duplicates. It then adds a system comment to each alert and identifies the paired alert in the comment as a *Possible Duplicate*.

External Entity-focused scenarios in Behavior Detection can generate alerts either on external identifiers (for example, external account ID) or on names of parties outside the bank. The logic of the scenarios only generates the name-focused alerts when the name has been found with multiple (or no) external identifiers. This check is made across all transactions, not just the transactions involved in a particular alert. As a result, a single run of an External Entity-focused scenario can generate alerts involving the exact same transactions, one alert focused on the external Party ID, and one alert focused on the external Party Name.

Using the Flag Duplicate Alerts Utility

The Flag Duplicate Alerts Utility looks at alerts that meet the following criteria:

- Entity focus (EN)
- Status of New (NW)
- Generated in the current running batch on the current date

The utility selects and compares alerts that meet the listed criteria above. It then determines whether generation of the alert is based on the same set of transactions for the same scenario, with different focuses (for example, one alert is an ID and the other is a Name). The utility flags these alerts as possible duplicates and adds a system comment in the Action History section of the Alert Details page (each alert cross-references the other). For example:

`Possible duplicate of alert` **`xxxxx`**.

Executing the Flag Duplicate Alerts Utility

Use the following procedure to execute the Flag Duplicate Alerts Utility.

To Execute the Flag Duplicate Alerts Utility

To execute the Flag Duplicate Alerts Utility, run the following script after the Alert Creator, Assigner, and Auto-Close processes (jobs) have completed:

```
<OFSBDF Installed Directory>/database/db_tools/bin/flag_duplicate_alerts.sh
```

The system writes log information for this process to the following location:

```
<OFSBDF Installed Directory>/database/db_tools/logs/run_stored_procedure.log
```

Notification

Notifications appear on the UI on the Home page and help alert users to items requiring their attention.

Notifications can be classified into two categories (depending on the method of generation):

- Event Based
- Batch Based

Event Based

These notifications are always associated with an event. Following are the event based notifications:

- **New Case Creation notification**: Whenever a user manually creates a new case, a notification is generated to the owner of the case and if owner is a pool then notification is generated to all the users who fall under that pool. If the user who created the case is also assigned as the owner, no notification is generated.
- **Re-assigned case notification**: Notification is generated to new owner of the case upon reassignment of the case. If the user who reassigned the case is also the new owner, no notification is generated. If the new owner is a pool then notification is generated to all users who are members of the organization represented by that pool.
- **Re-assigned alerts notification**: Notification is generated to the new owner of the Alert upon reassignment of the alert. If the user who reassigned the alert is also the new owner, no notification is generated. If the new owner is a pool then notification is generated to all users who are members of the organization represented by that pool.
- **Alert Data Transfer Unsuccessful**: In Asynchronous alert data transfer mode, if 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.

The above notifications are generated after the complete execution of Batch (provide the batch name) and can be seen in the Notification Grid in landing page. Each user sees the notification which is relevant to him.

Note: You can set the parameter of near due date and display of notification from the Manage Parameters screen. (Refer to section *Configuring Notifications* in the *Configuration Guide,* for more information).

Push E-mail Notification

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 or cases that they own since the last time this notification job was run. This notification also identifies any alerts and cases that have been assigned to the user that the user has not yet opened.
- OverDue: This notification identifies alerts and cases 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 (For example, 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, use the following shell script:

`<OFSBDF Installed Directory>/database/db_tools/bin/run_push_email.ksh [notification list]`

If you do not include any 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file. The java process runs the queries associated with the desired notifications and sends e-mail to the users. By default, the system sends e-mail using unauthenticated SMTP, however it also supports authenticated SMTP, authenticated or unauthenticated SMTPS and Microsoft Exchange.

When the notification runs, the date and timestamp for the notification is stored in the file `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/notification.config`

Warning: This file must not be edited.

To avoid corruption of this file, do no 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 95. 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

Three files are used to configure Push E-mail notification. Configuration for connectivity and mail format parameters are modified in:

`<OFSBDF 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:

`<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/NotificationDetails.xml`

The queries that are run against the database for notification are configured in:

```
<OFSBDF Installed
Directory>/database/db_tools/mantas_cfg/etc/xml/QBD_Notification.xml
```

The following sections provide configuration guidance for each of these files.

To Configure General Notification Properties

Table 96 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:

```
<OFSBDF Installed Directory>/changePasswords.sh email.smtp.password
```

```
<OFSBDF Installed Directory>/changePasswords.sh email.exchange.password
```

Table 96. 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.	mantas@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.	

Table 96. Push E-mail Notification Configurable parameters (Continued)

Property	Description	Sample Value
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.	Mantas
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.	Mantas.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). Behavior Detection Framework does not provide a copy of this product, it must be purchased separately. After you have purchased JEC, place the jec.jar in the <OFSBDF Installed Directory>/database/db_tools/lib directory, and copy your jeclicense file into <OFSBDF Installed Directory>/database/db_tools/mantas_cfg.

In addition to the configuration parameters identified in Table 96 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.

To Configure Notifications

The list of notifications to be delivered are configured in the <OFSBDF Installed Directory>/database/db_tools/mantas_cfg/NotificationDetails.xml file.

Figure 35 shows a sample of `NotificationDetails.xml` file for illustration.

```
<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 35. 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 97. Additional Elements of `NotificationDetails.xml` file

Element	Description
Title HTMLTitle	The title is a title for the section.

Table 97. Additional Elements of `NotificationDetails.xml` file

Element	Description
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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/etc/xml/QBD_Notification.xml`.

Configuring Notification Queries

The queryDefs for the default notifications are configured in`<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/etc/xml/QBD_Notification.xml`.

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 (for example, `QBD_CustomNotification.xml`). Figure 36 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 36. 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 98. 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. References to the MANTAS and BUSINESS schemas are replaced with the appropriate schema names for your environment.
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"/>
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 <OFSBDF 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:

<OFSBDF 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 `<OFSBDF 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:

`<OFSBDF Installed Directory>/database/db_tools/logs/refresh_temp_table.log`

Using Refreshing Temporary Tables

The MINER schema defines these tables; the tables have corresponding views that are used to populate them. Prior to running these patterns, run the `refresh_temp_table.sh` script. The script has the following calling signature:

`refresh_temp_table.sh <table_name> <view_name>`

where:

- `table_name` identifies the name of the table to populate.
- `view_name` identifies the name of the view to run to populate the table.

This procedure deletes all records in the target table prior to running the view to populate it. It then estimates statistics for the newly populated table. This procedure logs into the database with the user that the `utils.miner.user` property identifies in the `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file.

Populate Temporary Tables for Scenarios

Scenarios typically depend on Data Ingestion to complete processing, however the IML-HiddenRelationships-dINST, ML-NetworkOfAcEn-fAC, and CST-LOSSES scenarios depend on population of Temp Tables to populate data. The Link Analysis scenario also depends on the network job creation before the sequence matcher part of the scenario runs.

IML-HiddenRelationships-dINST

To populate the temporary tables for IML-HiddenRelationships-dINST scenario, follow the steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel):

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_JRNL TMP_HIDREL_NT_JRNL_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_WIRE TMP_HIDREL_NT_WIRE_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_ACTAXID TMP_HIDREL_NT_ACTAXID_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_ACADDR TMP_HIDREL_NT_ACADDR_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_ACPHONE TMP_HIDREL_NT_ACPHONE_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_ACEMAIL TMP_HIDREL_NT_ACEMAIL_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_ACPSWRD TMP_HIDREL_NT_ACPSWRD_VW
   ```

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
   TMP_HIDREL_NT_INST TMP_HIDREL_NT_INST_VW
   ```

```
<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIREACBENE TMP_HIDREL_NT_WIREACBENE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_WIREACORIG TMP_HIDREL_NT_WIREACORIG_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACTAXID TMP_HIDREL_NT_CUACTAXID_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACADDR TMP_HIDREL_NT_CUACADDR_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACPHONE TMP_HIDREL_NT_CUACPHONE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_HIDREL_NT_CUACEMAIL TMP_HIDREL_NT_CUACEMAIL_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698616.
3. Execute the scenario job. The product job template ID is 116200024.

ML-NetworkOfAcEn-fAC

To populate the temporary tables for ML-NetworkOfAcEn-fAC scenario, follow the steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel):

```
<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTADDR TMP_NETACENCU_NT_ACCTADDR_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTEMAIL TMP_NETACENCU_NT_ACCTEMAIL_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTPHONE TMP_NETACENCU_NT_ACCTPHONE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTPSWRD TMP_NETACENCU_NT_ACCTPSWRD_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_ACCTTAXID TMP_NETACENCU_NT_ACCTTAXID_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACADDR TMP_NETACENCU_NT_CUACADDR_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACEMAIL TMP_NETACENCU_NT_CUACEMAIL_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACPHONE TMP_NETACENCU_NT_CUACPHONE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_CUACTAXID TMP_NETACENCU_NT_CUACTAXID_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_JRNL TMP_NETACENCU_NT_JRNL_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIREACBENE TMP_NETACENCU_NT_WIREACBENE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIREACORIG TMP_NETACENCU_NT_WIREACORIG_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_NETACENCU_NT_WIRETRXN TMP_NETACENCU_NT_WIRETRXN_VW
```

2. Execute the link analysis/network generation job. The product job template ID is 114698120.
3. Execute the scenario job. The product job template ID is 114698631.

FR-NetworkOfAcEn-fAC

To populate the temporary tables for FR-NetworkOfAcEn-fAC scenario, follow these steps:

1. Execute these refresh temporary table processes (these commands can be run in parallel.):

```
<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_NT_ACCTADDR TMP_FRNTWRK_NT_ACCTADDR_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_ACCTEMAIL TMP_FRNTWRK_ACCTEMAIL_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_ACCTPHONE TMP_FRNTWRK_ACCTPHONE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_ACCTPSWRD TMP_FRNTWRK_ACCTPSWRD_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_ACCTTAXID TMP_FRNTWRK_ACCTTAXID_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_CUACADDR TMP_FRNTWRK_CUACADDR_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_CUACEMAIL TMP_FRNTWRK_CUACEMAIL_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_CUACPHONE TMP_FRNTWRK_CUACPHONE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_CUACTAXID TMP_FRNTWRK_CUACTAXID_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_JRNL TMP_FRNTWRK_JRNL_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_WIREACBENE TMP_FRNTWRK_WIREACBENE_VW

<OFSBDF Installed Directory>/database/db_tools/bin/refresh_temp_table.sh
TMP_FRNTWRK_WIREACORIG TMP_FRNTWRK_WIREACORIG_VW

<OFSBDF 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 the steps:

1. Execute this refresh temporary table process:

```
<OFSBDF 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:
   `<OFSBDF Installed Directory>/database/db_tools/run_p_uncvrdlongsales_ew.sh`
2. Execute the scenario job.

Truncate Manager

The Data Ingestion subsystem calls the `run_truncate_manager.sh` script to truncate tables that require complete replacement of their data.

Logs

The `log.category.TRUNCATE_MANAGER.location` property in the `<OFSBDF 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:

`<OFSBDF Installed Directory>/database/db_tools/logs/truncate_manager.log`

Using the Truncate Manager

The `run_truncate_manager.sh` script takes the table name as an argument; the table must exist in the BUSINESS schema (which the `schema.business.owner` property identifies). The script logs into the database using the user that the `truncate.database.username` property specifies in the `<OFSBDF 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.

ETL Process for Threshold Analyzer Utility

For inserting and updating records into the `KDD_TA_ML_DATA`, `KDD_TA_BC_DATA`, and `KDD_TA_TC_DATA` tables, there are two shell scripts that are used to call the database procedures. These are:

- `run_insert_ta_utility.sh` – This script calls the `P_TA_ML_INSERT_BREAKS`, `P_TA_BC_INSERT_BREAKS`, and `P_TA_TC_INSERT_BREAKS` procedures, which insert data into the `KDD_TA_ML_DATA`, `KDD_TA_BC_DATA`, and `KDD_TA_TC_DATA` tables, respectively, based on the `CREAT_TS` of the alerts in relation to the `LAST_RUN_DT` from `KDD_TA_LAST_RUN` (values for `RUN_TYPE_CD` are `ML_I`, `BC_I`, and `TC_I`). There is one optional parameter (`DEBUG_FL`) for this shell script (defaults value to `FALSE`). If you provide a value of TRUE as an argument then information (insert commands) is also loaded into the `KDD_TA_INS_DEBUG` table. It also updates the `LAST_RUN_DT` column in the `KDD_TA_LAST_RUN` table (values for `RUN_TYPE_CD` are `ML_I`, `BC_I`, and `TC_I`) with the date (sysdate) the procedure was last run.
- `run_update_ta_utility.sh` – This script calls the `P_TA_ML_UPDATE`, `P_TA_BC_UPDATE`, and `P_TA_TC_UPDATE` procedures, which update `QLTY_RTNG_CD` in the `KDD_TA_ML_DATA`,

KDD_TA_BC_DATA, and KDD_TA_TC_DATA tables, respectively, for any *Review* closed since the last run based on LAST_RUN_DT from KDD_TA_LAST_RUN (values for RUN_TYPE_CD are ML_U, BC_U, and TC_U). The CLS_CLASS_CD value from KDD_REVIEW is used as the new QLTY_RTNG_CD. There are no parameters needed for this shell script. It also updates the LAST_RUN_DT column in the KDD_TA_LAST_RUN table (values for RUN_TYPE_CD are ML_U, BC_U, and TC_U) with the date (sysdate) the procedure was last run.

The log for these scripts is written in the run_stored_procedure.log file under the <OFSBDF 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 <OFSBDF Installed Directory>/database/db_tools/bin directory.

You can run this utility anytime. (In other words, it is not necessary to run this utility during specific processing activities.)

Process to Deactivate Expired Alert Suppression Rules

The following shell script should be executed in order to deactivate Alert Suppression Rules that have expired based on the current system date:

```
-- run_upd_suppression_recs.sh
```

This script should be run as the last step in batch processing just prior to ending the batch. It is important that this script is run after post-processing has been completed (that is, not before the Alert Suppression job is executed).

CHAPTER 8 Administrative Utilities

OFSBDF 6.2.1 provides utilities that enable you to set up or modify a selection of database processes. This chapter focuses on the following topics:

- About Administrative Utilities
- Data Analysis Tool
- Get Dataset Query with Thresholds Utility
- Scenario Migration Utility
- Alert Correlation Rule Migration Utility
- Investigation Management Configuration Migration Utility
- Watch List Service
- Alert Processing Web Services
- Password Manager Utility
- Update 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:

- **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.

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.

Data Analysis Tool

The Data Analysis Tool enables you to determine how well a customer has populated the Production Data Model. By reviewing the quality of data in each of the tables that the schema identifies, the Data Analysis Tool indicates how well the supplied data can support scenarios. The tool does not make "judgments" about data quality. Rather, it provides a repeatable way to run a set of analytical queries across the data. You can then use the results to direct further analysis.

The Data Analysis Tool:

- Counts all table rows in the schema.
- Identifies unique values and their distribution against the table.
- Determines the number of null occurrences for a specified column.
- Determines the number of padded spaces that occur for a specified column.
- Checks referential integrity between tables.

The following sections provide instructions for using the tool:

- Configuring the Data Analysis Tool
- Using the Data Analysis Tool
- Logs
- Troubleshooting the Data Analysis Tool

The tool provides its results in either a text or Hypertext Markup Language (HTML) file. You can then use these results to direct an investigation for data quality.

Note: To use the Data Analysis Tool effectively, you must have basic knowledge of Structured Query Language (SQL) and Extensible Markup Language (XML).

Configuring the Data Analysis Tool

The Data Analysis Tool uses the `install.cfg` and `analysis.xml` (or similar) configuration files. You edit either file in a text editor such as vi. To produce well-formed XML files, however, you should edit the XML file in a validating XML editor.

To Configure General Tool Properties

Behavior Detection deploys the Data Analysis Tool as one of the utilities under the database tools. Basic configuration for these tools is through the `install.cfg` file that resides in `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg`.

The following table provides the configuration instructions for the properties that the Data Analysis Tool uses in the `install.cfg` file.

Table 99. Configuration Instructions for the `install.cfg` File

Property	Description	Example
`database.driverName`	Database connection driver that the utility is to use.	`database.driverName =oracle.jdbc.driver.OracleDriver`
`utils.database.urlName`	Database connection string that the Data Analysis Tool is to use.	`utils.database.urlName =jdbc:oracle:oci: @PROD_DB`
`schema.business.owner`	Database user for the BUSINESS schema.	`schema.business.owner=BUSINESS`

Table 99. Configuration Instructions for the `install.cfg` File (Continued)

Property	Description	Example
`schema.market.owner`	Database user for the MARKET schema.	`schema.market.owner= MARKET`
`dat.database.username`	User name for the database. The Data Analysis Tool connects to the database as the INGEST_USER for the appropriate privileges.	`dat.database.username= INGEST_USER`
`dat.database.password`	Password for the database. This is set by the Password Manager Utility.	
`dat.analysis.input`	Path and name for the XML input file. By default, this is the `analysis.xml` file under the `<OFSBDF 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*, on page 214.

To Configure the Analysis XML File

The `analysis.xml` configuration file specifies the queries that you can use to analyze the data that the database schema provides. You can perform the following types of queries:

- Distinct Values for Fields of Interest Analysis
- Null and Padded Space Count Analysis
- Join Counts Analysis for referential integrity between two or more tables.
- Other Queries as configured.

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 100. 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 100 lists. When using the EMPTY operator, the generated SQL statement includes the field name, a space, and the text within the value string. As such, representation of more complex operations is possible.

An AND operator joins any existing, multiple <CONSTRAINT> elements.

When adding date constraints as in the first example above, you must specify the date in the same format as the database's NLS Date Format (Oracle recommends `DD-MON-YYYY` as the default format).

Distinct Values for Fields of Interest Analysis

Identifying the table and one or more column combinations of interest provides a combination of distinct values and number of occurrences in the table. The following code illustrates the required structure of this analysis within the following elements:

```
<ANALYSIS>
  <TABLES>
    <analysis for distinct values occurs here>
  </TABLES>
</ANALYSIS>
```

The name attribute of the <TABLE> element identifies the table against which this analysis is applied. The <VALUES> element identifies targeted columns. The field attribute of the <COLUMN> element sets each database column.

Application of filters to an analysis is possible if the <CONSTRAINT> element identifies the filter. The following code illustrates the structure for using a filter:

```
<TABLE name="table name">
<!-- get distinct value for one column -->
  <VALUES>
```

```
        <COLUMN field="column name"/>
           <!-- Constraint feature is optional.
                May contain one or more constraints. -->
            <CONSTRAINT field="column name" operator="operator"
                           value="filter value" />
      </VALUES>
    <!-- get distinct value for many columns -->
      <VALUES>
        <COLUMN field="column name"/>
        <COLUMN field="column name"/>
           <!-- Constraint feature is optional.
                May contain one or more constraints. -->
           <CONSTRAINT field="column name"
            operator="operator"value="filter value" />
      </VALUES>
    </TABLE>
```

The following XML code illustrates use of a filter:

```
    <ANALYSIS>
      <TABLES>
        <TABLE name="ACCT">
          <VALUES>
            <COLUMN field="ACCT_TYPE1_CD"/>
            <COLUMN field="ACCT_TYPE2_CD"/>
          </VALUES>
        </TABLE>
        <TABLE name="CUST">
           <VALUES>
             <COLUMN field="CUST_TYPE_CD"/>
             <CONSTRAINT field="DATA_DUMP_DT" operator="EQ"
              value="15-NOV-2006" />
           </VALUES>
         </TABLE>
      </TABLES>
    <ANALYSIS>
```

This XML code executes the following queries:

```
    select ACCT_TYPE1_CD, ACCT_TYPE2_CD, count(1)
    from ACCT
    group by ACCT_TYPE1_CD, ACCT_TYPE2_CD

    select CUST_TYPE_CD, count(1)
```

```
from CUST
where DATA_DUMP_DT='15-NOV-2006'
group by CUST_TYPE_CD
```

Null and Padded Space Count Analysis

Null and padded space count analysis provides the number of occurrences for null values and padded spaces for a particular field in a table. You perform this analysis by identifying the table and one or more columns of interest. The null analysis feature has the following limitations:

- The feature is optional.
- The field identified for the specified table can be analyzed only once within the <NULLS> element per table.
- The filtering feature for the null analysis is optional and can have multiple constraints.

The structure to perform this analysis is:

```
<ANALYSIS>
   <TABLES>
     <!-- analysis for null counts occurs here -->
   </TABLES>
</ANALYSIS>
```

Within the <TABLE> element, the name attribute identifies the table to be analyzed. The targeted columns are identified within the <NULLS> element. The field attribute in the <NULL> element sets each column name. Apply filters to the analysis within the <CONSTRAINT> element. The following code illustrates the structure for the a null and padded space count analysis:

```
<TABLE name="table name">
<!-- May contain one or more columns -->
   <NULLS><!-- With no constraints -->
     <NULL field="column name"/><!-- With constraints -->
     <NULL field="column name">
       <!-- Constraint feature is optional.
            May contain one or more constraints. -->
        <CONSTRAINT field="column name" operator="operator"
                        value="filter value" />
     </NULL>
   </NULLS>
</TABLE>
```

The following XML code sample is an example of the correct structure:

```
<TABLE name="ACCT">
   <NULLS>
     <NULL field="ACCT_TYPE1_CD"/>
     <NULL field="RGSTN_TYPE_CD">
       <CONSTRAINT field="DATA_DUMP_DT" operator="EQ"
                        value="15-NOV-2006" />
```

```
      </NULL>
    </NULLS>
<TABLE name="ACCT">
```

This code executes the following queries:

```
SELECT sum(case when ACCT_TYPE1_CD is null then 1 else 0 end)as NULL_CT0,
sum(case when ACCT_TYPE1_CD <> ltrim(rtrim(ACCT_TYPE1_CD))
then 1 else 0 end) as SPACE_CT0,
sum(case when RGSTN_TYPE_CD is null
and DATA_DUMP_DT='15-NOV-2006' then 1 else 0 end) as NULL_CT1,
sum(case when RGSTN_TYPE_CD <> ltrim(rtrim(RGSTN_TYPE_CD))
and DATA_DUMP_DT='15-NOV-2006' then 1 else 0 end) as SPACE_CT1
FROM ACCT a
```

Join Counts Analysis

A join identifies the relationship between two tables by common fields. Checking for join counts determines the referential integrity between two or more tables. Determine join counts as follows:

- Simple join between two or more tables (Refer to *Simple Join* on page 275, for more information).
- Simple join between two or more tables with filter restriction (Refer to *Simple Join with Filter Restriction* on page 276, for more information).
- Join count of distinct values for specific column (Refer to *Join Count by Distinct Column* on page 277, 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 100 on page 272, for more information.

The structure is organized in the same manner as a Simple Join with an added <FILTER> element. The following code illustrates the structure:

```
<MULTIJOIN>
   <JOIN>
     <LEFT  table="table name" column="column" />
```

```
      <RIGHT table="table name" column="column" />
    </JOIN>
    <!-- Optional. May contain one or more filters. -->
    <FILTER table="table name" column="column" operator=
                "operator" value="filter value" />
</MULTIJOIN>
```

The <FILTER> element is optional in the join analysis. Multiple filters can be applied to a join. The AND operator is appended to each filter condition upon creation of the query. The following XML code illustrates the use of a filter with a simple join analysis:

```
<ANALYSIS>
  <JOINS>
    <MULTIJOIN>
      <JOIN>
        <LEFT table="ACCT" column="ACCT_INTRL_ID" />
        <RIGHT table="CUST_ACCT" column="ACCT_INTRL_ID" />
      </JOIN>
      <FILTER table="ACCT" column="DATA_DUMP_DT"
                operator="GTE" value="01-NOV-2006" />
      <FILTER table="ACCT" column="DATA_DUMP_DT"
                operator="LTE" value="05-NOV-2006" />
    </MULTIJOIN>
  </JOINS>
</ANALYSIS>
```

This code executes the following query:

```
select count(1) from ACCT a, CUST_ACCT b
where a.ACCT_INTRL_ID=b.ACCT_INTRL_ID
and a.DATA_DUMP_DT>='01-NOV-2006' and a.DATA_DUMP_DT<='05-NOV-2006'
```

To filter for values that are null or not null, set the operator to EMPTY and the value to IS NULL or IS NOT NULL, respectively.

Join Count by Distinct Column

To determine a join count of the number of distinct values for a specified column within the joined tables, include the <DISTINCT_COUNT> element as content to the <MULTIJOIN> element. The targeted table and its column are set to the table and column attributes, respectively. The following sample demonstrates integration of the <DISTINCT_COUNT> element in the analysis:

```
<MULTIJOIN>
  <JOIN>
    <LEFT table="table name" column="column" />
    <RIGHT table="table name" column="column" />
  </JOIN>
  <!-- Optional. Can only have one DISTINCT_COUNT within
       the MULTIJOIN element. -->
  <DISTINCT_COUNT table="table name" column="column" />
```

```
</MULTIJOIN>
```

The <DISTINCT_COUNT> element is optional in the join analysis.

The following XML sample code illustrates use of the <DISTINCT_COUNT> element:

```
<ANALYSIS>
  <JOINS>
    <MULTIJOIN>
      <JOIN>
       <LEFT table="ACCT" column="ACCT_INTRL_ID" />
       <RIGHT table="CUST_ACCT" column="ACCT_INTRL_ID" />
      </JOIN>
      <FILTER table="ACCT" column="DATA_DUMP_DT" operator=
                  "EQ" value="02-NOV-2006" />
     <DISTINCT_COUNT table="ACCT" column="ACCT_TYPE_CD" />
    </MULTIJOIN>
  </JOINS>
</ANALYSIS>
```

This sample code executes the following query:

```
select count(DISTINCT a.ACCT_TYPE_CD)
from ACCT a, CUST_ACCT b
where a.ACCT_INTRL_ID=b.ACCT_INTRL_ID and a.DATA_DUMP_DT='02-NOV-2006'
```

Other Queries

The Data Analysis Tool also supports providing SQL queries directly in the analysis XML file. A query has two components: the query title and the query itself. As queries often contain characters that are "reserved" in XML, you should follow the example below for "escaping" the SQL to ensure that it does not become corrupted.

```
<QUERIES>
     <SQLQUERY title="title">
      select col1, col2 from some_table
      where some_condition
     </SQLQUERY>
</QUERIES>
```

The following XML sample code illustrates use of the <QUERIES> element:

```
<ANALYSIS>
  <QUERIES>
    <SQLQUERY title="FO Transaction Roles"><![CDATA[ select
       FOT.mantas_PRODUCT_TYPE_CD,
       FOTPS.PARTY_ROLE_CD, count(1) as RoleCt
       from FO_TRXN_STAGE FOT, FO_TRXN_PARTY_STAGE FOTPS
       where FOT.TRXN_INTRL_ID = FOTPS.TRXN_INTRL_ID
       group by FOT.mantas_PRODUCT_TYPE_CD, FOTPS.PARTY_ROLE_CD
       order by 1, 2]]></SQLQUERY>
 </QUERIES>
</ANALYSIS>
```

This code runs the query in the <SQLQUERY> element and writes the results to the output file. For SQL queries, the results are always in HTML. Your code can contain any number of <SQLQUERY> elements. The system runs each query in sequence after the other components of analysis are complete.

SQLQUERY Element Rules

Several cautions and notes are specific to the <SQLQUERY> element:

- If your query contains characters that XML standards reserve (for example, > or <), you must place your query within a CDATA block.
- Verify that no white space exists between the SQL query opening tag and the CDATA tags (for example, <![CDATA[. . .) and the closing tag (for example, . . .]]>).
- Processing extracts column headers in the output from the SQL query itself. When performing calculations in return columns, it is best to alias the return columns for output.
- Line breaks and comments in the SQL are acceptable, but you should use /* */ style comments in lieu of single-line comments for safety.
- The tool does not perform any schema-name substitution. Therefore, verify that any schema names match the database contents. The database user (for example, INGEST_USER) has aliases for most tables you may need to analyze. Thus, running the tool as INGEST_USER should prevent you from needing schema names in queries.

Using the Data Analysis Tool

After editing the configuration files, you can run the Data Analysis Tool as a foreground or background process.

The following table lists the XML input files delivered for use with the Data Analysis Tool.

Table 101. Data Analysis Tool XML Input Files

File	Description
analysis_aml.xml	Analysis configuration specific for data required by Anti-Money Laundering scenarios and Ingestion Manager operations to support them.
analysis_aml_ui.xml	Analysis configuration specific for data displayed in support of Anti-Money Laundering scenarios.
analysis_iaml.xml	Analysis configuration specific for data required by Institutional Anti-Money Laundering scenarios and Ingestion Manager operations to support them.
analysis_iaml_ui.xml	Analysis configuration specific for data displayed in support of Institutional Anti-Money Laundering scenarios.
analysis_bc.xml	Analysis configuration specific for data required by Broker Compliance scenarios and Ingestion Manager operations to support them.
analysis_bc_ui.xml	Analysis configuration specific for data displayed in support of Broker Compliance scenarios.

You can also create your own files using the provided files as a template. Place files that you create in the mantas_cfg directory that the DTD can locate. If you place your files in a different directory, you must modify the DTD reference in the XML files to qualify the path to the DTD.

To Run the Data Analysis Tool

Go to the <OFSBDF Installed Directory>/database/db_tools/bin directory and 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 102. 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 101) 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:

```
<OFSBDF 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 281, 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 103. Data Analysis Report Output

Section	Description
Table Count Summary	Contains the row count of each table in the configured database excluding the KDD, archive, and temp tables.
Field Distribution Summary Table	Groups by table the unique values for the identified fields and number of times each value occurs in the table. This summary table appears only in the report if the analysis for Distinct Values for Fields of Interest and Its Count was configured in the XML file. In addition, quotes enclose any values with padded spaces to identify spaces in the value.
Null Summary Count Table	Groups by table the number of nulls present and values with padded spaces for the identified fields in each table. This summary table only appears in the report if the analysis for Null and Padded Space Count has been configured in the XML file.
Referential Integrity Table Summary	Displays the join analysis, the number of rows returned between the joined tables, and the table count for each table being joined. This summary only appears in the report if the analysis for Join Counts has been configured in the XML file.
Query Results	Displays the results of queries specified in the QUERIES section of the analysis file.
SQL Report	Lists all of the SQL run to produce the other sections of the report.
Error Report	Displays any errors that occurred when any of the queries were performed.

Troubleshooting the Data Analysis Tool

Table 104 lists common Data Analysis Tool errors and their solutions.

Table 104. Troubleshooting Data Analysis Tool Errors

Error Message	Cause	Solution
java.io. FileNotFoundException <path & filename>	The system cannot find the file specified.	Verify the `install.cfg` file indicates the correct path.
java.lang. RuntimeException: Tables `<table 1>` and `<table 2>`	Tables `<table 1>` and `<table 2>` are already joined in this fashion.	In the `analysis.xml` file, remove duplicate join contents in the `<JOIN>` element.

Get Dataset Query with Thresholds Utility

Processing uses the Get Dataset Query with Thresholds Utility to store a dataset query in the Behavior Detection database with the threshold names and not with the threshold values. When the Behavior Detection engine executes a scenario, it substitutes the correct threshold values in the SQL query before submitting it to the database. Tracking of the query that executes in the database occurs only through the Behavior Detection engine log file when it runs in trace mode.

Using the Get Dataset Query With Thresholds Utility

Processing extracts the dataset query and uses it as input for tuning and execution plan generation.

Note: This utility does not recursively substitute thresholds in child datasets. Therefore, if a dataset being extracted has a reference to another dataset, manual extraction of that dataset must also occur.

The following table describes the parameters to provide with the `get_dataset_query.sh` script:

Table 105. Get Dataset Query Variables

Parameter	Description
`Dataset ID`	Unique identifier of the dataset for retrieval.
`Threshold Set ID`	Unique identifier of the threshold set for retrieval.

Executing the Get Dataset Query with Thresholds Utility

The following section provides instructions to execute the Get Dataset Query with Thresholds Utility.

To Execute the Get Dataset Query with Thresholds

To execute the Get Dataset Query with Thresholds Utility, follow the steps:

1. After the Alert Creator process completes, execute the `get_dataset_query.sh` script as follows:

   ```
   <OFSBDF 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:

   ```
   <OFSBDF Installed Directory>/database/db_tools/bin/get_dataset_query.sh <Dataset ID> <Threshold Set ID> query.sql
   ```

Scenario Migration Utility

You use the Scenario Migration Utility to migrate scenarios, datasets, networks, and associated metadata from the development environment to the production environment.

To provide a list of scenarios, datasets, or networks, you edit the `scnros.cfg`, `dataset.cfg`, or the `network.cfg` files prior to scenario extraction or loading.

The Scenario Migration Utility creates and migrates the following metadata files:

- **Scenarios:** The `<scenario catalog identifier>.<scenario id>.xml` file contains scenario metadata for core Behavior Detection tables. It also may contain scenario metadata for optional tables.
- **Datasets:** The `<dataset idDS>.xml` file contains dataset metadata for core Behavior Detection tables.
- **Networks:** The `<network>NW.xml` file contains network metadata for core Behavior Detection tables.

Note: When the Scenario Migration Utility extracts these files, you can version-control them or store them in the Oracle client's archival system.

To help avoid accidental loading of a scenario into the incorrect environment, the Scenario Migration utility enables you to *name* your source and target environments. On extract, you can specify the environment name to which you plan to load the scenario. If you attempt to load it to a different environment, the system displays a warning prompt.

Logs

The Scenario Migration Utility produces two log files (Figure 37 on page 286): `load.log` and `extract.log`. These files reside in the following location:

`<OFSBDF 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

The <OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg file contains common configuration information that Scenario Migration and other utilities require for processing. Figure 37 provides sample information from the install.cfg file that is specific to this utility.

```
################## SCENARIO MIGRATION CONFIGURATION ######################
#### GENERAL SCENARIO MIGRATION SETTINGS

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

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

#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/ECM6.2/ECM6.2_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=ECM62_B06_CASE

# 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
```

(Continued on next page)

(Continued from previous page)

```
extract.system.id=
# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
extract.schema.case=ECM62_B06_CASE
extract.schema.business=${schema.business.owner}
extract.schema.market=${schema.market.owner}
extract.user.miner=${load.user.miner}
extract.miner.password=${utils.miner.password}

# File Paths for Extract

#Specify the full path in which to place extracted scenarios
extract.dirname=/scratch/ofsaaapp/ECM6.2/ECM6.2_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/ECM6.2/ECM6.2_B06/BDP62_B06/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 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=${schema.mantas.owner}
load.schema.case=ECM62_B06_CASE
load.schema.business=${schema.business.owner}
```

(Continued on next page)

```
(Continued from previous page)
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/ofsaaapp/ECM6.2/ECM6.2_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 37. 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` shows (Refer to Table 106 on page 286). 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 106. 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, <OFSBDF Installed Directory>/database/db_tools/mantas_cfg/dep.cfg.
`sm.release`	Version of the Scenario Migration Utility.

Caution: OFSBDF 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 107. Scenario Extraction Parameters

Parameter	Description
extract.database.username	User used to connect to the database when extracting scenarios (DB_UTIL_USER).
extract.database.password	Password for the above user.
extract.conn.driver	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
extract.conn.url	Database connection string that the Scenario Migration Utility is to use.
extract.system.id	System from which the scenario was extracted.
extract.schema.mantas	MANTAS schema owner in the database into which extraction of the scenarios occurs (MANTAS).
extract.schema.business	Business schema owner in the database into which extraction of the scenarios occurs (BUSINESS).
extract.schema.market	Market schema owner in the database into which extraction of the scenarios occurs (MARKET).
extract.user.miner	DATA MINER schema owner in the database into which extraction of the scenarios occurs (KDD_MNR).
extract.miner.password	Password for the above user.
extract.dirname	Full path to the target directory where the utility writes extracted metadata (<OFSBDF Installed Directory>/database/ db_tools/data).
extract.backup.dir	Full path to the target directory where the utility writes backups of the extracted metadata (<OFSBDF 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 108. 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.

Table 108. Scenario Load Parameters (Continued)

Parameter	Description
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	MANTAS schema owner in the database in which loading of the scenario occurs (MANTAS).
load.schema.business	BUSINESS schema owner in the database in which loading of the scenario occurs (BUSINESS).
load.schema.market	MARKET schema owner in the database in which loading of the scenario occurs (MARKET).
load.user.miner	DATA MINER schema owner in the database in which loading of the scenario occurs (KDD_MNR).
load.miner.password	Password for the above user.
load.threshold.update	Threshold values from the incoming scenario. • Selecting N retains the threshold values from the 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 Metadata

To extract scenario, dataset, and network metadata, follow these steps:

1. Navigate to the following directory:
   `cd <OFSBDF Installed Directory>/db_tools`
2. Edit the metadata configuration files with identifying information for the scenarios, datasets, or networks for extraction:
   - `<scnro_ctlg_id>` in the `scnros.cfg` file

     and/or

   - `<scnro_ctlg_id>.<scnro_id>` in the `scnros.cfg` file

Note: Providing both `<scnro_ctlg_id>` and `<scnro_id>` in the `scnros.cfg` file allows finer granularity when extracting scenarios. If you provide both a scenario catalog ID and a scenario ID on a line, you must separate them with a period.

   - `<data_set_id>` in the `dataset.cfg` file
   - `<network_id>` in the `network.cfg` file
3. Execute the `sm_extract.sh` script in this order:
   a. Enter `sm_extract.sh dataset` to extract dataset metadata.
   b. Enter `sm_extract.sh scenario` to extract scenario metadata.
   c. Enter `sm_extract.sh network` to extract network metadata.

Loading Scenario Metadata

The `sm_load.sh` script loads translated XML table data files into the target database.

To Load Scenario Metadata

To 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 <OFSBDF 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_ctlg_id>` in the `scnros.cfg` file

Note: Providing both `<scnro_ctlg_id>` and `<scnro_id>` in the `scnros.cfg` file allows finer granularity when loading scenarios. You must separate values with a period per line.

- `<data_set_id>` in the `dataset.cfg` file
- `<network_id>` in the `network.cfg` file

3. Copy the XML files you plan to load into the directory that the load.dirname specifies in the `install.cfg` file.
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 OFSBDF system.

Caution: Not following the recommended best practices while loading scenarios to the targeted system may cause one or more sequence ID conflicts to occur, and your scenario will not be loaded. Once a conflict occurs, the metadata in the target environment needs to be corrected before the scenario can be successfully loaded.

To execute the recommended best practices, you should have an intermediate level knowledge of the scenario metadata, and be familiar with scenario patterns, thresholds, threshold sets, and so on. Basic SQL are required, as well as access privileges to the MANTAS schema. You must also be able to update records through SQLPLUS or a similar DB utility.

Process Overview

Scenario metadata is stored in many tables, with each table using a unique sequence ID for each of its records. If scenarios, thresholds, and scoring rules are modified in multiple environments using the same sequence ID range, then conflicts may occur when you migrate scenarios to these environments. To prevent conflict, you must set different sequence ID ranges in each of the environments.

The recommended best practices contain two basic points:

- Make changes in only one environment
- Separate the sequence ID ranges

Best Practices

Prepare to implement the recommended best practices before installing OFSBDF. Once the application is installed you should execute these steps to avoid scenario migration problems.

Make changes in only one environment

1. Only make changes to scenarios, thresholds, threshold sets, and scoring rules in the source environment.
2. Test and confirm your changes in the source environment.

3. Extract scenarios from the source environment and migrate them to all of your target environments.

Conflicting sequence IDs are often the cause errors when you migrate a scenario, so it is important to separate the sequence ID range.

Separate Sequence ID ranges

1. Review the `MANTAS.KDD_COUNTER` table, which contains all sequence ID ranges and current values.
2. Start your sequence ID ranger at 10,000,000 and separate each environment by 10,000,000. The OFSBDF 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 109, Table 110, and Table 111 list sequences involved and sample values.

Table 109. Environment 1 (Development)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
KDD_ATTR	ATTR_ID_SEQUENCE	10000000	10000000	19999999
KDD_AUGMENTATION	AGMNT_INSTN_ID_SEQ	10000000	10000000	19999999
KDD_DATASET	DATASET_ID_SEQUENCE	10000000	10000000	19999999
KDD_JOB	JOB_ID_SEQ	10000000	10000000	19999999
KDD_LINK_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	10000000	10000000	19999999
KDD_PTTRN	PTTRN_ID_SEQ	10000000	10000000	19999999
KDD_RULE	RULE_ID_SEQ	10000000	10000000	19999999
KDD_SCNRO	SCNRO_ID_SEQ	10000000	10000000	19999999
KDD_SCORE	SCORE_ID_SEQ	10000000	10000000	19999999
KDD_SCORE_HIST	SCORE_HIST_SEQ_ID_SEQ	10000000	10000000	19999999
KDD_TSHLD	TSHLD_ID_SEQ	10000000	10000000	19999999
KDD_TSHLD_HIST	HIST_SEQ_ID_SEQ	10000000	10000000	19999999
KDD_TSHLD_SET	TSHLD_SET_ID_SEQ	10000000	10000000	19999999

Table 110. 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 111. 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
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

Table 111. Environment 3 (PROD) (Continued)

TABLE_NM	SEQUENCE_NAME	CURRENT_VALUE	MIN_VALUE	MAX_VALUE
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.

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 OFSBDF tables. To avoid accidental loading of correlation rules into the incorrect environment, the Alert Correlation Rule Migration Utility enables you to *name* your source and target environments. On extract, you can specify the environment name to which you plan to load the correlation rule. If you attempt to load it to a different environment, the system displays a warning prompt.

Logs

The Alert Correlation Rule Migration Utility produces two log files (Figure 38 on page 296): `load.log` and `extract.log`. These files reside in the following location:

`<OFSBDF 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 the Alert Correlation Rules Migration Utility
- Extracting Alert Correlation Rule
- Loading Alert Correlation Rule

Configuring the Alert Correlation Rules Migration Utility

The `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Alert Correlation Rule Migration and other utilities require for processing. Figure 38 provides sample information from the `install.cfg` file that is specific to this utility.

```
################# CORRELATION RULE MIGRATION CONFIGURATION ######################


#### GENERAL CORRELATION RULE MIGRATION SETTINGS
# Specify the name of the configuration file containing the names of correlation rules to be
migrated. This property is specific to the Correlation Rule Migration Utility
corrRuleMig.CorrRuleFileNm=/users/mantas/database/db_tools/mantas_cfg/corrRule.cfg


#### EXTRACT (These properties are shared by Correlation Rule Migration Utility with the
Scenario Migration Utility)

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

# Specify the jdbc driver details for connecting to the source database
extract.conn.driver=${database.driverName}
extract.conn.url= jdbc:oracle: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 38. 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 `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg` shows (Refer to Table 112 on page 297). 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 112. 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. <OFSBDF 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.

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: <OFSBDF Installed Directory>/database/db_tools/mantas_cfg/corrRule.cfg

Configuring Alert Correlation Rule Extraction

The following table describes alert correlation rule extraction parameters.

Table 113. Alert Correlation Rule Extraction Parameters

Parameter	Description
extract.database.username	User to use to connect to the database when extracting alert correlation rules (DB_UTIL_USER).
extract.database.password	Password for the above user.
extract.conn.driver	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
extract.conn.url	Database connection string that the Alert Correlation Rule Migration Utility is to use.
extract.system.id	System from which the alert correlation rule was extracted.
extract.schema.mantas	MANTAS schema owner in the database into which extraction of the alert correlation rule occurs (MANTAS).
extract.dirname	Full path to the target directory where the utility writes extracted metadata (<OFSBDF Installed Directory>/database/ db_tools/data).
extract.backup.dir	Full path to the target directory where the utility writes backups of the extracted metadata (<OFSBDF Installed Directory>/ database/db_tools/data/temp).
caseschema.schema.owner	Points to the CASE schema for validating the case-related data defined in the correlation rules while extraction/loading of the rules into the database

Configuring Alert Correlation Rule Load

The following table describes alert correlation rule load parameters.

Table 114. Alert Correlation Rule Load Parameters

Parameter	Description
`load.database.username`	User to use to connect to the database when loading alert correlation rules (DB_UTIL_USER).
`load.database.password`	Password for the above user. This is set by the Password Manager Utility.
`load.conn.driver`	Database connection driver that the utility is to use (oracle.jdbc.driver.OracleDriver).
`load.conn.url`	Database connection string that the Alert Correlation Rule Migration Utility is to use.
`load.schema.case`	ECM CASE schema name. (CASEMNG) This information is not only used by Correlation Migration Utility, but also by database utilities which interact with the CASE schema.
`load.schema.mantas`	MANTAS schema owner in the database in which loading of the alert correlation rule occurs (MANTAS).
`load.system.id`	Name that is assigned to the system into which this instance of Alert Correlation Rule Migration loads metadata. The system compares the value for this setting to the target system in the metadata file.
`load.dirname`	Directory from which the system loads alert correlation rule(s) XML files.
`verify.target.system`	Check target name upon loading metadata files. ● Setting to N prevents Alert Correlation Rule Migration from checking the load.system.id against the target system specified when the alert correlation rule was extracted. ● Setting to Y enables this check. If the target environment in the XML file does not match the setting for load.system.id or the target environment is present in XML file but the load.system.id is blank then the system prompts you for an appropriate action. You can then continue with load or abandon the load, and you can apply the same answer to all other files in the session of Alert Correlation Rule Migration or allow the utility to continue prompting on each XML file that has a mismatch.
`corrRuleMig.loadHistory`	Load audit trail records on load. ● Setting to N prevents Alert Correlation Rule Migration Utility from loading the audit trail records from the XML file into the database. ● Setting to Y enables the system to load the audit trail records from the XML file into the database. **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 the 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 <OFSBDF 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 <OFSBDF 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 CASE schema table KDD_CASE_TYPE_SUBTYPE.
- The attribute value for subtype attribute in the XML tag <Case/> should exist in the CASE_SUB_TYPE_CD column of the CASE schema table KDD_CASE_TYPE_SUBTYPE.
- The attribute value for <subClassTagLevel1/> should exist in the CASE_SUB_CLASS_LVL1_CD column of the CASE schema table KDD_SUBCLASS1.
- The attribute value for <subClassTagLevel2/> should exist in the CASE_SUB_CLASS_LVL2_CD column of the CASE schema table KDD_SUBCLASS2.
- The CDATA section of the XML tag <AlertAttrOperations/> is validated as follows:
  - The valid operations should be one of BOTH, TO and FROM.
  - The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.
  - The TO and FROM alert operations can be used only to compare alert attribute values to each other, and not to a literal.
  - The FROM alert operation should always precede the TO alert operation.
  - The BOTH alert operator must be used to compare alert attribute values to a literal.
  - The expression can be nested to any arbitrary length provided it confirms to a general syntax:
    Operand Operator Operand [Logical_Operator Operand Operator Operand]
    For example,
    a) BOTH.SCORE_CT >= 0
    b) BOTH.SCORE_CT >= 0 AND FROM.SCORE_CT = TO.SCORE_CT

Note: A space character is expected between each Operand and Operator combination.

- The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
- The alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_REVIEW` and `KDD_REVIEW_FINANCIAL`.

- The CDATA section of the XML tag `<AlertCorrAttrOperations>` is validated as follows:
  - The Correlation Alert operation should be CORR.
  - The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.

Note: The `SCNRO_ALERT_CT` attribute works 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 MANTAS schema tables `KDD_ALERT_CORR` and `KDD_ALERT_CORR_SCNRO`.

Loading Alert Correlation Rule

The `sm_load.sh` script loads translated XML table data files into the target database.

To load alert correlation rule metadata, follow the 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 <OFSBDF 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 <OFSBDF 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 `CASE` schema table `KDD_CASE_TYPE_SUBTYPE`.
- The attribute value for *subtype* attribute in the XML tag `<Case/>` should exist in the `CASE_SUB_TYPE_CD` column of the `CASE` schema table `KDD_CASE_TYPE_SUBTYPE`.
- The attribute value for `<subClassTagLevel1/>` should exist in the `CASE_SUB_CLASS_LVL1_CD` column of the `CASE` schema table `KDD_SUBCLASS1`.
- The attribute value for `<subClassTagLevel2/>` should exist in the `CASE_SUB_CLASS_LVL2_CD` column of the `CASE` schema table `KDD_SUBCLASS2`.
- The CDATA section of the XML tag `<AlertAttrOperations/>` is validated as follows:
  - The valid operations should be one of BOTH, TO and FROM.
  - The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.
  - The TO and FROM alert operations can be used only to compare alert attribute values to each other, and not to a literal.
  - The FROM alert operation should always precede the TO alert operation.
  - The BOTH alert operator must be used to compare alert attribute values to a literal.
  - The expression can be nested to any arbitrary length provided that it confirms to a general syntax:
    Operand Operator Operand [Logical_Operator Operand Operator Operand]
    For example,
    a) `BOTH.SCORE_CT >= 0`
    b) `BOTH.SCORE_CT >= 0 AND FROM.SCORE_CT = TO.SCORE_CT`

Note: A space character is expected between each Operand and Operator combination.

  - The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
  - The alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_REVIEW` and `KDD_REVIEW_FINANCIAL`.
- The CDATA section of the XML tag `<AlertCorrAttrOperations>` is validated as follows:
  - The Correlation Alert operation should be CORR.
  - The valid operators can be =, !=, <, >, <=, >=, IN, NOT IN, AND and OR.

- The expression can be nested to any arbitrary length provided that it confirms to a general syntax: Operand Operator Operand [Logical_Operator Operand Operator Operand]
  For Example:
  a) `CORR. SCNRO_ID >= 0`
  b) `CORR.SCNRO_ID >= 0 AND CORR.SCNRO_ID = CORR.SCNRO_ID`

Note: A space character is expected between each Operand and Operator combination.

- The precedence of an operation may be depicted using a pair of parenthesis '(' and ').'
- The Correlation Alert attributes provided should be a valid column name from the MANTAS schema tables `KDD_ALERT_CORR` and `KDD_ALERT_CORR_SCNRO`.

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 OFSBDF as well as allows you to move configuration metadata between installations of OFSBDF. 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 need to be manually migrated.

Logs

The Investigation Management Configuration Migration Utility produces two log files—`load.log` and `extract.log`. These files reside at the following location:

`<OFSBDF 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 <OFSBDF 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 39 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 39. 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 115 on page 304). 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 115. 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 116. 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 117. 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.
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 Meta-data 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`

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.

Watch List Service uses three scripts to start, stop, and re-read the Watch List tables in case of changes. These scripts are placed in `ingestion_manager/scripts` folder. The scripts are:

- `startWatchList.sh` script: To start the watch list.
- `shutdownWatchList.sh` script: To stop the watch list.
- `initWatchList.sh` script: This script causes the Watch List Service to re-read the Watch List tables in case there have been any changes to it while the service has been running.

There is a polling agent that runs as part of the web service that will query the Watch List tables on a configurable interval (default of 10 seconds). This is to done to keep the service in sync with the Watch List Management Utility

Alert Processing Web Services

The Alert Processing Web service provides the ability to execute additional processing steps during a call to the existing PostAlert service operation, currently delivered with Investigation Management. Details on this service can be found in the *Services Guide*.

Instructions on Administering the Alert Processing Services

Alert Processing Service provides two scripts, one to start the service and one to stop the service. These scripts can be found in the <OFSBDF Installed Directory>/services/scripts directory. Details of the scripts are as follows:

- `startWebServices.sh`: Run this script to start the web service.
- `shutdownWebServices.sh`: Run this script to stop the web service.

Password Manager Utility

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

<OFSBDF 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 needs to be re-entered in order to be accepted. If it is not necessary to change a given password, press the Enter key to skip to the next password. The password that was skipped was not changed. The following are the users for which the script prompts for passwords, depending on what subsystems have been installed:

- Data Ingest User

- Database Utility User
- 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:

```
<OFSBDF Installed Directory>/changePasswords.sh <property name>
```

For example,

```
<OFSBDF 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 need to be entered and it is not possible to skip a password.

```
<OFSBDF Installed Directory>/changePassword.sh all
```

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

- Web Application User
- Data Miner User

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

Update Oracle Sequences

The OFSBDF framework uses Oracle sequences for BDF datamap component. To this end, OFSBDF provides the ability to maintain the Oracle sequences used in the Behavior Detection Framework. This utility must be compulsorily run by clients who are upgrading from Informatica to BDF atleast 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 OFSBDF to 6.2.1 version just need to 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.

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: Please don't make any modifications to the `KDD_ORACLE_SEQUENCE` metadata table unless it is being specifically asked to from the Oracle support team.

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

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

Clients who are upgrading from Informatica to OFSBDF 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.

APPENDIX A Logging

This appendix describes the mechanism that OFSBDF uses when logging system messages.

- About System Log Messages
- Message Template Repository
- Logging Levels
- Logging Message Libraries
- Logging Configuration File

About System Log Messages

The Common Logging component provides a centralized mechanism for logging Behavior Detection messages, in which the system places all log messages in a single message library file.

In the event that a log file becomes very large (one gigabyte or more), the system creates a new log file. The naming convention is to add *.x* to the log file's name (for example, `mantas.log`, `mantas.log.1`, `mantas.log.2`, so forth).

Note: The log file size is a configurable property; section *Log File Sizes* on page 315 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 118 outlines the logging levels that the Common Logging component supports.

Table 118. Logging Levels

Severity (Log Level)	Usage
Fatal	Irrecoverable program, process, and thread errors that cause the application to terminate.
Warning	Recoverable errors that may still enable the application to continue running but should be investigated (for example, failed user sessions or missing data fields).
Notice (default)	High-level, informational messaging that highlights progress of an application (for example, startup and shutdown of a process or session, or user login and logout).
Diagnostic	Fine-grained diagnostic errors—used for viewing processing status, performance statistics, SQL statements, etc.
Trace	Diagnostic errors—use only for debugging purposes as this level enables all logging levels and may impact performance.

The configuration file specifies enabling of priorities in a hierarchical fashion. That is, if Diagnostic is active, the system enables the Notice, Warning, and Fatal levels.

Logging Message Libraries

Some Behavior Detection subsystems produce log output files in default locations. The following sections describe these subsystems.

Administration Tools

The following file is the message library for the Administration Tools application:

```
$FIC_WEB_HOME/AM/admin_tools/WEB-INF/classes/conf/mantas_cfg/etc/
mantas_admin_tools_message_lib_en.dat
```

All messages numbers that this log contains must be within the range of 50,000 - 89,999.

Database

The following file is the message library for the Database:

```
<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/etc/
mantas_database_message_lib_en.dat
```

All messages numbers that this file contains must be within the range of 250,000 - 289,999.

Scenario Manager

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

```
<OFSBDF Installed Directory>/behavior_detection/toolkit/mantas_cfg/etc/
mantas_toolkit_message_lib_en.dat
```

All messages numbers that this section contains must be within the range of 130,000 - 169,999.

Services

The following file is the message library for the Services:

`<OFSBDF Installed Directory>/services/server/webapps/mantas/WEB-INF/classes/conf/mantas_cfg/etc/mantas_alert_management_message_lib_en.dat`

All messages numbers that this section contains must be within the range of 210,000 - 249,999.

Alert Management/Case Management

The following logs contain the message library for both Alert and Case Management applications:

Web server Logs

The following file is the message library for the Web server logs:

`$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:

`<OFSBDF 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:

- Administration Tools:
  `$FIC_WEB_HOME/AM/admin_tools/WEB-INF/classes/conf/mantas_cfg/install.cfg`
- Database:
  `<OFSBDF Installed Directory>/database/db_tools/mantas_cfg/install.cfg`
- Scenario Manager:
  `<OFSBDF Installed Directory>/behavior_detection/toolkit/mantas_cfg/install.cfg`
- Behavior Detection: `<OFSBDF Installed Directory>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg`
- Alert Management/Case Management:

- Web Server logs:
  Logging levels can be configured in the below mentioned file:

  `$FIC_WEB_HOME/conf/RevLog4jConfig.xml`

  In below mentioned tag.

  ```
  <root>
  <priority value ="debug" />
  <appender-ref ref="ConsoleAppender1"/>
  </root>
  ```

  - Below mentioned logger levels are available:

    DEBUG

    INFO

    WARN

    SEVERE

    FATAL

- Application Server logs:
  Logging levels can be configured in the below mentioned file:

  `$FIC_WEB_HOME/conf/RevLog4jConfig.xml`

  ```
  <root>
  <priority value ="debug" />
  <appender-ref ref="ConsoleAppender1"/>
  </root>
  ```

  - Below mentioned logger levels are available:

    DEBUG

    INFO

    WARN

    SEVERE

    FATAL

- Services: `<OFSBDF Installed Directory>/services/server/webapps/mantas/WEB-INF/classes/conf/mantas_cfg/install.cfg<OFSBDF Installed Directory>/services/mantas_cfg/install.cfg`
- Ingestion Manager: `<OFSBDF 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 need to see the SQL statements being
# executed.
# TRACE logging is used for debugging during development. Also only turn on
# TRACE if needed.
#log.fatal=true
#log.warning=true
log.notice=true
#log.diagnostic=true
#log.trace=true

# Specify whether logging for a particular level should be performed
# synchronously or asynchronously.
log.fatal.synchronous=false
log.warning.synchronous=false
log.notice.synchronous=false
log.diagnostic.synchronous=false
log.trace.synchronous=true

# Specify the format of the log output. Can be modified according to the format
# specifications at:
# http://logging.apache.org/log4j/docs/api/org/apache/log4j/PatternLayout.html
# NOTE: Because of the nature of asynchronous logging, detailed information
# (class name, line number, etc.) cannot be obtained when logging
# asynchronously. Therefore, if this information is desired (i.e. specified
# below), the above synchronous properties must be set accordingly (for the
```

(Continued on next page)

(Continued from previous page)

```
# levels for which this detailed information is desired). Also note that this
# type of detailed information can only be obtained for Java code.
log.format=%d [%t] %p %m%n
# Specify the full path and file name of the message library.
log.message.library=@WORKFLOW_LOG_MESSAGE_LIB_FILE@

# Specify the full path to the categories.cfg
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
```

Figure 40. 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 119 identifies the valid values for this property.

Table 119. 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 120 identifies the configurable properties for logging in an Oracle client's environment.

Table 120. 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 315 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 315 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 315 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 120. 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 (for example, FDT or MDT) a category. You can configure the destination of log messages for each component which Table 119 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 OFSBDF releases, but the text of the message can change. If a message ID does change, you can Refer to the appropriate `readme.txt` file for information about updated IDs.

APPENDIX B OFSBDF Software Updates

This appendix describes the application of OFSBDF software updates in Oracle Financial Services Behavior Detection Framework:

- OFSBDF Software Updates - Hotfix
- Hotfix Effect on Customization

OFSBDF 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 OFSBDF. 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/BDF

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 BDF Datamap Details

This appendix lists the BDF datamaps used in OFSAAI and a brief explanation of the each datamap.

BDF Datamaps

The following table provides a list of datamaps and description for each datamap. These datamaps are listed in order.

Table 121. BDF 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 preprocessors 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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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 populates Account Management Stage table using Service team, Investment Advisor, Employee To Account table.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
10114	Security_CIRRatingUpd	This datamap derives the column CIRRating and updates back to Security table.
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).

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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_InstructionDerivedEntDerivedAdd	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_NameMatchStageInsert	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_LinkStage	This datamap writes link-stage associations to various link tables in Derived Account Address Table.
10570	WatchListStagingTable2_WatchListStage2AcctExistence	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_WatchListStage2CBExistence	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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
10590	WatchListStagingTable2_WatchListStage2CustExistence	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.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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.

Table 121. BDF Datamaps (Continued)

Datamap Number	Datamap Name	Description
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.
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 121. BDF 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 121. BDF 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 121. BDF 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 121. BDF 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 121. BDF 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 121. BDF 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 121. BDF 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.
60190	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.

APPENDIX D 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 122. BDF 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 122. BDF 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 122. BDF 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 122. BDF 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 122. BDF 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
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
10930	SettlementInstruction_EntityAcctivityRiskUpd	NA	X	NA	X	NA

Table 122. BDF Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
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
11150	WireTransaction_TrustedFlagsUpd	X	X	X	X	NA

Table 122. BDF Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
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 122. BDF Datamaps

Datamap Number	Datamap Name	AML	Fraud	Insurance	AML Brokerage	Broker Compliance
50170	CustomerBalance_ActiveOTCTradeCt Upd	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	*	*	*	*	*

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

APPENDIX E Configuring Admin Tools

This appendix provides instructions on how to configure the Admin Tools feature.

Follow these steps for Admin 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 **FCCM** from the LHS menu.
3. Click **FCCM** and then click the **Administration** Menu and select the **Manage Parameters**.
4. Click **Common Parameters**.
5. In the Parameter Category drop-down, select **Used for Design**.
6. In the Parameter Name drop-down, select **Admin Tools**.
7. Set the Attribute 2 Value as follows: <PROTOCOL>://<AdminTools_WEB_SERVER_NAME> :<PORT>

Where:

- <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.

Note: Placeholder variables are mentioned between angle brackets. Update the placeholders with actual value.

APPENDIX F Mapping 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 template.
   - 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 123. 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 123. 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 123. 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:
   - Delete from KDD_ACTVY_TYPE_RSTRN where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')
   - Delete from KDD_ACTVY_TYPE_REVIEW_STATUS where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')
   - Delete from KDD_SCNRO_CLASS_ACTVY_TYPE where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')

- Delete from KDD_ROLE_ACTIVITY_TYPE where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')
- Delete from KDD_ACTIVITY_TYPE_CD where ACTVY_TYPE_CD in ('<sample action1>','<sample action2>')
- Commit the transactions.

Use the following tables to decide the actions to be unmapped.

Table 124. 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 124. 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 G Ingestion Using FSDF Datamaps

The Oracle Financial Services Data Foundation (OFSDF) is an analytical data warehouse platform for the Financial Services industry.

OFSDF combines an industry data model for Financial Services along with a set of management and infrastructure tools that allows Financial Services Institutions to develop, deploy, and operate analytical solutions spanning key functional areas in Financial Services, including:

- Enterprise Risk Management
- Enterprise Performance Management
- Customer Insight
- Financial Crime and Compliance Management

OFSDF is a comprehensive data management platform that helps institutions manage the analytical data life cycle from sourcing to reporting and business intelligence/BI using a unified, consistent platform and toolset.

FSDF Datamaps

The following provides a list of datamaps. These datamaps are listed in alphabetic order.

- Account
- Account Address
- Account Balance
- Account Customer Role
- Account Email Address
- Account Group
- Account Group Member
- Account Phone
- Account To Correspondent
- Account To Customer
- Anticipatory Profile
- Back Office Transaction
- Country
- Customer
- Customer Address
- Customer Country
- Customer Email Address

- Customer Identification Document
- Customer Phone
- Customer To Customer Relationship
- Customer To Markets Served
- Customer To Products Offered
- Employee
- Employee To Account
- Front Office Transaction
- Front Office Transaction Party
- Watch List
- Watch List Entry

Each of the FSDF datamap names corresponds to the respective FCCM DIS table. For example, the datamap *Account* extracts the information from corresponding FSDF table, and then transforms and loads into the FCCM Account table. For more information about the FCCM DIS tables, refer to the *Data Interface Specification (DIS)*.

The Reference Table Detail (REF_TABLE_DETAIL) in the Business schema is used to store code translation details. This table is loaded by the Oracle client.

In this example, the following columns are impacted:

- **CODE_SET_ID**: Contains the column name for which the translation is required prefixed with FSDF_. For example, the SRC_SYS_CD column will be FSDF_SRC_SYS_CD.
- **CODE_VAL1_NM**: Contains the Code Values and descriptions stored in FSDF as received from the source data.
- **CODE_DESC_TX**: Contains the Standard Code Values used by the FCCM application (For example, for the SourceSystem column (SRC_SYS_CD) the Standard Code Value is 'MAN' for FSDF description 'MANTAS'.

In this example the country code (CODE_SET_ID) "AUS" is being translated as AU.)

Table 125. Example: Reference Table Detail

CODE_SET_ID	CODE_VAL1_NM	CODE_VAL2_NM	CODE_DESC_TX	CODE_ADDL_INFO_TX
FSDF_CNTRY_CD	AUS		AU	
FSDF_CNTRY_CD	IND		IN	
FSDF_SRC_SYS_CD	FLEXCUBE at Citi US		FLX	
FSDF_SRC_SYS_CD	MANTAS		MAN	

For more information on BDF Datamaps functionality, refer to Chapter 4, *BDF Datamaps*, on page 93. For more information about populating the FSDF Stage schema, refer to the *Oracle Financial Services Data Foundation Guide*.

Parameter Configuration

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

Table 126. bdf.xml file parameters

Property Name	Description	Default
DIS.Source	Indicates the source of DIS records. Valid values are FILE for a DIS file and FSDW for one or more FSDW tables. In the current version, only FSDW is supported for a select set of DIS tables.	FSDW
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
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

Executing FSDF Datamap

The FSDF datamap uses BDF datamap functionality for processing. To execute the FSDF datamap, the BDF process uses the `<OFSBDF Installed Directory>/bdf/scripts/execute.sh script` to invoke individual components.

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

For example:

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

After running FSDF Ingestion, check the ingestion logs for any errors using the following path:

```
<OFSBDF Installed Directory>/bdf/logs
```

If there are any errors in ingestion, correct the errors and rerun the ingestion.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W