Oracle® Retail Xstore Suite 19.0/Oracle® Retail Merchandising Suite 19.1.000

Implementation Guide

Release 19.0

F32771-04

July 2022

ORACLE

Oracle Retail Xstore Suite 19.0/Oracle Retail Merchandising Suite 19.1.000 Implementation Guide, Release 19.0

F32771-04

Primary Author: Gerlinde Rust

enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

5 Integration Considerations

6 RTLog Generator On Premise

7 RTLog Generator Cloud

A Appendix: Xstore to Sales Audit Mapping Details

List of Figures

List of Tables

Send Us Your Comments

Oracle Retail Xstore Suite 19.0/Oracle Retail Merchandising Suite 19.1.000 Implementation Guide

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

This implementation guide describes the implementation steps that you should take when integrating the Xstore Suite with the Merchandising applications.

Audience

This Implementation Guide is intended for the integrators and implementation staff, as well as the retailer's IT personnel.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following Release documents:

- Oracle Retail Merchandising System documentation set
- Oracle Retail Price Management documentation set
- *Oracle Retail Xstore Suite Implementation and Security Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 19.0) or a later patch release (for example, 19.0.1). If you are installing the base release or additional patches, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Help Center (docs.oracle.com) Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Help Center (docs.oracle.com) at the following URL:

https://docs.oracle.com/en/industries/retail/index.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Help Center (docs.oracle.com)

Oracle Retail product documentation is available on the following web site:

https://docs.oracle.com/en/industries/retail/index.html

(Data Model documents can obtain them through My Oracle Support.)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Overview

This document describes integration between the Xstore suite of solutions and the Merchandising suite of solutions. For Merchandising, this document covers functionality supported in both the on premise and cloud service versions of the following products:

On Premise Solution	Cloud Service Solution
Oracle Retail Merchandising System (RMS)	Oracle Retail Merchandising Foundation Cloud Service (RMFCS)
Oracle Retail Sales Audit (ReSA)	Oracle Retail Merchandising Foundation Cloud Service (RMFCS)
Oracle Retail Pricing (RP)	Oracle Retail Pricing Cloud Service (RPCS)

For the Xstore Suite, this document covers functionality supported in the following products:

On Premise Solution	Cloud Service Solution
Oracle Retail Xstore Office	Oracle Retail Xstore Office Cloud Service
Oracle Retail Xstore Point of Service	NA

This integration is facilitated in part by the Oracle Retail integration solutions:

- Oracle Retail Integration Cloud Service for SaaS implementations
- Oracle Retail Integration Bus for on premise implementations

  Both the on premise and SaaS versions of the integration solutions include the integration components that are leveraged in the integration: bulk-data integration (BDI), message-based integration (RIB), and a centralized repository of foundation and pricing information that is accessed by service (Omnichannel Data Service or OCDS).

The integration of the Merchandising Suite and the Xstore Suite consists of two major data flows:

- Foundation and price data from Merchandising and Pricing flow into the Omnichannel Data Service (OCDS).

Xstore Office communicates directly with OCDS to request data-changes reported to OCDS by Merchandising and Pricing. When changes to Merchandising and Pricing data are detected, Xstore .mnt files are automatically generated and distributed for data loading into Xcenter and Xstore databases.

- Point of Service transactions from Oracle Retail Xstore Point of Service to the Merchandising Foundation Cloud Service Sales Audit module.

In combination, these data flows represent the round trip of data between the stores and headquarters. New items, other foundation data, and prices from headquarters are communicated to Xstore. Sales and returns from Xstore are communicated to Merchandising, where these transactions impact inventory. Merchandising further integrates summarized sales and inventory information from Xstore to other Oracle Retail applications, such as Planning and Analytics.

The details of the integration are covered in the remaining sections of this guide:

- Chapter 2, "Data Flow from Merchandising to Xstore using Omnichannel Data Service (OCDS)": This chapter describes the flow of data from the Merchandising applications to Omnichannel Data Service to the Xstore Suite.
- Chapter 3, "Transaction Flow from Xstore to Sales Audit": This chapter describes the flow of transactions from Xstore Point of Service to Sales Audit.
- Chapter 4, "OCDS Integration Configuration": This chapter provides information on the configuration changes that can be made for the integration.
- Chapter 5, "Integration Considerations": This chapter covers functional and technical points about the integration that need to be taken into consideration when implementing the integration.
- Chapter 6, "RTLog Generator On Premise": This chapter covers how to install, deploy, and configure the RTLog Generator application.
- Chapter 7, "RTLog Generator Cloud": This chapter covers the RTLog Generator Cloud.
- Appendix A, Appendix: Xstore to Sales Audit Mapping Details: This appendix provides tables that describe the mappings.

2

Data Flow from Merchandising to Xstore using Omnichannel Data Service (OCDS)

This chapter covers the data flow from Merchandising and Pricing to OCDS, where Merchandising and Pricing data can be requested by Xstore Office for loading into the Xcenter and Xstore databases.

OCDS can provide the Xstore Suite the following categories of data:

- Merchandise hierarchy
- Organizational hierarchy
- Store (including addresses)
- Dimension type (derived from item differentiator, or diff usage)
- Dimension value (derived from item diff usage)
- Items and Item Location
- Enterprise Item and Enterprise Item Location
- Item UPC
- Value Added Tax (VAT) rules and item associations
- Related items
- Initial prices
- Price changes
- Promotions
- Clearance prices

Conceptual Data Flow

Oracle Omnichannel Cloud Data Service (OCDS) is a data hub, enabling the Merchandising applications to share information with the Oracle Retail Omnichannel applications. OCDS is composed of three major components:

- BDI Batch Job Admin - Enables the flow of data into OCDS using the Oracle Bulk Data Integration (BDI) technology. Job Admin has a User Interface (UI) to support the management of BDI batch jobs.
- RIB Injector - Enables the flow of data into OCDS from the Oracle Retail Integration Bus (RIB).

- ORDS Web Services - Enables the data contained in OCDS to be accessed by Omnichannel applications, such as the Xstore Suite, through the use of RESTful web services.

Figure 2–1 illustrates the major system components that make up OCDS, and the interactions of the applications major actors.

Figure 2–1 OCDS Component Diagram

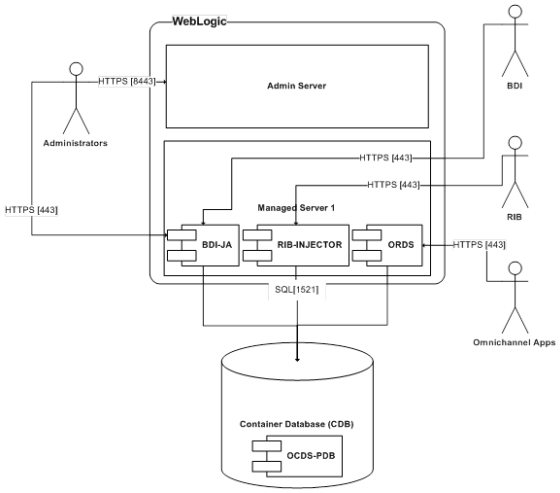
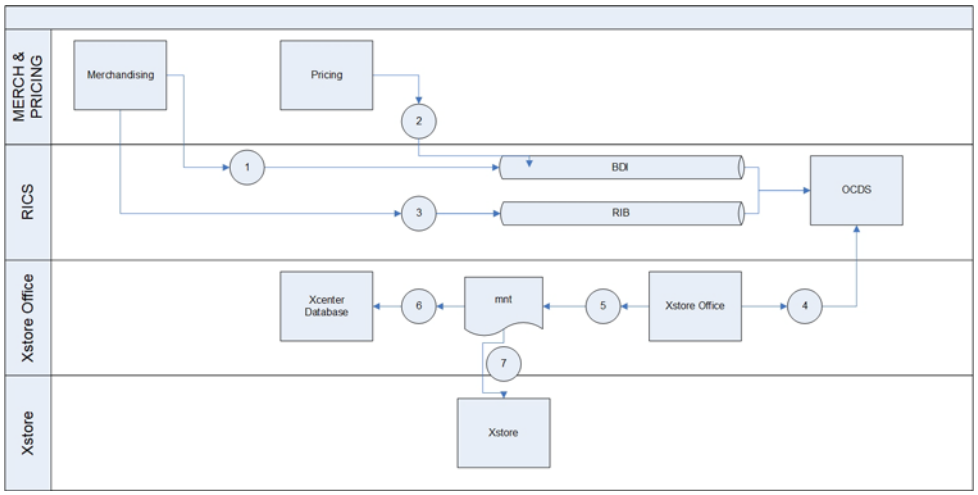


Figure 2–2 Conceptual Data Flow from Merchandising and Pricing to Xstore POS



The following steps describe the flow in Figure 2–2.

1. OCDS receives an initial load of Merchandising foundation data using BDI as the data transport. This is generally a one-time push of data over BDI into OCDS.
2. OCDS starts to receive, on-going, regularly scheduled pricing and promotion data using BDI as the data transport.
3. OCDS starts to receive, on-going, near-real-time updates of Merchandising data using the RIB as the data transport.
4. Xstore Office starts polling OCDS, at a regularly scheduled interval, to check for updates (for example, additions, deletions, and modifications) of Merchandising

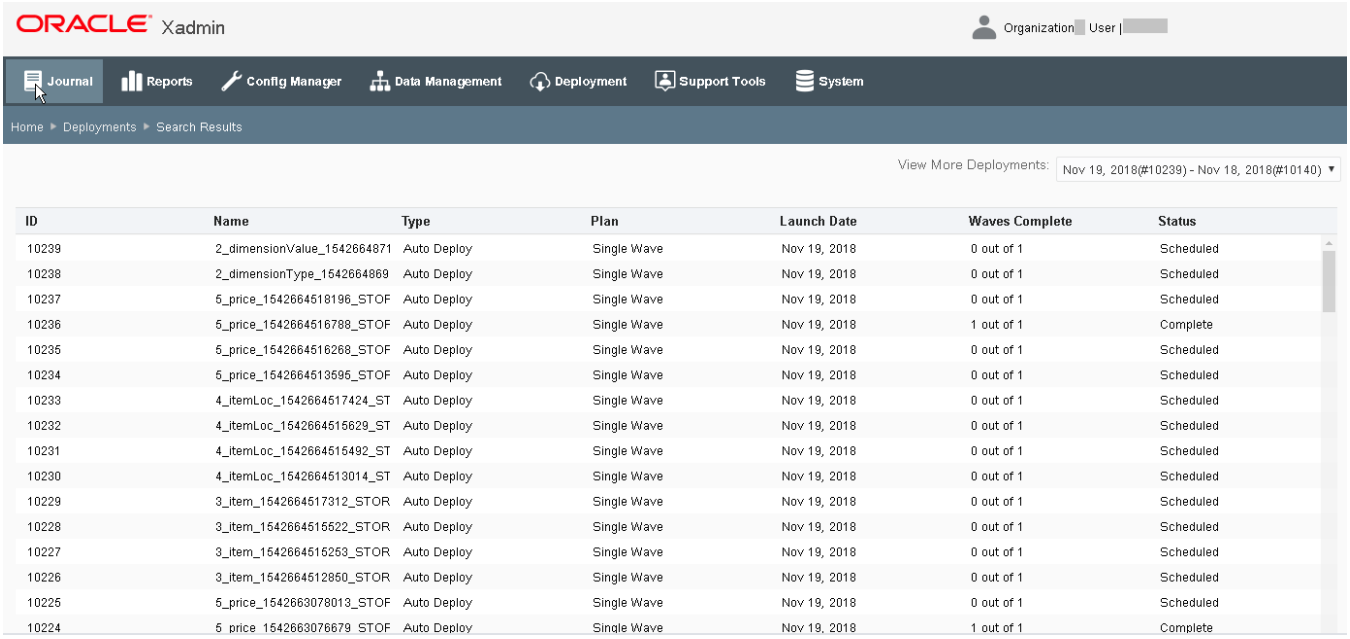
and Pricing data used by the Xstore Suite. Xstore Office communicates requests for changes to OCDS data by calling the OCDS REST web services.

5. When changes to OCDS data are detected, Xstore Office generates .mnt files containing the commands to update Xstore Suite databases. When .mnt files are generated, they are automatically deposited into the Xstore Office auto-drop folder for data-loading and distribution.
6. If any detected OCDS updates necessitate updating the Xcenter database then the appropriate .mnt file will be automatically dataloaded.
7. If any detected OCDS updates necessitate updating the Xstore database then the appropriate .mnt files will be deployed to the store where they can be dataloaded either immediately or at store close.

Xstore Office Deployments

When the OCDS integration is enabled, Xstore Office will automatically generate .mnt files with instructions for updating store databases when Merchandising or Pricing data changes are detected. Deployments of generated .mnt files, to be loaded at the store, are automatically created for either immediate or scheduled distribution. Each deployment status of the files is displayed in the Xstore Office Deployments screen.

Figure 2–3 Xstore Office Deployments



ID	Name	Type	Plan	Launch Date	Waves Complete	Status
10239	2_dimensionValue_1542664871	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10238	2_dimensionType_1542664869	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10237	5_price_1542664518196_STOF	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10236	5_price_1542664516788_STOF	Auto Deploy	Single Wave	Nov 19, 2018	1 out of 1	Complete
10235	5_price_1542664516268_STOF	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10234	5_price_1542664513595_STOF	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10233	4_itemLoc_1542664517424_ST	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10232	4_itemLoc_1542664515629_ST	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10231	4_itemLoc_1542664515492_ST	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10230	4_itemLoc_1542664513014_ST	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10229	3_item_1542664517312_STOR	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10228	3_item_1542664515522_STOR	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10227	3_item_1542664515253_STOR	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10226	3_item_1542664512850_STOR	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10225	5_price_1542663078013_STOF	Auto Deploy	Single Wave	Nov 19, 2018	0 out of 1	Scheduled
10224	5_price_1542663076679_STOF	Auto Deploy	Single Wave	Nov 19, 2018	1 out of 1	Complete

Phased Rollout of Stores [CLOUD ONLY]

Xstore Office's Integration Manager supports the use of Store Collections when configuring an OCDS integration. By associating a Store Collection with an OCDS Integration a retailer can identify which stores require data from OCDS. This feature can facilitate a phased-store rollout of OCDS data to a subset of existing stores in the Merchandising System. An OCDS integration using a Store Collection will limit the creation and deployment of mnt files to only those stores defined by the Store Collection.

Adding another store to an existing OCDS integration will enable incremental changes to start flowing to the store. A manual refresh from OCDS should be requested for the added store to seed it with a full set of OCDS data.

Manual Refresh of an Xstore Database from OCDS

The OCDS integration is designed to be fully automated; under normal conditions no manual steps are required to have Merchandising and Pricing data flow from OCDS into a store database. However, Xadmin's Data Publisher can be used to regenerate and redeploy .mnt files with OCDS data to a store if exceptional circumstances necessitate the refreshing of an Xstore database.

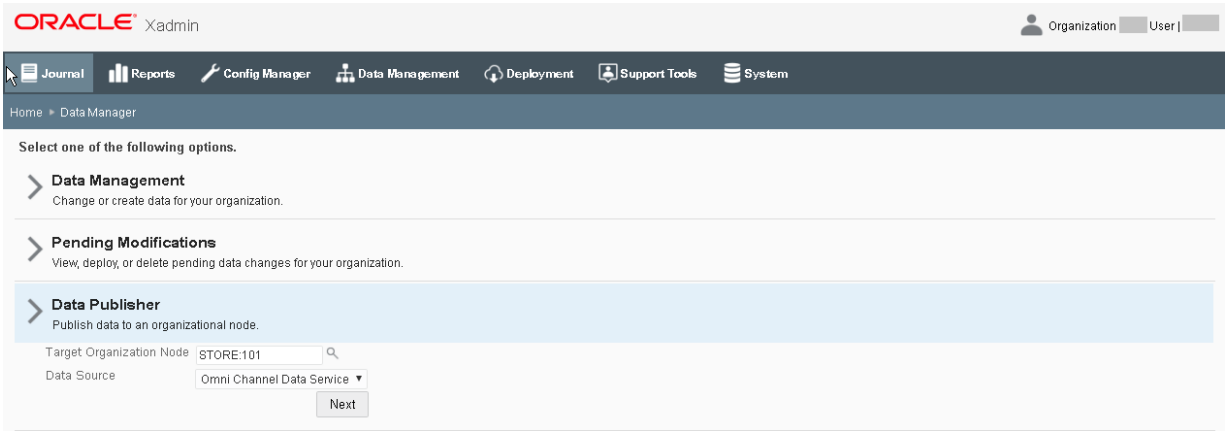
Use of the Data Publisher to replenish one or more types of OCDS data at a store will result in the purging of all existing OCDS-sourced data, followed by the loading of a full set of the most recent OCDS data.

When the OCDS integration is enabled in Xstore Office, the DataManager screen's "Data Publisher" option will include a "Data Source" drop down list, which includes the list option: Omni Channel Data Service.

To publish OCDS data to one or more stores:

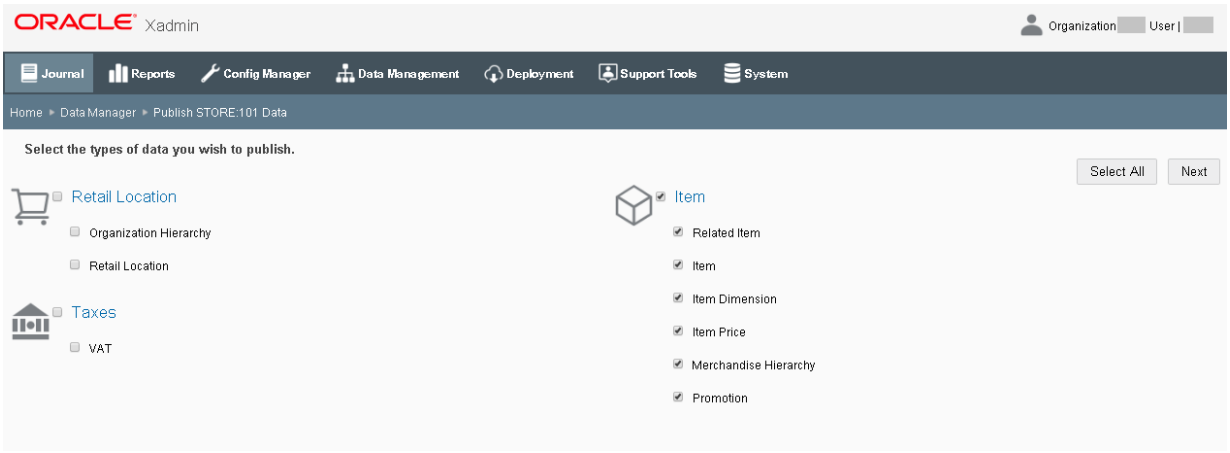
1. Choose the desired Organization Node for the target stores, select Omni Channel Data Service, and click **Next.**

Figure 2–4 Xstore Office - Data Publisher Option



2. Choose the type of data you wish to publish, then click **Next**.

Figure 2–5 Xstore Office - Data to Publish



3. Select a Download Priority, Immediate or Store Close, and click **Deploy**.

Figure 2–6 Xstore Office - Target Organization Node Pop Up

3

Transaction Flow from Xstore to Sales Audit

Xstore is the source of Point of Sale (POS) transactions, including but not limited to the following:

- Sales
- Returns
- Voids
- Cash management transactions
- Many store activity transactions

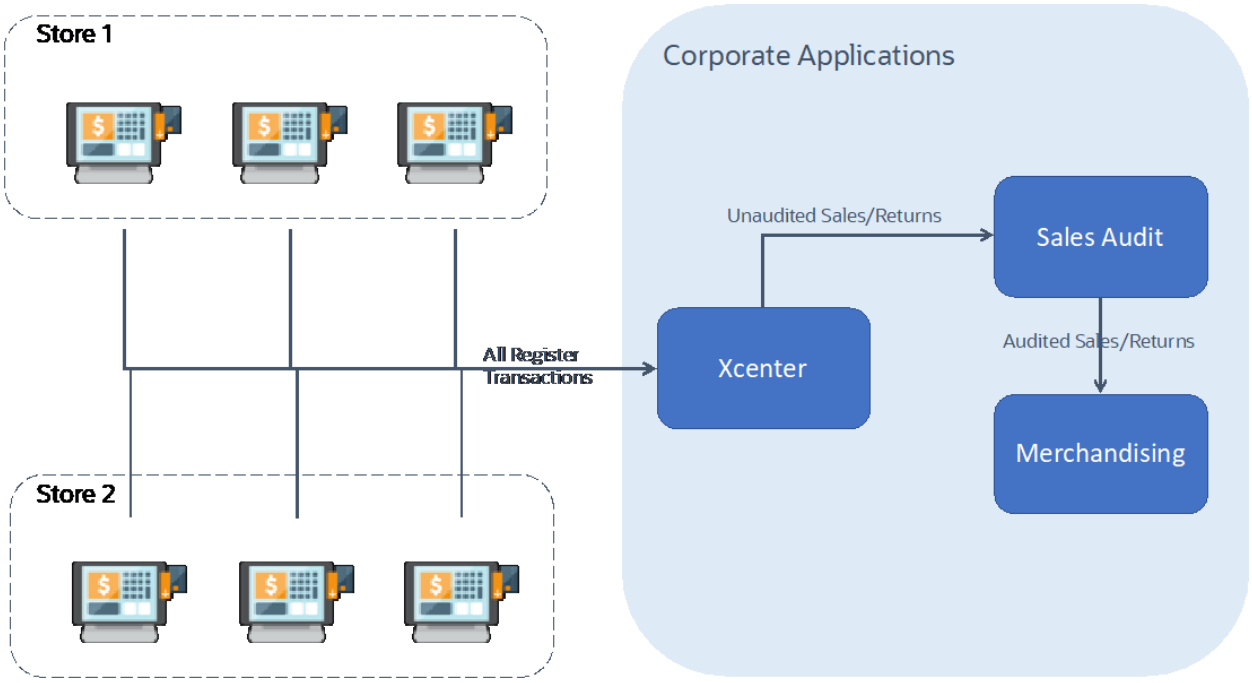
All transactions produced in Xstore are sent to Sales Audit. Sales Audit processing is primarily concerned with transactions that alter inventory or contain payment. Sales Audit loads other types of Xstore transactions (such as entering training mode, gift registry creation, and so on) into an OTHER transaction type for full visibility and to avoid gaps in the transactions sequence, but will not out of the box perform any audit functions on these OTHER types of transactions.

Sales Audit validates Xstore transactions that impact inventory (such as sales, returns, and customer orders) and exports the information to Merchandising to record the full financial and inventory impact.

Conceptual Data Flow

Figure 3–1 illustrates the transaction flow from Xstore to Sales Audit.

Figure 3–1 Xstore to Sales Audit Transaction Flow



The following steps describe the flow shown in Figure 3–1:

1. All Xstore registers replicate, or persist, all transactions to Xcenter. Note that this includes both customer related transactions (sale, return, void, and so on) and cash management/store operation transactions (paid in, no sale, change to training mode, and so on). Xcenter uses these transactions for activities such as cross location returns.
2. Xcenter broadcasts all transactions to Sales Audit in the form of RTLogs generated multiple times per day. For more information, see "Sales Audit saimptlog/i".
3. After successful totaling and auditing, Sales Audit sends all sale/return transactions to Merchandising, where the transactions impact perpetual inventory. For detailed information about uploadsales_all.ksh, see *Oracle Retail Merchandising Operations Guide, Volume 1 - Batch Overviews and Designs*.

Note: Integrating sales and returns data directly to Merchandising, bypassing Sales Audit, is not a supported integration.

Technical Implementation

The technical implementation of the data from Xcenter/Xstore to Sales Audit consists of three main components:

- Xstore Broadcaster
- Xstore RTLog Generator
- Sales Audit saimptlog/i

Xstore Broadcaster

The broadcast system in Xcenter provides a means to transmit POSLog data to other systems. The data is transmitted just as Xcenter receives it from the registers through the replication system, which is approximately in real-time. The temporal ordering of the POSLog data is also preserved, just as it is with the replication system.

There are a few systems which the base version of Xcenter can readily broadcast data to, simply by making configuration changes.

For more detailed information, see the following documents:

- Retail Reference Architecture available on My Oracle Support
- *Oracle Retail Xstore Technical Guide* available on My Oracle Support
- *Oracle Retail Xstore Suite Implementation Guide*

Xstore RTLog Generator

RTLog generator is a component that collects and aggregates broadcaster transactions and transforms them to the RTLog file format.

The on premise RTLog generator is packaged with Xstore, but is generally deployed in the same file system as Sales Audit.

For more information, see Chapter 6.

Sales Audit saimptlog/i

Sales Audit is the gateway for POS transactions to integrate to Oracle Retail headquarter systems. There are two Sales Audit sub-processes that can upload POS files:

- saimptlogi.c is used when loading sales incrementally throughout the day from stores, instead of just once per day. It validates files and directly inserts the transactions into the Sales Audit tables. This includes (as necessary) creating errors for the auditors to research and correct.
- saimptlog.c is used for the once a day import of data from stores. It validates POS files and creates Sql*Loader Files. This includes (as necessary) creating errors for the auditors to research and correct. A subsequent Sql*Load process loads the transactions and errors into the Sales Audit tables.

saimptlog and saimptlogi are built with the same shared code and vary only in their approach to physically loading data into the database. The programs are collectively referred to as saimptlog/i.

There are a number of regular prerequisites in the Sales Audit batch schedule which must be completed before POS transactions can be loaded. For more information about supporting batch jobs, see *Oracle Retail Merchandising Operations Guide, Volume 1 - Batch Overviews and Designs*.

For more detailed information about saimptlog/i and the RTLog file format, see the following documents:

- *Oracle Retail Merchandising Operations Guide, Volume 1 - Batch Overviews and Designs*

4

OCDS Integration Configuration

This chapter describes configuration options for the OCDS integration.

Xstore Office: OCDS Integration Communication

Xstore Office requests changes to Merchandising and Pricing data in OCDS by calling the REST web services.

[ON PREMISE ONLY] The Xstore Office xcenter.properties file contains properties for the OCDS URL and service path prefix.

[CLOUD ONLY] The properties for the OCDS integration are set up through the Xadmin Integration Management feature and are stored in the Xstore Office Cloud Service database.

> **Note:** The Integration Management feature only available for Xstore Office Cloud Service.

Table 4–1 Connection Properties

Properties	Description
`ocds.connectionURL`	A URL describing protocol, host name, and port of the OCDS Web Services. Example: `https://{hostname}:{port}`
`ocds.connection.servicePath.prefix`	The root service path for OCDS. Example: `/ords/{system}/omnichannel/v1`
`ocds.connection.username`	The encrypted username used to authenticate web service communication with OCDS. This property is only necessary for on-prem installations, cloud installations store this information in the Oracle Enterprise Manager (OEM).
`ocds.connection.password`	The encrypted password used to authenticate web service communication with OCDS. This property is only necessary for on-prem installations, cloud installations store this information in the Oracle Enterprise Manager (OEM).

Xstore Office: Omnichannel Data Service Configuration Options

The OCDS system settings are used to enable and configure the integration.

[ON PREMISE ONLY] The Xstore Office xcenter.properties file contains properties for the OCDS configuration options.

[CLOUD ONLY] Use the Xadmin Integration Management feature to set OCDS configuration options.

Table 4–2 OCDS Configuration Options

Setting	Description
OCDS Retain Job History	The number of days to retain OCDS Job History data in the Xadmin database.
OCDS Scheduled Job Interval Minutes	This is the frequency with which Xadmin requests data from OCDS. The value should be greater than or equal to 30 minutes.
OCDS On Demand Job Interval Minutes	This is the maximum amount of time that will elapse between when an on-demand job is created and is executed. The value should be greater than or equal to 5 minutes.
OCDS Orphan Data Protection Offset Seconds	This is the minimum amount of time data must age in OCDS before it can be visible to Xcenter. The offset helps to prevent related-data from becoming orphaned due to system latency. The value should be greater than or equal to 30 seconds.
OCDS Records Per Request Limit	This specifies the number of records to request in calls to OCDS. If no limit is defined then OCDS will determine the maximum number of records.
OCDS Retail Location: Till Accountability	This specifies if Retail Locations/Stores created from OCDS data are to use Till Accountability or not.
OCDS Retail Location: Default Locale	This specifies the Locale to use in Retail Locations created from OCDS data when a LANG_ISO_CODE is not defined for a location. This setting defines the value assigned to a Retail Location in the Xstore database if the store was not assigned a Locale in Merchandising
OCDS Item: Tax Group ID for non-taxable Items	This specifies the Tax Group ID assigned to non-taxable items created from OCDS data.
OCDS Item: Include future date to determine VAT code with the greatest active date?	This specifies if future date should be included to determine VAT code with the greatest active date.
OCDS VAT Rounding Code	Specifies how the system will round when calculating the tax amount. Valid values are CEILING, DOWN, FLOOR, HALF_DOWN, HALF_EVEN, HALF_UP, and UP. For example, `ocds.vatRoundingCode=HALF_UP`
OCDS VAT Rounding Digits	Specifies how many decimal places to round to when calculating the tax amount.The minimum rounding digit is 0 and the maximum rounding digit is 10. For example, `ocds.vatRoundingDigits=2`
OCDS VAT Rounding at Transaction Level	Specifies the ability to round at transaction level when calculating the tax amount. When setting is set to `true`, the calculated tax will be rounded at the transaction level. For example, `ocds.vatRoundingAtTransLevel=true` When setting is set to `false`, the calculated tax is rounded at the line item level.

Merchandising Chain to Xstore Organization Mapping

The Xstore Suite provides the opportunity to organize data by Organization, where centralized and store-level data can be isololated according to a retail organization

structure. The Merchandising Suite supports the use of one or more Chains, where a chain can be used to group various store formats.

The OCDS Integration in the On Premise and Cloud Xstore Office requires that at least one Merchanising Chain be mapped to an Xstore Organization. This configuration must not be changed once the mapping is established.

OCDS Database Tables

There are three database tables in the Xadmin database that are exclusively used for integration with OCDS. See the *Oracle® Retail Xstore Point-of-Service Software Database Dictionary* for complete details on these tables.

Table 4–3 OCDS Database Tables

Table Name	Description
OCDS_JOB_HISTORY	When enabled in Xadmin, an OCDS job is executed by scheduled-interval or on-demand, to detect foundation data changes in OCDS server and extract them out into .mnt files. An entry in this table records the status, start/end time and other information of a job executed.
OCDS_ON_DEMAND	An entry in this table represents an on-demand OCDS job request. On-demand jobs can be system generated (as in the case of new store detection), or user generated (from the Xadmin UI).
OCDS_SUBTASK_ DETAILS	An OCDS job executes a list of subtasks. Each subtask represents a foundation data area to detect and extract out changes. The table defines metadata for all subtasks. For a scheduled job, each and every active subtask is executed. For an on-demand job, a subset of subtasks specified for the job is executed.

The only table that requires seed data is the OCDS_SUBTASK_DETAILS; the other two tables will populate during OCDS operations.

Seed Data for VAT and Non-VAT Organizations

The OCDS_SUBTASK_DETAILS table contains metadata that controls how .mnt files are named and deployed for data-loading into Xstore Suite databases. In general, the out-of-the box seed data values for this table are appropriate for most organizations; however, if not using VAT then the ACTIVE column for the VAT SUBTASK_ID should be set to 0.

Figure 4–1 OCDS Subtask Details Table

ORGANIZATION_ID	SUBTASK_ID	FILENAME_PREFIX	QUERY_BY_CHAIN	QUERY_BY_ORG_NODE	DESTINATION	DOWNLOAD_TIME	APPLY_IMMEDIATELY	ACTIVE	FAMILY
1	CLEARANCE_PRICE_CHANGE	5_clrpc	0	1	ALL	IMMEDIATE	1	1	PRICE
1	DIMENSION_TYPE	2_dimensionType	0	0	ALL	STORE_CLOSE	0	1	DIMENSION
1	DIMENSION_VALUE	2_dimensionValue	0	0	ALL	STORE_CLOSE	0	1	DIMENSION
1	ENTERPRISE_ITEM	3_itemCorp	0	0	XCENTER_ONLY	(null)	0	1	ITEM
1	ITEM	3_item	0	1	XSTORE_ONLY	STORE_CLOSE	0	1	ITEM
1	ITEM_LOC	4_itemLoc	0	1	ALL	STORE_CLOSE	0	1	ITEM
1	ITEM_NODE	4_itemNode	0	1	XCENTER_ONLY	(null)	0	0	ITEM
1	MERCH_HIERARCHY	2_merchHier	0	0	ALL	STORE_CLOSE	0	1	MERCH_HIERARCHY
1	ORG_HIERARCHY	1_orgHier	1	0	ALL	STORE_CLOSE	0	1	ORG_HIERARCHY
1	PRICE	5_price	0	1	ALL	IMMEDIATE	1	1	PRICE
1	PROMOTION	5_promo	0	1	XSTORE_ONLY	STORE_CLOSE	0	1	PROMOTION
1	REGULAR_PRICE_CHANGE	5_regpc	0	1	ALL	IMMEDIATE	1	1	PRICE
1	RELATED_ITEM	4_relatedItem	0	1	ALL	STORE_CLOSE	0	1	RELATED_ITEM
1	RETAIL_LOCATION	2_retailLoc	1	0	ALL	STORE_CLOSE	0	1	RETAIL_LOCATION
1	VAT	2_vat	0	0	ALL	STORE_CLOSE	0	1	VAT

Download: Immediate vs Store Close

The OCDS_SUBTASK_DETAILS table includes the column DOWNLOAD_TIME, which specifies when .mnt files should be downloaded for data loading at the store. The column contains null for those OCDS Subtasks that only populate the Xcenter database. The out-of-the-box seed data is configured so that only Pricing data will flow to the store immediately.

5

Integration Considerations

This chapter provides the considerations that should be accounted for when implementing these solutions to minimize errors in data movement between solutions, as well as to call out some functional differences in the solutions that may limit the use of functionality in one or the other solutions.

Foundation Data

There are a number of basic data elements that are common between the two solutions but which are not part of the integration. This is because they are generally a one-time set up at initial implementation with only infrequent updates afterward. However, because this data is foundational to how the solutions work, it is critical that they are set up properly. These data elements fall into a couple different categories:

- Seed Data
- Transaction Details
- Currency Exchange Rates

Seed Data

Seed data refers to data that is loaded into both solutions on implementation by Oracle Retail provided install scripts. These are coordinated between solutions as part of the base installation, but if any updates are made in one solution to add or remove items, the corresponding change should be made in the other solution. Data elements that fall into this category are:

- Currency codes
- Country codes
- Units of measure

Transaction Details

The mapping of transaction details from Xstore POSlog to Sales Audit RTLog depends on the mappings of valid values. These mappings are detailed in Appendix A, "Appendix: Xstore to Sales Audit Mapping Details". It is critical that the mappings are complete. If additional valid values are configured for Xstore in the RTLogMappingConfig.xml, they must also be configured for Sales Audit for the appropriate code types.

Similar to seed data, some initial data is provided for the data entities in this category, but this is an area that is more commonly configured for retailers based on their specific business processes. On initial implementation, the configurations in both

Xstore and Sales Audit should be made to be in synch, with any changes made post-implementation continuing to be made in both solutions. The entities in this category include:

- Transaction Types
- Tender Types
- Tender Total IDs
- Item Types
- Reason Codes
- Item Statuses
- Sales Types

See the Appendix A, "Appendix: Xstore to Sales Audit Mapping Details" for details on configuring and mapping these entities.

Currency Exchange Rates

Exchanges rates for currencies are not one of the things integrated between Merchandising and Xstore, as Merchandising is not considered the system of record for this information at a retailer - generally that comes from the financials solution. However, if you require currency exchange rates in Xstore, then it expected that the same source of data used for exchange rates in Merchandising will also be used to load those rates into Xstore, in order to ensure both solutions are operating with the same information and to prevent a financial impact from occurring due to differences in the rates used. Tender exchange transactions that occur in Xstore, where a customer is given USD in exchange for CAD, will be mapped to the transaction type OTHER in Sales Audit.

Stores

By default, Xstore is configured to allow four digit store IDs, but it can be configured to hold up to 5 digit store numbers in the SequenceConfig.xml. Although Merchandising can hold up to a 10 digit store ID, when integrating with Xstore, it is strongly recommended that only four or five digit location IDs are used. Custom modifications would be required to Xstore to support larger store IDs.

Additionally, latitude and longitude information that is used by Xstore to determine nearby stores for its inventory lookup function are not available as part of the integration from Merchandising. If you wish to use this functionality in Xstore, the record type, RETAIL_LOCATION_COORDINATES, is available to DataLoader to populate the latitude and longitude of stores using the .mnt format.

All stores in Merchandising that you expect to get a sales file from Xstore should be set up to have Integrated Sales flag set to checked (Yes) as part of the store setup. Additionally, the store level attribute Unique Transaction Number should be set to Register for all Xstore integrated stores. These are both used by Sales Audit to know which stores to expect to receive files and how to audit the data in the files.

Merchandise Hierarchy

Xstore supports up to 4 levels of the merchandise hierarchy, which will be populated by the bottom four levels of the merchandise hierarchy from Merchandising - group, department, class, and subclass. In Merchandising, the class ID displayed to users is unique only when combined with its department ID. Similarly, the displayed subclass

ID is only unique when combined with its department and class. However, instead of using the composite key in the integration with Xstore, the unique key that is held in the Merchandising tables for class and subclass used in the OCDS and is written into .mnt files. This unique ID is not visible to users of Merchandising.

The following is the Xstore Merchandise Hierarchy Spring bean configuration that should be used to map the bottom four Merchandising application's levels to the Xstore's four levels:

```
<bean id="merchandiseHierarchyInfo"
class="dtv.pos.common.MerchHierarchyLevelInfo">
      <property name="numberOfLevelsAvailable" value="4" />
      <property name="level1Code" value="GROUP" />
      <property name="level2Code" value="DEPARTMENT" />
      <property name="level3Code" value="CLASS" />
      <property name="level4Code" value="SUBCLASS" />
</bean>
```

Note: When overriding the default Xstore Merchandising Hierarchy levels, the default resource bundle text values should also be overridden accordingly.

Items

This section lists considerations regarding items.

Merchandise Items

Physical merchandise items should be mastered in Merchandising and use the integration described in this document to flow the data to Xstore. Xstore Office should not be used to create physical items in order to prevent errors when loading sales data into Sales Audit where the item being sold or returned cannot be identified and accounted for in Merchandising.

Non-Merchandise Items

If using non-merchandise items, such as warranties, fees, and services, in Xstore, special attributes are required that are not available in Merchandising. Therefore to configure these items, the following approach is required:

1. Create the non-merchandise item in the Xstore Office UI, specifying the required attributes to control its behavior in Xstore.
2. Create an item in Merchandising with the same ID as that created in Xcenter. The item created in Merchandising should be set up as a non-merchandise item to prevent it from being re-exported to Xstore.

The creation of the item in Merchandising will prevent any errors from occurring in the auditing process. Any maintenance on the non-merchandise items should occur in Xstore Office going forward.

To allow end users to create non-merchandise items, but be prevented from creating or editing merchandise items in Xstore, the `CFG_MERCH_ITEMS` privilege should not be granted to any users. The merch items option will still be on the screen, but it will not be accessible.

Kit/Pack Items

Kits, or pack items in Merchandising, are items that contain multiple components but are sold as a single unit. As part of the standard item integration, Xstore does not import the component level information from Merchandising, so these items will appear as standard items in Xstore and the component details will not be available.

Differentiators

Differentiators are used in Merchandising to define how a transaction level item (for example, SKU) differs from its parent (for example, style). For example, a differentiator might be a color, size, or flavor for an item. In Xstore, differentiators are called dimensions. Merchandising supports up to 4 differentiators/dimensions for items, while Xstore can support only three. It is strongly recommended that the 4th differentiator is not used when implementing Merchandising with Xstore, as it will be ignored in the integration.

Additionally, in Merchandising an item can be assigned differentiators without having a parent (style) associated with it. This could be used for hardline or grocery items to indicate the color or size of an item for reporting purposes, for example. However, in Xstore dimensions are primarily used to allow a user to determine the sellable SKU by entering a style ID and selecting the valid dimensions (usually color and size). Therefore, if an item does not have a parent, the dimensions sent from Merchandising will be ignored and will not be visible in Xstore.

Product Restrictions

Product restrictions can be set up in Merchandising to indicate limitations on certain products. For example, a restriction may be set up to limit alcohol from being sold to customers under a certain age. Product restrictions are not currently supported in the integration to Xstore. Custom integration would be required to communicate this information from Merchandising to Xstore.

Related Items

Merchandising has a concept of related items that can be used to define items that are substitutes for one another, or that could be used to cross-sell or up-sell to a customer when purchasing the main item. Substitute items from Merchandising are mapped to the Xstore substitute items to indicate items that may be substituted or offered in place of another item.

The cross-sell and up-sell types of related items are mapped to Xstore's Attached Items and configured as prompt-to-attach. Only transaction level related items are used by Xstore. Those created at the parent item level (for example, style) in Merchandising are ignored.

Other Item Attribute Notes

- Item Restocking - unlike Xstore, Merchandising does not have a flag that indicates whether an item is subject to an item restocking fee, nor the ability to define what an item's fee would be. Therefore, Xstore would not have the ability to prompt for a restocking fee during returns.
- Xstore can support prorated refunds for items, but to do so requires specific attributes sent for an item, which are not currently available in Merchandising. Therefore, this function would not be available in Xstore.

- Merchandising has the ability for retailers to extend the available item attribution by creating user defined attributes and custom flex attributes. Although included in the available data from Merchandising, these are currently not used by Xstore.
- Translated item descriptions are available from Merchandising as part of the integration but are not currently used by Xstore. Xstore uses the item level description (which is communicated in the primary Merchandising language) for Xoffice and the item/location level descriptions for the store in Xstore. If you have the requirement to send item descriptions in different languages to your stores, it is recommended that the item/location level description in Merchandising be updated to show the localized item description.

Tax

This section describes considerations regarding taxes.

Value Added Tax (VAT)

Merchandising integration includes VAT rates and the regions in which the stores have been classified for companies with operations in geographies where this type of tax is applicable. For retailers that have operations in both VAT and non-VAT regions - such as stores in the US and Canada - non-VAT regions are configured as exempt in Merchandising and communicated as such to Xstore. For more information on configuration for VAT in Xstore, see the *Oracle Retail Xstore Technical Guide*.

When Merchandising sends VAT rate updates for an item, it also includes the active date for the rate to be applicable. Retailers sometimes enter new VAT rates in advance for future planning. However, Xstore currently does not support an active date for VAT code and will ignore the active date sent, which means any new codes will go into effect immediately. Therefore, it is recommended that retailers enter the VAT code changes in Merchandising only when needed.

Note: Buying from a VAT store and returning to a non-VAT store (and vice versa) is not supported in Xstore.

US Sales Tax

Merchandising does not provide US Sales tax information to Xstore; it is assumed that product tax groups are imported into Xstore from a third-party system using Xstore Point of Service DataLoader and .mnt files.

In standalone mode, DataLoader has to be executed twice, first to import Merchandising files and second to import this .mnt file. If they are placed together in the download directory, .mnt files always get loaded first.

In an integrated environment with Xstore Office/Xenvironment, the retailer has to drop the Merchandising .zip file first, and wait until that .zip file is processed by Xstore Office before dropping this .mnt file. This guarantees the .mnt file is imported into Xcenter after all Merchandising files, and is staged for store deployment with a deployment ID greater than the deployment ID of the Merchandising files.

After loading Merchandising data, the following additional steps are required to configure sales tax using the .mnt file format:

1. Set up sales tax rules. To set up a simple rate based tax rule, use existing record types TAX_LOCATION, TAX_AUTHORITY, TAX_GROUP, TAX_GROUP_RULE, and TAX_RATE_RULE to populate tax tables tax_tax_loc, tax_tax_authority, tax_

tax_group, tax_tax_group_rule, and tax_tax_rate_rule. For more details on tax rule configuration, see the TAXING section in the *Oracle Retail Xstore Point of Service Host Interface Guide* available on My Oracle Support.

2. Set up retail store and tax location mapping in table tax_rtl_loc_tax_mapping using existing record type TAX_RETAIL_LOCATION_MAPPING. For more details on this record type, see the TAXING section in the *Oracle Retail Xstore Point of Service Host Interface Guide* available on My Oracle Support.
3. ITEM_TAX_GROUP is used to update the item record in the itm_item_options table with sales tax group ID. This .mnt file has to be imported after the Merchandising data import. There is no built-in mechanism in DataLoader or Xstore Office to ensure this ordering. It has to be enforced by retailer manually.

Note: There is also a configuration of tax data that must be done in Sales Audit when using the US Sales Tax configuration. See Appendix: Xstore to Sales Audit Mapping Details for more details.

Inventory

Inventory functionality in Xstore should be disabled when implemented with Merchandising. No inventory information is integrated between Xstore and Merchandising other than sales related data and it is assumed store inventory is managed in another application, such as Oracle Retail Store Inventory Management (SIM) or Store Inventory and Operations Cloud Service (SIOCS), which is also integrated with Merchandising. Therefore, when these systems are all part of a retailer's implementation, the .sim entry in the configuration path should be used in Xstore to turn off Xstore inventory functionality. Inventory integration outside of sales and returns between Merchandising and Xstore is not supported.

Serialized Inventory

Merchandising supports the concept that an item can be a serialized item in one store, but not in another, however in Xstore, the designation for whether or not an item is serialized is held at the item level, so there is not any differentiation by store. This means that if the serialized flag actually varies by location for an item in Merchandising, the last location to be dataloaded by the integration code sets the item level serialized flag in Xstore.

Note: Merchandising does not support serialized inventory at this time. It only flags items as being serialized or not.

Customer Orders

When customer orders are initially captured in Xstore, the Xstore RTLog generator sets the Fulfillment order number in the RTLog to UNKNOWN, as the fulfillment order number is not known at the time the order is created, because information has not yet been sent to the order management system.

In-Store Orders

Orders taken in the store on behalf of a customer that do not go through an Order Management System (OMS) for fulfillment will include only a customer order number,

but not a fulfillment order number when it the transactions related to it are integrated to Sales Audit.

Recognition of a Sale

For customer orders, Xstore can be configured to recognize a sale at either the time the order is place or at the time of pickup. Integration with Merchandising requires that this configuration be time of pickup, which corresponds to when inventory is decremented from the store, in order to prevent out of synch issues between actual store inventory and what is shown in Merchandising.

In order to configure this in Xstore, the following settings should be set to false (which is the default) under both `<Layaway>` and `<SpecialOrder>` in `SystemConfig.xml` (whose settings are also controllable in Xadmin):

```
<Layaway>
<BookAsSaleOnSetup dtype="Boolean">false</BookAsSaleOnSetup>
<SpecialOrder>
<BookAsSaleOnSetup dtype="Boolean">false</BookAsSaleOnSetup>
```

Pricing

In both Merchandising and Pricing the data type for retail prices is NUMBER(20,4), but in Xstore, the standard is to use a data type of NUMBER(17,6). This applies to the following item prices:

- Selling Unit Retail (from Merchandising and Pricing)
- Manufacturer's Recommended Retail (from Merchandising)

If an Pricing retail value is over 17 digits, dataloading into Xstore will fail. Non-failing records from the same file will continue to be loaded.

Multi-Unit Pricing

Pricing and Xstore have different approaches to multi-unit pricing. Xstore converts multi-unit prices to single price, but cannot accurately do this without rounding information which varies from store to store. As such, Pricing's regular price update information for multiple units is not supported in this integration.

Promotions

There are certain features of promotions that are supported in Pricing that are not supported in Xstore. These features should be configured off in Pricing to prevent users from creating promotions that cannot be executed. These are the features that should be configured off:

Allow Reward List Exclusions on Transaction Offers

This Pricing system option should be unchecked, as Xstore cannot support receiving reward list exclusions for transaction offers.

Allow Supplier Site and Brand for Offer Item Selection

This Pricing system option should be unchecked, as Xstore does not have information about the brand or supplier for an item; therefore, adding or excluding items by this type of criteria on a promotion cannot be supported in the integration.

Pricing Application - Promotion Offer Distribution Rules (OFDR)

Pricing allows users to defined how the discount should be distributed to items on a customer's purchase by selecting a distribution rule as part of the promotional offer creation. However, Xstore always applies the discount to the "get" items on an offer. Therefore, this code should be configured to only allow the Get Items option to be selected by users. See also the *Oracle Retail Pricing Implementation Guide* for more details.

Sales Audit

This section describes configurations that should be made in Sales Audit, as well as some limitations that exist.

First, in addition to configuring the store setup as described above, there are some Sales Audit system options that must be configured in a particular way when integrating with Xstore:

- Transaction Appended with Workstation ID – this should be set to unchecked (No); when checking for duplicate transactions from Xstore, Sales audit will look at the register field in the transaction, along with the transaction number.
- Credit Card Masking Character – this should be set to * when integrating with Xstore, as that is how Xstore masks credit card numbers.
- Balancing Level - when integrated with Xstore, Sales Audit should be configured with a balancing level of Register and Xstore will always sends the workstation ID as the register.

Store Data Configuration

Sales Audit requires configuration to be set up for each store where you expect to import data. For all Xstore stores, you'll need to include two lines for each store configuration – one for Sales Import and one for Item Level Tax. Sales Import is the normal store sales import configuration, and the Item Level Tax configuration is used in validation of the import file to look for tax at the item level instead of the transaction level. This is how tax data will always be sent from Xstore, regardless of tax type.

Register-level Balancing

Xstore workstation and Sales Audit register are equivalent concepts; however Sales Audit does not have an entity equivalent to the Xstore till, which means that Xstore cannot be configured for till-level balancing when integrated with Sales Audit.

Sales Person

In Xstore, the sales person field length can be up to 60 characters in length, but Sales Audit only allows up to 10 characters. Retailers should, as a business process, not use Xstore sales person IDs with more than 10 characters.

Additionally, Xstore allows multiple sales associates at the line item level, however Sales Audit only supports one. Therefore only the transaction level sales associate is exported to Sales Audit.

Tender Types

Xstore supports a tender type of Home Office Check, which is not supported by Sales Audit. Retailers using this integration should not use the Xstore Home Office Check

tender type. See Appendix: Xstore to Sales Audit Mapping Details, for more information on tender type mapping between solutions.

Coupons

Bounce back coupon number length in Xstore can be 60 characters long, but Sales Audit only allows 40 characters. If retailers want to use the integration, they should as a business process, not use IDs with more than 40 characters.

For more Sales Audit related considerations and configurations, see Appendix: Xstore to Sales Audit Mapping Details in this document.

File Format

The RTLOG file that is sent from Xstore to Sales Audit uses a version that does not include the following components in the file format:

- Transaction Header level:
  - Reference number 28-31
- Transaction Item level:
  - Fulfillment Location Type
  - Fulfillment Location

For more information on how these fields are used in Sales Audit, see t*he Oracle Retail Merchandising Operations Guide – Volume 1*.

Employee IDs

A new employee can be created using the Employee Maintenance function in Xstore. By default, the employee ID is generated automatically based on the employee.seq, for example, 0219001000009. The first four digits are the store ID, the next three digits are the register ID, and the last six digits are the sequence ID. If this new employee is selected as a sales associate, an exception will be thrown in the RTLogGenerator, since the length of a sales person ID, defined in the RTLogFormatConfig file is ten, whereas the length of the auto generated employee ID is 13. In Sales Audit, an associate ID cannot be over ten characters long.

6

RTLog Generator On Premise

This chapter describes how to install, deploy, and configure the On Premise RTLog Generator application. The On Premise RTLog Generator application can be used with both the cloud or on premise Merchanding/Pricing applications.

RTLog Generator is a Java and XML based web application that exposes a Spring-JAXWS implemented SOAP web service. It is distributed as a web archive along with a configuration .zip file ready to be deployed on an Oracle WebLogic 12c server.

This chapter uses Microsoft Windows path format as the example for paths.

Configuration

The RTLog Generator application is shipped with a configuration .zip file (rtlog-gen-config.zip) which should be used to externally configure and extend the RTLog Generator's functionality.

> **Note:** Bounce the WebLogic server after making any configuration level changes.

Starting from Xstore release 18.0, the RTLog Generator application is shipped with two configuration zip:

- `rtlog-generator-config-resa-cs.zip`
- `rtlog-generator-config-resa-onprem.zip`

To integrate with Sales Audit on cloud, rtlog-gen-config-resa-cs.zip should be used to externally configure and extend the RTLog Generator's functionality.

To integrate with Sales Audit on premise, rtlog-gen-config-resa-onprem.zip should be used to externally configure and extend the RTLog Generator's functionality.

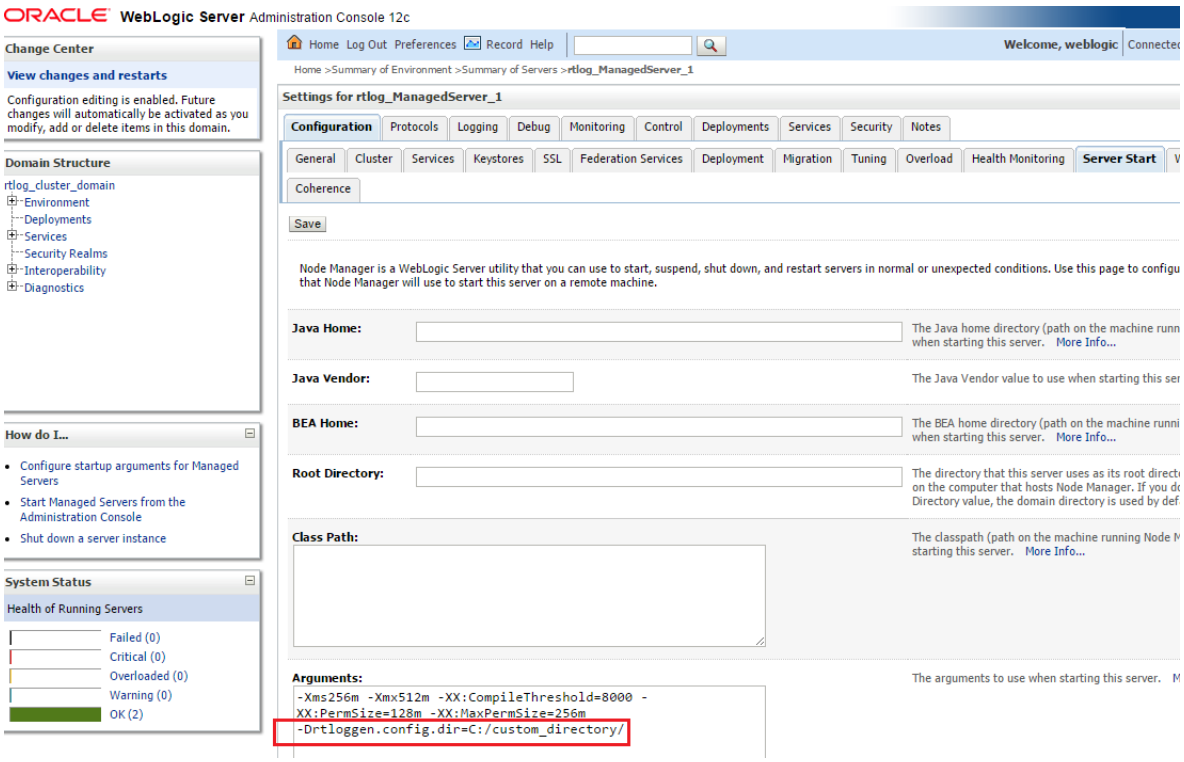
To set up the external configuration features:

1. Extract the configuration file's content into the *C:\<rtlog-generator-config>* directory if installing on Microsoft Windows or */usr/local/<rtlog-generator-config>* on Linux OS. These directories are the default locations where the RTLog Generator application will look for the configuration files. These default locations can be overridden/changed by using one of the following ways:
   - Pass a JVM argument to the server startup script and bounce the server:

     -Drtloggen.config.dir=C:/*<custom_directory>*/

If the WebLogic domain is created with a Node manager, the same argument can be passed from the Administration Console in the Arguments field. See Figure 6–1.

Figure 6–1 Administration Console Configuration Page



- Specify the context-param field in the RTLog Generator WAR file. This requires opening up the WAR file and making the required changes. Update the web.xml file as shown in the following example:

```
<context-param>
    <param-name>rtlog.generator.config.home</param-name>
    <param-value>C:/<custom_directory>/</param-value>
</context-param>
```

Figure 6–2 Example of context-param Field Update

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns
   http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
3      <display-name>RTLOG-GENERATOR</display-name>
4      <context-param>
5          <param-name>contextConfigLocation</param-name>
6          <param-value>/WEB-INF/classes/applicationContext.xml</param-value>
7      </context-param>
8      <!-- Customizable external location for RTLog config files-->
9      <context-param>
10         <param-name>rtlog.generator.config.home</param-name>
11         <param-value>C:/custom_directory/</param-value>
12     </context-param>
13     <!--Customizable RTLog generator app_name. If not unspecified, it remains "rtlog-generator". This
14     <context-param>
15         <param-name>rtlog.generator.application.name</param-name>
16         <param-value />
17     </context-param>
18     <!--Customizable external log4j xml file. Specify just the file name without any extension. By de
19     <context-param>
20         <param-name>rtlog.generator.config.log4j</param-name>
21         <param-value />
22     </context-param>
```

The JVM argument takes the precedence over the default location, that is, C:*<rtlog-gen-config>*. If either of the two does not exist, the context parameter is used. If nothing is specified, the RTLog Generator application will fail on startup with error messages in the server logs.

2. Once the configuration file is extracted to the configured directory, verify the following files:

   - `rtlogconfig.properties`:

     This file contains three properties (key value pairs):

     - `processingDir`: This directory path specifies the location that RTLog Generator will use to build its RTLog files as it receives data from Xstore Office. This directory needs to be created manually.
     - `resaFileDropDir`: This directory path specifies the destination for the RTLog files this system is producing. It should be configured to the location where Sales Audit is looking to receive the RTLog files. This directory needs to be created manually.
     - `clusterNodeNumber`: This property should only be enabled when running in a clustered environment. For more information, see "WebLogic Cluster Setup".

     Following is an example of the three properties:

     ```
     processingDir = C:/RTLOG_Weblogic/Output/Store/RTLOGS
     resaFileDropDir = C:/RTLOG_Weblogic/Output/ReSA
     clusterNodeNumber = 1
     ```

   - `RTLogFormatConfig.xml`:

     This file specifies the format of the RTLog record as specified by Sales Audit. You do not make any changes to this file.

   - `rtlog-generator-log4j.xml`: This file configures the logging levels for the RTLog Generator application.

   - `RTLogMappingBean.xml`:

     This is a spring configuration XML file that provides metadata for the `FieldMapper` and Record Accessor beans which get injected into the RTLog Generator business logic classes. The following example is an excerpt from this file:

Figure 6–3 RTLogMappingBean.xml File Excerpt

```
<!-- exportability mappers -->
<bean id="retailTrnDetailExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.RetailTransactionDetailExportabilityMapper" />
<bean id="retailTrnItemExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.RetailTransactionItemExportabilityMapper" />
<bean id="retailTrnItemDiscountExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.RetailTransactionItemDiscountExportabilityMapper" />
<bean id="retailTrnItemTaxExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.RetailTransactionItemTaxExportabilityMapper" />
<bean id="retailTrnTenderExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.RetailTransactionTenderExportabilityMapper" />
<bean id="controlTrnTenderExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.ControlTransactionTenderExportabilityMapper" />
<bean id="controlTrnTotalExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.ControlTransactionTotalExportabilityMapper" />
<bean id="tenderExchangeTrnTenderExportabilityMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.TenderExchangeTransactionTenderExportabilityMapper" />
<bean id="tillAccountabilityTransactionTypeMapper"
      class="oracle.retail.stores.exportfile.rtlog.fieldmappers.TillAccountabilityTransactionTypeMapper" />
<!-- RTLog record accessors -->
<bean id="FileHeaderAccessor" class="oracle.retail.stores.exportfile.rtlog.accessors.AccessFileHeader" />
<bean id="TransactionHeaderAccessor" class="oracle.retail.stores.exportfile.rtlog.accessors.AccessTransactionHeader" />
```

- `RTLogMappingConfig.xml`:

  The RTLog Generator application relies heavily on the XML-based mapping which provides extensibility and a way to maintain/upgrade features for the application. This file can be used to override all the field values for either mapping strategy:

  - `FieldMapperThenValueMapping`:

    The `RecordValue` attribute values as shown in the following example can be changed:

```
<MAP sourceField="tenderId" targetRecord="TransactionHeaderTotal"
targetField="ReferenceNumber1"
mappingStrategyOrder="FieldMapperThenValueMapping"
fieldMapper="trnHeaderTotalMapper">
 <VALUE_MAPPINGS handleNotFound="success"> <VALUE_MAPPING
sourceValue="GIFT_CERTIFICATE" RecordValue="GIFTCERT" />
 <VALUE_MAPPING sourceValue="HOUSE_ACCOUNT" RecordValue="HACCNT" />
 <VALUE_MAPPING sourceValue="ISSUE_STORE_CREDIT" RecordValue="ISTCRDT"
/>
 <VALUE_MAPPING sourceValue="ISSUE_MERCHANDISE_CREDIT_CARD"
RecordValue="IMCCARD" />
 <VALUE_MAPPING sourceValue="ISSUE_XPAY_GIFT_CARD"
RecordValue="IXPAYGC" />
 <!--For e.g above given value can be changed as shown here.-->
 <VALUE_MAPPING sourceValue="ISSUE_XPAY_GIFT_CARD" RecordValue="SAMPLE_
IXPAYGC" />
 <VALUE_MAPPING sourceValue="MALL_CERTIFICATE" RecordValue="MALLCERT"
/>
 <VALUE_MAPPING sourceValue="MERCHANDISE_CREDIT_CARD"
RecordValue="MCCARD" />
 <VALUE_MAPPING sourceValue="PAYPAL" RecordValue="PAYPAL" />
 <VALUE_MAPPING sourceValue="COUPON" RecordValue="QPON" />
 <VALUE_MAPPING sourceValue="ROOM_CHARGE" RecordValue="ROOMCHAG" />
 <VALUE_MAPPING sourceValue="RELOAD_XPAY_GIFT_CARD"
RecordValue="RXPAYGC" />
 <VALUE_MAPPING sourceValue="RELOAD_MERCHANDISE_CREDIT_CARD"
RecordValue="RMCCARD" />
```

```
 <VALUE_MAPPING sourceValue="STORE_CREDIT" RecordValue="STCRDT" />
 <VALUE_MAPPING sourceValue="XPAY_GIFT_CARD" RecordValue="XPAYGC" />
 </VALUE_MAPPINGS>
</MAP>
```

Figure 6–4 RTLogMappingConfig.xml Field Mapper Example 1

```
<MAP sourceField="tenderId" targetRecord="TransactionHeaderTotal" targetField="ReferenceNumber1"
     mappingStrategyOrder="FieldMapperThenValueMapping" fieldMapper="trnHeaderTotalMapper">
 <VALUE_MAPPINGS handleNotFound="success"> <VALUE_MAPPING sourceValue="GIFT_CERTIFICATE" RecordValue="GIFTCERT" />
 <VALUE_MAPPING sourceValue="HOUSE_ACCOUNT" RecordValue="HACCNT" />
 <VALUE_MAPPING sourceValue="ISSUE_STORE_CREDIT" RecordValue="ISTCRDT" />
 <VALUE_MAPPING sourceValue="ISSUE_MERCHANDISE_CREDIT_CARD" RecordValue="IMCCARD" />
 <VALUE_MAPPING sourceValue="ISSUE_XPAY_GIFT_CARD" RecordValue="IXPAYGC" />
 <!--For e.g above given value can be changed as shown here.-->
 <VALUE_MAPPING sourceValue="ISSUE_XPAY_GIFT_CARD" RecordValue="SAMPLE_IXPAYGC" />
 <VALUE_MAPPING sourceValue="MALL_CERTIFICATE" RecordValue="MALLCERT" />
 <VALUE_MAPPING sourceValue="MERCHANDISE_CREDIT_CARD" RecordValue="MCCARD" />
 <VALUE_MAPPING sourceValue="PAYPAL" RecordValue="PAYPAL" />
 <VALUE_MAPPING sourceValue="COUPON" RecordValue="QPON" />
 <VALUE_MAPPING sourceValue="ROOM_CHARGE" RecordValue="ROOMCHAG" />
 <VALUE_MAPPING sourceValue="RELOAD_XPAY_GIFT_CARD" RecordValue="RXPAYGC" />
 <VALUE_MAPPING sourceValue="RELOAD_MERCHANDISE_CREDIT_CARD" RecordValue="RMCCARD" />
 <VALUE_MAPPING sourceValue="STORE_CREDIT" RecordValue="STCRDT" />
 <VALUE_MAPPING sourceValue="XPAY_GIFT_CARD" RecordValue="XPAYGC" />
 </VALUE_MAPPINGS>
</MAP>
```

- No `mappingStrategyOrder` and `fieldMapper` attributes are defined.

  The `RecordValue` attribute values shown in the following example can be changed or a new value can be added:

```
<MAP sourceField="reason" targetRecord="TransactionHeader"
targetField="ReasonCode">
 <VALUE_MAPPINGS handleNotFound="nextMapping">
 <VALUE_MAPPING sourceValue="PI1" RecordValue="PI1"/>
 <VALUE_MAPPING sourceValue="PI2" RecordValue="PI2"/>
 <VALUE_MAPPING sourceValue="PI3" RecordValue="PI3"/>
 <VALUE_MAPPING sourceValue="PO1" RecordValue="PO1"/>
 <VALUE_MAPPING sourceValue="PO2" RecordValue="PO2"/>
 <VALUE_MAPPING sourceValue="PO3" RecordValue="PO3"/>
 <VALUE_MAPPING sourceValue="PO4" RecordValue="PO4"/>
 <VALUE_MAPPING sourceValue="PO5" RecordValue="PO5"/>
<VALUE_MAPPING sourceValue="SAMPLE" RecordValue="SAMPLE_VALUE"/>
 </VALUE_MAPPINGS>
 </MAP>
```

Figure 6–5 RTLogMappingConfig.xml Field Mapper Example 2

```
<MAP sourceField="reason" targetRecord="TransactionHeader" targetField="ReasonCode">
<VALUE_MAPPINGS handleNotFound="nextMapping">
<VALUE_MAPPING sourceValue="PI1" RecordValue="PI1"/>
<VALUE_MAPPING sourceValue="PI2" RecordValue="PI2"/>
<VALUE_MAPPING sourceValue="PI3" RecordValue="PI3"/>
<VALUE_MAPPING sourceValue="PO1" RecordValue="PO1"/>
<VALUE_MAPPING sourceValue="PO2" RecordValue="PO2"/>
<VALUE_MAPPING sourceValue="PO3" RecordValue="PO3"/>
<VALUE_MAPPING sourceValue="PO4" RecordValue="PO4"/>
<VALUE_MAPPING sourceValue="PO5" RecordValue="PO5"/>
<VALUE_MAPPING sourceValue="SAMPLE" RecordValue="SAMPLE_VALUE"/>
</VALUE_MAPPINGS>
</MAP>
```

- `spring-scheduler.xml`:

  It is the most commonly modified file in the RTLog Generator application. It is used to configure the scheduled interval for publishing the RTLog files. In the case of trickle polling, the default interval should be 15 minutes, however, keeping a larger interval (at least greater than or equal to 15 minutes) is recommended as configuring with a smaller interval might affect the performance.

Figure 6–6 spring-scheduler.xml Example

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:p="http://www.springframework.org/schema/p"
       xmlns:task="http://www.springframework.org/schema/task" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
           http://www.springframework.org/schema/task
           http://www.springframework.org/schema/task/spring-task-3.0.xsd">

<task:scheduled-tasks scheduler="rtlogScheduler">

<!-- To publish files once every 10 minutes = 600000 milliseconds 15 minutes = 900000 milliseconds
     1 hour = 3600000 milliseconds in fixed-delay below.
     It is not supported if fixed-delay is less than three second (3000 milliseconds). -->

<task:scheduled ref="rtLogFilesPublisher" method="publishFilesToReSA" fixed-delay="900000" />

<!-- You can also use "cron syntax". This simplistic example publishes files once every 5 minutes -->
<!-- <task:scheduled ref="rtLogFilesPublisher" method="publishFilesToReSA" cron="0 */5 * * * ?"/> -->
</task:scheduled-tasks> <task:scheduler id="rtlogScheduler" />

<task:annotation-driven />

</beans>
```

Note: For more information on how to customize the RTLog Generator, see the *Retail Xstore - RTLog Generator Extension Guidelines (Doc ID 2174095.1)* on `https://support.oracle.com`.

Deployment

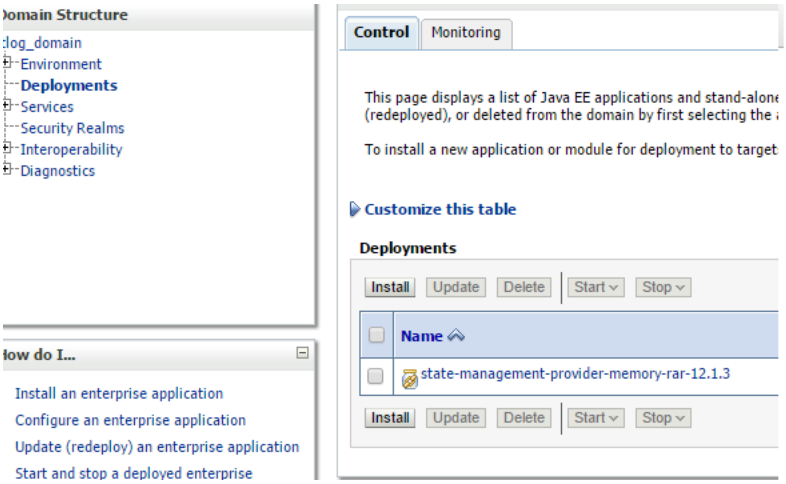
If you are deploying in a cluster, first set up a WebLogic cluster. For more information, see "WebLogic Cluster Setup".

This section covers the deployment in both a clustered and non-clustered environment.

To deploy the RTLog Generator application:

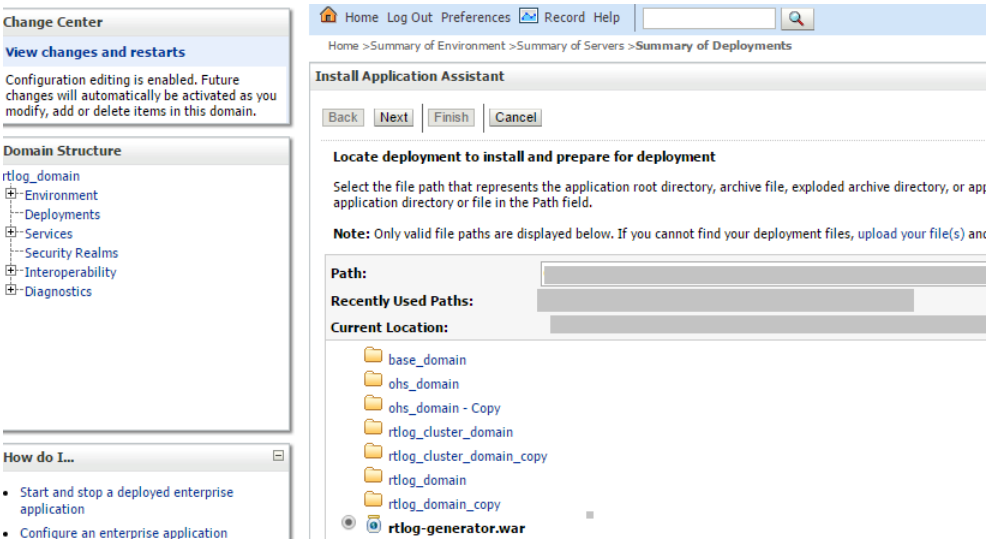
1. Log in to the WebLogic 12 Server Administration Console (http://<*hostName*>:<*port*>/console).
2. Click the Deployment link from the left navigation menu.
3. Click **Install**.

Figure 6–7 Administration Console Control Page



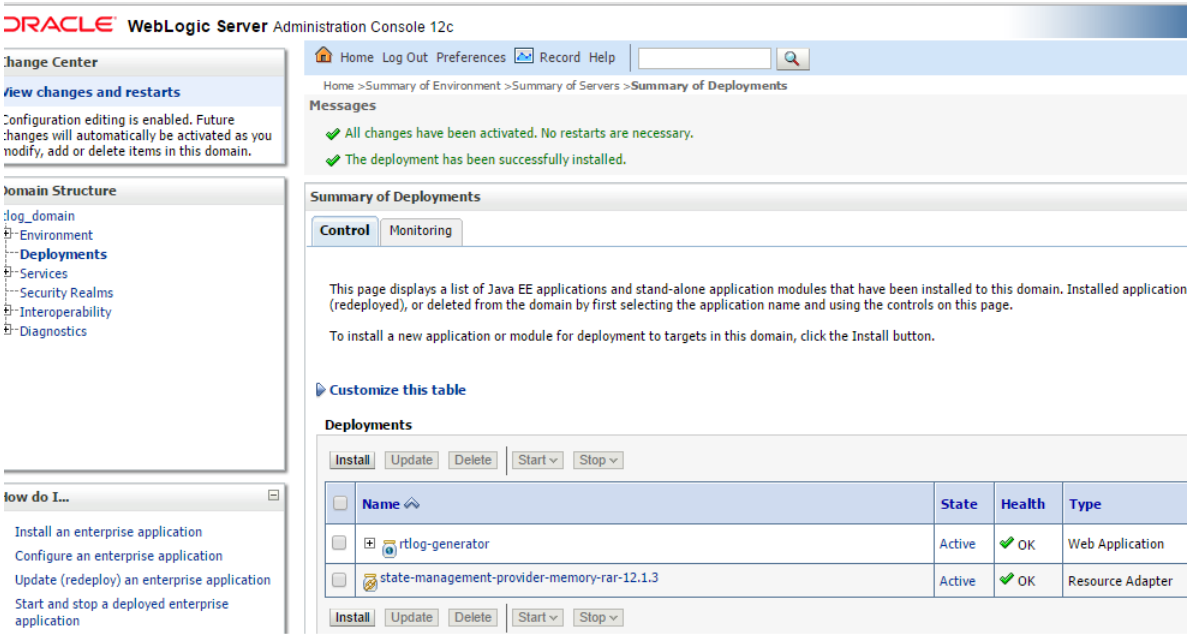
4. Navigate to the rtlog-generator.war file directory. Select the rtlog-generator.war option.

Figure 6–8 Administration Console Install Application Assistant Page



5. Click **Next** and then **Finish**. Once deployed, RTLog Generator should be listed as one of the deployed applications as shown in Figure 6–9.

Figure 6–9 Administration Console Summary of Deployments



Once the deployment is complete, following are the next steps:

- To deploy on a cluster, see "Deployment of the RTLog Generator Application on a Cluster".
- To enable security for the RTLog Generator application, see "Security Configuration". When deploying in a non-clustered environment, continue at this section.

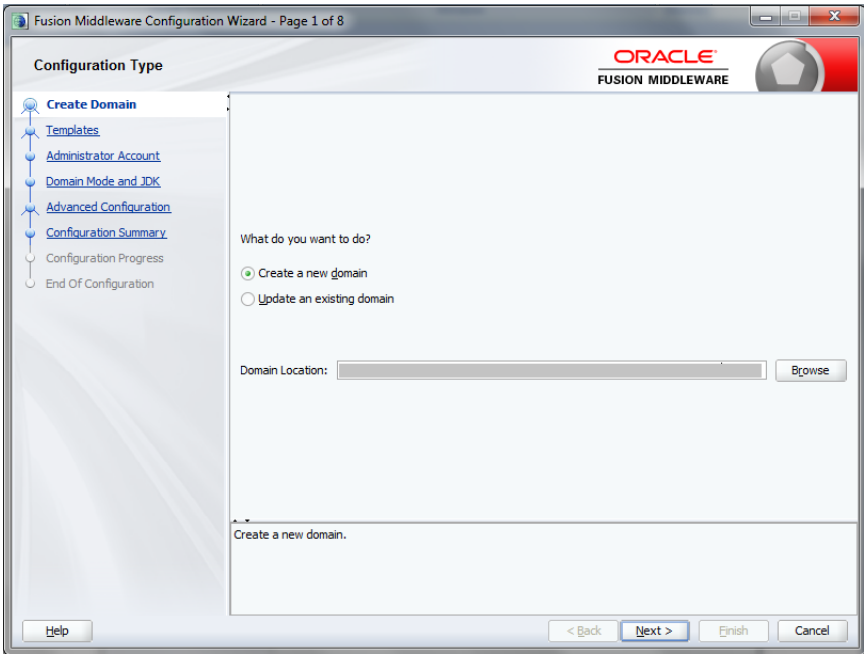
WebLogic Cluster Setup

Note: WebLogic 12c must be installed on all the clustered machines and the exact same installed directory location must be used on all the machines.

To set up the cluster to use RTLog Generator:

1. Start the WebLogic configuration wizard on one machine where the Administration server needs to reside.
2. On the Configuration Wizard Configuration Type page, select **Create a new domain**. Enter or browse to the location for the domain. Click **Next**.

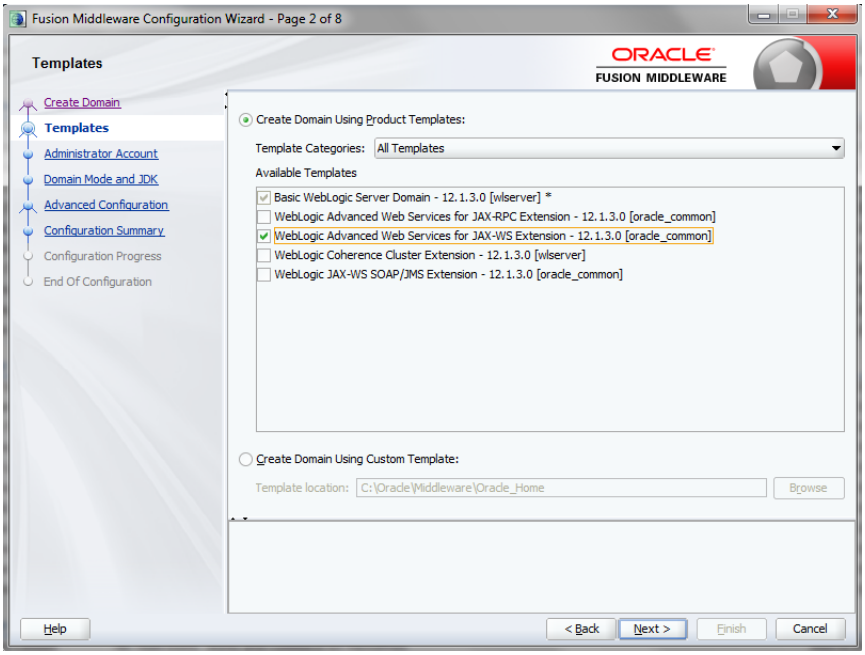
Figure 6–10 Configuration Wizard Configuration Type Page



3. On the Templates page, select the supported products and click **Next**. It is recommended to select the following:

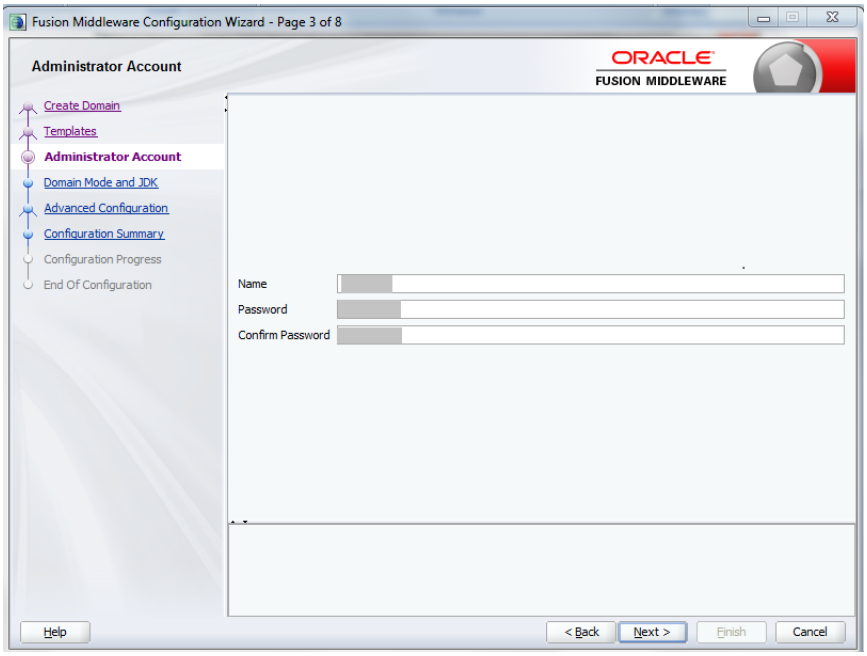
   `WebLogic Advanced Web Services for JAX-WS Extension - 12.1.3.0 [oracle_common]`

Figure 6–11 Configuration Wizard Templates Page



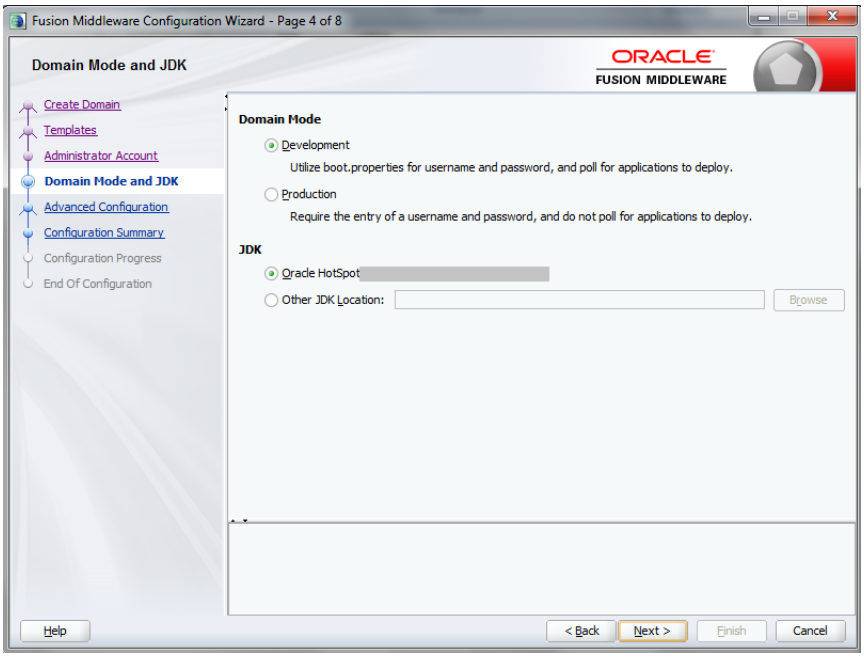
4. On the Administrator Account page, enter the Administrator user name and password. Enter the password a second time to confirm. Click **Next**.

Figure 6–12 Configuration Wizard Administrator Account Page



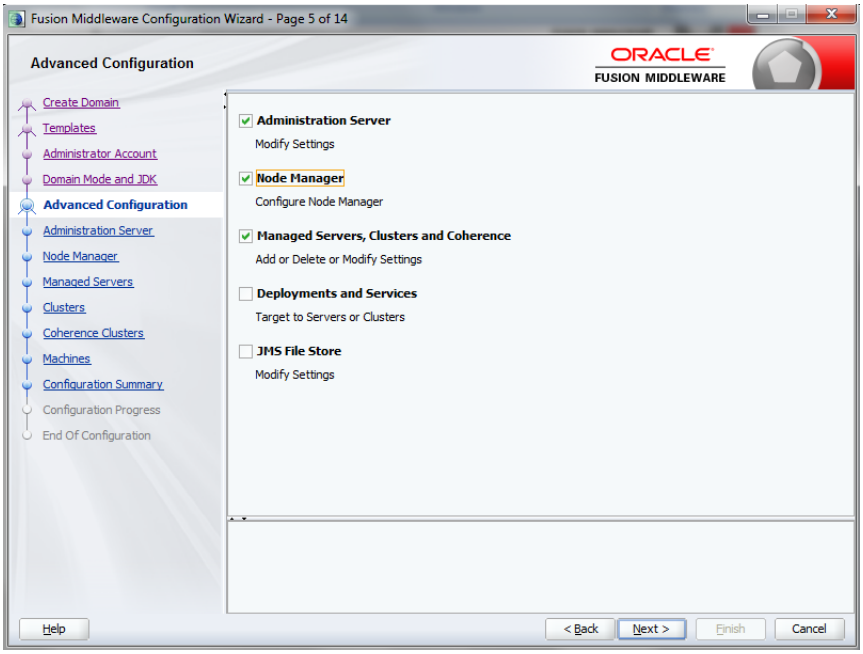
5. On the Domain Mode and JDK page, select either Development or Production mode. For production mode, you need to manually create the boot.properties file. Click **Next**.

Figure 6–13 Configuration Wizard Domain Mode and JDK



6. On the Advanced Configuration page, select the Administration Server, Node Manager, and Managed Servers, Clusters and Coherence options. Click **Next**.

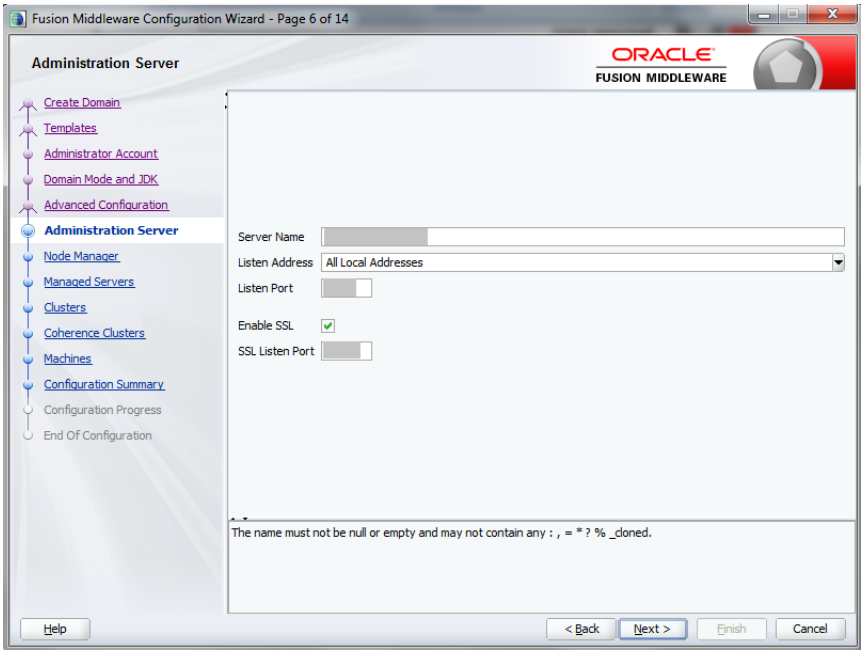
Figure 6–14 Configuration Wizard Advanced Configuration Page



7. On the Administration Server page, enter the values to configure the administration server. The administrator server controls all the managed servers that are part of the cluster.

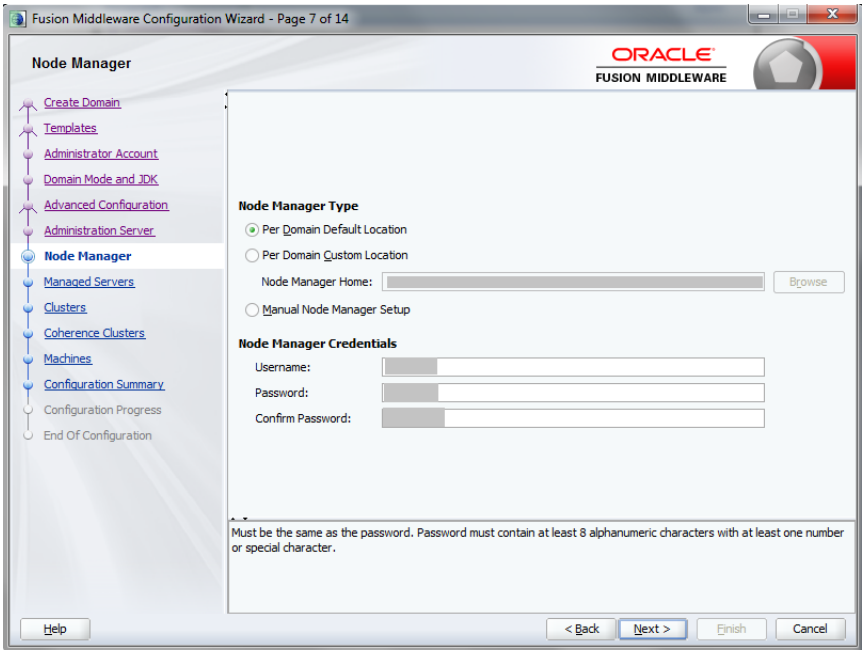
   Enter the server name, select Enable SSL, and enter the listen ports. For the listen address, enter the Machine_1 IP address. Machine_1 will be part of the cluster and will have the administrator server running on it. Click **Next**.

Figure 6–15 Configuration Wizard Administration Server Page



8. On the Node Manager page, do not change the default node manager settings. For the credentials, enter weblogic as the user name and enter the password. Click **Next**.

Figure 6–16 Configuration Wizard Node Manager Page



9. On the Managed Servers page, add and configure each managed server:

   a. For the listen address, enter the IP address of the managed server. Do not select All local Addresses.

   b. rtlog_ManagedServer_1 will be running on Machine_1 in this configuration. Enter the Machine_1 IP address for the server.

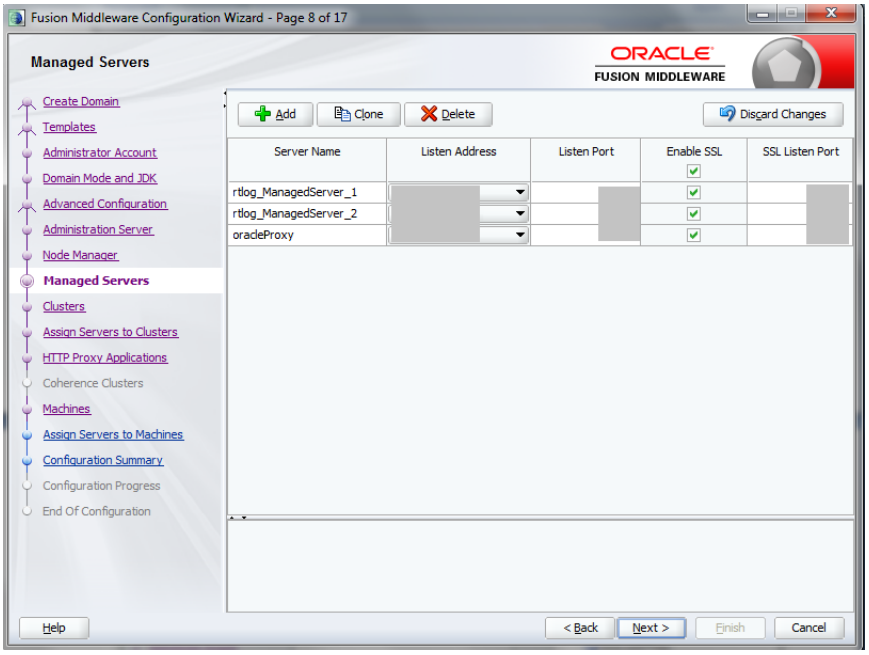
   c. rtlog_ManagedServer_2 will be running on Machine_2 in this configuration. Enter the Machine_2 IP address for this server.

   d. oracleProxy is running on Machine_1, but is not a part of the cluster. It is an Oracle proxy HTTP cluster servlet used for failover and load balancing purposes. Enter the Machine_1 IP address for this server.

   e. Enable SSL for all the managed servers.

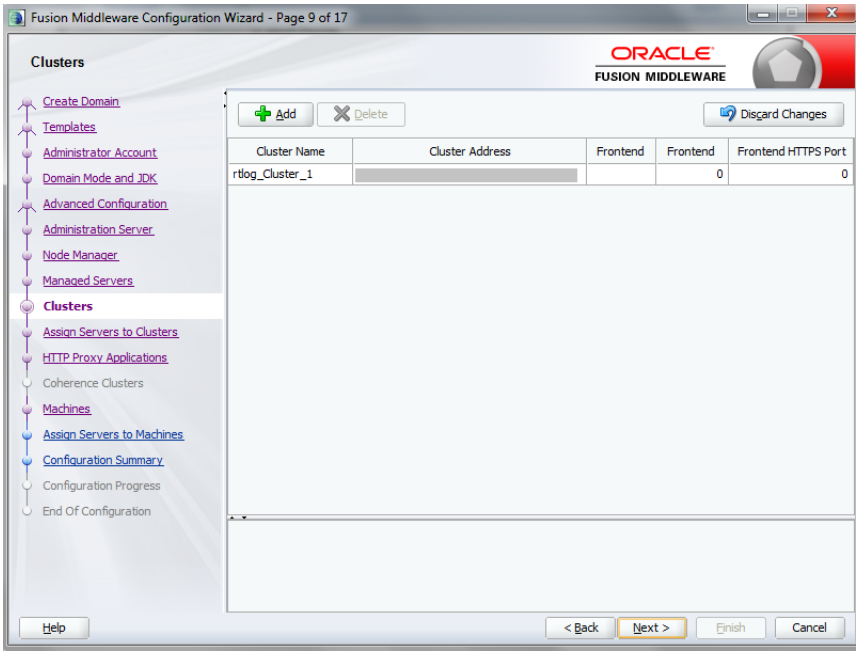
   f. Click **Next**.

Figure 6–17 Configuration Wizard Managed Servers Page



10. On the Clusters page, add and configure the cluster. Enter the cluster name followed by the cluster address, that is, IP address1:port1, IP address2:port2, so on. Click **Next**.

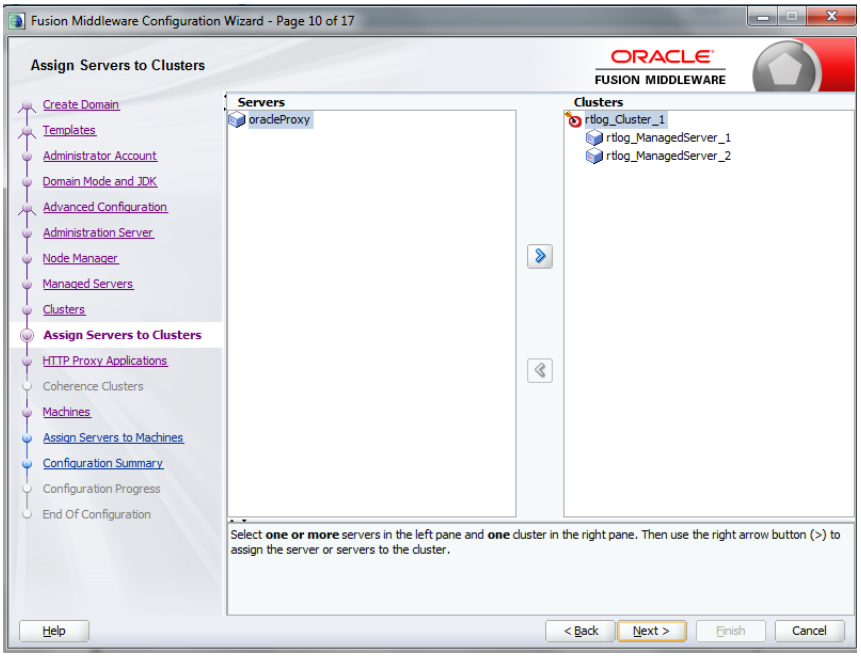
Figure 6–18 Configuration Wizard Clusters Page



11. On the Assign Servers to Cluster page, assign the managed servers to the cluster. and click **Next**.

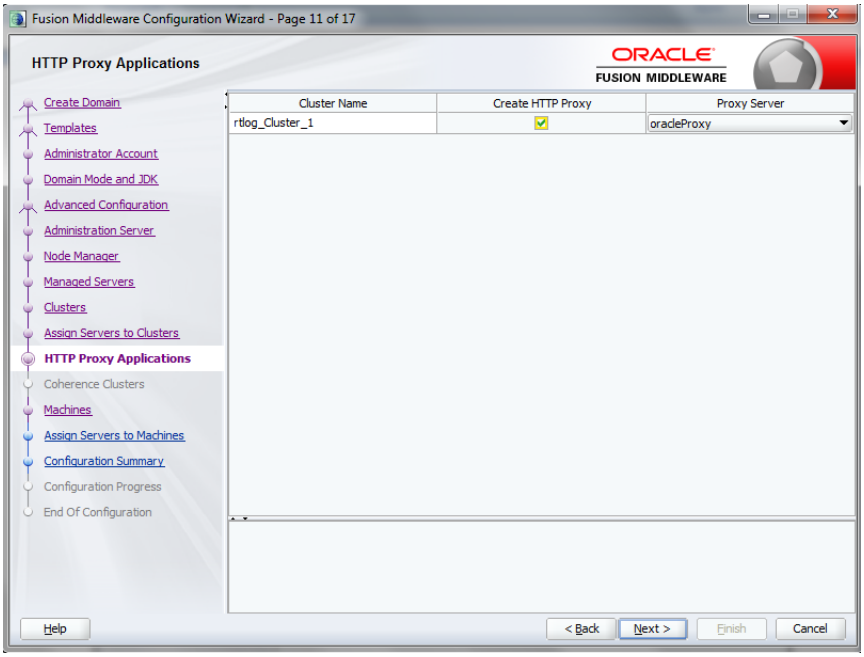
> **Note:** Do not include the Oracle Proxy as part of the cluster.

Figure 6–19 Configuration Wizard Assign Servers to Clusters Page



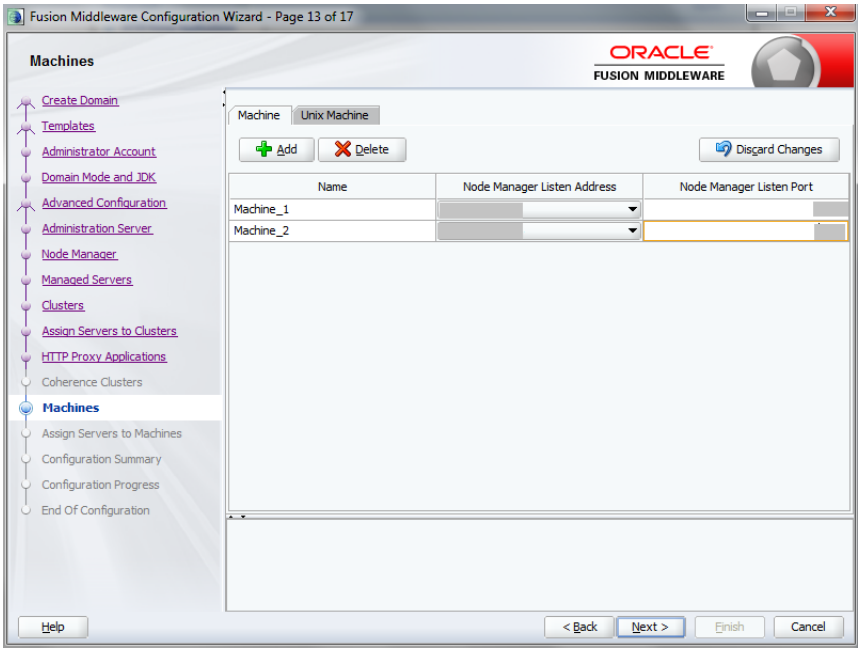
12. On the HTTP Proxy Applications page, select Create HTTP Proxy and then select the server from the drop-down list. By default, it should have already been selected. Click **Next**.

Figure 6–20 Configuration Wizard HTTP Proxy Applications Page



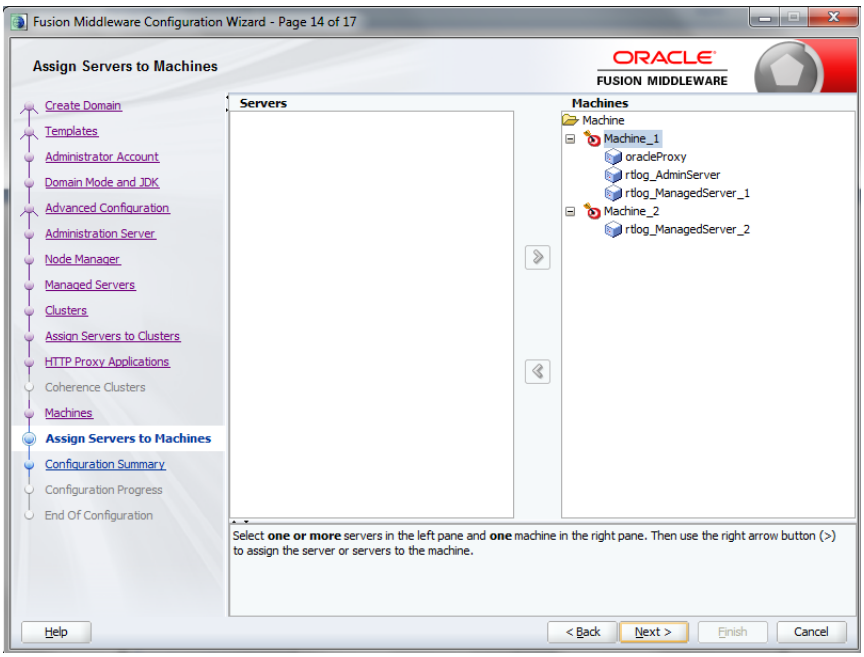
13. On the Machines page, add and configure each machine. To add Machine_1 and Machine_2, click **Add** and enter the respective IP addresses. This configuration is for setting up the Node managers on both the machines. Since these node managers are physically separated, you can select the same host. Click **Next**.

Figure 6–21 Configuration Wizard Machines Page



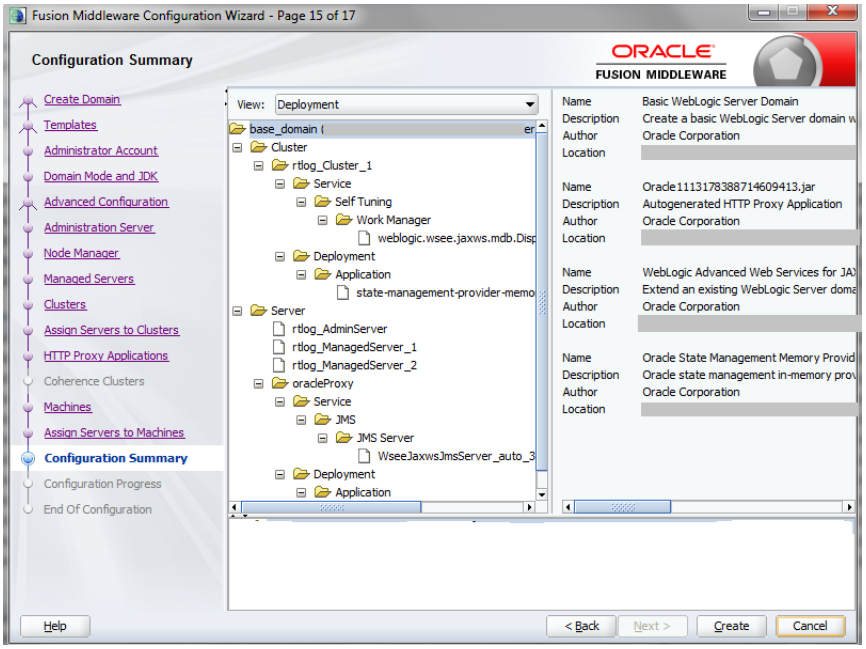
14. On the Assign Servers to Machines page, assign the servers to the machines. In this example, Oracle proxy (load balancer), Administration server, and one managed server are configured on Machine_1. Another managed server is configured on Machine_2. Click **Next**.

Figure 6–22 Configuration Wizard Assign Servers to Machines Page



15. On the Configuration Summary page, verify the selected configuration. Click **Create**. The domain is created.

Figure 6–23 Configuration Wizard Configuration Summary Page



To complete the configuration of the cluster:

1. Start and stop the node manager. You can find the start up script inside the newly created domain, that is, the <*rtlog_clust_domain*>\bin directory.
2. In the nodemanager.properties file, set SecureListener=false. This file is found in the <*rtlog_clust_domain*>\nodemanager directory.
3. Edit the <*rtlog_clust_domain*>\config\config.xml file. Use plain communication for the node managers by updating the communication type for the node managers as shown in the following example:

```
<machine>
       <name>Machine_1</name>
       <node-manager>
      <name>Machine_1</name>
      <nm-type>Plain</nm-type>
      <listen-address>xxx.x.xxx.xx</listen-address>
    </node-manager>
 </machine>
 <machine>
    <name>Machine_2</name>
    <node-manager>
      <name>Machine_2</name>
      <nm-type>Plain</nm-type>
      <listen-address>xxx.x.xxx.xxx</listen-address>
    </node-manager>
  </machine>
```

4. If the <*rtlog_clust_domain*> is created with the production mode option:
   a. Run <*rtlog_clust_domain*>\startWeblogic.cmd for the first time. This creates the servers folders under the domain. Enter the administration user name and password.
   b. Create a folder named security under the <*rtlog_clust_domain*>\servers\Admin server.

c. Create the boot.properties file with the following entries under the security folder:

```
password=%admin_server_password%
username=%admin_server_username%
```

%admin_server_password% and %admin_server_username% are the administrator password and user name.

d. After making these changes, if there are any running processes, shut down all the processes.

5. Pack the created domain:

a. Stop both the Node manager and Admin Server if not already stopped. Use the packing utility to pack the domain on the machine. This utility is found in the following location:

<*WL_HOME*>\wlserver\common\bin\pack.cmd

Run the following command:

pack.cmd -domain=<*WL_HOME*>\user_projects\domains\rtlog_cluster_domain -template=<*WL_HOME*>\user_projects\domains\rtlog_cluster_domain\rtlog_cluster_domain.jar -template_name="RTLog C domain"

This command creates a jar named rtlog_cluster_domain.jar by packing the complete domain into it. Copy the rtlog_cluster_domain.jar to Machine_2 and unpack it.

b. Create a <*user_templates*> directory on the remote machine and copy the rtlog_cluster_domain.jar file to this location. Run the following command:

unpack.cmd -template=<*WL_HOME*>\user_projects\domains\<*user_templates*>\rtlog_cluster_domain.jar -domain=<*WL_HOME*>\user_projects\domains\rtlog_cluster_domain

c. Start the Administration server and node manager on Machine_1.

6. To enroll the remote (Machine_2) node manager:

a. Run the WebLogic scripting utility. This utility can be found at the following location: <*WL_HOME*>\wlserver\common\bin\as wlst.cmd

b. Start the node manager on this machine, in this example, Machine_2. The node managed must be started before connecting to the Machine_1 Admin server.

c. Run the following command:

connect ('adminServer_username', 'adminServer_password','t3://Machine_1_IPAddress:Admin_server_unsecured_port')

d. Once the connect command shows the connection completed successfully, run the following command:

nmEnroll ('<*WL_HOME*>/user_projects/domains/<*rtlog_cluster_domain*>','<*WL_HOME*>/user_projects/domains/<*rtlog_cluster_domain*>/nodemanager')

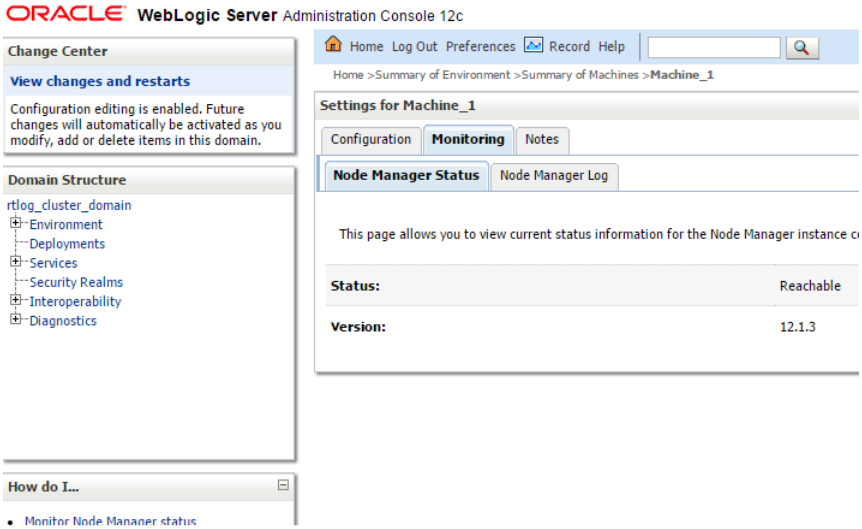
e. When the command completes successfully, run exit ().

Note: Repeat Step 6 for all the remote machines that will be in the cluster on which managed servers will be running. This step used Machine_2 as the example.

---

7. Log in to the Administration Server console and make sure all the node managers are reachable. This can be found under Machines. Repeat this step for all the clustered machines to ensure all of them are reachable.

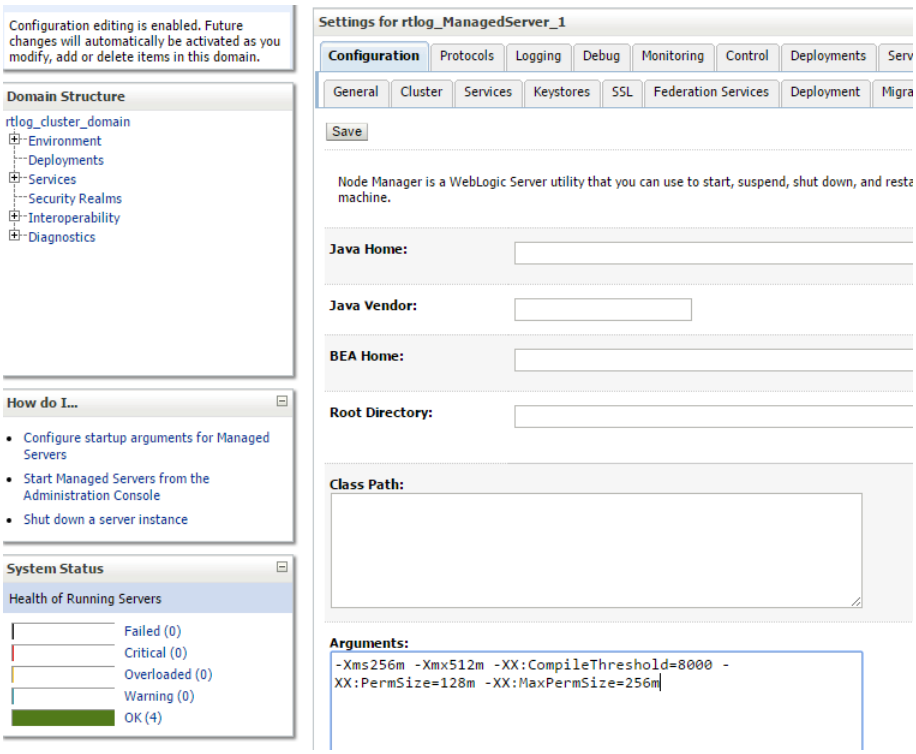
Figure 6–24 Administration Console Settings Page



8. For each managed server, select the Server Start tab. In the Arguments text box, add the following if it does not already exist:

```
-Xms512m -Xmx512m -XX:CompileThreshold=8000 -XX:PermSize=512m
-XX:MaxPermSize=512m
```

Figure 6–25 Administration Console Configuration Page



If you want to configure the non-default external RTLog configuration directory, include an additional JVM argument:

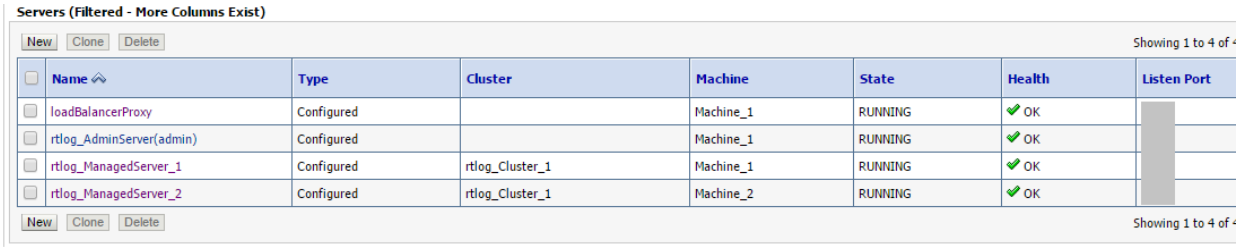
-Drtloggen.config.dir=C:/<*rtlog-gen-config_1*>/

Note: The server-start arguments only work when you are using a NodeManager. If you do not have a NodeManager, specify the JVM argument in the start up scripts. You can also configure the same ext directory location in the RTLog Generator WAR's context-param. For more information, see "Configuration".

---

9. Start all the managed servers including the Oracle proxy. Figure 6–26 shows an example of the list of managed servers.

Figure 6–26 Administration Console List of Servers



Deployment of the RTLog Generator Application on a Cluster

To deploy the application:

1. Oracle proxy creates a web application by creating the web.xml and weblogic.xml files which can be found in the following directory:

   <*WL_HOME*>\user_projects\domains\<*rtlog_cluster_domain*>\apps\OracleProxy4_rtlog_Cluster_1_oracleProxy\WEB-INF

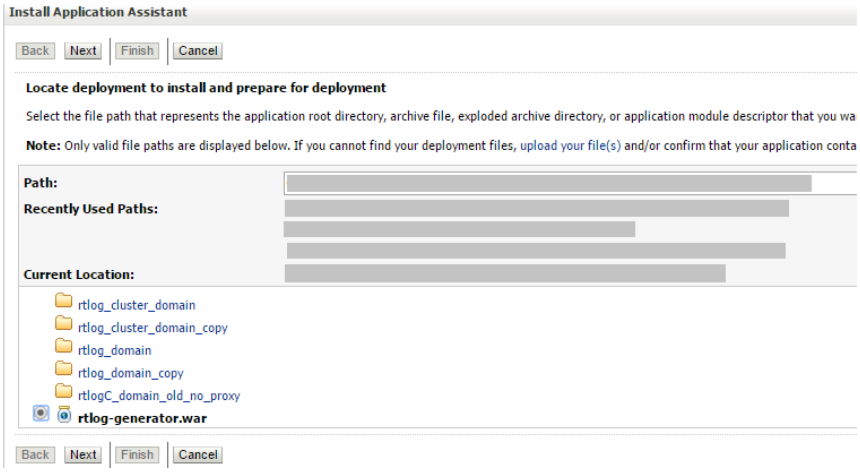
   You can modify the configurations provided in these two files and redeploy the application from the console by pointing it to this directory, that is, WEB-INF.

2. Navigate to the Administration Console home page and click Deployments in the left navigation menu. Figure 6–27 shows an example of the page before deploying the RTLog Generator application.

Figure 6–27 Administration Console Deployments Page

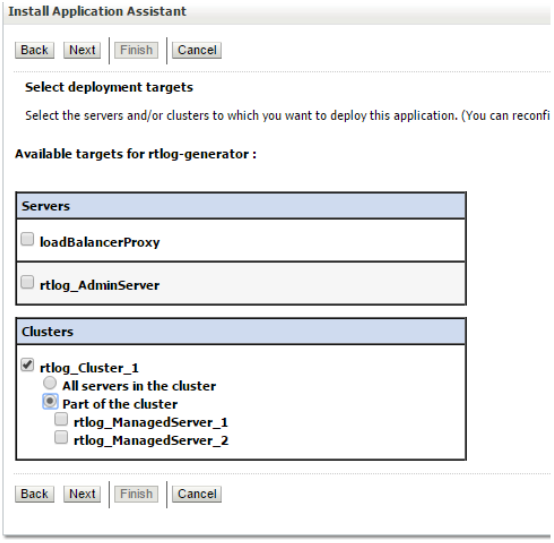
3. Click **Install**. The Install Application Assistant page appears. Select the path to the RTLog Generator WAR directory. Select the rtlog-generator.war option. Click **Next**.

Figure 6–28 Administration Console Install Application Assistant Page



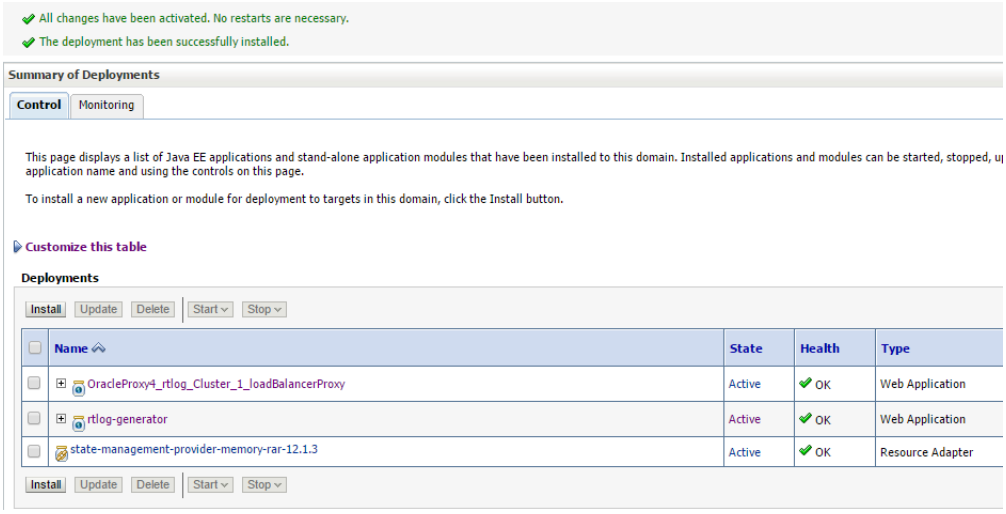
4. Select only the managed servers and click **Next** to finish the deployment.

Figure 6–29 Install Application Assistant Select Deployment Targets Page



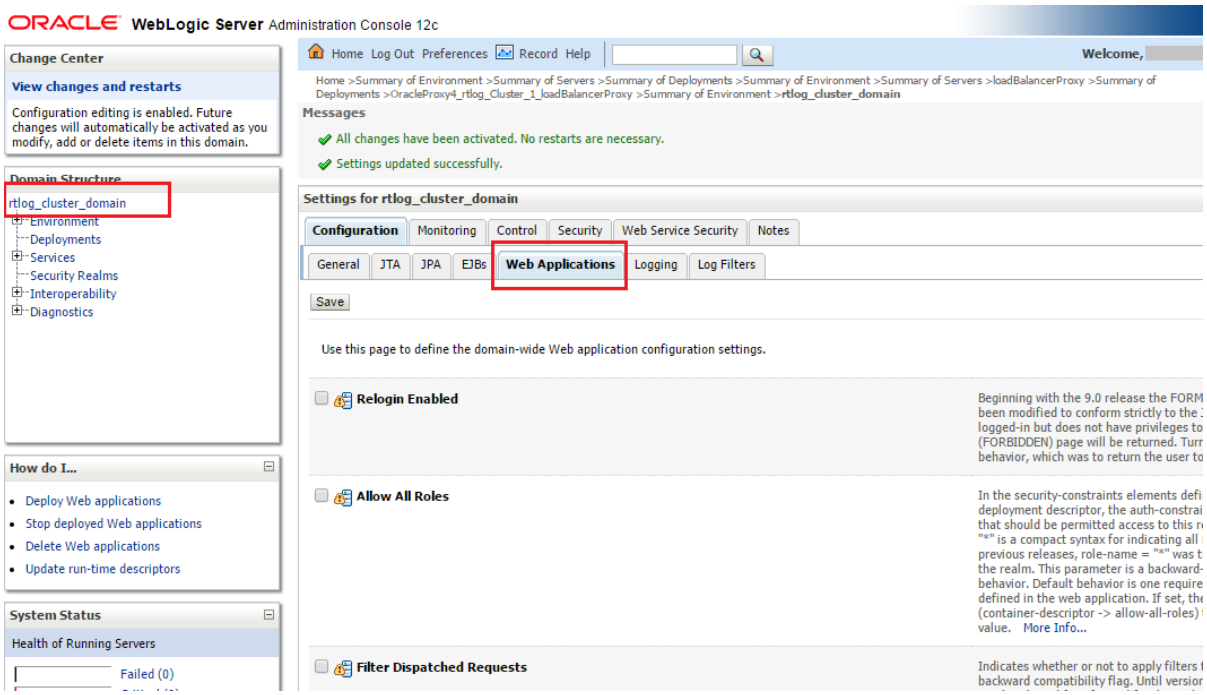
After it is successfully deployed, the RTLog Generator application appears in the Summary of Deployments page.

Figure 6–30 Summary of Deployments Page



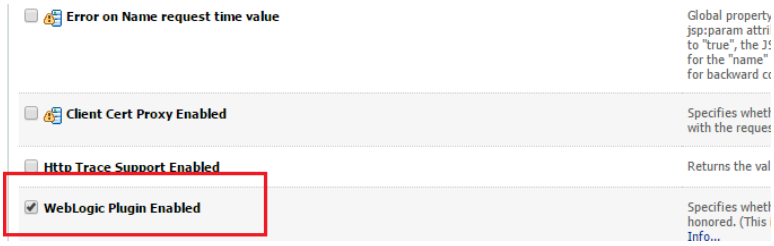
5. To enable container and transport level security, see "Security Configuration".
6. To enable the WebLogic Plugin Enabled parameter from the cluster domain:
   a. Click the <*rtlog_cluster_domain*> link in the left navigation menu. Navigate to the Web Application tab.

Figure 6–31 Administration Console Settings Page



   b. Scroll down the page and select WebLogic Plugin Enabled. Click **Save**.

Figure 6–32 WebLogic Plugin Enabled Parameter



Security Configuration

The RTLog Generator application is secured by leveraging two levels of security:

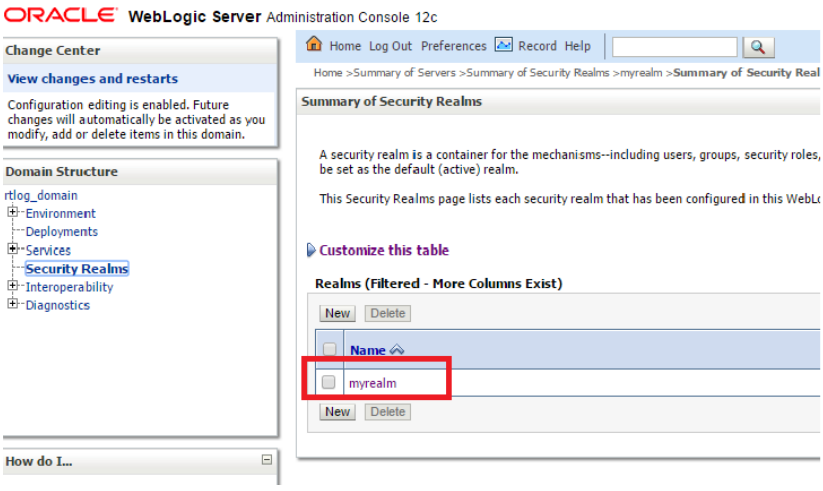
- Container level security: Basic HTTP authentication by setting up the security realm in WebLogic. To configure this security, see "Container Level Security".
- Transport level security: SOAP requests are sent over the secured protocol (HTTPS) by configuring the keystore/truststore in the WebLogic domain and importing the public certificate into Xstore Office's (client) truststore. To configure this security, see "Transport Level Security".

Container Level Security

The following steps assume that a domain has been created with secure port (HTTPS) enabled. To configure container level security:

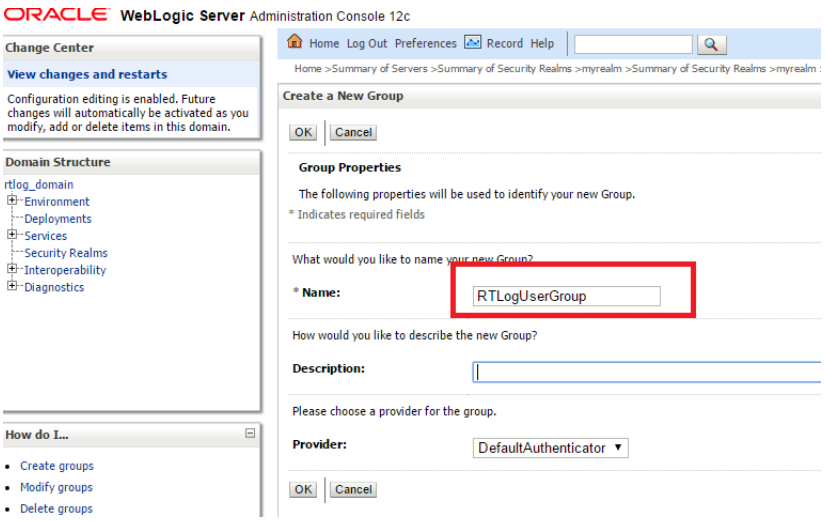
1. Start the WebLogic server and log in to Administration Console.
2. Click Security Realms in the left navigation menu.

Figure 6–33 Administration Console Summary of Security Realms Page



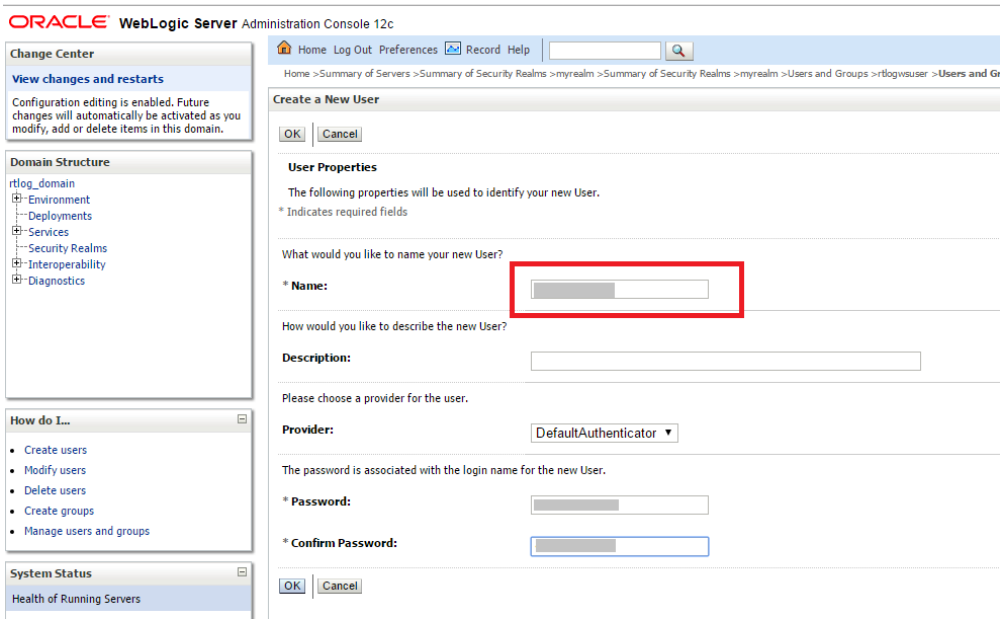
3. In the list of realms on the Summary of Security Realms page, select myrealm.
4. Select Users and Groups and then the Groups tab. To create a new group, click **New**. Enter a group name, for example RTLogUserGroup, and click **OK**.

Figure 6–34 Create a New Group Page



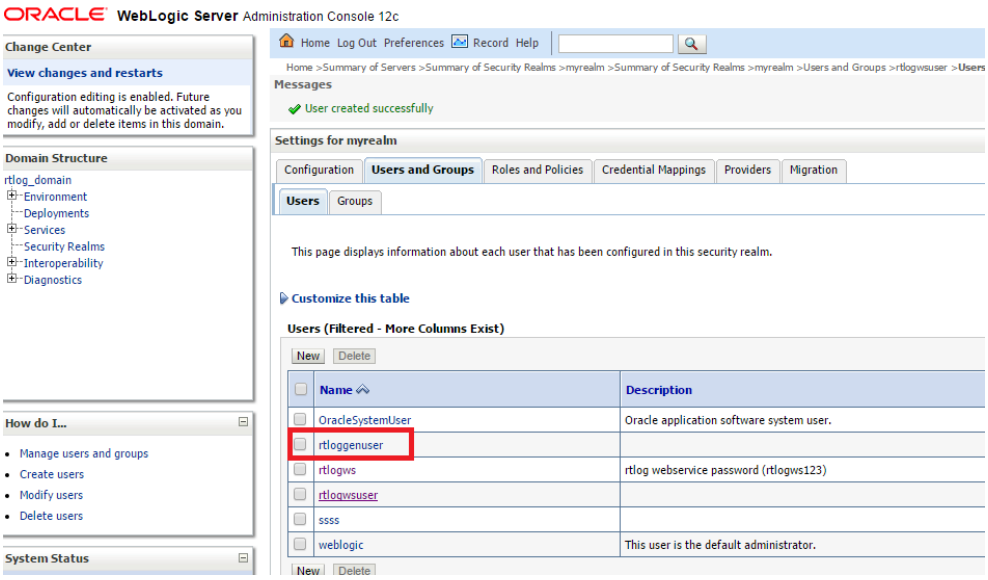
5. Select the Users tab and click **New**. Enter a user name and password and click **OK**.

Figure 6–35 Create a New User Page



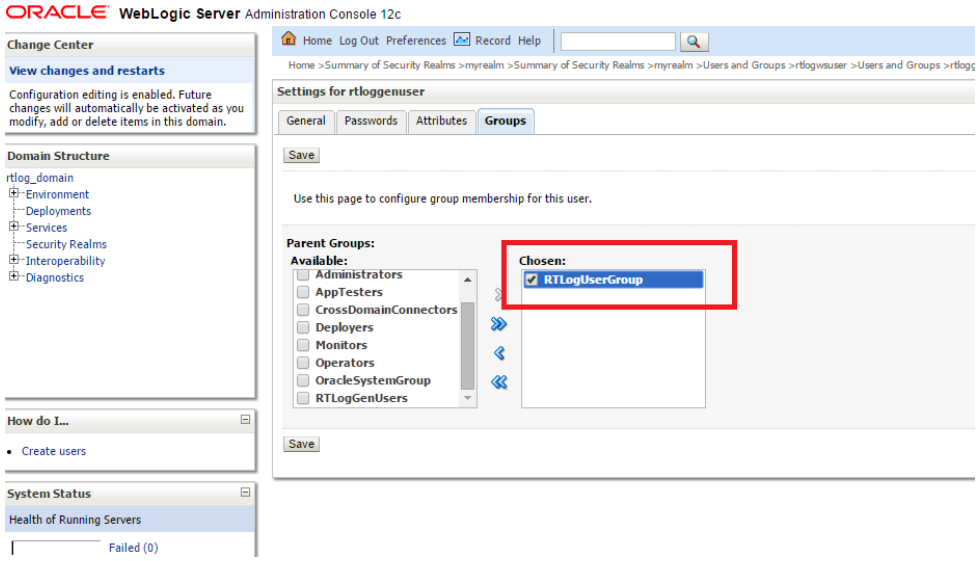
6. In the list of users, click the newly created user.

Figure 6–36 Users Page



7. Select the Groups tab. Assign this user to the same group created in Step 4.

Figure 6–37 User Settings Page



8. Enter the same user name and password created in Step 5 into Xstore Office's broadcaster configuration for the RTLog Generator Web service.

   You should try the MrJaxWsPortProxyFactoryBean bean and create the encrypted values for the user name and password using the String Encryption Utility. For more information, see the *Oracle Retail Xstore Point of Service Implementation Guide*.

Figure 6–38 Example of MrJaxWsPortProxyFactoryBean Update

```
<bean id="ReSA_Broadcaster_jaxws_weblogic"
      class="com.micros_retail.xcenter.broadcast.MrJaxWsPortProxyFactoryBean" >

  <property name="endpointAddress" value="https://hostname:7002/rtlog-generator/service" />
  <property name="serviceInterface" value="com.micros_retail.xcenter.poslog.poslogobj.v2.PoslogObjReceiverApi" />
  <property name="wsdlDocumentUrl" value="classpath:wsdl/generic_poslog_object_v2/PoslogObjReceiverApiService.wsdl" />
  <property name="namespaceUri" value="http://v2.ws.poslog.xcenter.dtv/" />
  <property name="serviceName" value="PoslogObjReceiverApiService" />
  <property name="portName" value="PoslogObjReceiverApiPort" />
  <property name="customProperties" ref="jaxwsCustomProperties" />
  <property name="encryptedUsername" value=            />
  <property name="encryptedPassword" value=            />
</bean>
```

Transport Level Security

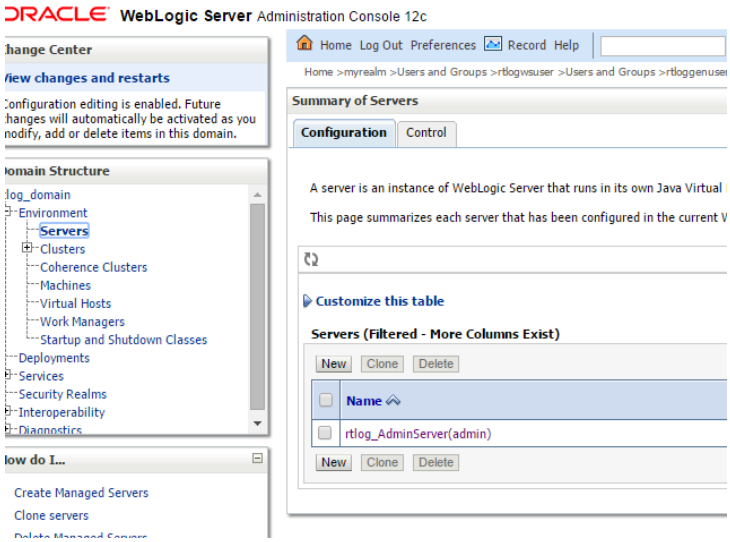
To configure transport level security:

1. Create keystore.jks using a keytool utility. For information on keytool utilities, see the *Oracle Retail Xstore Point of Service Implementation Guide*.
2. Export the public certificate into a truststore.jks file. These files are needed to configure the custom key and trust store for Step 3.

> **Note:** In a clustered environment, import all the public certificates into one truststore file and configure all the instances of the server, including HttpClusterServlet proxy, to use the same truststore file.

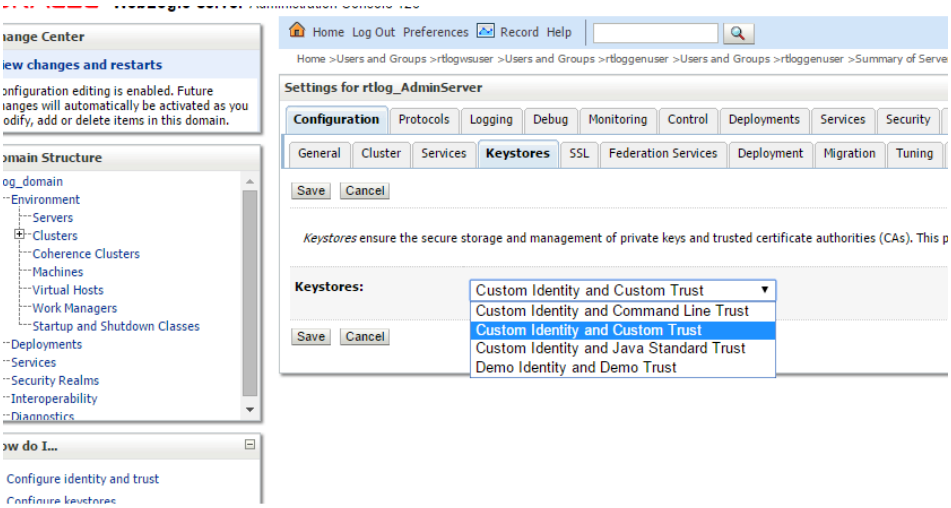
3. Log in to the WebLogic console. Click Environment and then the Servers link from the left navigations menu.

Figure 6–39 Administration Console Servers Page



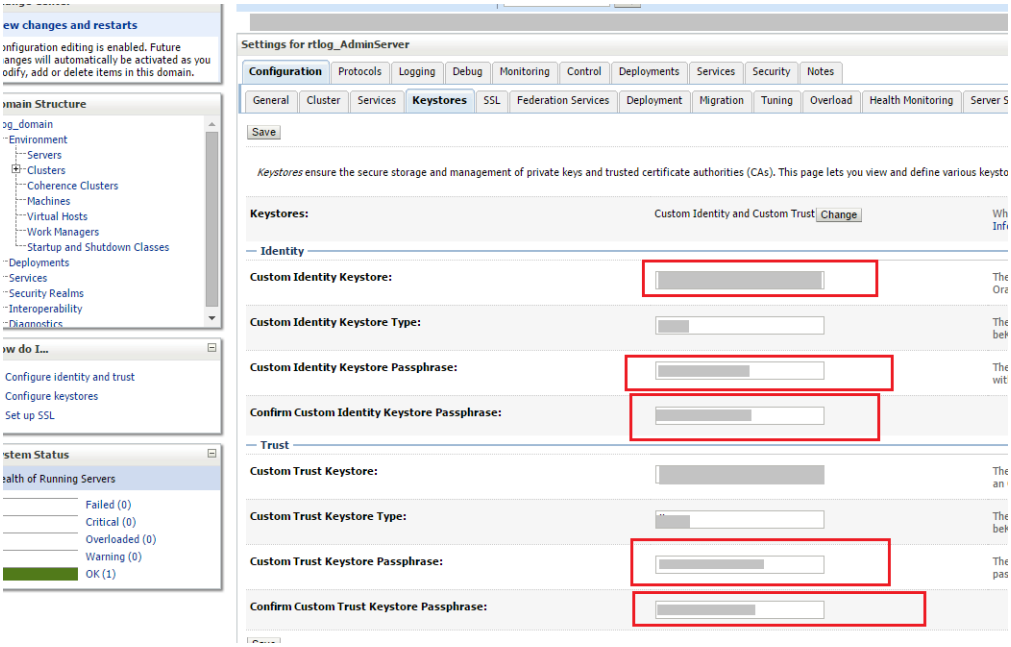
4. Click **Change**. Select Custom Identity and Custom Trust. Click **Save**.

Figure 6–40 Keystores Settings



5. Click the linked name for the Administration Server. The page containing the settings for the Administration Server appears. Select the Keystores tab.

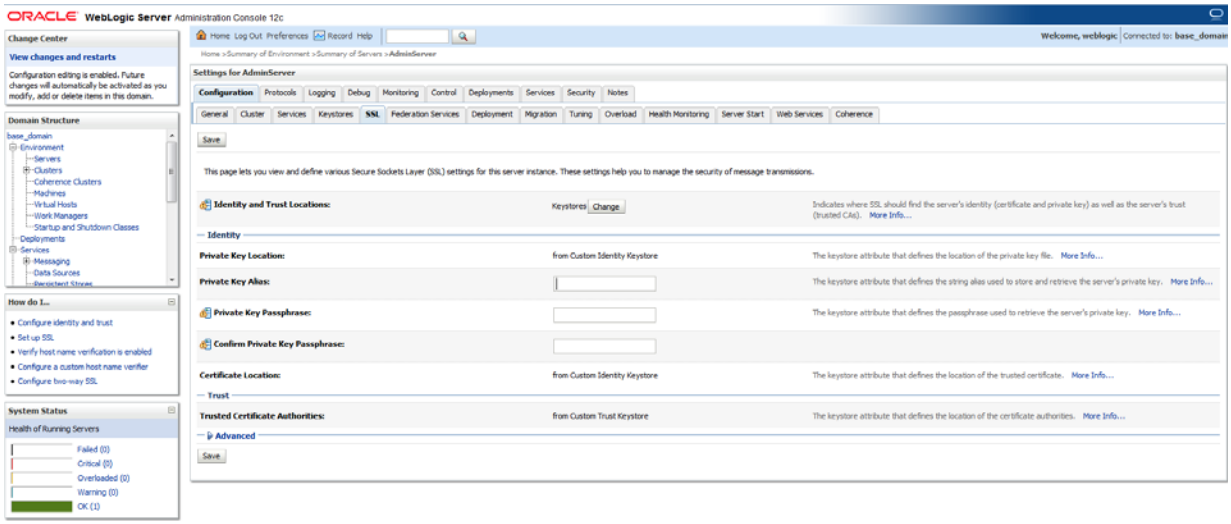
Figure 6–41 Settings for the Administration Server



6. Enter the path to keystore.jks, including the file name, and enter the custom Identity Keystore passphrase you created for the keystore. Repeat this for trustore.jks, but enter the appropriate passphrase for the truststore. For an example, see Figure 6–41.

7. Switch to the SSL tab. Enter the alias name and private keyphrase as created during the certificate generation. To save the changes, click **Save**.

Figure 6–42 Save Settings for Administration Server



Note: For a clustered environment, disable the non-SSL port for the HttpClusterServlet proxy.

Complete the Security Configuration

Test both the container and transport level security using SOAPUI.

To set up the unlimited strength JCE files:

1. Download and install the correct version of the unlimited strength JCE files. For more information, see the *Oracle Retail Xstore Point of Service Implementation Guide*.
2. Configure WebLogic 12c with the Xstore suite of product's supported cipher suites. To configure it, update the <*domain*>\<*domain_name*>\config\config.xml file and add the following inside the ssl block:

```
<ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA</ciphersuite
<ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384</ciphersuite>
<ciphersuite>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384</ciphersuite>
```

3. Disable the schema validation in WebLogic by passing the JVM argument in the WebLogic startup script:

   -Dweblogic.configuration.schemaValidationEnabled=false

4. Xstore Office's RTLog Generator broadcaster end point should be configured to use the secured (HTTPS) URL for configuring the container level security section:

```
<property name="endpointAddress"
value="https://<hostname>:7002/rtlog-generator/service" />
```

   The endpoingAddress property is defined at xcenter-spring-beans.xml under Xcenter external configuration directory\xcenter-config. There are two required modifications:

   - Modify broadcasterManager bean in the file by uncommenting the line below.

```
<ref bean="ReSA_Broadcaster"/>
```

- Configure endpointAddress of the ReSA_Broadcaster_jaxws bean.

7

RTLog Generator Cloud

This chapter describes the RTLog Generator component of Xstore Office Cloud. The RTLog Generator Cloud application can be used with both the cloud or on premise Merchanding/Pricing applications.

RTLog Generator Cloud

This chapter describes how configure the RTLog Generator application deployed on cloud.

The RTLog Generator on cloud is a Java and XML based web application that exposes a Spring-JAXWS implemented SOAP web service and JAXRS implemented REST web services. It is distributed as a web archive along with a configuration .zip file ready to be deployed on an Oracle WebLogic 12c server. It is usually deployed alongside the other Xstore office cloud applications.

Configuration

The RTLog Generator cloud application can be configured in the following way.

Customize the RTLog Generator's mapping configuration via REST services.

Note: For more information on how to customize the RTLog Generator, see the Configuration section in Chapter 6, "RTLog Generator On Premise" and the *Retail Xstore - RTLog Generator Extension Guidelines (Doc ID 2174095.1)* on `https://support.oracle.com`.

Integration

This section describes the RTlog Generator Cloud integration.

Updating Mapping Configuration

RTLog Generator Cloud application provides three REST services to retrieve, update and delete the `RTLogMappingConfig.xml` file. All the three services point to the URL at

https://*<hostname>/rtlog-generator/rest/config/file/v1/RTLogMappingConfig*

A new property `configUploadDir` is added to `rtlogconfig.properties`.

If RTLog generator is deployed on cloud, its mapping configuration file `RTLogMappingConfig.xml` is not accessible to a user. To customize the mapping, restful APIs are provided to upload a customized `RTLogMappingConfig.xml` to override the default out-of-box one. This directory specifies the upload directory to host the customized mapping file. In a cluster environment with multiple RTLogGen nodes, all nodes must be configured to point to the same config upload directory on shared file system. This is to ensure that once a mapping file is uploaded, it is visible to all the nodes.

```
####
#### (uncomment and configure this if and only if you are setting up a cluster of
RTLogGen nodes on cloud)
####
####configUploadDir = DIR/rtlogconfig_upload
```

Table 7–1 REST Services related to the RTLogMappingConfig.xml

HTTP Protocol	Security Protocol	Response Type	Description
GET	OAuth2	application/xml	Returns the active RTLogMappingConfig.xml file. If the customer has not uploaded a customized configuration xml file yet, provides a copy of the default mapping configuration XML file that is provided with the deployment.
PUT	OAuth2	application/json	Customer submits the updated RTLogMappingConfig.xml file as the request body. Returns JSON that contains the number of bytes in the uploaded XML file.
DELETE	OAuth2	No content	If the customer has uploaded a configuration XML file previously, it will be deleted and HTTP 200 status is returned. If there is no customized RTLogMappingConfig.xml file active yet, HTTP 204 status is returned. The default RTLogMappingConfig.xml that is part of the deployment will resume being the active mapping configuration.

The examples below show how to retrieve and update the RTLogMappingConfig.xml.

Example 7–1 Get active RTLogMappingConfig.xml - Get Current RTLog Mapping Configuration

```
$ curl -H "Authorization: Bearer <token>"
https://<rlog-generator-host>/rtlog-generator/rest/config/file/v1/RTLogMappingConf
ig"  > RTLogMappingConfig.xml
```

Example 7–2 Update RTLogMappingConfig.xml - Update the RTLog Mapping Configuration

```
$ curl -H "Authorization: Bearer <token>" -X PUT -T "/path/to/mapping/file"
https://<rlog-generator-host>/rtlog-generator/rest/config/file/v1/RTLogMappingConf
ig"
```

Similar to the example above, using the -X option with the value of `DELETE` will delete any customer uploaded mapping configuration XML file.

Retrieving Published RTLog Files

RTLog Generator Cloud application's ability to provide a mechanism to retrieve the published RTLog files varies depending on the type of Sales Audit application that it is integrated with.

- For Merchandising on cloud, SFTP process is used to transfer the files
  - SFTP credentials to connect to the Merchandising's SFTP directory on cloud are made available to the RTLog Generator Cloud deployment team.
  - SFTP connectivity utilizes public/private key based authentication.
  - Cloud Application Management is responsible for SFTP credentials rotation.
- For Sales Audit on premise, REST service provides the way to download the files.

For Cloud Release 19.1, there is no coupling of RTLog delivery target (ReSA on cloud or ReSA on-premises) with delivery method (SFTP or rest service). RTLog generator's chef script allows a user to make two separate choices:

- If the delivery target is ReSA on cloud or ReSA on-premises.
- If the preferred delivery method is through SFTP or REST.

Security Configuration

RTLog Generator's web services are secured by requiring HTTPS protocol for transport layer security and require OAuth2 authentication for application level security. All of the Xoffice applications on cloud including the RTLog Generator have a valid OAuth Client (Application) registered with a specific tenant of the Oracle Identity Cloud Service. Sales Audit is required to do the same in order to communicate with the RTLog Generator application via REST web services.

OAuth2 authentication is a two-step process.

- Acquire a valid OAuth2 Bearer token using the IDCS or OCI IAM Client Credentials.
- Provide the token value in the HTTP Authorization header for all of the web service requests until the token's validity is expired.

Acquiring IDCS or OCI IAM Token

In order to acquire a valid IDCS or OCI IAM token, the following information is needed beforehand.

- IDCS or OCI IAM tenant host information to build the URL for requesting a token
  - https://*<IDCS_TENANT_HOST>/oauth2/v1/token*
- `ClientID` and `ClientSecret` for the RTLog Generator Client App (that is Sales Audit).
- A command line utility or any software that can make HTTP requests with the ability to setup specific header values
  - "curl" in Linux environments
- Access to a command/utility to encode the credentials in base64 format.
  - "base64" command in Linux environments
  - "certutil" command in Windows environments

The following example shows how to request a token using the curl command line tool in a Linux environment. Ensure to replace the `clientID`, `clientSecret` and `IDCS_TENANT_HOST` with the appropriate values.

Example 7–3 Request IDCS Oauth2 Token - OAuth2 Token Request

```
$ curl -i -H "Authorization: Basic $(echo -n clientID:clientSecret | base64)" -H
"Content-Type: application/x-www-form-urlencoded;charset=UTF-8" https://<IDCS_
TENANT_HOST>/oauth2/v1/token -d "grant_type=client_credentials&scope=urn:opc:idm:_
_myscopes__"
```

You may generate Base64 encoded text of the `"clientID:clientSecret"` ahead of the request and use it directly in the curl command for the Basic Authorization header value. The following example shows the response that contains the token.

Example 7–4 IDCS Oauth2 Token Response - OAuth2 Token Response

```
{"access_token": "<oauth2_token>",
"token_type": "Bearer",
"expires_in": 3600 }
```

The response above shows the token value and the expiration time in seconds. Usually, the token is a sequence of random characters of varying length up to a maximum of 16K.

Provide IDCS or OCI IAM Authentication

The following example shows how to provide the OAuth2 token while communicating with RTLog Generator REST services. The following example shows how to request the current active RTLogMappingConfig.xml file. Please make sure to replace the `"<token>"` with a valid OAuth2 token acquired in the last step and provide the correct RTLog Generator Host value.

Example 7–5 Provide Oauth2 Token - Provide OAuth2 Token for REST Services

```
$ curl -i -H "Authorization: Bearer <token>"
"https://<rlog-generator-host>/rtlog-generator/rest/config/file/v1/RTLogMappingCon
fig"
```

A

Appendix: Xstore to Sales Audit Mapping Details

The mapping from the Xstore POSLog format to the Sales Audit RTLog format is defined in the Xstore configuration file RTLogMappingConfig.xml. This appendix provides details on the following ode types used by each of the applications that are used in this mapping:

- Transaction Types
- Tender Types
- Tender Totals
- Item Types
- Sales Audit Reason Codes
- Item Status/and Sales Types
- Customer ID Types
- Sales Audit Tax Codes
- Reference Codes/Labels
- Sale Return Transaction

Transaction Types

Both Xstore and Sales Audit support a number of similar transactions types, but each solution uses different codes for these transaction types. The table below shows how the transaction types in Xstore map to the transaction types and sub-transaction types used in Sales Audit. See also the *Oracle Retail Sales Audit Implementation Guide* for more details on these codes.

Table A–1 Transaction Type Mapping

Xstore Transaction Type	Sales Audit Transaction Type TRAT	Sales Audit Sub-Transaction Type TRAS	Description
ACCOUNT_LOOKUP	OTHER	OTHER	ACCOUNT_LOOKUP transactions are passed from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit.
BALANCE_INQUIRY	OTHER	OTHER	BALANCE_INQUIRY transactions are passed from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit.
CREDIT_ APPLICATION	OTHER	OTHER	CREDIT_APPLICATION transactions are passed from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit.
ESCROW	OTHER	OTHER	ESCROW transactions are passed from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit.
EXCHANGE_RATE	OTHER	OTHER	EXCHANGE_RATE transactions are passed from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit.
GNRIC	OTHER	OTHER	GNRIC transactions are passed from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit.
INVENTORY_ CONTROL	OTHER	OTHER	INVENTORY_CONTROL transactions are mapped from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit. Xstore should be configured so that inventory control transactions are not generated, and therefore not sent to Sales Audit.
INVENTORY_ SUMMARY_COUNT	OTHER	OTHER	INVENTORY_SUMMARY_COUNT transactions are mapped from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit. Xstore should be configured so that inventory summary count transactions are not generated, and therefore not sent to Sales Audit.
MOVEMENT_ PENDING	OTHER	OTHER	MOVEMENT_PENDING transactions are mapped from Xstore to Sales Audit for full visibility audit, but not otherwise implemented in Sales Audit. Xstore should be configured so that inventory summary count transactions are not generated, and therefore not sent to Sales Audit.
NO_SALE	NOSALE	NOSALE	NA
POST_VOID	PVOID	VOID	NA

Table A–1 (Cont.) Transaction Type Mapping

Xstore Transaction Type	Sales Audit Transaction Type TRAT	Sales Audit Sub-Transaction Type TRAS	Description
RETAIL_SALE (can be mapped to multiple Sales Audit transaction types depending on other conditions)	SALE	SALE	Regular transaction.
	NOSALE	SUSPND	Suspend transaction.
	VOID	CANCEL	Cancel transaction.
	VOID	CANCEL	Cancel orphaned transaction.
SESSION_CONTROL	OTHER	OTHER	Issue till.
	OTHER	OTHER	Assign till/assign till tender transfer.
	OTHER	OTHER	Attach till.
	OTHER	OTHER	Remove till.
	OTHER	OTHER	Return till.
SYSTEM_CLOSE	CLOSE	CSTORE	Close store.
SYSTEM_OPEN	OPEN	OSTORE	Open store.
TENDER_CONTROL (can be mapped to multiple Sales Audit transaction types depending on other conditions)	OPEN	OTILL	Begin till count.
	CLOSE with TOTAL /OTHER	CTILL with CTILLT /OTHER	Till closing count (register accountability/till accountability).
	CLOSE and TOTAL	CTILL and CTILLT	Till reconcile. Each counted tender type has a corresponding TOTAL and CTILLT as a THEAD.
	PAIDIN	PITILL	Pay in.
	PAIDOU	POTILL	Pay out.
	OTHER	AUDIT	Till audit.
	PULL	PUTILL	Mid-day deposit. Place funds in store bank.
	OTHER	BANK	Bank deposit.
	LOAN	LOTILL	Till loan (cash transfer).
	PULL	PUTILL	Pick up till (cash pickup).
	OTHER	OTHER	Open store bank.
	OTHER	OTHER	Store bank reconcile.
TENDER_EXCHANGE	PAIDIN	PITILL	NA
TILL_CONTROL	OTHER	OTHER	NA
TIMECLOCK	OTHER	OTHER	Employee clock in.
	OTHER	OTHER	Employee clock out.
TRAINING_MODE_ ENTRY	OTHER	NTRAIN	NA
TRAINING_MODE_ EXIT	OTHER	XTRAIN	NA
WORKSTATION_ CLOSE	CLOSE	CREG	NA

Table A–1 (Cont.) Transaction Type Mapping

Xstore Transaction Type	Sales Audit Transaction Type TRAT	Sales Audit Sub-Transaction Type TRAS	Description
WORKSTATION_ COMPLETE_REMOTE_ CLOSE	CLOSE	CRGRC	NA
WORKSTATION_OPEN	OPEN	OREG	NA
WORKSTATION_ START_REMOTE_ CLOSE	OTHER	CRGRC	NA
GIFT_REGISTRY	OTHER	OTHER	Assign gift registry (register operation)
	OTHER	OTHER	Reissue gift registry (register operation)
RAIN_CHECK	OTHER	OTHER	Redeem rain check.
BATCH_CLOSE	OTHER	OTHER	Credit and debit settlement.
REOPEN	REOPEN	NA	Used to reopen a previously closed store day in Sales Audit.

Tender Types

In order to communicate the tender type used on transactions, a specific mapping is used between Xstore and Sales Audit. The details below outline how the tenders in Xstore map to the Sales Audit tender groups and types. See also the *Oracle Retail Sales Audit Implementation Guide* for information on configuration of tender types and tender type groups.

Table A–2 Tender Type Mapping

Xstore		Xstore POS Log Tender Group Type		Sales Audit RTLog	
Tender Type Code	Tender Typel D	Tender Type	Tender ID	Tender Type Group	Tender Type ID
CURRENCY	USD_ CURRENCY	Cash	USD_ CURRENCY	CASH	If primary 1000, if alternate 1010.
	AUD_ CURRENCY	Cash	AUD_ CURRENCY	CASH	If primary 1000, if alternate 1010.
	CAD_ CURRENCY	Cash	CAD_ CURRENCY	CASH	If primary 1000, if alternate 1010.
	EUR_ CURRENCY	Cash	EUR_ CURRENCY	CASH	If primary 1000, if alternate 1010.
	GBP_ CURRENCY	Cash	GBP_ CURRENCY	CASH	If primary 1000, if alternate 1010.

Table A–2 (Cont.) Tender Type Mapping

Xstore		Xstore POS Log Tender Group Type		Sales Audit RTLog	
Tender Type Code	**Tender Typel D**	**Tender Type**	**Tender ID**	**Tender Type Group**	**Tender Type ID**
CREDIT_CARD	VISA	CreditDebit	VISA	CCARD	3000
	MASTERCARD	CreditDebit	MASTERCARD	CCARD	3010
	AMERICAN_ EXPRESS	CreditDebit	AMERICAN_ EXPRESS	CCARD	3020
	DINERS_CLUB	CreditDebit	DINERS_CLUB	CCARD	3040
	DISCOVER	CreditDebit	DISCOVER	CCARD	3030
	JCB	CreditDebit	JCB	CCARD	3090
	DEBITCARD	CreditDebit	DEBITCARD	DCARD	8000
ACCOUNT	HOUSE_ ACCOUNT	dtv:Account	HOUSE_ ACCOUNT	CCARD	3120
	A new type of credit card	CreditDebit	A new type of credit card	CCARD	Map to UNKNW.
CHECK	CHECK	Check	CHECK	CHECK	If primary 2000, if foreign 2050.
TRAVELERS_ CHECK	USD_ TRAVELERS_ CHECK	dtv:TravelersC heck	USD_ TRAVELERS_ CHECK	CHECK	If primary 2020, if foreign 2060.
	CAD_ TRAVELERS_ CHECK	dtv:TravelersC heck	CAD_ TRAVELERS_ CHECK	CHECK	If primary 2020, if foreign 2060.

Table A–2 (Cont.) Tender Type Mapping

Xstore		Xstore POS Log Tender Group Type		Sales Audit RTLog	
Tender Type Code	**Tender Typel D**	**Tender Type**	**Tender ID**	**Tender Type Group**	**Tender Type ID**
VOUCHER	GIFT_ CERTIFICATE	Voucher	GIFT_ CERTIFICATE	VOUCH	If primary 4030, if foreign 4100.
	ISSUE_GIFT_ CERTIFICATE	Voucher	ISSUE_GIFT_ CERTIFICATE	VOUCH	If primary 4030, if foreign 4100.
	ISSUE_ MERCHANDIS E_CREDIT_ CARD	Voucher	ISSUE_ MERCHANDISE _CREDIT_CARD	VOUCH	4050
	ISSUE_STORE_ CREDIT	Voucher	ISSUE_STORE_ CREDIT	VOUCH	4050
	ISSUE_XPAY_ GIFT_CARD	Voucher	ISSUE_XPAY_ GIFT_CARD	VOUCH	4040
	MALL_ CERTIFICATE	Voucher	MALL_ CERTIFICATE	VOUCH	4060
	MERCHANDIS E_CREDIT_ CARD	Voucher	MERCHANDISE _CREDIT_CARD	VOUCH	4050
	RELOAD_ MERCHANDIS E_CREDIT_ CARD	Voucher	RELOAD_ MERCHANDISE _CREDIT_CARD	VOUCH	4050
	RELOAD_ XPAY_GIFT_ CARD	Voucher	RELOAD_ XPAY_GIFT_ CARD	VOUCH	4040
	STORE_ CREDIT	Voucher	STORE_CREDIT	VOUCH	If primary 4050, if foreign 4090.
	XPAY_GIFT_ CARD	Voucher	XPAY_GIFT_ CARD	VOUCH	4040
COUPON	COUPON	Manufacturer Coupon	COUPON	QPON	5000
	ROOM_ CHARGE	CreditDebit	ROOM_ CHARGE	VOUCH	4050
CREDIT_CARD	PAYPAL	TBD	PAYPAL	PAYPAL	3075
HOME_OFFICE_ CHECK	HOME_ OFFICE_ CHECK	NA	NA	Not supported in this solution. Home office check tenders should not be used in Xstore if it is integrated with Sales Audit.	

Tender Totals

Sales Audit uses tender totals to compare the total amount of a tender reported from the store (Accounted For) with the amount that it should have reported (Accountable For) based on store activity in a day. Totals are fully customer configured in Sales Audit. However, when integrating with Xstore, a few specific totals are expected to be created to total tenders. The totals that should be created and their ID are outlined in the table below, along with the Xstore tender type used for the total.

Table A–3 Total Tender ID Mapping

Xstore		Sales Audit RTLog
TenderType	**TenderID**	**Total ID**
CURRENCY	USD_CURRENCY	CASH
	AUD_CURRENCY	CASHAC
	CAD_CURRENCY	CASHAC
	EUR_CURRENCY	CASHAC
	GBP_CURRENCY	CASHAC
TRAVELERS_CHECK	USD_TRAVELERS_CHECK	TCHECK
	AUD_TRAVELERS_CHECK	TCHECKAC
	CAD_TRAVELERS_CHECK	TCHECKAC
	EUR_TRAVELERS_CHECK	TCHECKAC
	GBP_TRAVELERS_CHECK	TCHECKAC
	MXN_TRAVELERS_CHECK	TCHECKAC
CREDIT_CARD	CREDIT_CARD	CCARD
VOUCHER	GIFT_CERTIFICATE	GIFTCERT
	MALL_CERTIFICATE	MALLCERT
	MERCHANDISE_CREDIT_CARD	MCCARD
	RELOAD_MERCHANDISE_CREDIT_CARD	RMCCARD
	RELOAD_XPAY_GIFT_CARD	RXPAYGC
	STORE_CREDIT	STCRDT
	XPAY_GIFT_CARD	XPAYGC
	ISSUE_XPAY_GIFT_CARD	IXPAYGC
	ISSUE_STORE_CREDIT	ISTCRDT
	ISSUE_MERCHANDISE_CREDIT_CARD	IMCCARD
ACCOUNT	HOUSE_ACCOUNT	HACCNT
COUPON	COUPON	COUPON

Item Types

There are a number of different item types that are used in Xstore, and these all need to be mapped to item types used by Sales Audit. The table below outlines how the types in Xstore map to the Sales Audit item types.

Table A–4 Item Type Mapping

Xstore Item Type	Sales Audit Item Type	Description
Alteration	NMITEM	Non-Merchandise Item
Deposit	NMITEM	Non-Merchandise Item
dtv:GiftCertificate	GCN	Voucher
dtv:NonMerchandise	NMITEM	Non-Merchandise Item

Table A–4 (Cont.) Item Type Mapping

Xstore Item Type	Sales Audit Item Type	Description
dtv:Payment	NMITEM	Non-Merchandise Item
Fee	NMITEM	Non-Merchandise Item
ItemCollection	ITEM	Item
Service	NMITEM	Non-Merchandise Item
Stock	ITEM	Item
Warranty	NMITEM	Non-Merchandise Item

Sales Audit Reason Codes

Xstore has a single set of reason codes, used both for reason codes, price override codes, and other modifications. Sales Audit separates these concepts into individual sets. Because reason codes can be mixed coming out of Xstore, Sales Audit has mapped some code values to multiple code types to avoid the possibility of errors.

For more information on each of these categories of reason codes, see the *Oracle Retail Sales Audit Implementation Guide*.

Reason Codes

First is a general reason code that may be entered for specific transaction types to provide more information about the context of the transaction. This type of reason code is mapped to Xstore miscellaneous reason codes.

Table A–5 Sales Audit Reason Codes

Xstore Reason Code	Sales Audit Reason Code	Description
PV1	PV1	Cashier Error
PV2	PV2	Supervisors Discretion
PV3	PV3	Customer Satisfaction
NS1	NS1	Making Change
NS2	NS2	Employee Check Cashed
NS3	NS3	Petty Cash In
NS4	NS4	Petty Cash Out
NS5	NS5	Spiff/Bonus Out 1
CF1	CF1	Holiday Adjustment
CF2	CF2	Register Down
PAID_IN	PI1	Change from Paid Out
PAID_IN	PI2	Found Money
PAID_IN	PI3	Drawer Loan 1
PAID_IN	TENDEX	Tender exchange
PAID_OUT	PO1	Stocks
PAID_OUT	PO2	Delivery

Table A–5 (Cont.) Sales Audit Reason Codes

Xstore Reason Code	Sales Audit Reason Code	Description
PAID_OUT	PO3	Postage
PAID_OUT	PO4	Contractor Services
PAID_OUT	PO5	Store Incentives

Return Reason Codes

Sales Audit return reason codes are used to provide the context of the return in Sales Audit. These map to the Xstore reason codes as follows:

Table A–6 Return Reason Codes

Xstore Reason Code	Sales Audit Reason Code	Description
RET1	RET1	Did not like
RET2	RET2	Better price somewhere else
RET3	RET3	Did not fit
RET4	RET4	Damaged
RET5	RET5	Exchange
RET6	RET6	Poor quality
RET41	RET41	Open box
RET42	RET42	Unusable
RET43	RET43	Repairable

Discount Reason Codes

Used in Sales Audit to indicate the valid discount types for sales and return transaction, this table shows how these are mapped to the Xstore reason codes used for returns:

Table A–7 Discount Reason Codes

Xstore Reason Code	Sales Audit Reason Code	Description
DC1	S	Incorrect Label
DC2	MS	Manager Discretion
DC3	CP	Price Guarantee
DC4	D	Damage Adjustment
NEW_PRICE_RULE	NEWPRC	New Price Rule
DOCUMENT	DOC	Document
MANUFACTURER_ COUPON	MCOUP	Manufacturer Coupon
REFUND_PRORATION	REFUND	Refund Proration
CALCULATED_ WARRANTY_PRICE	CALWAR	Warranty Price

Item Price Override Reason Codes

This grouping of reason codes is used to hold the valid price override reason codes that are expected by Sales Audit. These map to the reason codes used in Xstore as follows:

Table A–8 Item Price Override Reason Codes

Xstore Reason Code	Sales Audit Reason Code	Description
AR_PR_1	AR_PR_1	Insufficient Funds
AR_PR_2	AR_PR_2	Wrong Amount
AR_PR_3	AR_PR_3	Wrong Amount
AR_PR_4	AR_PR_4	Wrong Invoice
COMMENT	NEWPRC	Other - Enter Comments
PC1	S	Incorrect Label
PC2	MS	Supervisors Discretion
PC3	CP	Competitive Price Match
PC4	D	Damage Adjustment
BASE_PRICE_RULE	BSPRC	Base Price Rule
PROMPT_PRICE_ CHANGE	PROMPT	Price Prompt
AUTHORIZED_ AMOUNT	AUTHMT	Authorized Amount

Item Status/and Sales Types

Valid values for item status in Sales Audit are:

V	Voided
S	Sale
R	Return
O	Other
ORI	Order Initiate
ORC	Order Cancel
ORD	Order Complete
LIN	Layaway Initiate
LCA	Layaway Cancel
LCO	Layaway Complete

Valid values for sales type in Sales Audit are:

R	Regular
I	In-Store Customer Order
E	External Customer Order

These two sets of codes are then mapped to the actions in Xstore as follows:

Table A–9 Item Status/ and Sales Type Mapping

Xstore Item	Xstore Action	Sales Audit Item Status	Sales Audit Sales Type
Regular Sale	Sale	S	R
	Return	R	R
	Void	S and V (two lines)	R
Layaway Item	Init	LIN	I
	Cancel	LCA	I
	Pickup	LCO	I
	Void	S and V (two lines)	I
Locate Order	Init	ORI	E
	Cancel	ORC	E
	Pickup	ORD	E
	Void when update or pickup	ORC	E
	Void when Init	S and V (two lines)	E
Special Order	Init	ORI	E
	Cancel	ORC	E
	Pickup	ORD	E
	Void when update or pickup	ORC	E
	Void when Init	S and V (two lines)	E
Work Order	Init	ORI	I
	Cancel	ORC	I
	Pickup	ORD	I
	Void when update or pickup	ORC	I
	Void when Init	S and V (two lines)	I
Pre-Sale	Init	ORI	I
	Cancel	ORC	E
	Pickup	ORD	E
	Void when update or pickup	ORC	E
	Void when Init	S and V (two lines)	E
On Hold	Init	ORI	I
	Cancel	ORC	I
	Pickup	ORD	I
	Void when update or pickup	ORC	I
	Void when Init	S and V (two lines)	I
Send Sale	Init	S	R
	Void when Init	S and V (two lines)	R

Customer ID Types

Sales Audit has a set of codes that it uses for validating the various forms of identification that can be presented at the store for a transaction. However, Xstore always sends just one type - Customer ID or CUSTID. This customer ID type is part of the default Sales Audit implementation and should be configured to "Used" when implementing with Xstore.

Sales Audit Tax Codes

When implementing using US Sales Tax for some or all stores, the tax code sent by Xstore will always be sent as TOTTAX. This will be validated against the list of valid non-VAT tax codes in Sales Audit. This code is included in the initial Sales Audit configuration, but care should be taken not to remove or configure that code type off when integrating with Xstore.

Reference Codes/Labels

Sales Audit has a flexible method of taking in reference fields from a POS at various levels of the transaction. These can be configured differently by transaction and sub-transaction type within Sales Audit. In general, these are not used in the base Xstore implementation without customization with a couple of exceptions. The reference fields that Xstore uses are as follows:

- Reference Number 1 (Transaction Header) - for the Day Close (DCLOSE) transaction type, this field will contain the file counter for the end of day
- Reference Number 1 (Transaction Header) - for the Tender Total (TOTAL) transaction type, contains the tender ID/type
- Reference Number 3 (Transaction Header) - incudes the employee ID if the sale was to an employee

For more on configuring reference fields in Sales Audit, see the *Oracle Retail Sales Audit Implementation Guide*.

Sale Return Transaction

This section describes sale return transaction mappings.

Transaction Header Mapping

This section describes mappings for the transaction header of sale return transactions. The transaction header format is defined at TransactionHeader RECODE_FORMAT in the RTLogFormatConfig.xml file.

A sale return transaction can be mixed with a sale line item, return line item, special order pickup line item, special order payment, work order line item, and so on. In this case, the sale and return transaction type and sub transaction type are defined as SALE. The actual types of line lines are specified at line item level.

The field Salesperson in Transaction Header is matched to -1. It means there is no Salesperson available at transaction level. Instead the field Salesperson in line item is populated, since it is possible to have different sales people for different line items in the same transaction.

OriginalTransactionNumber and Orig_reg_no are only populated for post void transactions. Mapper PostVoidOriginalTransactionInfoMapper indicates the mapping logic.

Line Item (TITEM) Mapping

This section describes mappings for line items of sale return transactions. The line item format is defined at TransactionItem RECODE_FORMAT in the RTLogFormatConfig.xml file.

The line item will not be exported for suspended, cancelled and post void transactions. It is disabled in retailTrnDetailrExportabilityMapper.

It would be one or more line item records in a sale return transaction.

There are two kinds of item types, physical/merchandise item (ITEM) or non physical/non merchandise item (NMITEM). Non physical items are service fee, payment, work order item, and so on.

The Item id of a physical item is set to Item field; the Item id for a non physical item is set to NonMerchandiceItem field of the RTLog record.

The default value of the ItemStatus field is S, which is set in the RTLogFormatConfig.xml file. If the line item is return, the value is set to R. Return reason code mapping can be found in /retailTransaction/lineItems/return source field VALUE_MAPPINGS.

The default value of SalesType for sale return line item is I (internal sale).

ItemVoidStatusMapper sets the void flag of the record to true if the line is voided. If a sale return line item record is voided, a clone of the sale return line item record will be created, the item status for the new record is set to "V". In this case the status of the two lines are "S" and "V" or "R" and "V". Sales Audit balances off a sale/return line with its voided line to zero.

Item Discount (IDISC) Mapping and Round off Discount Mapping

This section describes mappings for item level discounts of sale return transactions. The item level discount format is defined at ItemDiscount RECODE_FORMAT in the RTLogFormatConfig.xml file.

The item level discount will not be exported for suspended, cancelled and post void transactions. It is disabled in retailTrnDetailExportabilityMapper.

It would be zero or more item discount records for a line item.

If it is a Deal discount, the RMSPromotionType is mapped to 9999, otherwise it is mapped to 1004.

The discount type of ItemDiscount is mapped from reasonCode and discountReasonCode.of RetailPriceModifierType Poslog object.

The values of DiscountReferenceNumber, PromoComponent and RMSPromotionType are mapped from PromotionIDMapper. If the retail price modifier is voided, DiscountReferenceNumber and PromoComponent are set to zero. If the retail price modifier is not voided and it is a deal promotion, it is mapped based on the following conditions:

- If it is an RPM promotion, DiscountReferenceNumber is set to RPM promotion id and PromoComponent is set to promotion component id.

- If it is not an RPM promotion, RMSPromotionType is set to 2000, DiscountReferenceNumber and PromoComponent are set to zero.

PromotionIDMapper implements this logic.

DiscountAmountMapper calculates unit discount amount for the ItemDicount RTLog IDISC record and also sets roundedOffAmount to the RTLog record. The value of unit discount amount is the total discount amount divided by item quantity rounding by half up with unit discount scale. The value of roundedOffAmount is the difference between total discount amount and unit discount amount multiplies item quantity. Note that the roundedOffAmount is not a field of IDISC, therefore it is not going to be exported.

A round off discount record (ItemRoundedOffDiscount) is always created after a discount record (ItemDiscount) if the roundedOffAmount is not equal to zero. In a discount record, if the roundedOffAmount exists, a round off discount record will be cloned from the discount record, otherwise a new round off discount record will be created. The value of discount amount of a round off discount record is the round off amount of the corresponding discount record. The value of the quantity in a round off discount record is always set to zero. RoundedOffDiscountAmountMapper is specially used for this mapping.

The format of ItemRoundedOffDiscount record and ItemDiscount record are exactly the same, that is IDISC. The reason for adding a round off discount record is that Poslog uses discountAmount and RTLog uses UnitDiscountAmount. If a quantity of a line item is not one, such as 3, the UnitDiscountAmount times 3 is not equal to discountAmount in Poslog. To balance it off, an extra discount line is introduced to set the round off amount to unit discount amount. It guaranties the total discount value is equal to sum of the two IDISC records.

Item Level Tax Mapping

This section describes mappings for item level tax of sale return transactions. The item level tax format is defined at ItemGlobalTax RECODE_FORMAT in the RTLogFormatConfig.xml file.

Tax lines will not be exported for suspended, cancelled and post void transactions. It is disabled in retailTrnDetailExportabilityMapper.

It would be one or more tax lines for one sale return line item. It depends on how many tax authorities are applied on this item.

RTLog requires four digits decimal for TaxRate in ItemGlobalTax record format. AmountMapper object sets the scale of tax rate to four to fit the need.

RTLog requires four digits decimal for TaxAmount in ItemGlobalTax record format. AmountMapper object sets the scale of tax amount to four to fit the need.

The value of TaxAmountSign field is P when tax amount is positive, N when tax amount is negative. It is done is SignMapper.

The RTLog Generator application supports either VAT or non VAT. It does not support both of them in the application instance. VAT and non VAT configurable settings can be found at the roundedOffTaxAmountMapper spring bean in the RTLogMappingBean.xml file.

TaxCode mappings are different for VAT and non VAT systems. If the RTLog Generator application is set to VAT, the tax group id maps to the tax code. If the RTLog Generator application is set to non VAT, the tax code is always "TOTTAX". ItemTaxCodeMapper implements this logic.

The value of TotalTaxAmount is blank if the tax is voided. Otherwise, the value of TotalTaxAmount is the sum of the item taxes from all item tax authorities. It is implemented at ItemTotalTaxAmountMapper.

The value of TaxableIndicator of ItemGlobalTax is a flag to indicate the item has tax or not. If the TaxAmountSign amount sign is zero, the value of TaxableIndicator is set to N, otherwise it is set to Y.

Tender Line Mapping

This section describes mappings for transaction tender of sale return transactions. The transaction tender format is defined at TransactionTender RECODE_FORMAT in the RTLogFormatConfig.xml file.

The tender line will not be exported for suspended, cancelled and post void transactions. It is disabled in retailTrnTenderExportabilityMapper.

The actual mappings of transaction tender type group and tender type id (sub type) are listed on sourcePath "/transaction/lineItems/tender" in the RTLogMappingConfig.xml file.

There may be zero or more tender line items within one sale return transaction.

The tender type id is different for base currency and foreign currency and is also different for base traveler's check and foreign traveler's check. Object tenderTypeIDMapper is responsible for doing the actual currency and traveler's check mapping. If the tender type is currency or traveler's check, mapping value will be found from the mapper, the corresponding VALUE_MAPPINGS will be skipped. Otherwise, it will do the value mapping in the VALUE_MAPPINGS.

RTLog requires four digits decimal for TenderAmount in TransactionTender record format. Mapper amountMapper sets the scale of tender amount to four to fit the need.

The value of TenderSign field will be P when tender amount is positive, N when tender amount is negative. It is done in SignMapper.

There is a special tender line which is used for transaction rounded off amount. For detail information, refer to the next section.

Rounded Off Amount

Transaction rounded amount is used to balance off a transaction. The amount comes from the following areas.

- Rounded amount from roundedTotal@RetailTransactionType Poslog object.
- Total tax amount from the sum of amount@TaxType of each tax LineItem of RetailTransactionType Poslog object.
- Total sale return line item tax amount from the sum of amount@TaxType of SaleType of LineItem of RetailTransactionType Poslog object.

Transaction rounded off amount = value of 1 + (value of 2 - value of 3).

RTLog exports line item taxes rather than exports transaction level taxes. The reason is that the payment amount in an order transaction includes a certain percentage of item price and tax amount. When doing a pickup, the transaction level tax is not zero, and the payment already includes the part of the tax. The total tax will be more than the value is supposed to be. The transaction will not balance off.

A special RECORD_FORMAT TransactionTenderRoundedOff is created for this purchase. The syntax of the format is exactly the same as the one for

TransactionTender. The TenderTypeGroup is always set to CASH and TenderTypeID is always 1020. TenderAmount of TransactionTenderRoundedOff is the transaction rounded of amount, TenderSign is the transaction rounded of amount sign. RoundedOffAmountMapper accumulates the rounded amounts based on the algorithm above and sets the value and sign to TenderAmount and TenderSign fields.

This special tender line will not be exported for suspended, cancelled and post void transactions. It is disabled in retailTrnDetailExportabilityMapper. It will not be exported when the total rounded amount is zero.