

**Oracle® Retail Xstore Suite 15.0/Merchandising
15.0**

Implementation Guide

Release 15.0

E64570-01

December 2015

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Primary Author: Bernadette Goodman

Contributing Author: Siobhan McMahon, Abhinav Sharma, Owen Horne, Sandy Gu

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

- (i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.
- (ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.
- (iii) the software component known as **Access Via**™ licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.
- (iv) the software component known as **Adobe Flex**™ licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR

Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, 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

Send Us Your Comments	ix
Preface	xi
Audience	xi
Documentation Accessibility	xi
Related Documents	xi
Customer Support	xii
Review Patch Documentation	xii
Improved Process for Oracle Retail Documentation Corrections	xii
Oracle Retail Documentation on the Oracle Technology Network	xiii
Conventions	xiii
1 Overview	
2 Data Flow from Merchandising to Xstore	
Conceptual Data Flow	2-1
Technical Implementation	2-2
RMS Foundation Data Extracts	2-3
RMS Batch Jobs	2-3
RPM Extracts	2-4
RPM Batch Jobs	2-5
Data Import Flow	2-5
Xstore DataLoader	2-5
DataLoader Detailed Description	2-6
Merchandising File Consumption by Location	2-9
3 Transaction Flow from Xstore to ReSA	
Conceptual Data Flow	3-1
Technical Implementation	3-2
Xstore Broadcaster	3-2
RTLog Generator	3-3
ReSA saimptlog/i	3-3

4	Configuration	
	RMS.....	4-1
	Diffs	4-1
	Item.....	4-2
	Merchandise Hierarchy	4-3
	Organizational Hierarchy	4-3
	Related Item	4-3
	Store.....	4-4
	VAT	4-5
5	Integration Considerations	
	Functional Considerations.....	5-1
	Technical Considerations.....	5-4
	RMS	5-4
	RPM.....	5-4
	Xstore	5-4
6	RTLog Generator	
	Configuration.....	6-1
	Deployment.....	6-6
	WebLogic Cluster Setup.....	6-8
	Deployment of the RTLog Generator Application on a Cluster	6-19
	Security Configuration.....	6-22
	Container Level Security	6-22
	Transport Level Security	6-25
	Complete the Security Configuration	6-28
7	Existing Functionality Gaps	
	RMS.....	7-1
	RPM.....	7-3
	Xstore to ReSA Integration.....	7-4
A	Appendix: POSLog to RTLog Mapping Details	
	Transaction Type Mapping.....	A-1
	Tender Type Mapping	A-3
	Total Tender ID Mapping	A-5
	Item Type Mapping.....	A-6
	Reason Code Mapping.....	A-7
	ReSA Reason Codes	A-7
	ReSA Return Reason Codes.....	A-8
	ReSA Discount Reason Codes	A-8
	ReSA Item Price Override Reason Codes	A-9
	Item Status/Sales Type Mapping.....	A-9

List of Figures

2-1	Conceptual Data Flow from Merchandising to Xstore Suite.....	2-2
2-2	RMS Foundation Data Bulk Export.....	2-3
2-3	RPM Extract Flow	2-4
2-4	DataLoader Flow.....	2-6
2-5	Example of Loaded RMS and RPM Files.....	2-10
3-1	Xstore to ReSA Transaction Flow	3-2
6-2	Example of context-param Field Update.....	6-2
6-5	RTLogMappingConfig.xml Field Mapper Example 2.....	6-6
6-7	Administration Console Control Page.....	6-7
6-8	Administration Console Install Application Assistant Page	6-7
6-10	Configuration Wizard Configuration Type Page.....	6-9
6-11	Configuration Wizard Templates Page	6-9
6-12	Configuration Wizard Administrator Account Page	6-10
6-13	Configuration Wizard Domain Mode and JDK.....	6-10
6-14	Configuration Wizard Advanced Configuration Page	6-11
6-15	Configuration Wizard Administration Server Page	6-11
6-16	Configuration Wizard Node Manager Page.....	6-12
6-17	Configuration Wizard Managed Servers Page.....	6-13
6-18	Configuration Wizard Clusters Page	6-13
6-19	Configuration Wizard Assign Servers to Clusters Page	6-14
6-20	Configuration Wizard HTTP Proxy Applications Page.....	6-14
6-21	Configuration Wizard Machines Page.....	6-15
6-22	Configuration Wizard Assign Servers to Machines Page.....	6-15
6-23	Configuration Wizard Configuration Summary Page	6-16
6-24	Administration Console Settings Page	6-18
6-25	Administration Console Configuration Page	6-19
6-27	Administration Console Deployments Page.....	6-20
6-28	Administration Console Install Application Assistant Page	6-20
6-29	Install Application Assistant Select Deployment Targets Page	6-21
6-30	Summary of Deployments Page	6-21
6-32	WebLogic Plugin Enabled Parameter	6-22
6-33	Administration Console Summary of Security Realms Page.....	6-23
6-34	Create a New Group Page	6-23
6-35	Create a New User Page	6-24
6-36	Users Page.....	6-24
6-37	User Settings Page.....	6-25
6-39	Administration Console Servers Page	6-26
6-40	Keystores Settings.....	6-26
6-41	Settings for the Administration Server	6-27

List of Tables

7-1	RMS to Xstore Functionality Gaps	7-1
7-2	RPM to Xstore Functionality Gaps	7-3
7-3	Xstore to ReSA Integration Functionality Gaps	7-4
A-3	Total Tender ID Mapping.....	A-6
A-4	Item Type Mapping	A-6
A-5	ReSA Reason Codes.....	A-7
A-6	ReSA Return Reason Codes.....	A-8
A-7	ReSA Discount Reason Codes.....	A-9
A-8	ReSA Item Price Override Reason Codes.....	A-9
A-9	ReSA Item Status/Sales Type Mapping	A-10

Send Us Your Comments

Oracle Retail Xstore Suite 15.0/Merchandising 15.0 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 15.0 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, 15.0) or a later patch release (for example, 15.0.1). 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 Technology Network 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 Technology Network at the following URL:

<http://www.oracle.com/technetwork/documentation/oracle-retail-100266.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 Technology Network

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

<http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html>

(Data Model documents are not available through Oracle Technology Network. You 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.
<i>italic</i>	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.

Overview

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

- Foundation and price data from Oracle Retail Merchandising System (RMS) and Oracle Retail Price Management (RPM) to Oracle Retail Xcenter and Xstore Office
- Point of Service transactions from Oracle Retail Xstore Point of Service to Oracle Retail Sales Audit (ReSA)

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"](#): This chapter describes the flow of data from the Merchandising applications to the Xstore applications.
- [Chapter 3, "Transaction Flow from Xstore to ReSA"](#): This chapter describes the flow of transactions from Xstore Point of Service to ReSA.
- [Chapter 4, "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"](#): This chapter covers how to install, deploy, and configure the RTLog Generator application.
- [Chapter 7, "Existing Functionality Gaps"](#): This chapter describes any functional gaps that exist between Xstore and the Merchandising applications.
- [Appendix A, Appendix: POSLog to RTLog Mapping Details](#): This appendix provides tables that list the mappings.

Data Flow from Merchandising to Xstore

This chapter covers the data flow from RMS and RPM to Xcenter/Xstore.

RMS is the source of foundation data. RMS foundation data sent to Xcenter/Xstore is limited to the following:

- Merchandise hierarchy
- Organizational hierarchy
- Store (including addresses)
- Diff IDs
- Diff groups
- Items
- VAT tax rules and item associations
- Related items

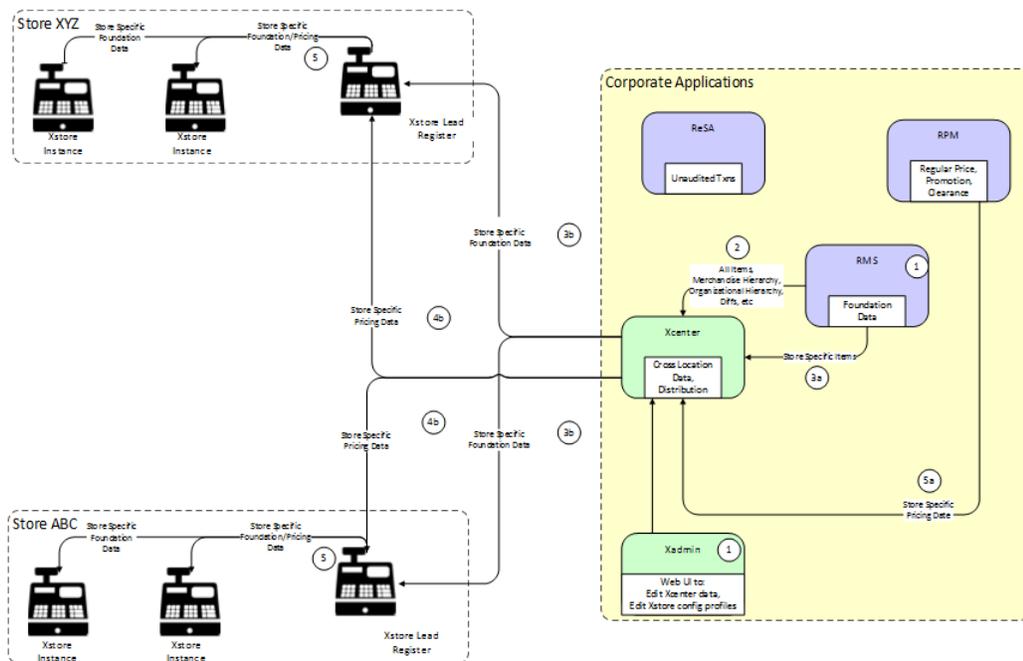
RPM is the source of pricing data. RPM pricing data sent to Xcenter/Xstore is limited to the following:

- Price changes
- Promotions
- Clearance prices

Conceptual Data Flow

[Figure 2-1](#) illustrates the data flow from the Merchandising applications to Xcenter/Xstore.

Figure 2–1 Conceptual Data Flow from Merchandising to Xstore Suite



The following steps describe the flow in Figure 2–1:

1. Manual process to set up some infrequently changing foundation data in both Xstore Office and RMS, for example, currency codes.
2. RMS produces foundation data consumed by Xcenter. Xcenter loads this foundation data to, among other things, facilitate cross location transactions. Xcenter loads foundation data to the appropriate lead registers at individual stores.
3. RMS produces store-specific foundation data consumed by Xcenter (3a). Xcenter does not load this data into its own repository, but instead distributes these files to the appropriate store lead registers (3b).
4. RPM produces store-specific pricing data consumed by Xcenter (4a). Xcenter does not load this data into its own repository, but instead distributes these files to the appropriate store lead registers (4b).
5. Lead register distributes store specific foundation data to all other Xstore instances in the store. This step occurs when the store is being closed during end of day processing.

Technical Implementation

The technical implementation of the foundation/price data from Merchandising to Xcenter/Xstore consists of three main components:

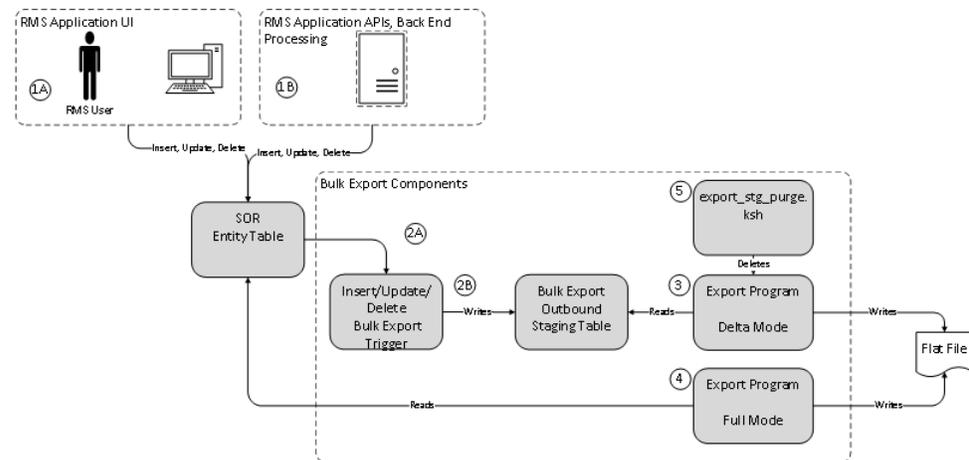
- [RMS Foundation Data Extracts](#)
- [RPM Extracts](#)
- [Xstore DataLoader](#)

RMS Foundation Data Extracts

RMS writes data to .dat files using a series of .ksh extract scripts. These scripts support both kill/fill (full) and delta processing. Many of these scripts also support creating files that apply either to all stores (for Xcenter, to facilitate cross store processing) or store-specific files. For an example, see [Figure 2-5](#).

There are some entity specific variations, but RMS uses a general pattern for foundation data bulk export as shown in [Figure 2-2](#).

Figure 2-2 RMS Foundation Data Bulk Export



The following steps describe the data bulk export flow shown in [Figure 2-2](#):

1. A business user using the RMS application UI (1A) or an API/Batch Process (1B) performs an insert/update/delete on a System of Record (SOR) table.
2. Trigger on SOR entity table fires on insert/update/delete (2A). Trigger writes new/changed/deleted information to the outbound staging table (2B).
3. In delta mode, the program reads the bulk export staging table to get recently created, modified, and deleted records and writes them to a file. Records are marked as exported.
4. In full mode, the program reads all current records from the SOR table and writes them to a file. Note that recently deleted records are not part of the data set.
5. `export_stg_purge.ksh` drops aged partitions from the export outbound staging tables.

Note: If bulk extract programs are not run for some time, it is possible that delta records will be purged without having been exported. It is important to run these jobs daily.

For more detailed information, see the following documents:

- Retail Reference Architecture available on My Oracle Support
- *Oracle Retail Merchandising System Operations Guide, Volume 2 - Batch Master Designs*

RMS Batch Jobs

The following batch jobs are used for the integration:

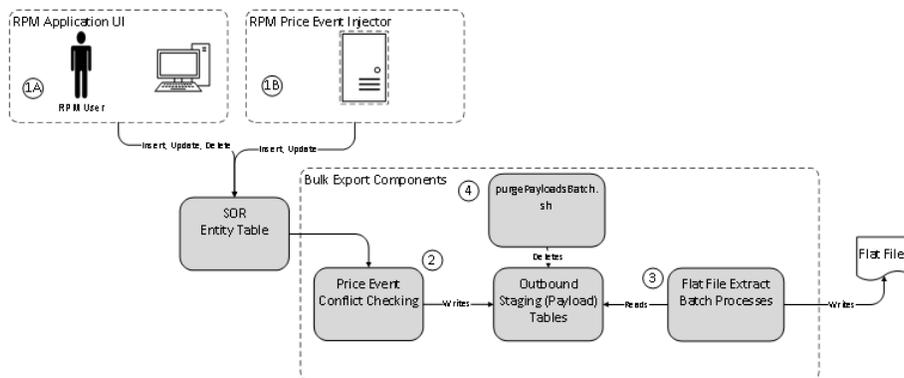
- export_merchhier.ksh
- export_orghier.ksh
- export_stores.ksh
- export_diffs.ksh
- export_diffgrp.ksh
- export_itemmaster.ksh
- export_itemloc.ksh
- export_itemvat.ksh
- export_relitem.ksh
- export_vat.ksh
- export_stg_purge.ksh

For more information, see the *Oracle Retail Merchandising System Operations Guide, Volume 2 - Batch Master Designs*.

RPM Extracts

RPM writes data to .dat files using a set of base processes. These base processes send pricing data to both Xcenter/Xstore and Oracle Retail Store Inventory Management (SIM). This ensures consistent price information across the Xstore applications. For an example, see [Figure 2-5](#).

Figure 2-3 RPM Extract Flow



The following steps describe the flow in [Figure 2-3](#):

1. A Pricing Analyst creates and approves price events (regular price changes, clearances, and promotion details) in the RPM UI (1A). Price events are created and approved in a bulk fashion using the Price Event Injector batch process (1B).
2. With data created on the SQR tables in RPM, the conflict checking process stages data on the outbound (payload) tables (2).
3. Data on the outbound staging tables is read by the flat file extraction batch processes to create delimited flat files. The associated outbound staging data is flagged as having been extracted as part of these batch processes (3).
4. `purgePayloadsBatch.sh` purges data from the staged outbound tables that has already been extracted. This purge batch is run in conjunction with the extract batch processes (4).

For more detailed information, see the following documents:

- Retail Reference Architecture available on My Oracle Support
- *Oracle Retail Price Management Operations Guide*

RPM Batch Jobs

The following batch jobs are used for the integration:

- RegularPriceChangePublishBatch
- PromotionPriceChangePublishBatch
- ClearancePricePublishBatch

For more information, see the *Oracle Retail Price Management Operations Guide*.

Data Import Flow

The following process describes the flow of the Merchandising data file import:

1. Xstore Office plays a central role in the Merchandising data importing. It periodically polls the configured auto-deploy directory on the file system. The interval is configurable, and is 15 minutes by default. For information on how to configure these settings, see the *Oracle Retail Xstore Office User Guide*.
2. Data files (.dat) generated by the RMS/RPM extract programs are delivered to Xstore Office's auto-deploy directory in the form of a zip-format archive file using the file extension .momzip. It is the System Integrator's responsibility to create the archive, and deliver it using a preferred file transfer protocol.
3. Once the archive is detected by Xstore Office, it regroups its content into deployments based on their targeted locations. For data files targeted to corporate, it invokes DataLoader immediately to import them into the Xcenter data source. For data files targeted to a store, it creates a deployment of these files to the store, and uploads them to an Apache Server for the store to download at close time. For details on the set of Merchandising files targeted to corporate or stores, see "[Merchandising File Consumption by Location](#)".
4. Once a store is closed, Xenvironment of the lead register pulls down the files from the Apache server and runs DataLoader to import all the files deployed into the store primary database.

Xstore DataLoader

DataLoader is the Xstore component responsible for translating RPM and RMS flat files into database data that can be used by Xstore. It can consume .mnt files from third-party sources, or the foundation data batch .dat files produced by RMS and RPM.

The DataLoader interacts with Xstore Office, Xcenter, Xenvironment, and Xstore Point of Service to provide a complete automated solution for the propagation of foundation data changes to the centralized and store-level databases used in an enterprise Xstore deployment. Xstore data not supplied by RPM and RMS can also be loaded by the DataLoader using its native .mnt format.

The DataLoader is designed to adapt flat files of data into relational data that Xstore can use. These flat files are referred to generically as data files within the DataLoader. Each field in a data file is delimited by a vertical bar (|) character. The DataLoader is configured to detect file types so it can process a data file's lines in distinct units of work appropriate for the type of file. For most file types, a unit of work corresponds to

a single line of flat file data; however, RPM promotion files are an example of a file type where a unit of work can consist of multiple lines of flat file data.

If a failure occurs during DataLoader processing of a data file, all SQL statements associated with the unit of work are rolled back and the error is logged. Processing continues with the next unit of work in the data file.

For more detailed information, see the following documents:

- Retail Reference Architecture
- *Oracle Retail Xstore Point of Service Host Interface Guide* available on My Oracle Support.

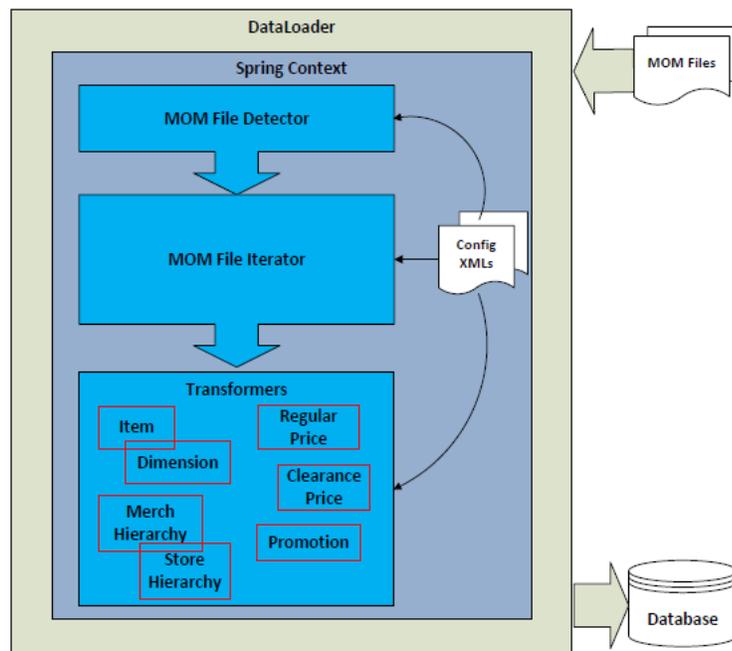
Both documents are available on My Oracle Support.

DataLoader Detailed Description

DataLoader is configured at the corporate and store level. It is responsible for detecting and sorting incoming data files, and iterating through them to convert each file record into persistable objects (DAO or SQLQuery) and writing them into the database. All its major components are spring loaded from dataloader-bean.xml, and are therefore highly customizable.

Figure 2–4 shows the flow of the DataLoader.

Figure 2–4 DataLoader Flow



The following sections describe each major execution phase.

File Detection

DataLoader is configured with a list of detectors to identify known file types that can be processed. Unknown file types are skipped and not processed. A Merchandising file detector is configured to identify all types of Merchandising data files and their meta data.

The detection is based solely on file names. Regular expressions are configured to perform pattern matching in file name to identify Merchandising file types and its meta data including target store ID, fill type, timestamp, and line count:

- Merchandising file type detection

A file name is matched against regular expressions configured to detect its Merchandising type. If no match is found, the file is not a Merchandising file type. The keys in the bean configuration are the Merchandising file types that Xstore/Xcenter care about.
- Target store detection

Target store ID is used by DataLoader as well as Xstore Office to determine the deployment target of a Merchandising file.

A file name is matched against regular expressions configured to detect its target store ID. Not all Merchandising file types have a target store ID configured:

 - If a store ID is not detected, the file is deployed to all stores and imported into Xcenter.
 - If a store ID is detected and is corp, the file is imported into Xcenter only.
 - If a store ID is detected and is not corp, it is deployed to the store. With the exception of Item Header batch, it is also imported into Xcenter.
- Timestamp detection

Timestamp is used by DataLoader to sort the files. For more details, see "[File Sorting](#)". A file name is matched against regular expressions configured to detect its timestamp.
- Line count detection

Line count is used by DataLoader to validate a file. If the number of lines in the file (excluding FHEAD and FTAIL) does not match the line count, a warning is logged. A file name is matched against regular expressions configured to detect its line count. Only RMS extracts support line count in their file names.

File Sorting

There are some data dependencies when importing RMS files into Xstore, such as related item detail that needs to be imported after the related item header. When DataLoader is called to import multiple Merchandising files in the same deployment, it applies sorting to the files before importing them.

A detector is configured to have a sorting strategy, which is used to sort all the files the detector detects. A Merchandising file sorting strategy bean is configured for the Merchandising file detector to perform sorting for all Merchandising files based on their file types. Files of the same Merchandising file type are sorted based on their timestamps. Out of the box the following sorting order is specified:

- Org Hierarchy
- Store
- Store Address
- Merchandise Hierarchy
- VAT
- Diff Group Head
- Diff Group Detail

- Diff
- Item Head
- Item Loc
- VAT Item
- Related Item Head
- Related Item Detail
- Regular Price Change
- Clearance Price Change
- Promotion Price Change

File Loading Dependency

Although the sorting strategy configuration lists all Merchandising file types, not all file types have file loading dependencies. The actual dependencies are shown in the following table:

File Type	Depends on
VAT Item	Item Loc
Store Address	Store
Related Item Detail	Related Item Head, Item Loc
Item Loc	Item Head
Item Head	Diff, Diff Group Detail, Diff Group Head, Merchandise Hierarchy
Diff Group Detail	Diff Group Head
Diff	Diff Group Detail, Diff Group Head

File Iteration and Transformation

DataLoader processes each file in the sorted order. It invokes a file iterator to process each file. A file iterator implements Java Iterator interface. During each iteration, it transforms flat file records into a list of IPersistable (DAO or SQLQuery) objects, and returns them.

A Merchandising file iterator is configured for each Merchandising file type. It processes lines between unit dividers as a data unit that should be transformed together:

- A single line iterator that expects each line in the file, other than FHEAD or FTAIL, is a data unit that gets transformed during each iteration. One and only one line in a unit is expected. An exception is raised if that is not the case.
- A multi-line iterator expects multiple lines to form a data unit that gets transformed together during each iteration. A unit may contain one or more lines. Out of the box, only promotion price change is configured to have a multi-line iterator.
- Unit dividers are lines that end a unit. They are configured as unit definitions for each Merchandising file type.

A Merchandising transformer is called to convert a unit of data from a flat file to a list of IPersistable (DAO or SQLQuery) objects. A transformer is configured for each Merchandising file type.

All Merchandising transformers implement the IMOMDataTransformer interface, which defines two APIs:

- The transform API is invoked by the iterator in each iteration. It does all the transformation to turn a unit of flat file data to a list of IPersistable objects to create, update, or delete foundation data records in database.
- The purgeData API is invoked once for a file by the iterator. It is only called if the file is for a full reload. It returns a list of IPersistable objects to remove all existing records sourced from Merchandising.

Persisting into the Database

DataLoader saves IPersistable objects to database in batches. A batch contains a list of AtomicPersistables objects. The maximum number of AtomicPersistables objects in a batch is configurable. An AtomicPersistables is a container of a group of IPersistable objects that must all be persisted or rolled back together as a unit. All IPersistable objects returned in one iteration are grouped into one AtomicPersistable object.

DataLoader first attempts to persist and commit all IPersistable objects from all AtomicPersistables objects in a batch together. If it fails, it tries to persist and commit IPersistable objects from one AutomiPersistables at a time. The number of succeeded and failed records are written to summary.ini files.

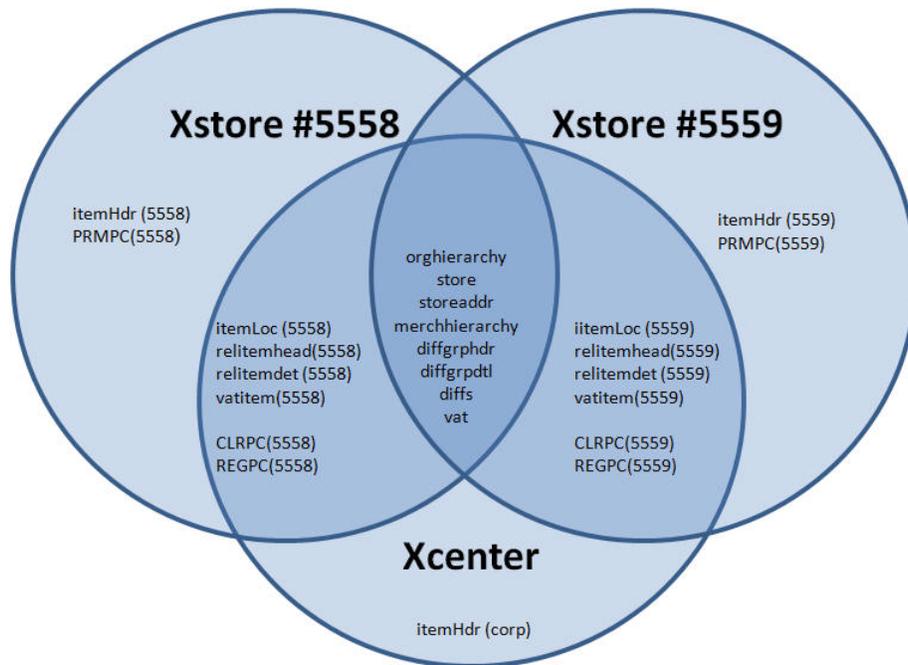
Merchandising File Consumption by Location

The files produced by the RPM and RMS extract programs containing data loaded into the Xcenter and Xstore databases comprise four data sets. A data file's targeted location is specified in its file name:

- Data loaded into Xcenter
- Data loaded into Xcenter and all stores
- Data loaded into one store
- Data Loaded into all stores

[Figure 2-5](#) illustrates the type of RMS and RPM files loaded at each location, using a two store chain example.

Figure 2-5 Example of Loaded RMS and RPM Files



Transaction Flow from Xstore to ReSA

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

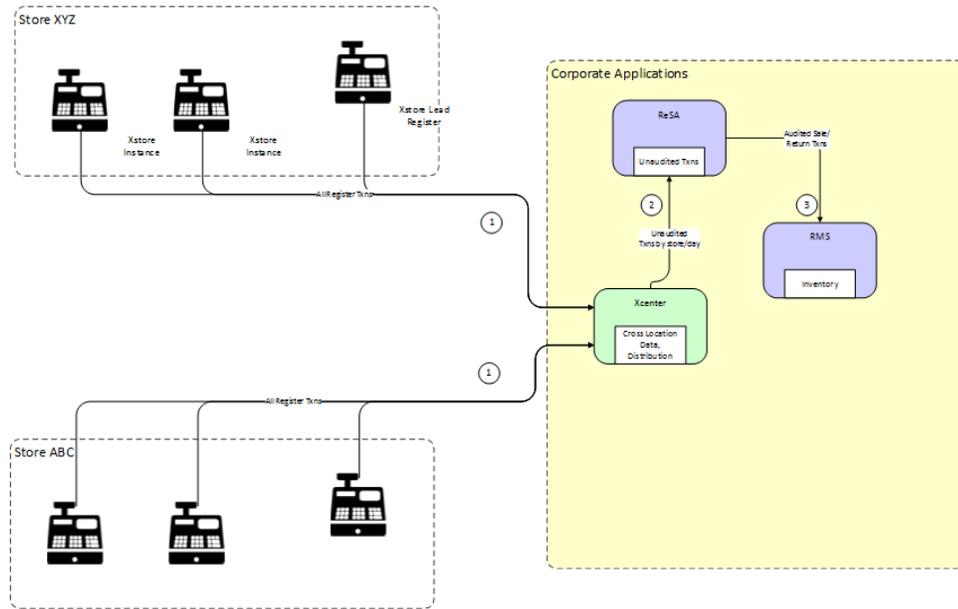
All transactions produced in Xstore are sent to ReSA. ReSA processing is primarily concerned with transactions that alter inventory or contain payment. ReSA loads other types of Xstore transactions (such as entering training mode, gift registry creation, and so on) into an OTHER transaction type for 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.

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

Conceptual Data Flow

[Figure 3-1](#) illustrates the transaction flow from Xstore to ReSA.

Figure 3–1 Xstore to ReSA 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 ReSA in the form of RTLogs generated multiple times per day. For more information, see "[ReSA saimptlog/i](#)".
3. After successful totaling and auditing, ReSA sends all sale/return transactions to RMS, where the transactions impact perpetual inventory. For detailed information about `uploadsales_all.ksh`, see *Oracle Retail Merchandising System Operations Guide, Volume 2 - Batch Master Designs*.

Technical Implementation

The technical implementation of the foundation/price data from Merchandising to Xcenter/Xstore consists of three main components:

- [Xstore Broadcaster](#)
- [RTLog Generator](#)
- [ReSA 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*

RTLog Generator

RTLog generator is a component that collects and aggregates broadcaster transactions and transforms them to the RTLog file format. The RTLog generator is packaged with Xstore, but is generally deployed in the same file system as ReSA.

For more information, see [Chapter 6](#).

ReSA saimptlog/i

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

- saimptlogi.c validates files and directly inserts the transactions into the ReSA tables. This includes (as necessary) creating errors for the auditors to research and correct.
- saimptlog.c 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 ReSA 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 ReSA batch schedule which must be completed before POS transactions can be loaded. For more information about supporting batch jobs, see *Oracle Retail Merchandising System Operations Guide, Volume 2 - Batch Master Designs*.

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

- Retail Reference Architecture available on My Oracle Support
- *Oracle Retail Merchandising System Operations Guide, Volume 2 - Batch Master Designs*

Configuration

This chapter provides information on the configuration changes that can be made for the integration with RMS. There are no configuration options available for integration with RPM.

RMS

For information on configuration for the integration with RMS for the following type of data, see the following:

- [Diffs](#)
- [Item](#)
- [Merchandise Hierarchy](#)
- [Organizational Hierarchy](#)
- [Related Item](#)
- [Store](#)
- [VAT](#)

Configuration is not available for all the data. If there is no available configuration, that is called out in the following sections.

The Xcenter and Xstore config.jar files contain a Spring bean configuration file used by the DataLoader. Configuration modifications are made in the following file:

```
dataloader/spring/dataloader_beans.xml
```

Diffs

Diffs (also known as Diff IDs) information provided by RMS is consumed by Xstore using the DataLoader. Diff is short for Differentiator and is similar in concept to Xstore's Dimension concept. Diffs data is the primary source of rms_diff_ids records representing an identifier record for a Diff.

RMS Diff information is communicated to Xstore in three separate extract files: Diff Group Header, Diff Group Detail, and Diffs. The rms_diff_group_head table is a staging table supporting the loading of ItemHeader records. The Diff staging tables are accessed by the DataLoader to generate Dimension data when an ItemHeader record is a Style or Style Item. Xstore and Xcenter do not directly access the rms_diff_group_head table.

Diff Group Header data is the primary source of `rms_diff_group_head` records representing a header record for a Diff Group. Diff Group Detail data is the primary source of `rms_diff_group_detail` records representing a detail record for a Diff Group.

RMS Diffs

The DataLoader uses `dtv.dataloader.mom.DiffTransformer` to process RMS Diffs records. This bean has one configuration option.

The prefix used for DIFF Groups generated by the DataLoader as a Default Diff Group for each Diff Type. The prefix must be four characters or less. If not provided, `DEF_` is used as the prefix value.

```
<bean id="diffTransformer" class="dtv.dataloader.mom.DiffTransformer" >
  <property name="defaultDiffGroupIdPrefix" value="****" />
</bean>
```

RMS Diff Group Header Records

The DataLoader uses `dtv.dataloader.mom.DiffGroupHeadTransformer` to process RMS Diff Group Header records. There are no configuration options for the `diffGroupHeadTransformer` bean.

```
<bean id="diffGroupHeadTransformer"
class="dtv.dataloader.mom.DiffGroupHeadTransformer" />
```

RMS Diff Group Detail Records

The DataLoader uses `dtv.dataloader.mom.DiffGroupDetailTransformer` to process RMS Diff Group Detail records. There are no configuration options for the `diffGroupDetailTransformer` bean.

```
<bean id="diffGroupDetailTransformer"
class="dtv.dataloader.mom.DiffGroupDetailTransformer" />
```

Item

Configuration is available for item header and item location.

Item Header

Item Header information provided by RMS is consumed by Xstore using the DataLoader. Item Header data is the primary source of `itm_item` records representing physical items. Records representing non-physical items are ignored.

The DataLoader uses `dtv.dataloader.mom.ItemHeadTransformer` to process RMS ItemHeader records. This bean has one configuration option.

The prefix used for DIFF Groups generated by the DataLoader as a Default Diff Group for each Diff Type. This value must match the property configuration used by the `diffTransformer` bean. The prefix must be four characters or less. If not provided, `DEF_` is used as the prefix value.

```
<bean id="itemHeadTransformer" class="dtv.dataloader.mom.ItemHeadTransformer" >
  <property name="defaultDiffGroupIdPrefix" value="****" />
</bean>
```

Item Location

Item Location information provided by RMS is consumed by Xstore using the DataLoader. Item Location data is the primary source of `itm_item_option` records representing store-specific attributes of physical items. Records representing non-physical items are ignored.

The DataLoader uses `dtv.dataloader.mom.ItemLocTransformer` to process RMS ItemLoc records. This bean has two configuration options:

- When an RMS ItemLoc record's `TaxableInd='N'`, the `TaxGroupId` of the Xstore item `ItemOptions` record is populated with the value configured for `nonTaxableTaxGroupId`.
- When `translateItemDescriptionEnabled` is true, a `com_translations` record is populated using the RMS ItemLoc records' Local Item Description.

Note: For anyone configuring Xcenter, this property should be configured with "true" when loading ItemLoc data into the Xstore database; the property should be configured with "false" when loading ItemLoc data into the Xcenter database.

```
<bean id="itemLocTransformer" class="dtv.dataloader.mom.ItemLocTransformer" >
  <property name="nonTaxableTaxGroupId" value="0" />
  <property name="translateItemDescriptionEnabled" value="true"/>
</bean>
```

Merchandise Hierarchy

RMS Merchandise Hierarchy information provided by RMS is consumed by Xstore using the DataLoader. RMS supports the following merchandise hierarchy levels:

COMPANY, DIVISION, GROUP, DEPARTMENT, CLASS, SUBCLASS

The DataLoader uses `dtv.dataloader.mom.MerchHierarchyTransformer` to process RMS Merchandise Hierarchy records. There are no configuration options for the `merchHierarchyTransformer` bean.

```
<bean id="merchHierarchyTransformer"
class="dtv.dataloader.mom.MerchHierarchyTransformer" />
```

Organizational Hierarchy

Organizational Hierarchy information provided by RMS is consumed by Xstore and Xcenter using the DataLoader.

The DataLoader uses `dtv.dataloader.mom.OrgHierarchyTransformer` to process RMS store records. The `orgHierarchyTransformer` bean has no configuration options.

```
<bean id="orgHierarchyTransformer"
class="dtv.dataloader.mom.OrgHierarchyTransformer" />
```

Related Item

RMS related items are communicated to Xstore in two separate extract files: Related Item Header and Related Item Detail. The `rms_related_item_head` table is a staging table supporting the processing of `RelatedItemDetail` records by the DataLoader. Xstore and Xcenter do not directly access the `rms_related_item_head` table.

Related Item Header

Related Item Header information provided by RMS is consumed by Xstore using the DataLoader. Related Item Header data is the primary source of `rms_related_item_head` records representing a header related Item record. RMS supports three types of related item relationships, Cross-Sell, Up-Sell, and Substitution. RMS Cross-Sell/Up-Sell

records are interpreted as Xstore Attached Items. RMS Substitution Items are interpreted as Xstore substitution items.

The DataLoader uses `dtv.dataloader.mom.RelatedItemHeadTransformer` to process RMS Related Item Header records. There is one configuration option for the `relitemHeadTransformer` bean. For the case where `itm_attached_items` records are created, which occurs when a related item header update event results in the change of relationship type from SUBS to CRSL/UPSL, the value assigned the `promptToAddMessageKey` column will be the configured value.

```
<bean id="relitemHeadTransformer"
class="dtv.dataloader.mom.RelatedItemHeadTransformer" >
  <property name="promptToAddMessageKey" value="_commonAttachedItemsPrompt" />
</bean>
```

Related Item Detail

Related Item Detail information provided by RMS is consumed by Xstore using the DataLoader. Related Item Detail data is the primary source of `itm_attached_items` and `itm_substitute_items` records representing a detail related Item record. RMS supports three types of related item relationships, Cross-Sell, Up-Sell, and Substitution. RMS Cross-Sell/Up-Sell records are interpreted as Xstore Attached Items. RMS Substitution Items are interpreted as Xstore substitution items.

The DataLoader uses `dtv.dataloader.mom.RelatedItemDetailTransformer` to process RMS Related Item Detail records. There is one configuration options for the `relitemDetailTransformer` bean. The value configured for the `promptToAddMessageKey` property is used to populate the `prompt_to_add_msg_key` column of the `itm_attached_items` table for Cross-Sell and Up-Sell records.

```
<bean id="relitemDetailTransformer"
class="dtv.dataloader.mom.RelatedItemDetailTransformer" >
  <property name="promptToAddMessageKey" value="_commonAttachedItemsPrompt" />
</bean>
```

Store

Store information provided by RMS is consumed by Xstore using the DataLoader.

The DataLoader uses `dtv.dataloader.mom.StoreTransformer` to process RMS store records. The following configuration options are available:

- Configure the `use_till_accountability_flag` value assigned to all records created in `loc_rtl_loc`.
- Configure the `location_type` value assigned to all record created in `loc_rtl_loc`. This property is optional. If not set, `location_type` will be null.

```
<bean id="storeTransformer" class="dtv.dataloader.mom.StoreTransformer">
  <property name="useTillAccountability" value="false" />
  <property name="locationType" value="STORE" />
</bean>
```

Store Address

Store Address information provided by RMS is consumed by Xstore using the DataLoader. RMS Store Address records are interpreted as an instructions to update the Address information of existing Store records. Each location in RMS can have many types of addresses. There is a configuration to specify which RMS `AddrType` is to be recognized as a store address. In addition, RMS supports multiple addresses for the same `addrType`. However, only the primary address is used to populate the address fields of an Xstore store record.

The DataLoader uses `dtv.dataloader.mom.StoreAddressTransformer` to process RMS store records. One configuration option is available.

Configure the `addrType` value. Only records having this Address Type value, and where the record's `PrimaryAddrInd` is true, are used as the source for a Store's Address information.

```
<bean id="storeAddressTransformer"
class="dtv.dataloader.mom.StoreAddressTransformer">
  <property name="addrType" value="01" />
</bean>
```

VAT

VAT information provided by RMS is consumed by Xstore and Xcenter using the DataLoader. RMS VAT information is used to populate the Xstore tax tables for when using VAT. RMS provides a VAT Item file containing the information used to populate an item's `TaxGroupId`.

The DataLoader uses `dtv.dataloader.mom.VATTransformer` to process RMS store records. The `VATTransformer` bean has three configuration options:

- A Boolean flag indicating if tax is calculated at the transaction level.
- A rounding code, such as `HALF_UP`, `HALF_DOWN`, and so on, set into `TaxAuthorityDAO`.
- The number of rounding digits set into `TaxAuthorityDAO`.

```
<bean id="VATTransformer" class="dtv.dataloader.mom.VATTransformer">
  <constructor-arg type="boolean" value="true" />
  <constructor-arg type="java.lang.String" value="HALF_UP" />
  <constructor-arg type="int" value="2" />
</bean>
```

VAT Item

VAT Item information provided by RMS is consumed by Xstore and Xcenter using the DataLoader. RMS VAT item information is used to populate an item's `TaxGroupId` with a VAT Code.

The DataLoader uses `dtv.dataloader.mom.VATItemTransformer` to process RMS store records. The `VATItemTransformer` bean does not have any configuration options.

```
<bean id="VATItemTransformer" class="dtv.dataloader.mom.VATItemTransformer" />
```

Integration Considerations

This chapter provides the following considerations that should be taken into account when implementing the integration:

- [Functional Considerations](#)
- [Technical Considerations](#)

For information on functionality gaps, see [Chapter 7](#).

Functional Considerations

This section contains functional information that should be considered for the implementation.

Sales Tax

RMS is not the system of record for sales tax.

RMS does not provide US Sales tax information to Xstore. This includes Xstore product tax groups. This integration assumes 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 RMS 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 RMS 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.

Inventory Functionality

Inventory functionality in Xstore should be disabled. This integration assumes that inventory functionality in Xstore is not implemented.

Seed Data

The integration requires that a number of very basic seed data values are the same in both Merchandising and Xstore. These data values are set to common values in both Merchandising and Xstore installation scripts. If changes are made in one system after installation, changes also need to be made in the other system. Synchronized seed data includes the following:

- Currency codes
- Country codes
- Units of measure

Currency Exchange

This integration assumes that Xstore and RMS are fed consistent currency exchange rates from the same source system.

Xstore tender exchange transactions (that is, where a customer is given USD in exchange for CAD) are mapped to the transaction type OTHER in ReSA. These transactions are to have a zero net financial impact, and are not reported to the General Ledger (GL) by ReSA and RMS.

Physical Items

Physical merchandise items should be mastered in RMS. Xstore Office UIs should not be used to create physical items.

Non-Merchandise Items

Non-merchandise items, including warranties, fees, services, and so on, should be fully mastered in Xstore Office:

- Non-merchandise items require special attributes for Xstore processing that do not exist in RMS. These attributes do exist in Xstore Office.
- Shell items with the same IDs should be manually created in RMS so that the items are valid on an ReSA upload.
- Note that changes made to these items in RMS are not consumed by Xcenter/Xstore. RMS exports all items, including non-merchandise items. The Xcenter/Xstore DataLoader will not load non-merchandise items.
- Any maintenance of non-merchandise items should occur using Xstore Office.

Latitude and Longitude in Lookup

The Xstore inventory lookup functionality, which helps store associates find inventory at nearby stores, requires that stores have latitude and longitude attributes. RMS does not have latitude and longitude. The record type, RETAIL_LOCATION_COORDINATES, is available to DataLoader to populate the latitude and longitude of stores using the .mnt format.

Batch Jobs

If retailers are using this integration, the following batch jobs do not support the integration and should not be scheduled to run:

- RMS batch jobs batch_orpos_extract.ksh, posdnld.pc, posrefresh.pc, poscdnld.pc, and posgpdnld.pc
- RPM jobs RPMtoORPOSPublishBatch.sh and RPMtoORPOSPublishExport.sh

For information on the batch jobs for the integration, see the *Oracle Retail Merchandising System Operations Guide, Volume 2 - Batch Master Designs*.

Populate Data at Implementation and Upgrade

At the time of implementation and upgrade, retailers should use the kill/fill export option in the RMS extracts to populate Xstore with RMS data. The upgrade process does not automatically republish all RMS data.

Also, retailers should run the staging batch for every store and then extract the data using the standard batch processes. RPM does not flag the files that would contain the refresh data as anything different than the normal delta data. The upgrade process does not automatically republish all RPM data.

In-Store Orders

The Xstore RTLog generator sets the Fulfillment order number in the RTLog to NULL. A fulfillment order number is not required because the order is completed within the store and does not require any participation from an order management system.

Mapping of Transaction Details

The mapping of transaction details from Xstore POSlog to ReSA RTLog depends on the mappings of valid values. These mappings are detailed in [Appendix A](#). It is critical that the mappings are complete. If additional valid values are configured for Xstore in RTLogMappingConfig.xml, they must also be configured for ReSA for the appropriate code types.

Recognition of a Sale

Xstore can be configured to recognize a sale at either the time of sale or time of pickup. Integration with Merchandising requires that this configuration be time of pickup, as that is the time that inventory decrements. (If set to time of sale, inventory would decrement early, causing downstream inventory correctness issues.)

The following setting needs to be set to false (which is the default) under both <Layaway> and <SpecialOrder> in SystemConfig.xml (whose settings are also controllable in Xadmin):

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

VAT

This integration allows RMS to be configured for VAT enabled (SVAT), but allows specific zones to be configured as VAT Exempt. This enables a single instance of RMS to support both VAT locations and non-VAT locations. The same Xcenter instance can support both VAT and non-VAT stores. For more information on configuration for VAT, see the *Oracle Retail Xstore Technical Guide*.

Technical Considerations

This section contains technical information that should be considered for the implementation.

RMS

To better integrate with downstream systems, including Xstore, RMS has a unique key to the CLASS and SUBCLASS tables. This means it is possible to identify a class or subclass with a single key value (instead of a composite key including dept/class or dept/class/subclass). The unique key is generated and not displayed to RMS end users.

RPM

RPM has an option to not allow users to use any complex promotions which includes all Multi-Buy promotions, all Threshold promotions, and all Transaction promotions and Finance promotions. This system option can be used to prevent end users from creating complex promotions which are not supported by Xstore including: Multi-Buy promotions with a reward of cheapest free, Multi-Buy promotions that use the OR qualifier between multiple buy and/or reward lists, Threshold promotions created with a qualification of "item level," and Finance promotions.

Note: Setting this system option to not allow creation of complex promotions limits the other types of Xstore supported promotions that the RPM user can create.

RPM_SYSTEM_OPTIONS.COMPLEX_PROMO_ALLOWED_IND

Xstore

For the Xstore Suite, consider the following:

- The ReSA/RMS and Xstore integration assumes that store inventory is being managed in SIM (or another similar store inventory management application). SIM provides updates to RMS on shipments and receipts and other inventory movements in the store, while Xstore provides this system with updates on sales and returns. 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 ReSA/RMS and Xstore is not supported.
- To allow end users to create non-merchandise items, but prevent users from creating or editing merchandise items, 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.
- By default, the following settings in the Xstore SystemConfig.xml file (whose settings are also controllable by the user in Xstore Office) are false. The settings should not be changed. If they are changed, sales and inventory information may post incorrectly to ReSA.

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

- Following is the Xstore Merchandise Hierarchy configuration:

```
<MerchHierarchy dtype="Default">  
  <NumberOfLevels dtype="Integer">4</NumberOfLevels>  
  <Level1Code dtype="String">GROUP</Level1Code>  
  <Level2Code dtype="String">DEPARTMENT</Level2Code>  
  <Level3Code dtype="String">CLASS</Level3Code>  
  <Level4Code dtype="String">SUBCLASS</Level4Code>  
</MerchHierarchy>
```

- RMS allows up to 10-digit store and warehouse IDs. By default, Xstore is configured to allow four digit store IDs. It is recommended that four or five digit location IDs are used in RMS. The size of Xstore store IDs can be configured in SequenceConfig.xml, but Xstore is only tested with four and five digit store IDs.

RTLog Generator

This chapter describes how to install, deploy, and configure the RTLog Generator application.

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.

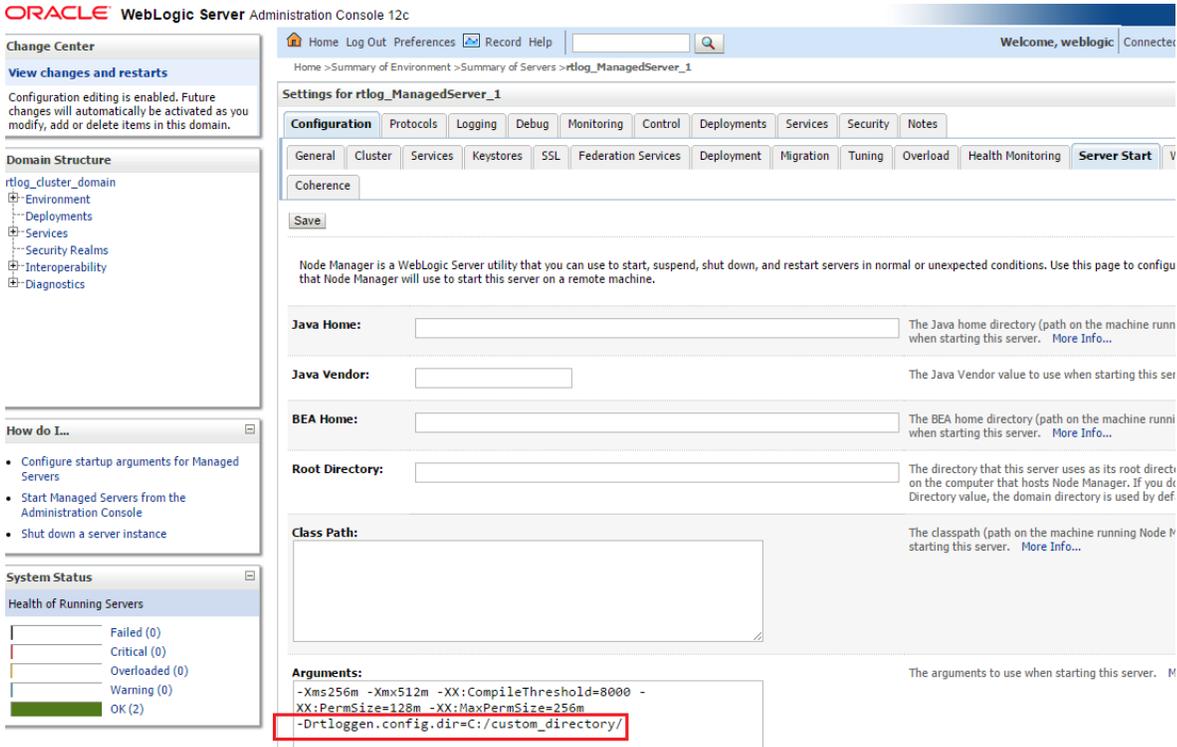
Note: Bounce the WebLogic server after making any configuration level changes.

To set up the external configuration feature:

1. Extract the configuration file's content into the C:\<rtlog-gen-config> directory if installing on Microsoft Windows or /usr/local/<rtlog-gen-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 ReSA 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 ReSA. 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="P01" RecordValue="P01" />
    <VALUE_MAPPING sourceValue="P02" RecordValue="P02" />
    <VALUE_MAPPING sourceValue="P03" RecordValue="P03" />
    <VALUE_MAPPING sourceValue="P04" RecordValue="P04" />
    <VALUE_MAPPING sourceValue="P05" RecordValue="P05" />
    <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>

```

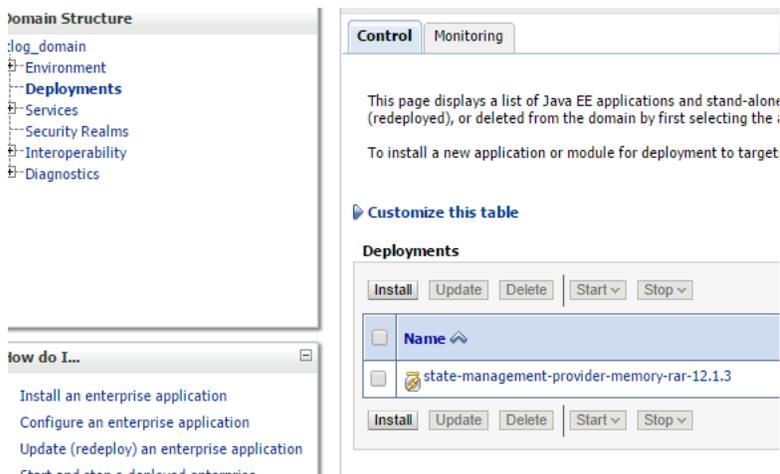
Deployment

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

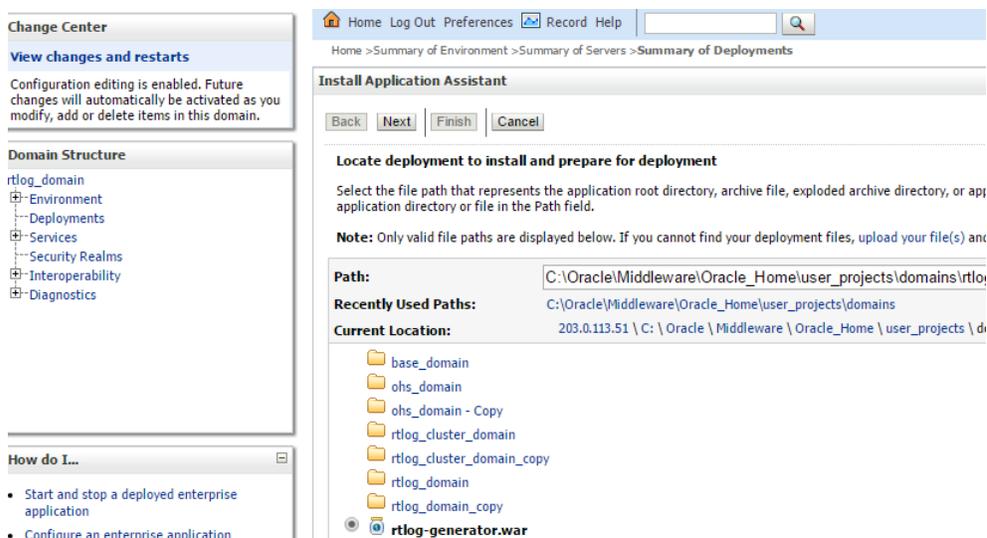
This section covers the deployment in both a clustered and non-clustered environment.

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

The screenshot shows the Oracle WebLogic Server Administration Console interface. The main content area is titled "Summary of Deployments" and includes a "Messages" section with two green checkmarks: "All changes have been activated. No restarts are necessary." and "The deployment has been successfully installed." Below this is a "Summary of Deployments" section with "Control" and "Monitoring" tabs. A text block explains that the page displays a list of Java EE applications and stand-alone application modules. A table titled "Deployments" is shown with columns for Name, State, Health, and Type. The table contains two entries: "rtlog-generator" (Web Application) and "state-management-provider-memory-rar-12.1.3" (Resource Adapter). Both are in an "Active" state with a "Health" of "OK".

Name	State	Health	Type
rtlog-generator	Active	OK	Web Application
state-management-provider-memory-rar-12.1.3	Active	OK	Resource Adapter

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

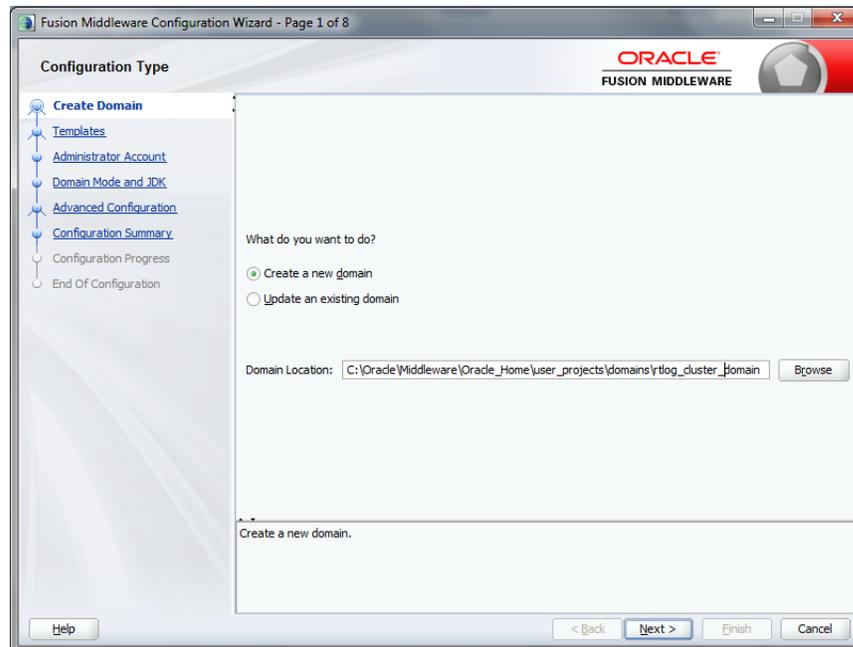
- To deploy on a cluster, see "[Deployment of the RTLog Generator Application on a Cluster](#)".
- To enable security for the RTLog Generator application, see "[Security Configuration](#)". When deploying in a non-clustered environment, continue at this section.

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.

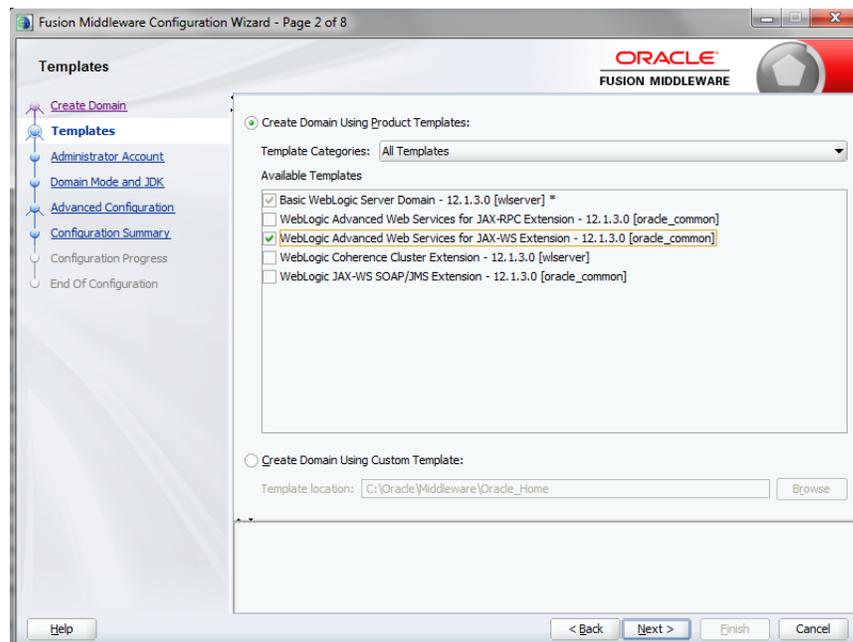
To set up the cluster to use RTLog Generator:

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

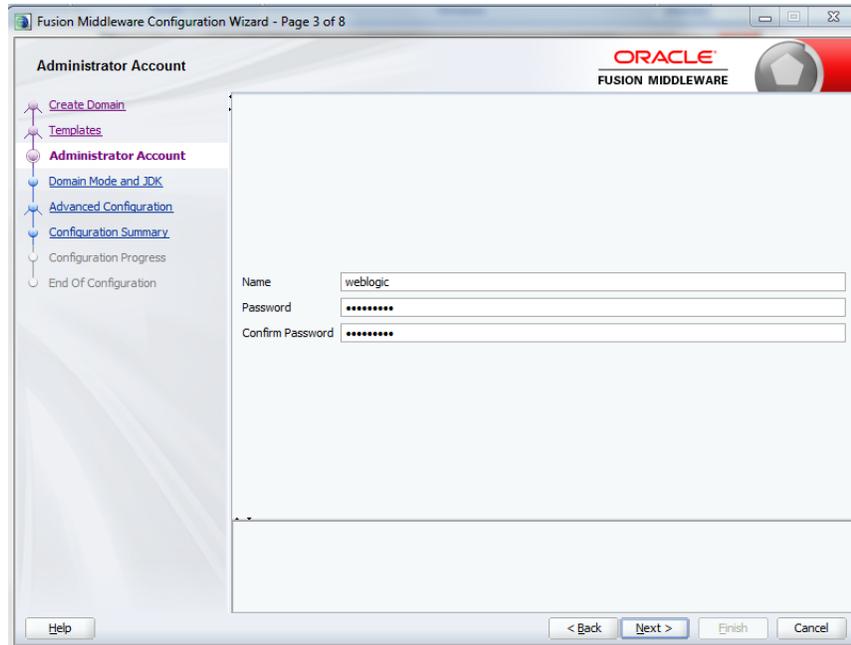
3. On the Templates page, select the supported products and click **Next**. It is recommended to select the following:

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

Figure 6–11 Configuration Wizard Templates Page

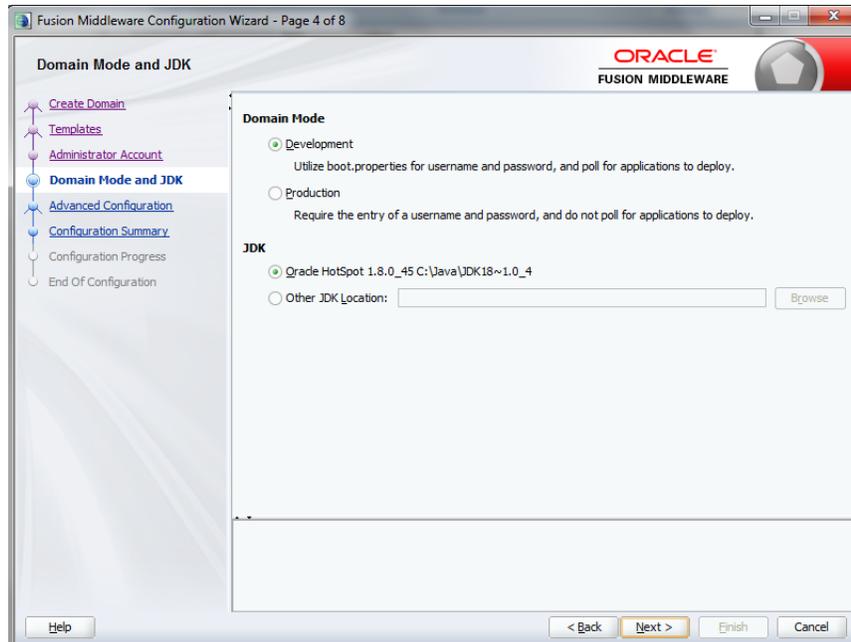
4. On the Administrator Account page, enter the Administrator user name and password. Enter the password a second time to confirm. Click **Next**.

Figure 6–12 Configuration Wizard Administrator Account Page

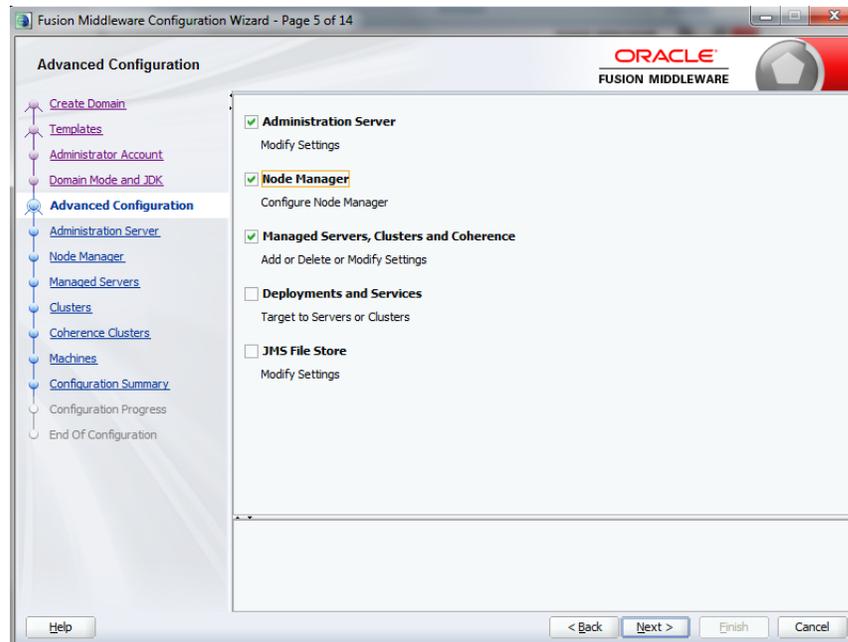


5. On the Domain Mode and JDK page, select either Development or Production mode. For production mode, you need to manually create the boot.properties file. Click **Next**.

Figure 6–13 Configuration Wizard Domain Mode and JDK

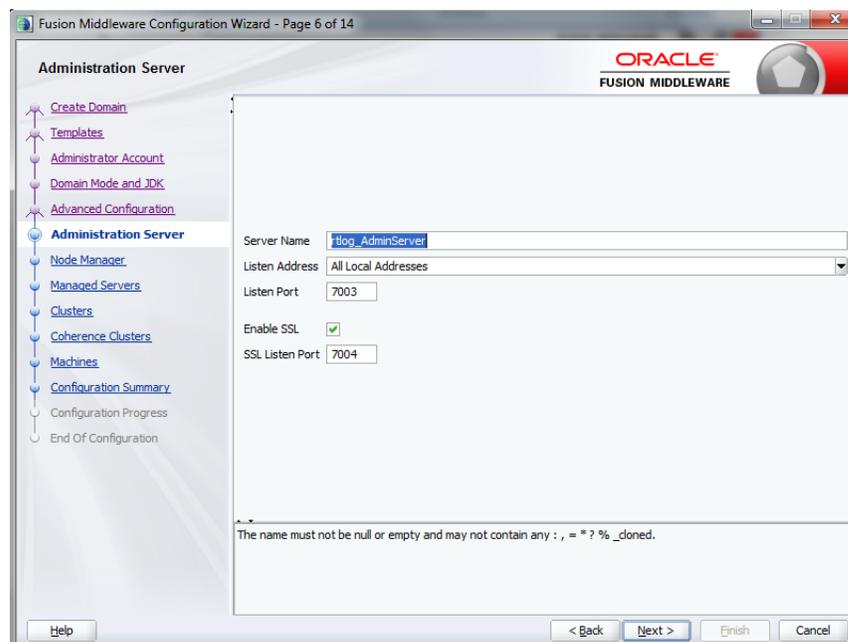


6. On the Advanced Configuration page, select the Administration Server, Node Manager, and Managed Servers, Clusters and Coherence options. Click **Next**.

Figure 6–14 Configuration Wizard Advanced Configuration Page

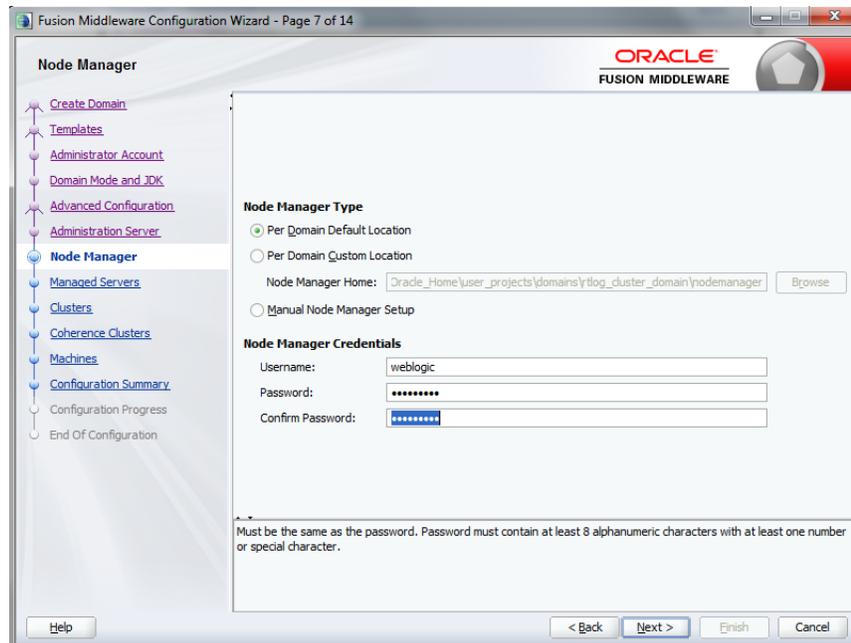
7. On the Administration Server page, enter the values to configure the administration server. The administrator server controls all the managed servers that are part of the cluster.

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**.

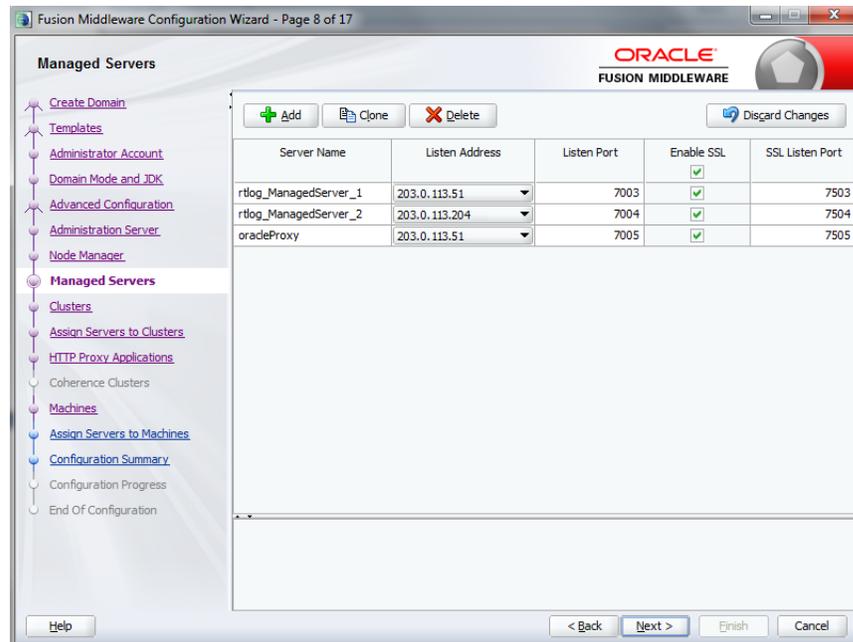
Figure 6–15 Configuration Wizard Administration Server Page

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**.

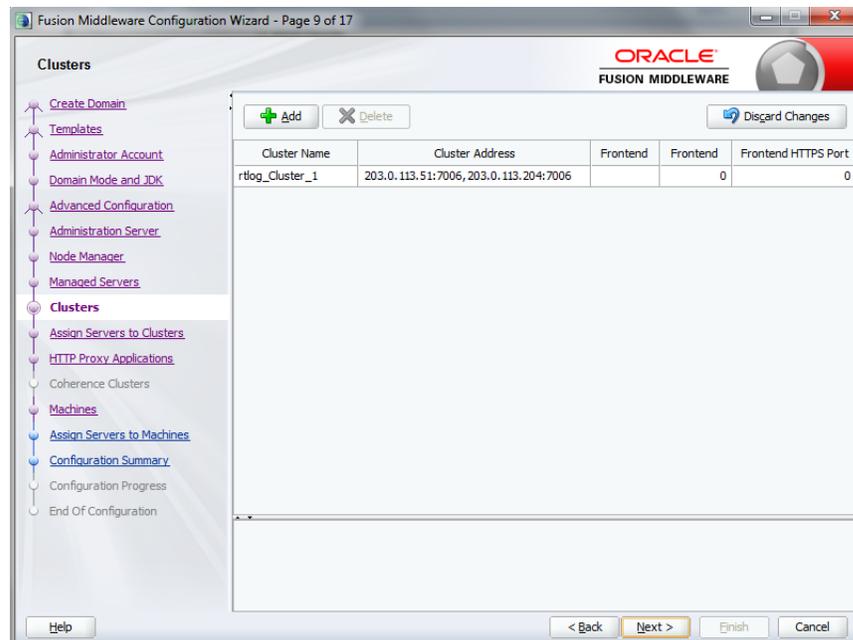
Figure 6–16 Configuration Wizard Node Manager Page



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

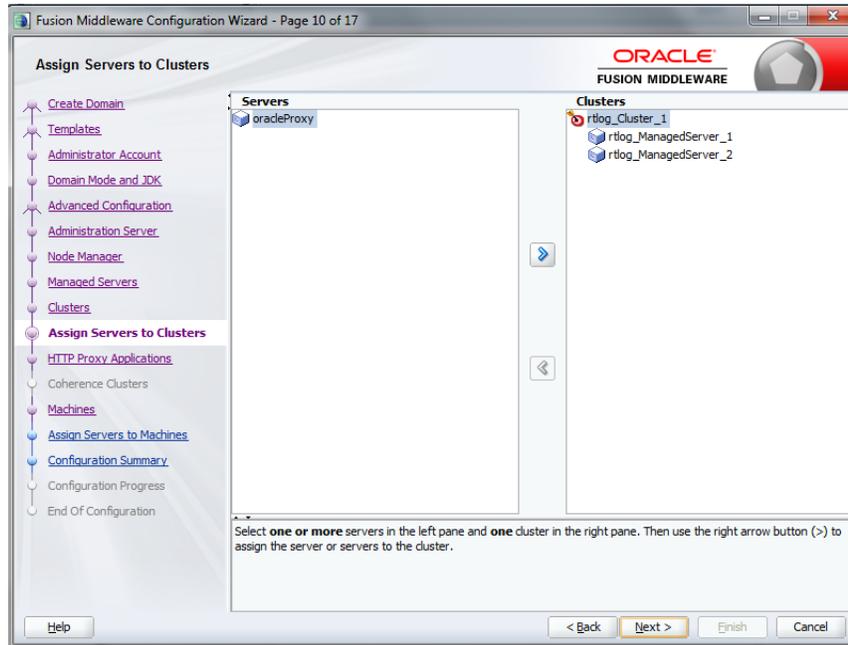
- On the Clusters page, add and configure the cluster. Enter the cluster name followed by the cluster address, that is, IP address1:port1, IP address2:port2, so on. Click **Next**.

Figure 6–18 Configuration Wizard Clusters Page

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

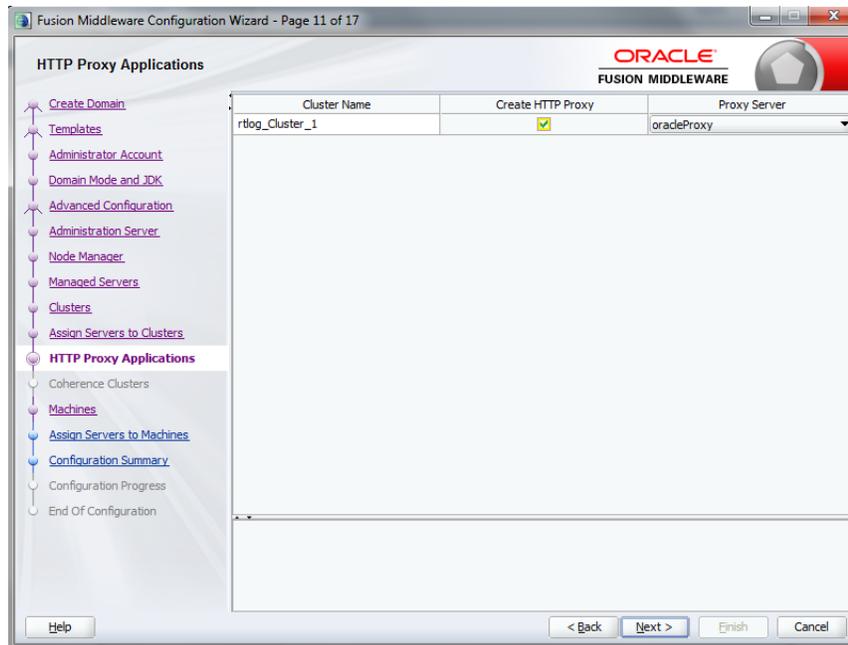
Note: Do not include the Oracle Proxy as part of the cluster.

Figure 6–19 Configuration Wizard Assign Servers to Clusters Page

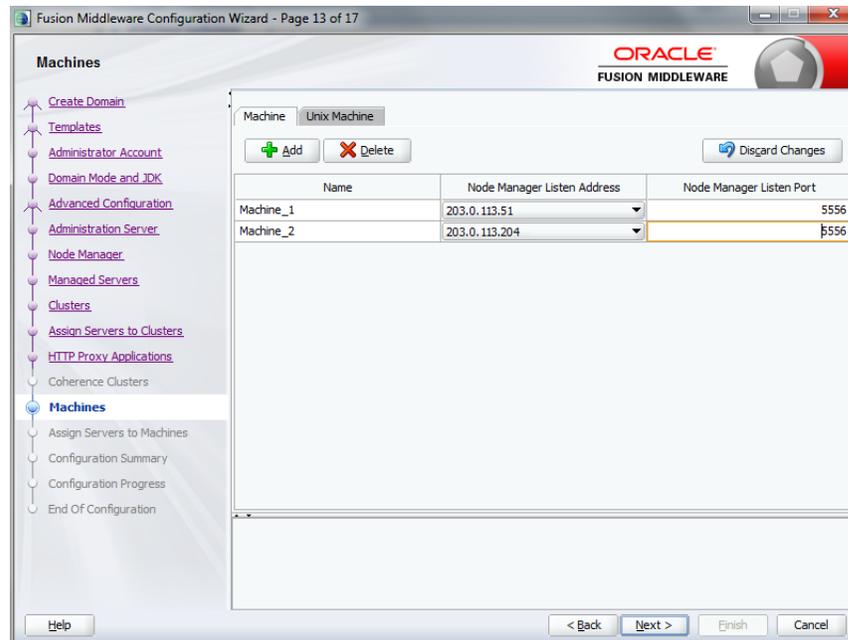


12. On the HTTP Proxy Applications page, select Create HTTP Proxy and then select the server from the drop-down list. By default, it should have already been selected. Click Next.

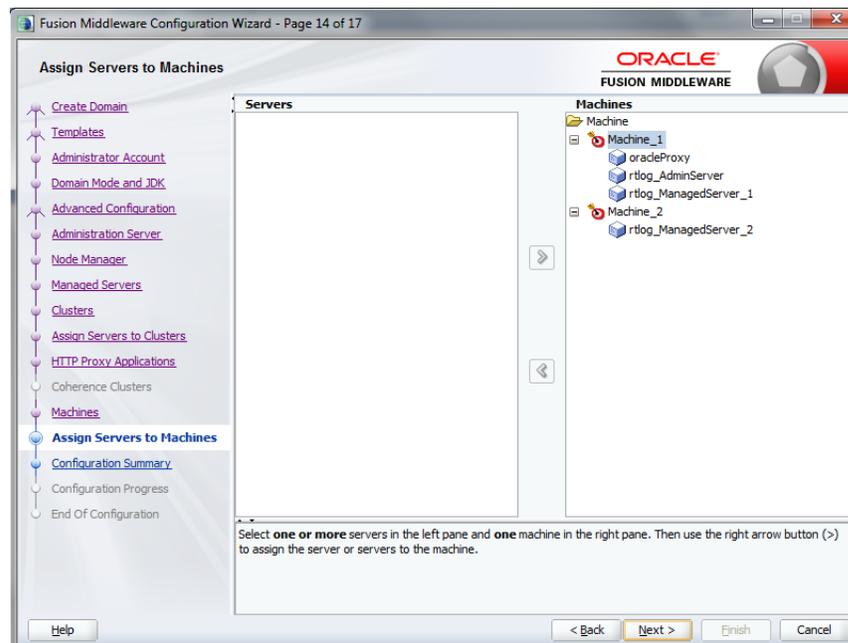
Figure 6–20 Configuration Wizard HTTP Proxy Applications Page



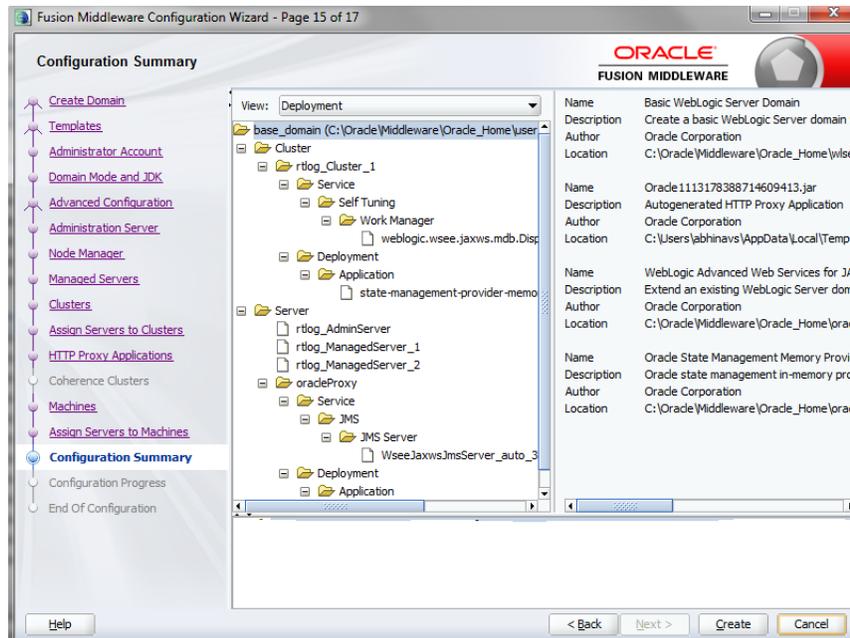
13. On the Machines page, add and configure each machine. To add Machine_1 and Machine_2, click **Add** and enter the respective IP addresses. This configuration is for setting up the Node managers on both the machines. Since these node managers are physically separated, you can select the same host.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

Figure 6–21 Configuration Wizard Machines Page

- 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

- 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>203.0.113.51</listen-address>
  </node-manager>
</machine>
<machine>
  <name>Machine_2</name>
  <node-manager>
    <name>Machine_2</name>
    <nm-type>Plain</nm-type>
    <listen-address>203.0.113.204</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')
```

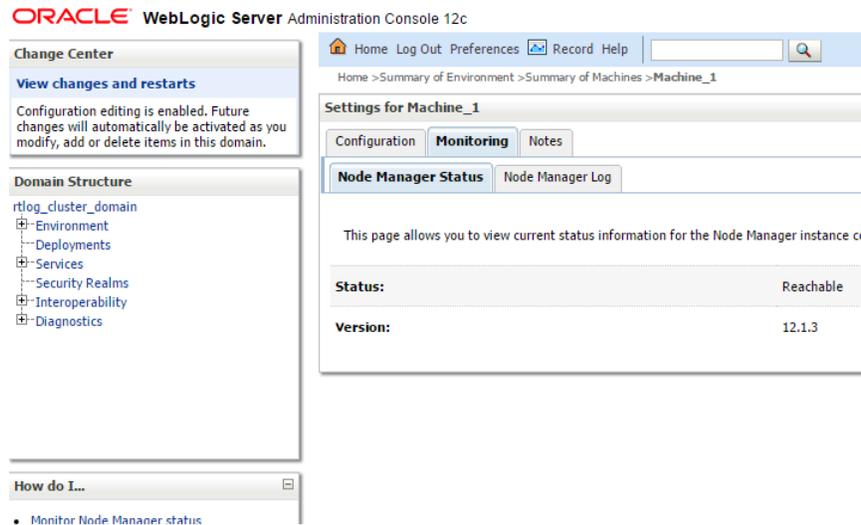
For example: `connect ('weblogic','weblogic1','t3://203.0.113.51:7003')`
 - d. Once the connect command shows the connection completed successfully, run the following command:


```
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](#)".

- 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

Servers (Filtered - More Columns Exist)							
Name	Type	Cluster	Machine	State	Health	Listen Port	
loadBalancerProxy	Configured		Machine_1	RUNNING	OK	7001	
rtlog_AdminServer(admin)	Configured		Machine_1	RUNNING	OK	7003	
rtlog_ManagedServer_1	Configured	rtlog_Cluster_1	Machine_1	RUNNING	OK	7005	
rtlog_ManagedServer_2	Configured	rtlog_Cluster_1	Machine_2	RUNNING	OK	7005	

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.

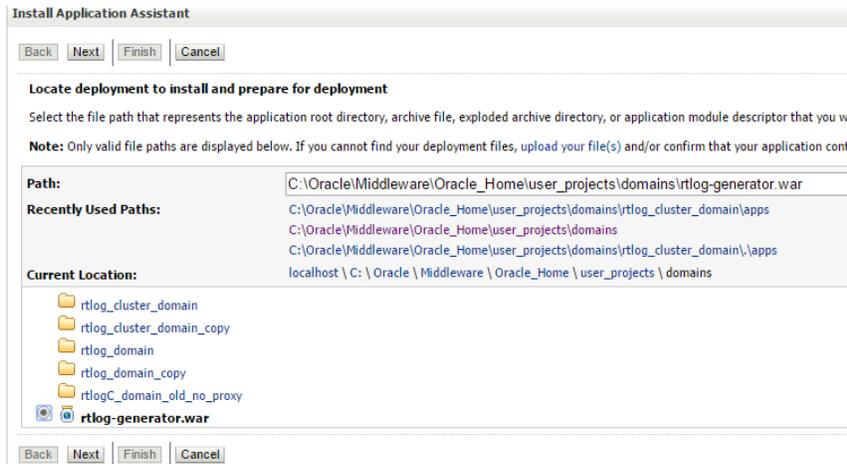
2. Navigate to the Administration Console home page and click Deployments in the left navigation menu. [Figure 6–27](#) shows an example of the page before deploying the RTLog Generator application.

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

Install Application Assistant

Back Next Finish Cancel

Select deployment targets

Select the servers and/or clusters to which you want to deploy this application. (You can reconfi

Available targets for rtlog-generator :

Servers

loadBalancerProxy

rtlog_AdminServer

Clusters

rtlog_Cluster_1

All servers in the cluster

Part of the cluster

rtlog_ManagedServer_1

rtlog_ManagedServer_2

Back Next Finish Cancel

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

Figure 6–30 Summary of Deployments Page

✔ All changes have been activated. No restarts are necessary.

✔ The deployment has been successfully installed.

Summary of Deployments

Control Monitoring

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, u application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install button.

[Customize this table](#)

Deployments

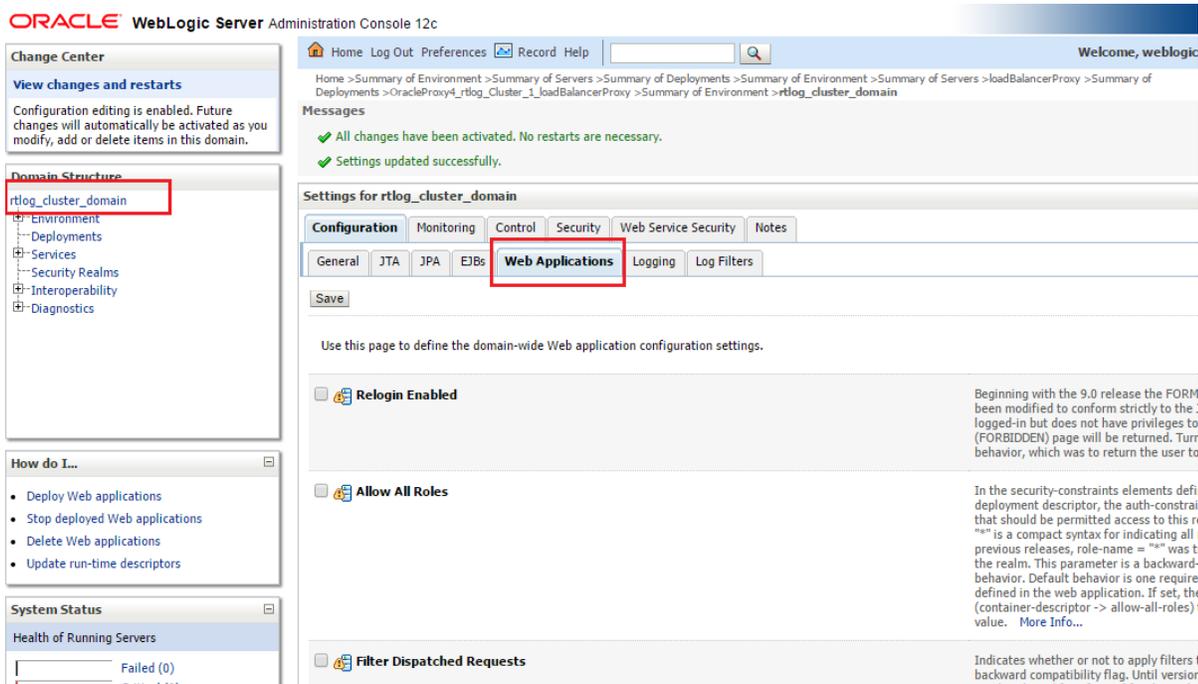
Install Update Delete Start Stop

Name	State	Health	Type
OracleProxy4_rtlog_Cluster_1_loadBalancerProxy	Active	✔ OK	Web Application
rtlog-generator	Active	✔ OK	Web Application
state-management-provider-memory-rar-12.1.3	Active	✔ OK	Resource Adapter

Install Update Delete Start Stop

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

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled "Summary of Security Realms". It contains a table with one row for a realm named "myrealm". The table has columns for "Name" and "Action". The "myrealm" entry is highlighted in blue, and a red rectangular box is drawn around it. Above the table, there are "New" and "Delete" buttons. Below the table, there are also "New" and "Delete" buttons. The left sidebar shows the "Domain Structure" with "Security Realms" selected. The top navigation bar includes "Home", "Log Out", "Preferences", "Record", and "Help".

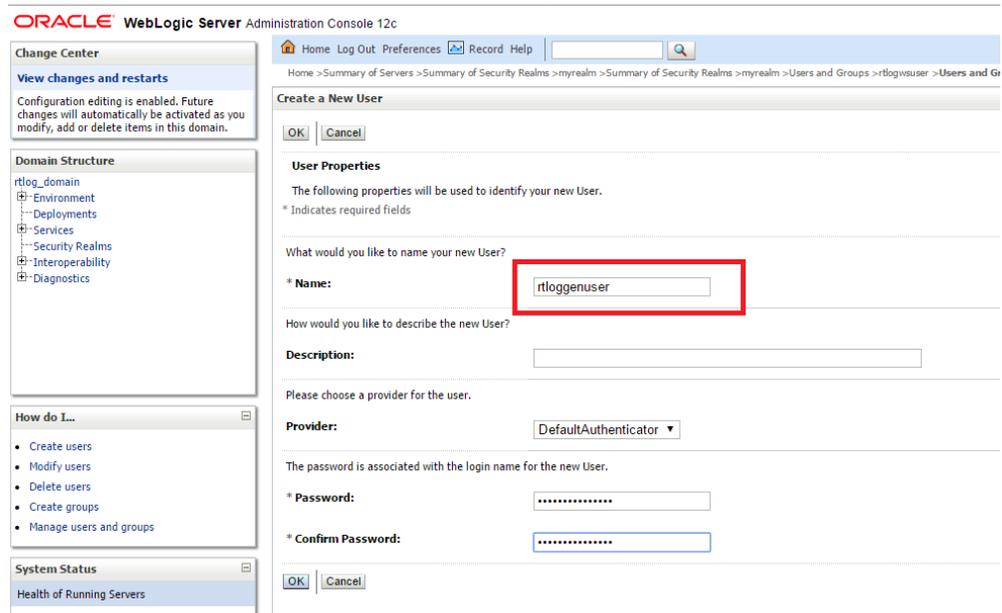
3. In the list of realms on the Summary of Security Realms page, select myrealm.
4. Select Users and Groups and then the Groups tab. To create a new group, click **New**. Enter a group name, for example RTLogUserGroup, and click **OK**.

Figure 6–34 Create a New Group Page

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled "Create a New Group". It contains a form with the following fields: "Name" (required), "Description", and "Provider". The "Name" field is filled with "RTLogUserGroup" and is highlighted with a red rectangular box. The "Description" field is empty. The "Provider" dropdown menu is set to "DefaultAuthenticator". The left sidebar shows the "Domain Structure" with "Security Realms" selected. The top navigation bar includes "Home", "Log Out", "Preferences", "Record", and "Help".

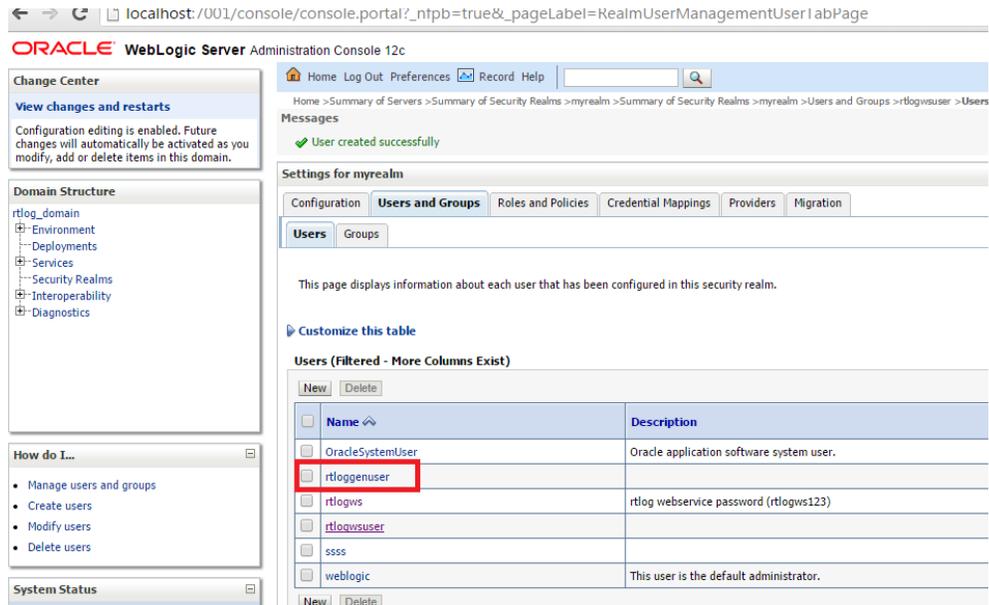
5. Select the Users tab and click **New**. Enter a user name and password and click **OK**.

Figure 6–35 Create a New User Page



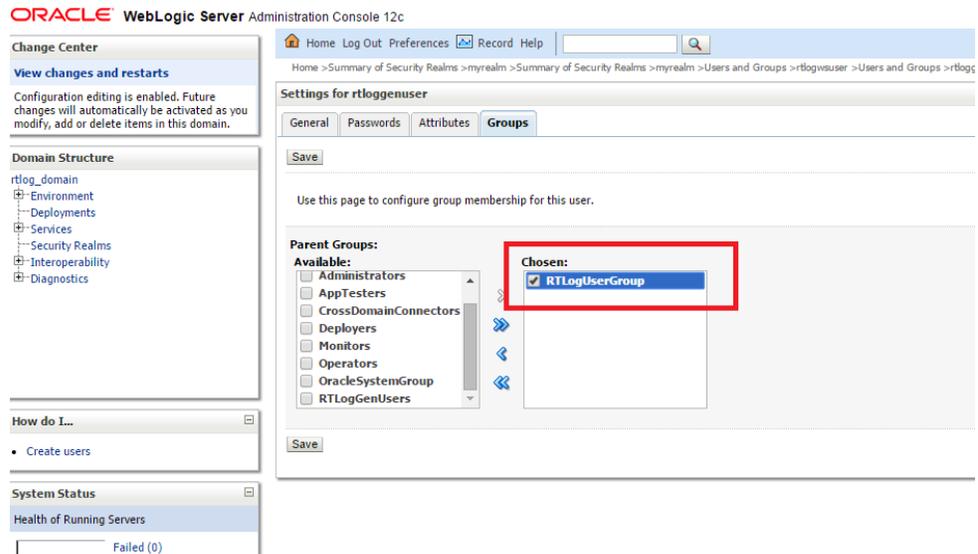
6. In the list of users, click the newly created user.

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

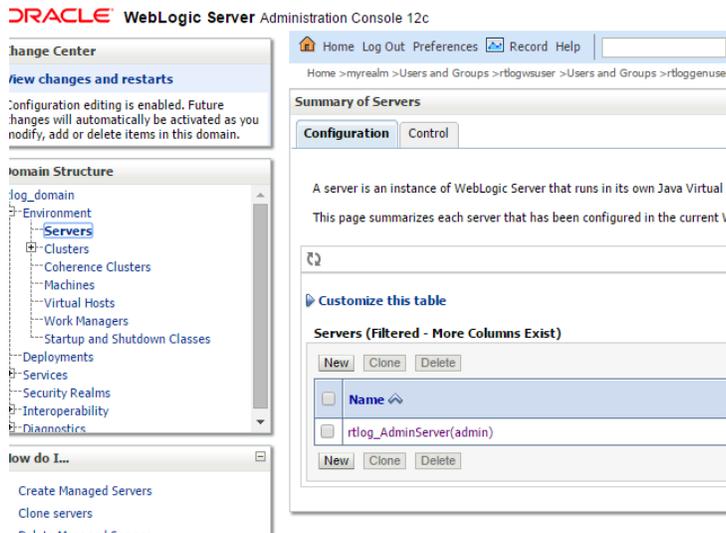
To configure transport level security:

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.

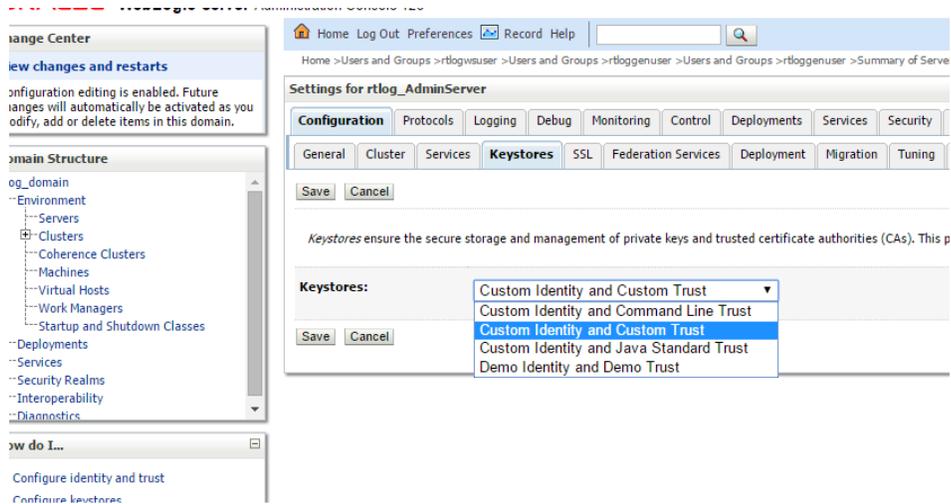
3. Log in to the WebLogic console. Click Environment and then the Servers link from the left navigations menu.

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

The screenshot displays the 'Settings for rtlog_AdminServer' page in the Oracle WebLogic Administration Console. The 'Keystores' tab is active. The 'Identity' section is expanded, showing the following configuration:

- Custom Identity Keystore:** C:\TASKS\IXstore_POC_JA\
- Custom Identity Keystore Type:** jks
- Custom Identity Keystore Passphrase:** [Masked]
- Confirm Custom Identity Keystore Passphrase:** [Masked]

The 'Trust' section is also expanded, showing the following configuration:

- Custom Trust Keystore:** C:\TASKS\IXstore_POC_JA\
- Custom Trust Keystore Type:** jks
- Custom Trust Keystore Passphrase:** [Masked]
- Confirm Custom Trust Keystore Passphrase:** [Masked]

A 'Save' button is located at the top left of the configuration area. The left sidebar shows the 'main Structure' tree with 'Environment' > 'Servers' > 'rtlog_AdminServer' selected. The 'System Status' section shows 'Health of Running Servers' with 'OK (1)'.

- 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.
- 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

The screenshot displays the 'Settings for AdminServer' page in the Oracle WebLogic Administration Console. The 'SSL' tab is active. The 'Identity and Trust Locations' section is expanded, showing the following configuration:

- Private Key Location:** from Custom Identity Keystore
- Private Key Alias:** [Empty field]
- Private Key Passphrases:** [Empty field]
- Confirm Private Key Passphrase:** [Empty field]
- Certificate Location:** from Custom Identity Keystore

The 'Trusted Certificate Authorities' section is also expanded, showing the following configuration:

- Trusted Certificate Authorities:** from Custom Trust Keystore

A 'Save' button is located at the bottom left of the configuration area. The left sidebar shows the 'Domain Structure' tree with 'base_domain' > 'Environment' > 'Servers' > 'AdminServer' selected. The 'System Status' section shows 'Health of Running Servers' with 'OK (1)'.

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 `endpointAddress` 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.

Existing Functionality Gaps

There are certain functionality gaps that exist in the integration that are not remedied at this time. This chapter describes these functional gaps.

RMS

Table 7-1 is a list of functionality gaps that exist for the RMS to Xstore integration.

Table 7-1 *RMS to Xstore Functionality Gaps*

Identified Functionality Gap
RMS does not provide enough information to sufficiently identify types of non-physical items. Non-physical items must still be set up in RMS; however the integration ignores them in the ItemHdr and ItemLoc files through the use of MerchandiseInd. Retailers are responsible for the population of all tables defining non-physical items.
RMS does not provide kit components for a kit header. Kits will appear as standard items in Xstore.
RMS supports up to four differentiators for each item. Xstore supports only three. RMS Diff4 is ignored. If the fourth differentiator is used in RMS, it will be exported from RMS, but Xstore DataLoader will not load the fourth value.
Xstore item records support four merchandise levels. The RMS Item Header records contain only three Merchandise Hierarchy levels: Department, Class, and Subclass. The Group Merchandise Hierarchy level is not included, and therefore cannot be mapped to an Xstore merchandise level.
Xstore can support up to four levels of merchandise hierarchy. RMS supports five levels. Levels above group in the RMS hierarchy will not be used in Xstore. For store operations, the lower four levels of the RMS merchandise hierarchy (group, department, class, subclass) are the most important, so these are the levels integrated into Xstore.
The data type and length of the RMS field MfgRecRetail is Number(20). When provided, the MfgRecRetail value is used to populate the itm_item table's list_price column, which can contain a max of Number(17,6).

Table 7-1 (Cont.) RMS to Xstore Functionality Gaps

Identified Functionality Gap
<p>It is possible in RMS to range a child item to a store without also ranging its parent to the store. RMS items can be associated with diffs without having a parent item (for example, a broom is red; red is an attribute of the item, but there is no parent for brooms and the same broom is not available in other colors). Xstore must receive from RMS the parent item's information in order to take advantage of Xstore Style features. Without a parent item available to create a Style in Xstore's database, Xstore will not display dimension values such as size, color, and width for an L1 item having Diff ID assignments.</p>
<p>In Xstore, a Style is an entry in the Item database that has properties such as size, color, width, and so. These properties are known as Dimensions in Xstore, and DIFFs in RMS. Multiple regular items may be associated with a Style ID. If you enter a Style ID in Xstore's Scan Item Prompt, the system prompts with additional item selection criteria by displaying dimension value choices. The Style ID, also known as the dimension system, is the RMS parent (or grandparent if one exists) of the transaction level item. If the RMS item used to represent a Style item in Xstore is not ranged to the store, items associated with the Style cannot be selected by scanning the Style ID. In addition, when a Style record does not exist in the database, the item dimension values are not displayed in the Xstore UI.</p>
<p>RMS items can be associated with DIFFs without having a parent item associated with one or more DIFF Groups. Without a parent item that can be set up in Xstore as an Item Style, these kinds of items will not be associated with a Dimension System.</p>
<p>RMS does not have a item restocking fee indicator.</p>
<p>The data type and length of the RMS field SellingUnitRetail is Number (20). When provided, the SellingUnitRetail value is used to populate the itm_item_prices table's price column, which can contain a maximum of Number (17,6).</p>
<p>In RMS, the data type of manufacture's recommended retail is NUMBER (20,4). In Xstore, the value is mapped to ITM_ITEM.LIST_PRICE, which is a NUMBER (17,6). If an RMS recommended retail value is over 17 digits, DataLoader into Xstore will fail for this record. Non-failing records from the same file will continue to be loaded.</p>
<p>RMS and SIM both support the concept that an item can be a serialized item in one store, but not in another. In Xstore, the SERIALIZED_ITEM_FLAG applies to an item in all locations. Serialized item is an item/location level flag in RMS, but an item level flag in Xstore. The Xstore DataLoader maps the item/location level information from RMS to the item level in Xstore. This means that if the serialized flag actually varies by location for an item in RMS, the last location to be processed by the integration code sets the item level serialized flag in Xstore.</p>
<p>RMS does not export Age and Quantity product restrictions.</p>
<p>RMS does not provide enough information in an ItemLoc record for the integration layer to identify above-transaction level items that do not need to be created as itm_item_options records. If above-transaction records are ranged to stores, which is likely, itm_item_options will be populated with records using Item IDs that do not exist in itm_item.</p>
<p>The ID displayed in the RMS UI for Class and Subclass is not the same ID as the numeric portion of the merchandise hierarchy displayed in the Xstore UI. RMS displays a display ID. Xstore's UI displays the RMS-supplied unique Class ID and Subclass ID (fields that are not displayed in RMS UI). The inconsistent display of identifiers in the two systems may cause some confusion.</p>
<p>RMS supports cross-sell and up-sell related item types. Xstore does not. RMS CRSL/UPSL records are mapped to Xstore's AttachedItems.</p>
<p>RMS supports creation of CRSL/UPSL items with above-transaction level parents; however this relationship is not useful in Xstore. In Xstore, both members of the relationship must be transaction level items.</p>
<p>Xstore's attached items can be auto-attached or prompt-to-attach. RMS does not have the same auto-attach concept, so attached items are always prompt-to-attach in Xstore.</p>

Table 7–1 (Cont.) RMS to Xstore Functionality Gaps

Identified Functionality Gap
RMS sends the VAT code for an item along with the active date for the VAT code to be applicable. Retailers use the active date for future planning. However, Xstore currently does not support an active date for VAT code: <ul style="list-style-type: none"> ■ Xstore will ignore the active date for VAT code. ■ Whenever the retailer sends the VAT code, the changes will be effective immediately. It is recommended that retailers send the VAT code change only when they need it.
Restocking fee is not maintained by RMS. Xstore Point of Service uses this to prompt for a restocking fee during returns.
Buying from a VAT store and returning to a non-VAT store (and vice versa) is not supported.
RMS product restrictions are not integrated to Xstore.
Packs are sent from RMS to Xstore as they are items. But pack details (the items that make up a pack) are not integrated. In Xstore, RMS packs are kits. Xstore does not have visibility to drill into kit components.
Xstore supports defining partial refunds for certain products based on item attribution. RMS does not export any attributes to support this Xstore functionality.
RMS does not manage coupons. Coupons should instead be managed in Oracle Retail Customer Engagement or another system that directly integrates with Xstore.
RMS UDAs and CFAS are not integrated to Xstore.
Xstore supports translation of some data; however, RMS only exports data in the primary integration language to stores. The exception to this is item description. RMS can hold a local item description at the item/location level. Retailers can use this to hold item descriptions in the store's local language or dialect.

RPM

Table 7–2 is a list of functionality gaps that exist for the RPM to Xstore integration.

Table 7–2 RPM to Xstore Functionality Gaps

Identified Functionality Gap
In RPM, the data type of selling unit retail is NUMBER (20,4). In Xstore, the value is mapped to ITM_ITEM_PRICE.PRICE, which is a NUMBER (17,6). If an RPM retail value is over 17 digits, DataLoader into Xstore will fail. Non-failing records from the same file will continue to be loaded.
RPM supports multi-buy promotions with a reward of cheapest free. Xstore does not support a cheapest free (Buy N and get Cheapest Free) promotion.
RPM and Xstore have different approaches to multi-unit pricing. RPM regular price update information for multiple units is not supported in this integration. Xstore converts multi-unit prices to single price, but cannot accurately do this without rounding information which varies from stores to stores.
RPM enables the execution of Finance Promotions. Xstore Point of Service does not support Finance Promotion components.
RPM integration includes customer segment information. Xstore does not support the integration.
RPM supports the integration of Multi-Buy promotions that use the OR condition between buy lists and/or reward lists. Xstore does not support the integration.
RPM supports the integration of threshold promotions with a qualifier of item level. Xstore does not support the integration.

Xstore to ReSA Integration

[Table 7-3](#) is a list of functionality gaps that exist for the Xstore to ReSA integration.

Table 7-3 Xstore to ReSA Integration Functionality Gaps

Identified Functionality Gap
<p>Till ID is not available for retail transactions. Xstore workstation and ReSA register are equivalent concepts. ReSA does not have an entity equivalent to the Xstore till.</p> <p>When integrated with Xstore, ReSA should be configured with a balancing level of Register. Xstore always sends the workstation ID as the register. The separate concept of a till is not supported in the integration, so Xstore cannot be configured for till-level balancing when integrated with ReSA.</p>
<p>Bounce back coupon number length in Xstore can be 60 characters long, but ReSA only allows 16 characters. If retailers want to use the integration, they should as a business process, not use IDs with more than 16 characters.</p>
<p>In Xstore, the Sales person field length can be 60 characters in length, but ReSA only allows 10 characters. Retailers should, as a business process, not use Xstore sales person IDs with more than 10 characters.</p>
<p>Commerce Anywhere is not supported. For Commerce Anywhere orders, the Xstore RTLog generator sets the Fulfillment order number in the RTLog to UNKNOWN. 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.</p>
<p>Xstore supports a tender type of Home Office Check. This tender type is not supported by ReSA. Retailers using this integration should not use the Xstore Home Office Check tender type.</p>
<p>Many Xstore transaction types and subtypes are exported as OTHER/OTHER. For more information on the types and subtypes, see Appendix A.</p>
<p>Xstore allows multiple sales associates at the line item level, however ReSA does not. Only the transaction level sales associate is exported to ReSA.</p>

Appendix: POSLog to RTLog Mapping Details

The mapping from the POSLog format to the RTLog format is defined in the Xstore configuration file RTLogMappingConfig.xml. This appendix provides details on the following mappings:

- [Transaction Type Mapping](#)
- [Tender Type Mapping](#)
- [Total Tender ID Mapping](#)
- [Item Type Mapping](#)
- [Reason Code Mapping](#)
- [Item Status/Sales Type Mapping](#)

Transaction Type Mapping

- The ReSA transaction type values are defined in code_type TRAT.
- The ReSA sub-transaction type values are defined in code_type TRAS.

[Table A-1](#) describes the Xstore to ReSA transaction type mapping.

Table A-1 Transaction Type Mapping

Xstore Transaction Type	ReSA Transaction Type TRAT	ReSA Sub-Transaction Type TRAS	Description
ACCOUNT_LOOKUP	OTHER	OTHER	ACCOUNT_LOOKUP transactions are passed from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA.
BALANCE_INQUIRY	OTHER	OTHER	BALANCE_INQUIRY transactions are passed from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA.
CREDIT_APPLICATION	OTHER	OTHER	CREDIT_APPLICATION transactions are passed from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA.
ESCROW	OTHER	OTHER	ESCROW transactions are passed from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA.

Table A-1 (Cont.) Transaction Type Mapping

Xstore Transaction Type	ReSA Transaction Type TRAT	ReSA Sub-Transaction Type TRAS	Description
EXCHANGE_RATE	OTHER	OTHER	EXCHANGE_RATE transactions are passed from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA.
GNRIC	OTHER	OTHER	GNRIC transactions are passed from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA.
INVENTORY_CONTROL	OTHER	OTHER	INVENTORY_CONTROL transactions are mapped from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA. Xstore should be configured so that inventory control transactions are not generated, and therefore not sent to ReSA.
INVENTORY_SUMMARY_COUNT	OTHER	OTHER	INVENTORY_SUMMARY_COUNT transactions are mapped from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA. Xstore should be configured so that inventory summary count transactions are not generated, and therefore not sent to ReSA.
MOVEMENT_PENDING	OTHER	OTHER	MOVEMENT_PENDING transactions are mapped from Xstore to ReSA for full visibility audit, but not otherwise implemented in ReSA. Xstore should be configured so that inventory summary count transactions are not generated, and therefore not sent to ReSA.
NO_SALE	NOSALE	NOSALE	NA
POST_VOID	PVOID	VOID	NA
RETAIL_SALE (can be mapped to multiple ReSA 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.

Table A-1 (Cont.) Transaction Type Mapping

Xstore Transaction Type	ReSA Transaction Type TRAT	ReSA Sub-Transaction Type TRAS	Description
TENDER_CONTROL (can be mapped to multiple ReSA 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
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.

Tender Type Mapping

- The ReSA tender type groups are defined in code_type TENT.

- The ReSA tenders are defined in the seeded data table POS_TENDER_TYPE_HEAD.

Table A-2 describes the Xstore to ReSA transaction tender type mapping.

Table A-2 Tender Type Mapping

Xstore		Xstore POS Log Tender Group Type		ReSA RTLog	
TenderTypeCode	TenderTypeID	Tender Type	Tender ID	TenderTypeGroup	TenderTypeID
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.
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:TravelersCheck	USD_TRAVELERS_CHECK	CHECK	If primary 2020, if foreign 2060.
	CAD_TRAVELERS_CHECK	dtv:TravelersCheck	CAD_TRAVELERS_CHECK	CHECK	If primary 2020, if foreign 2060.

Table A-2 (Cont.) Tender Type Mapping

Xstore		Xstore POS Log Tender Group Type		ReSA RTLog	
TenderTypeCode	TenderTypeID	Tender Type	Tender ID	TenderTypeGroup	TenderTypeID
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_MERCHANDISE_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
	MERCHANDISE_CREDIT_CARD	Voucher	MERCHANDISE_CREDIT_CARD	VOUCH	4050
	RELOAD_MERCHANDISE_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
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 ReSA.	

Total Tender ID Mapping

Table A-3 describes the ReSA mapping for the total ID record in the transaction header.

Table A-3 Total Tender ID Mapping

Xstore		ReSA 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 Type Mapping

ReSA tender type values are defined in code SAIT and used in the following:

- RTLOG TITEM record, item type field
- Sa_tran_item.item_type

Table A-4 describes the Xstore item type mapping.

Table A-4 Item Type Mapping

Xstore Item Type	ReSA Item Type	Description
Alteration	NMITEM	NA
Deposit	NMITEM	NA
dtv:GiftCertificate	GCN	Gift Card and Gift Certificate

Table A-4 (Cont.) Item Type Mapping

Xstore Item Type	ReSA Item Type	Description
dtv:NonMerchandise	NMITEM	NA
dtv:Payment	NMITEM	NA
Fee	NMITEM	NA
ItemCollection	ITEM	NA
Service	NMITEM	NA
Stock	ITEM	NA
Warranty	NMITEM	NA

Reason Code Mapping

Xstore has a single set of reason codes, used both for reason codes, price override codes, and other modifications. ReSA separates these concepts into individual sets used in different RTLog fields and saved to different database table/fields. Because reason codes can be mixed coming out of Xstore, ReSA has mapped some code values to multiple code types to avoid the possibility of errors.

ReSA Reason Codes

ReSA reason codes:

- Code type REAC

Note: ReSA supports a number of other transaction level reason codes. Only reason codes related to Xstore integration are listed here.

- SA_TRAN_HEAD.REASON_CODE
- Used for further information on a number of transaction types.
- Mapped to Xstore miscellaneous reason codes.

Table A-5 describes the reason code mapping.

Table A-5 ReSA Reason Codes

Xstore Reason Code	ReSA 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

Table A-5 (Cont.) ReSA Reason Codes

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

ReSA Return Reason Codes

ReSA return reason codes:

- Code type = SARR
- SA_TRAN_ITEM.RETURN_REASON_CODE

[Table A-6](#) describes the return reason code mapping.

Table A-6 ReSA Return Reason Codes

Xstore Reason Code	ReSA 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

ReSA Discount Reason Codes

ReSA discount reason codes:

- Code type SADT

Note: ReSA supports a number of other discount types. Only discount types related to Xstore integration are listed here.

- SA_TRAN_DISC.DISC_TYPE

[Table A-7](#) describes the discount reason code mapping.

Table A-7 ReSA Discount Reason Codes

Xstore Reason Code	ReSA 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

ReSA Item Price Override Reason Codes

ReSA item price override reason codes:

- Code type ORRC
- SA_TRAN_ITEM.OVERRIDE_REASON

[Table A-8](#) describes the item price override reason code mapping.

Table A-8 ReSA Item Price Override Reason Codes

Xstore Reason Code	ReSA 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/Sales Type Mapping

ReSA item status:

- Code type SASI

- SA_TRAN_ITEM.ITEM_STATUS

Valid values for the ReSA item status are shown in the following table:

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

ReSA sales type:

- Code type SASY
- SA_TRAN_ITEM.SALES_TYPE

Valid values for the ReSA sales type are shown in the following table:

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

Table A-9 describes the item status and sales type mapping.

Table A-9 ReSA Item Status/Sales Type Mapping

Xstore Item	Xstore Action	ReSA Item Status	ReSA 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

Table A-9 (Cont.) ReSA Item Status/Sales Type Mapping

Xstore Item	Xstore Action	ReSA Item Status	ReSA Sales Type
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	ORI	E
	Cancel	ORC	E
	Pickup	ORD	E
	Void when update or pickup	ORC	E
	Void when Init	S and V (two lines)	E

