

Oracle® Retail Returns Management

Operations Guide

Release 1.0

November 2006

Copyright © 2006, Oracle. All rights reserved.

Primary Author: Graham Fredrickson

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Preface	xi
Audience.....	xi
Related Documents	xi
Customer Support	xi
 1 Introduction	
What Is Oracle Retail Returns Management?.....	1-1
Concept Of A Return In Oracle Retail Returns Management	1-2
Context Model.....	1-4
Oracle Retail Returns Management Actors.....	1-5
Tax Responsibility In Oracle Retail Returns Management	1-5
 2 Configuration	
Starting Up The Application.....	2-1
Importing Data.....	2-1
Establishing A Store Hierarchy	2-2
XML Tags In Store Hierarchies	2-2
Importing The Store Hierarchy	2-3
Adding Stores To The Database.....	2-3
Editing The Store Hierarchy	2-4
Scheduling Post-processors.....	2-4
Configuring Security	2-4
Configuring Browser Options For Security	2-5
 3 Technical Architecture	
General Technologies And Frameworks	3-1
Architectural Styles And Patterns	3-1
Conceptual Modules.....	3-3
Enabling Technologies.....	3-5
J2EE	3-5
Struts	3-5
Axis.....	3-5
Web-Based User Interface.....	3-5
Physical Module View	3-6
User Interface Layer.....	3-7

Consumer Adapter Layer	3-8
Service Layer	3-9
Data Layer	3-10
Messaging	3-11

4 Functional Design And Overview

Conceptual Service Flow	4-1
Conceptual Data Flow	4-3
Historical Data	4-4
Engine Data: Policies, Rules, And Return Activities	4-5
Functional Assumptions	4-6
Functional Overviews	4-6
Return Tickets Functional Overview	4-6
Exception Files Functional Overview	4-6
Messages And Responses Functional Overview	4-7
Policies And Exceptions Functional Overview	4-7
Analytic Engine Functional Overview	4-7

5 Integration Methods And Communication Flow

Methods Of Contact	5-1
Oracle Retail Returns Management Messages	5-1
Sample XML For Return Transaction Scenarios	5-2
Point Of Return To Returns Management — Initial Return Request	5-2
Returns Management To Point Of Return — Return Response: Need Positive ID	5-6
Point Of Return To Returns Management — Second Return Request	5-8
Returns Management To Point Of Return — Second Return Response	5-10
Point Of Return To Returns Management — Return Result From Second Response	5-12
Point Of Return To Returns Management — Void Return	5-13
Offline Return Result	5-14
Implementation Decisions	5-15
Asynchronous Versus Synchronous Communication	5-16
XML Versus JavaBean Messages	5-16
Web Service Versus Enterprise JavaBeans And Remote Method Invocation Call	5-16
Elements	5-17
Return Request	5-17
Return Response	5-18
Return Result	5-18
Adding A New Rule	5-19
Environment entries In ejb-jar.xml	5-20
Return Ticket Formatting Entries	5-20
Auditing Entries	5-21
Web Service Interface	5-21
Relationship Of Oracle Retail Returns Management Data To ARTS Transaction Data	5-22

6 Exception Files

Exception File And Count Calculation	6-1
--	-----

Definition Of Return, For Calculation.....	6-2
Exceptions	6-3
Customer Exceptions	6-3
Cashier Exceptions	6-3

A Extension Guidelines

Element Location And Schema Definition	A-1
Element Usage And Retrieval	A-2

List of Examples

5-1	Initial Return Request.....	5-2
5-2	Return Response Requesting Positive ID	5-6
5-3	Second Return Request	5-8
5-4	Second Return Response.....	5-10
5-5	Return Result	5-12
5-6	Void Return Result	5-13
5-7	Offline Return Result.....	5-14
A-1	Message Extension.....	A-1
A-2	XML Message Using The MessageExtension	A-2
A-3	Searching The MessageExtension Elements	A-2

List of Figures

1-1	Oracle Retail Returns Management Decisions Process	1-1
1-2	Oracle Retail Returns Management Context Model	1-4
3-1	Oracle Retail Returns Management Architectural Layers	3-2
3-2	Oracle Retail Returns Management Conceptual Modules.....	3-3
3-3	Oracle Retail Returns Management Web-based User Interface	3-6
3-4	Oracle Retail Returns Management Physical Module View	3-7
3-5	Oracle Retail Returns Management Consumer Adapter Layer	3-8
3-6	Oracle Retail Returns Management Service Layer.....	3-9
3-7	Oracle Retail Returns Management Data Layer.....	3-10
4-1	Oracle Retail Returns Management Conceptual Service Flow	4-2
4-2	Oracle Retail Returns Management Conceptual Data Flow	4-4
4-3	Oracle Retail Returns Management Policies and Rules	4-5

List of Tables

1-1	Oracle Retail Returns Management Actors.....	1-5
2-1	Tags In A Store Hierarchy	2-3
5-1	XSD Locations.....	5-2
5-2	Required Elements By Return Request.....	5-17
5-3	Required Elements By Return Response	5-18
5-4	Required Elements By Return Result Use Case.....	5-18
5-5	Return Ticket Format <env-entry>.....	5-20
5-6	Audit Target <env-entry>.....	5-21
5-7	Web Service Methods	5-21
6-1	Exception Counting Examples.....	6-2
A-1	MessageExtension Locations.....	A-1

Preface

The purpose of this document is to guide deployers of Oracle Retail Returns Management through the back end administration process. This document includes information about configuring the application as well as architectural information and information necessary to integrate or extend Oracle Retail Returns Management.

Audience

The audience for this document consists of all deployers, administrators, and developers of Oracle Retail Returns Management.

Related Documents

For more information, see the following documents in the Oracle Retail Returns Management Release 1.0 documentation set:

- *Oracle Retail Returns Management User Guide*
- *Oracle Retail Returns Management Install Guide*
- *Oracle Retail Returns Management Release Notes*

Customer Support

- <https://metalink.oracle.com>

When contacting Customer Support, please provide:

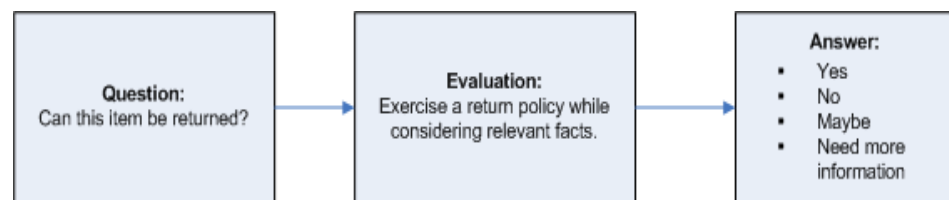
- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to recreate
- Exact error message received
- Screen shots of each step you take

Introduction

Oracle Retail Returns Management is a centralized system designed to monitor and control the return of retail merchandise. Control is provided through a flexible set of rules that determine if a particular item is returnable. Monitoring is provided through pattern watches, enabling a retailer to uncover unusual return patterns indicative of fraud, poor product quality, and so forth.

At its most basic, Oracle Retail Returns Management enables you to centralize the knowledge and decision making of what is and what is not returnable.

Figure 1–1 Oracle Retail Returns Management Decisions Process



What Is Oracle Retail Returns Management?

Oracle Retail Returns Management is packaged as a stand alone product.

Oracle Retail Returns Management provides:

- A flexible and configurable set of rules
- The ability to collect differing rules into multiple policies
- The ability to assign different policies to different situations (for instance, one policy might apply to receipted items, another policy might apply to non-receipted items)
- A decision engine to initiate the policies
- A defined application-program interface (API) for evaluation of returnability
- A defined API for post-return information gathering
- A web-based user interface for administration

Concept Of A Return In Oracle Retail Returns Management

Occasionally, a customer might buy an item from a retailer and then decide that they no longer want the item. This could be for any number of reasons:

- Dissatisfaction with quality
- Finding a better price somewhere else
- Buying the wrong size

Most retailers allow customers to return items they have purchased under certain conditions. Conditions might include that the item was bought within the last 90 days, that the item is in an unopened state, or that the customer has a receipt. Additionally, a retailer might decide to charge a restocking fee, issue a return merchandise authorization (RMA) and a call tag, provide a discount on the customer's next purchase (in case of a quality problem), or other actions based around the return. Finally, returns can happen at different places in a retailer, such as at a point of sale, a separate returns desk, or at a remote call center.

With the act of returning, there are several steps that a retailer must go through.

- A retailer must determine what merchandise is being returned.
- A retailer must decide if the merchandise is returnable.
- A retailer must record that the item was returned, which affects financial and inventory calculations.
- Once the retailer accepts a return, the retailer must physically move the item to some place (such as placing it in a returns cage, or issuing pickup instructions to a carrier in the case of a remote call center). Afterwards, the item might undergo further actions, such as being returned to the vendor or destroyed.

Of these many aspects of the return process, Oracle Retail Returns Management focuses mainly on the conditions for return, sometimes referred to in this document as returnability. Returnability is determined by the rules and policies configured in Oracle Retail Returns Management. See "Engine Data: Policies, Rules, And Return Activities" on page 5 of the "Functional Design and Overview" chapter. Oracle Retail Returns Management can associate metadata with return policies so Oracle Retail Returns Management can decide to use different policies in different situations. Each policy has its own set of rules which defines returnability for that situation, for example, non-receipted returns might be more restrictive than receipted returns.

A policy is composed of one or more rules. Each policy has associated metadata that enables the service layer to choose the most appropriate policy for the current item in question.

Oracle Retail Returns Management does not prescribe the point of return, what happens before or after a customer initiates a return, or the financial and inventory ramifications of the returns process. Furthermore, Oracle Retail Returns Management isolates itself from the majority of data found in the retail enterprise and restricts itself to knowing a prescribed set of facts. It is this set of facts that Oracle Retail Returns Management uses when evaluating a policy and determining if a product is returnable.

Oracle Retail Returns Management has been isolated to this degree in order to make Oracle Retail Returns Management applicable to a wide variety of situations. As long as a point of return can communicate with Oracle Retail Returns Management, it is immaterial where that point of return is located and what kind of point of return it is (register, returns desk, and so forth). The point of return provides the majority of the data that Oracle Retail Returns Management needs to make its decision, so Oracle

Retail Returns Management is shielded from the format of transaction data in a retail enterprise.

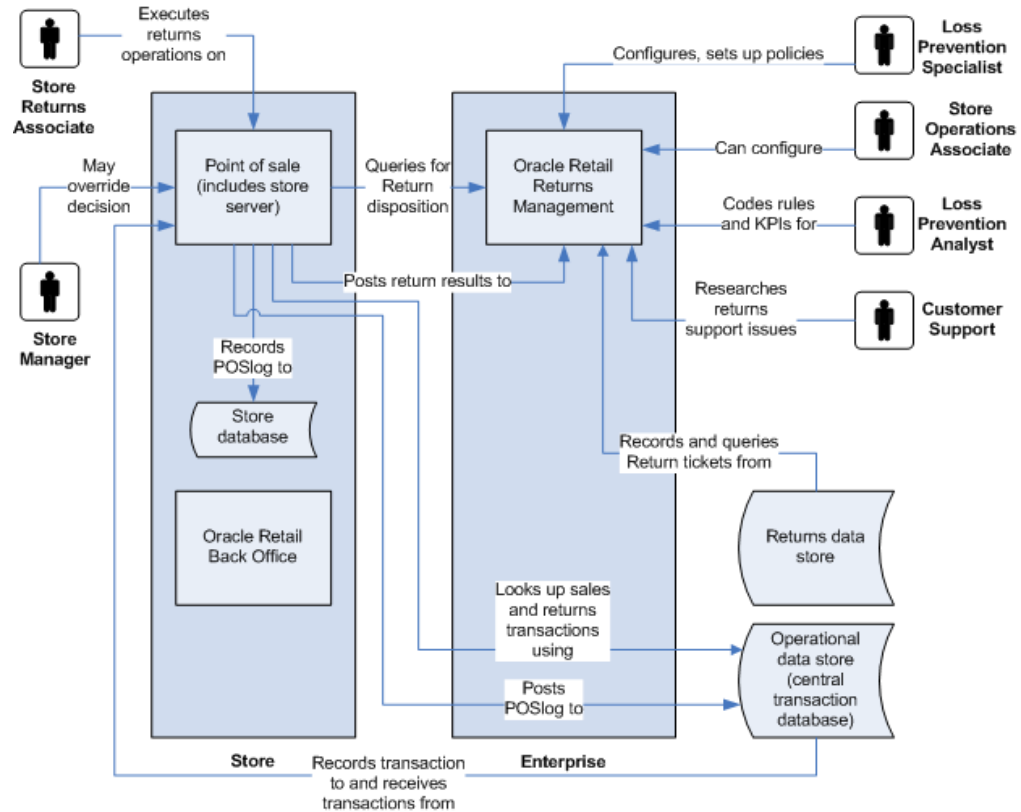
By configuring the rules and policies in Oracle Retail Returns Management, the retailer can enforce the same return policies across the enterprise. The return policies can be centrally administered. Since the policies are not compiled code, they can be quickly updated. Finally, Oracle Retail Returns Management records the steps it makes for each decision, allowing a customer to ask exactly why a return was accepted or declined.

As stated previously, Oracle Retail Returns Management focuses mainly on the conditions for return. The other main focus of Oracle Retail Returns Management is that it keeps a record of what was actually returned to the enterprise. This information is rolled up in both the return ticket data (there is a ticket for each return, each ticket having one or more line items corresponding to items on the return) as well as a list of exception activity. The exception file records unusual activity that might be fraudulent. Using these records, Oracle Retail Returns Management provides decision support to the enterprise. The retail enterprise can monitor these records to determine the volume of returns, the type of items being returned, and patterns of fraudulent behavior.

Context Model

The following diagram identifies how Oracle Retail Returns Management exists with other existing Oracle Retail products. Also included in the context are the actors mentioned in [Table 1-1, "Oracle Retail Returns Management Actors" on page 5](#).

Figure 1-2 Oracle Retail Returns Management Context Model



Oracle Retail Returns Management Actors

The following tables lists the actors that Oracle Retail Returns Management expects to interact with. Although most of the actors are users, some items such as the point of return are expected to interact with Oracle Retail Returns Management without direct human intervention.

Table 1–1 Oracle Retail Returns Management Actors

Interaction	Actors					
	Store			Corporate		
	Sales Associate	Point of return	Store Manager	Business analyst	Customer Service Rep	Loss Prevention Specialist Software Developer
Request return	X				X	
Update return information		X				
Develop return policies				X		X
Monitor exception behaviors			X			X
Monitor what is being returned			X	X		X
Audit a specific returns decision				X	X	

Tax Responsibility In Oracle Retail Returns Management

Oracle Retail Returns Management evaluates data provided from the point of return as well as centrally stored historical data and provides a recommendation to the point of return for the handling of a potential return. Because Oracle Retail Returns Management is not transactional in nature, Oracle Retail Returns Management has no tax responsibility. All of the tax responsibility belongs to the point of return when the return transaction is created and processed.

Configuration

This chapter discusses required and optional configuration steps that must be performed after Oracle Retail Returns Management is deployed. This chapter also discusses how to run the application after it has been configured.

Starting Up The Application

You must run the application in order to perform some final configuration tasks, such as importing parameters. To run Oracle Retail Returns Management, verify that the application is available in the Application Server environment. Then, access the following URL from a browser, specifying the correct application server host name and port number:

```
http://<app-server-hostname>:<app-server-port#>/returnsmanagement/
```

Importing Data

Within Oracle Retail Returns Management, select **Data Management** to display a list of data import options. You can import data immediately or schedule an import for later. See "Data Management" in the *Oracle Retail Returns Management User Guide*.

The following types of data can be imported:

- POSlog transaction file
- Parameters and application parameters
Parameters are values that set options and behavior for Oracle Retail applications. They are passed among the applications as XML files using JMS communication.
- Store hierarchy
See See "Establishing A Store Hierarchy" on page 2 of this chapter.

Establishing A Store Hierarchy

Returns Management displays store data according to a hierarchy you define with an XML format. To establish a store hierarchy, start with the initial store hierarchy file, storeHierarchy.xml, provided by Oracle Retail. Edit this file to reflect the proper hierarchy and import the file. To change the store hierarchy, edit the XML file and reimport it.

Note: Ad-hoc groups (created and managed through the **Admin->Store Directory->Store Groups** interface in Oracle Retail Returns Management) are a separate hierarchy at the root level. You can have any number of groups, but they cannot be nested; all the groups remain at the top level. You do not edit XML to define ad-hoc groups.

Work with Oracle Retail to define your store hierarchy. A sample hierarchy might look like this:

- Company
 - Region
 - * Division

Any hierarchy can be duplicated in Oracle Retail Returns Management. You can include stores at any level of the hierarchy, and you can name the levels according to your own business practices.

XML Tags In Store Hierarchies

The following is an example of a store hierarchy and illustrates the XML tags that are used to define the hierarchy.

```
<storehierarchy>
  <hierarchy-list>
    <hierarchy>
      <level-list>
        <level>
          additional nested levels
        </level>
      </level-list>
      <node-list>
        <node>
          additional nested nodes
          <store /> any node may contain one or more stores
        </node>
      </node-list>
    </hierarchy>
  </hierarchy-list>
</storehierarchy>
```

The following table describes the tags used in a store hierarchy.

Table 2–1 Tags In A Store Hierarchy

Tag	Definition
storehierarchy	This wraps the entire hierarchy data structure.
hierarchy-list	This wraps the collection of hierarchies. It is possible to have more than one store hierarchy, although this functionality is not currently used.
hierarchy	This wraps a particular hierarchy. It requires function and name attributes.
level-list	This wraps the list of hierarchy levels.
level	This defines a level in the hierarchy. It requires ID, name, and parent attributes. The level with parent 0 is the root of the hierarchy. Levels must be nested to show their relationship.
node	This establishes a particular point in the hierarchy. A node can contain stores or other nodes. It requires level and name attributes. Create the hierarchy of regions by nesting nodes.
store	This refers to a particular store. It requires an ID attribute that corresponds to the store ID number in the database. Each store must have a database record. A store tag can be empty (/>).

Importing The Store Hierarchy

You can import a store hierarchy from the Admin section of Oracle Retail Returns Management.

To import a store hierarchy:

1. Launch the application.
2. Select **Admin**.
3. Select **Store Directory**.
4. Select **Store Hierarchy**.
5. Click **Browse** to locate your store hierarchy file.
6. Click **Import**. Oracle Retail Returns Management displays the message **Import Store Hierarchy Complete** and displays the contents of the imported file to confirm the importation.

You can also perform this function from the Data Management page.

Adding Stores To The Database

Add stores to the database before attempting to add them to your hierarchy. The store hierarchy does not establish stores in the system; it merely tells the system where an existing store record fits in the enterprise. A store which exists in the hierarchy but not in the database could cause confusion or errors; for example, you could run a search with the store as one of the criteria, but no data would be found.

To add a store to the database, use SQL to add it to the PA_STR_RTL table.

Editing The Store Hierarchy

Start by editing a sample store hierarchy file provided by Oracle Retail. Edit the attributes and tags provided, copying tags where needed to preserve the structure and syntax of the existing file. Only a few of the tags require editing.

To edit a hierarchy:

1. Establish the levels in your hierarchy by creating level tags and giving them appropriate names.
2. Add nodes that replicate all the nodes in your hierarchy.
3. Add stores to appropriate nodes.
4. Import the store hierarchy file again.

Scheduling Post-processors

After installation, you must schedule post-processor jobs as part of the configuration process. Post-processors create summary data for use in reporting.

To schedule regular post-processor jobs for Oracle Retail Returns Management:

1. Click the **Data Management** tab.
2. From the list of import options, click **Available Imports**.
3. From the available imports, click the **Schedule** link adjacent to the Returns Dashboard Processor. Oracle Retail Returns Management displays the Job Schedule page.
4. Choose the **Scheduled** option. Oracle Retail Returns Management displays additional scheduling options.
5. For Begin Date, enter the current date.
6. Check the **Repeating** box.
7. Leave the **No End** radio button selected.
8. Set a Repeating option.
9. Enter a Run Time in the appropriate box and click **Add**.
10. Click **Next**. Oracle Retail Returns Management displays a Notification page.
11. Add the e-mail addresses of anyone you want to be notified.
12. Click **Next**. The Distribution Summary screen is displayed.
13. Click **Submit Job**. The Distribution Confirmation screen is displayed.
14. Click **Done**.

Configuring Security

Oracle Retail Returns Management has many individual security access points. This enables you to control the functionality to which any particular end user has access. You can also control workflow through approval permissions, enabling some employees to schedule tasks which others must approve.

For more information about security roles, see "Security and Errors" in the *Oracle Retail Returns Management User Guide*.

Configuring Browser Options For Security

When multiple users use the same systems, savvy end-users could use the browser history trail to access data not appropriate for their access levels. You must configure the browsers used to access the application to remove access to the history function and the address bar to prevent this.

For more information about configuring security, see "Configuring Security" in the *Oracle Retail Returns Management Installation Guide*.

Technical Architecture

This chapter presents a concise description of the system's architecture. The system is considered a collection of run time behaviors, a set of software modules, and a member of a larger group of external systems and actors.

General Technologies And Frameworks

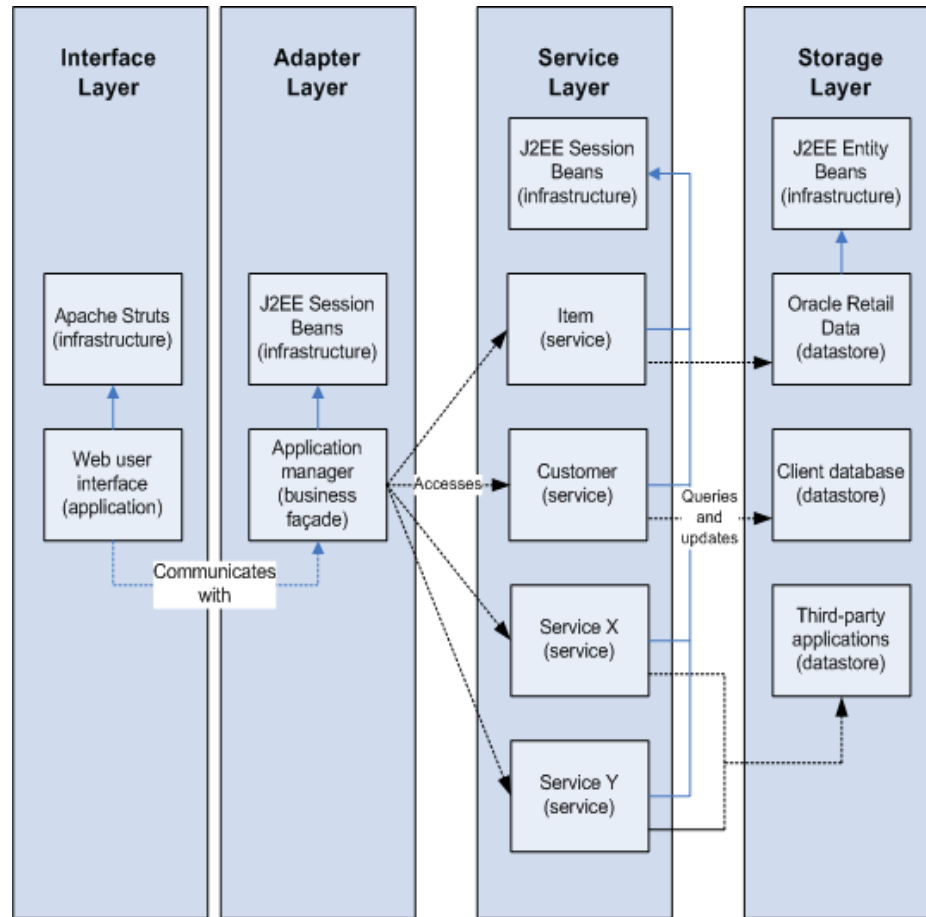
This section describes technologies and frameworks that are used by Oracle Retail Returns Management. These assets are not unique to Oracle Retail Returns Management but are key to its implementation.

Architectural Styles And Patterns

The following information describes the architectural styles and patterns of Oracle Retail Returns Management and its component pieces.

Architectural Layers

The architectural layers design style is not unique to Oracle Retail Returns Management. It is a shared architecture across all of Oracle Retail's web applications.

Figure 3–1 Oracle Retail Returns Management Architectural Layers

This diagram shows the break down of the user interface and business logic across the application. The design uses both a model-view-controller pattern as well as a façade pattern to hide the implementation of business logic. The façade not only applies to the user interface, but to the other pieces of business logic as well.

As with all object construction, the goal is to reduce the dependency between the objects so that code might be more freely modified without adversely affecting other parts of the application.

Apache Struts is used at the graphical user interface layer to provide both a clear separation of the controller, view, and model as well as providing a well known technology for ease of extension. At the façade layer, an application manager is implemented using J2EE session beans to provide a coarse-grained view of business logic to the user interface. Each manager communicates directly with one or more services located in the service layer that provide fine-grained business operations. These service beans are also implemented using J2EE session beans. Each service bean can then communicate with other services or down to persistent storage in the data layer. By abstracting the storage away from the other layers, this not only enables the design to leverage J2EE entity beans but also allows for disparate storage mediums for integrating with third party data stores.

Interface modules

Functionality to interact with an outside actor, such as the user or the point of return. These modules include:

- **Web Service Adapter**

The point of return is expected to talk to Oracle Retail Returns Management using a web service. An adapter is provided to enable this functionality out of the box.

- **JSP UI**

The user interface is HTML based, powered by JavaServer Pages (JSP) and Struts.

Service modules

Functionality that provides the core services offered by Oracle Retail Returns Management. These modules include:

- **Return Evaluation**

This is where the initial "Is this item returnable?" question is asked.

- **Final Result Processing**

This is where Oracle Retail Returns Management consumes results to maintain historical data.

- **Analytic Engine**

The decision engine that evaluates Oracle Retail Returns Management rules.

Administrative modules

Functionality to administer, monitor, and examine Oracle Retail Returns Management behavior and data. These modules include:

- **Auditing**

Provides the capability to examine the steps that went into a return decision.

- **Policy Rule and Maintenance**

Enables the user to add, modify, or delete policies and the rules which comprise them.

Data modules

Functionality tied to persistent storage. These modules include:

- **Audit Log**

The steps recorded by the engine as it processes a policy.

- **Policies**

Collections of rules that are bound to certain conditions, for example, some policies apply only to receipted items.

- **Rules**

Each rule evaluates a return-related question, for example, "How many returns has this customer attempted in the past week?" Rules are responsible for indicating to the point of return whether an item is returnable or not.

- **Return Tickets**

Oracle Retail Returns Management stores each return request for later manipulation by Final Result Processing.

- **Exception File**

Tables that store information about exceptional behavior, for example, penalty boxes, which are created when a customer has violated some inappropriate level of behavior.

Enabling Technologies

J2EE

Oracle Retail Returns Management is built using the technologies of the Java 2 Enterprise Edition stack.

Struts

Oracle Retail Returns Management uses the Apache Struts project to present its Java Server Pages in a J2EE compliant container. For more information, go to <http://struts.apache.org/>

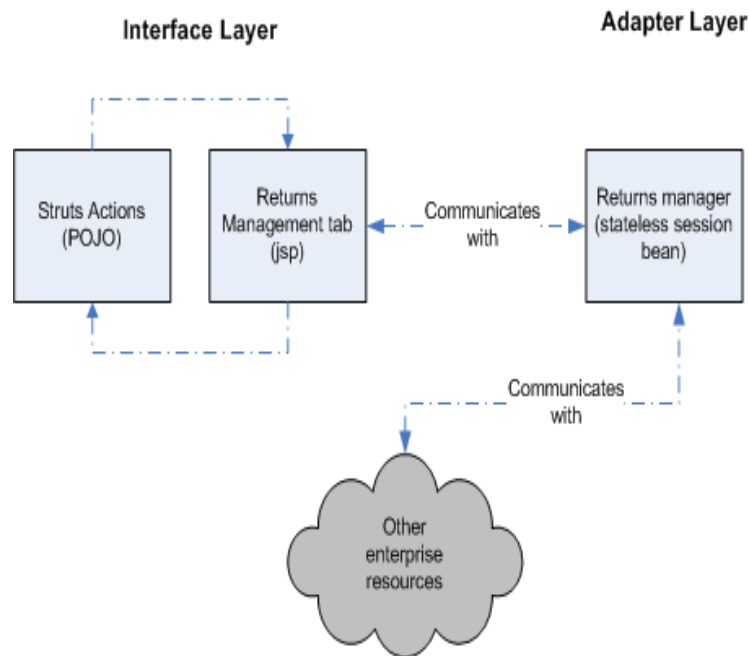
Axis

Oracle Retail Returns Management uses Apache Axis to provide a container-neutral way of presenting web services. For more information, go to <http://ws.apache.org/axis/>

Web-Based User Interface

The user interface for Oracle Retail Returns Management can be divided into two classes of components:

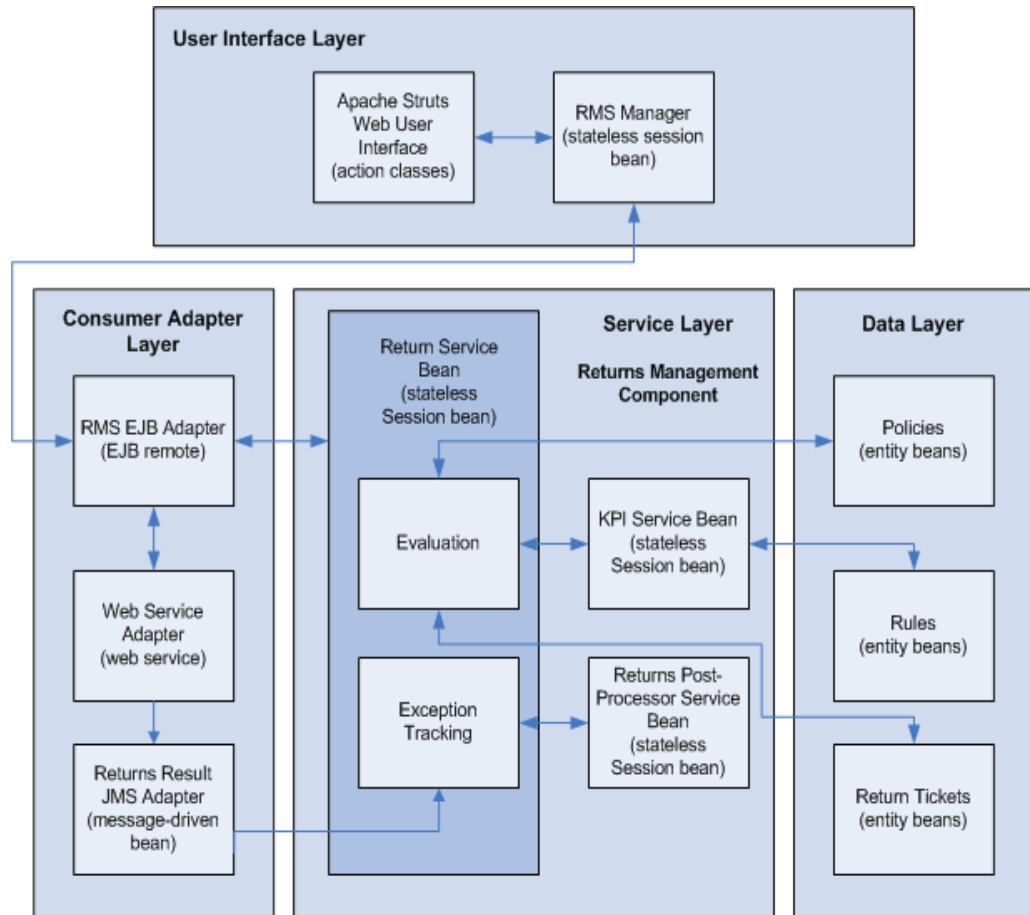
- JSPs and Action Classes
- The Returns Manager

Figure 3–3 Oracle Retail Returns Management Web-based User Interface

This diagram shows the general web user interface diagram as it relates to Oracle Retail Returns Management in particular. Here we see that the Interface Layer is composed of the JSPs attached to the Oracle Retail Returns Management tab as well as their backing action classes. These actions attempt to provide user interface-level functionality such as flow control and rudimentary data checking. The class at the façade layer, the Returns Manager, exposed numerous coarse-grained methods to enable the Action classes to retrieve key performance indicators (KPIs) — also known as return activities, search the exception file, and other administrative tasks. The Returns Manager then talks to whatever resources it needs to provide the necessary information to the Interface Layer. Since all of this work is hidden behind the façade, the Returns Manager has a great deal of flexibility in deciding how to perform a certain task with minimal impact to the client classes in the interface layer.

Physical Module View

The conceptual module view divided the system into modules based off of the functionality provided. The physical module view divides the modules along the notion of module type. For instance, rather than showing policy maintenance as a separate service, policy maintenance is included in the larger Returns Service.

Figure 3–4 Oracle Retail Returns Management Physical Module View

This diagram broadly divides the modules into four groups:

- The user interface layer, responsible for the web-based user interface.
- The consumer adapter layer, responsible for communication with Oracle Retail Returns Management.
- The service layer, which provides the heavy lifting of Oracle Retail Returns Management functionality.
- The data layer, which provides access to persistent storage.

User Interface Layer

As mentioned previously, the Oracle Retail Returns Management user interface is a web-based system implemented using Struts and Tiles. The presentation layer consists of a large number of JSPs, forms, actions, and other artifacts of the Struts system. Behind this presentation layer resides the Returns Manager façade, which provides the Struts actions with access to the business logic of the application.

In the Model-View-Controller paradigm, Struts provides all three pieces:

- Actions provide the model.
- JSPs are the view.
- Struts classes and configuration files provide the controller.

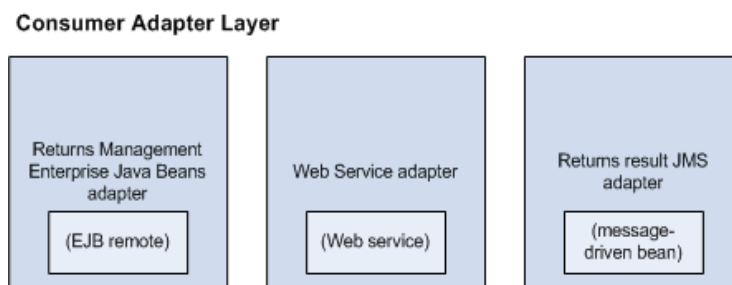
However, the action classes are not the model used by Oracle Retail Returns Management. The action classes exist primarily to marshall data from the presentation layer down to the business logic, and to provide coarse-grained flow control over a business process. It is important to realize that the real model of Oracle Retail Returns Management has little to do with the action classes. The real model is implemented behind the Manager façade in the service and data layers.

JavaServer Pages (JSPs) are text files which correspond to the normal JavaServer Page formatting restrictions. Actions and forms are normal Java classes while the Manager is implemented as a stateless session bean.

Consumer Adapter Layer

The consumer adapter layer provides the different interfaces into the Oracle Retail Returns Management system. This layer is specifically split from the service layer in the design to decouple the interface of communication from the implementation classes. Therefore, regardless of underlying changes in how Oracle Retail Returns Management is implemented, the interface expected by clients can remain static. The separation provides a well-defined contract with which service consumers can interact, and enables future custom adapters to be integrated with minimal effort.

Figure 3–5 Oracle Retail Returns Management Consumer Adapter Layer



There are three ways to communicate with the Oracle Retail Returns Management system:

- Return Request and Return Response are exposed using a Web Services interface.
- Return Results are collected using a JMS queue.
- The EJB remote interface, which enables arbitrary methods to be invoked on the service bean.

For more information about communicating with Oracle Retail Returns Management, see [Chapter 5, "Integration Methods And Communication Flow"](#).

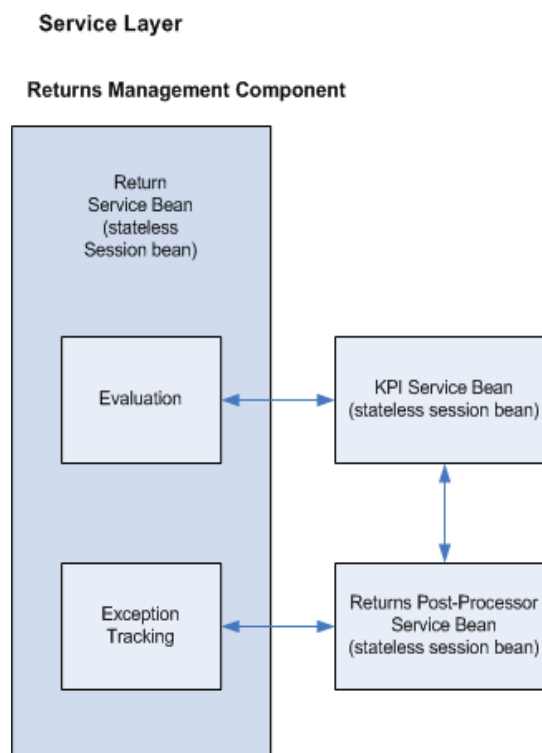
The first two interfaces mentioned enable flexible client implementations while providing clear interfaces for interaction. The EJB remote interface is available when some system needs to operate with Oracle Retail Returns Management outside of the normal API-type transactions, for example, the Returns Manager makes liberal use of the remote interface for inquiring and maintaining Oracle Retail Returns Management data.

Service Layer

The term service is used in two different ways when describing Oracle Retail Returns Management.

- The first way is the abilities provided by Oracle Retail Returns Management in its role as a service in a service-oriented architecture. These abilities are restricted to the two interfaces of evaluation (return request) and exception tracking (return result).
- The second way is used to describe the interoperable commerce services that form the core of Oracle Retail Returns Management and Central Office functionality. These are the services that live in the service layer. These services are generally not client accessible. They provide discrete business functions available to other services and Application Managers living in the façade layer.

Figure 3–6 Oracle Retail Returns Management Service Layer



When describing Oracle Retail Returns Management in terms of service-oriented architecture, the services provided by Oracle Retail Returns Management are implemented primarily in one class, the Return Service Bean. These services are exposed in the Consumer Adapter Layer for access from client routines.

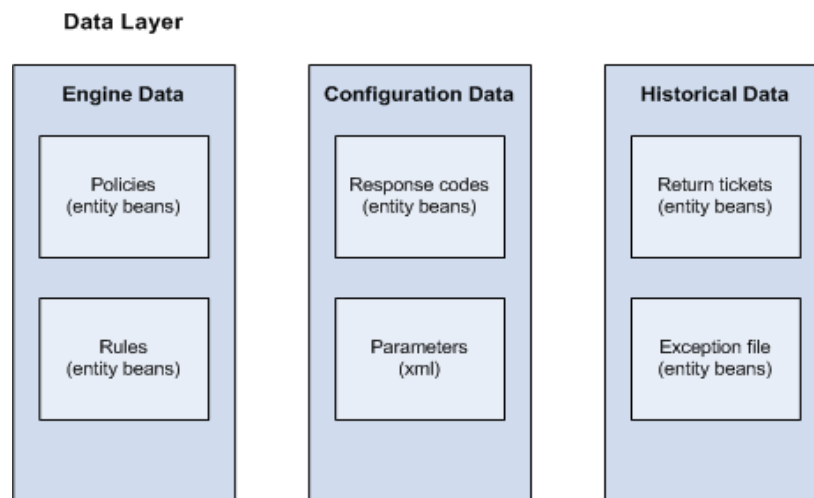
When describing Oracle Retail Returns Management in terms of its commerce service modules, then not only would the return service bean be included but also the modules which exist to provide support to Oracle Retail Returns Management, such as the KPI Service and the Returns Post-Processor Service.

Data Layer

For purposes of discussion, this document splits the data used by Oracle Retail Returns Management into three types:

- Configuration data used to control behavior.
- Historical data used to record information. See “Historical Data” on page 4 of [Chapter 4, "Functional Design And Overview"](#).
- Engine data used to determine returnability. See “Engine Data: Policies, Rules, And Return Activities” on page 5 of [Chapter 4, "Functional Design And Overview"](#).

Figure 3–7 Oracle Retail Returns Management Data Layer



These types of data are generally stored in an RDBMS and accessed using an entity bean layer. Each general type is covered briefly in the following sections.

Configuration Data

Oracle Retail Returns Management has a list of valid response codes that can be returned to a client. This list is maintained in the data store and is accessed using entity beans. Oracle Retail Returns Management also maintains a list of receipt messages which are accessed in a similar fashion.

Following an established pattern in Oracle Retail products, Oracle Retail Returns Management uses an XML parameter file to maintain a list of configuration options and choices that can be modified for a particular customer deployment. These parameters control a variety of behaviors as well as providing a place for some of the data used during processing (for instance, the list of acceptable tenders).

Note: For more information on specific parameters, see the Parameter Names and Values Addendum in the `_resources` directory provided with your Oracle Retail Returns Management documentation.

Messaging

This section describes the interface of the two main services provided by Oracle Retail Returns Management:

- Evaluation of the return request
- Processing of the return result

These services are expected to be invoked from an external source, usually the point of return.

In order to provide language neutrality, these two services are accessed in a stateless fashion using XML documents. Return request is a synchronous message, that is, the invoker is expected to wait on a return response message. Return result is an asynchronous message that can be invoked in a fire-and-forget fashion.

Although both services are exposed using a web service interface, a message-driven bean exists to collect return result messages asynchronously as a best practice. More details about messaging are provided in [Chapter 5, "Integration Methods And Communication Flow"](#).

Functional Design And Overview

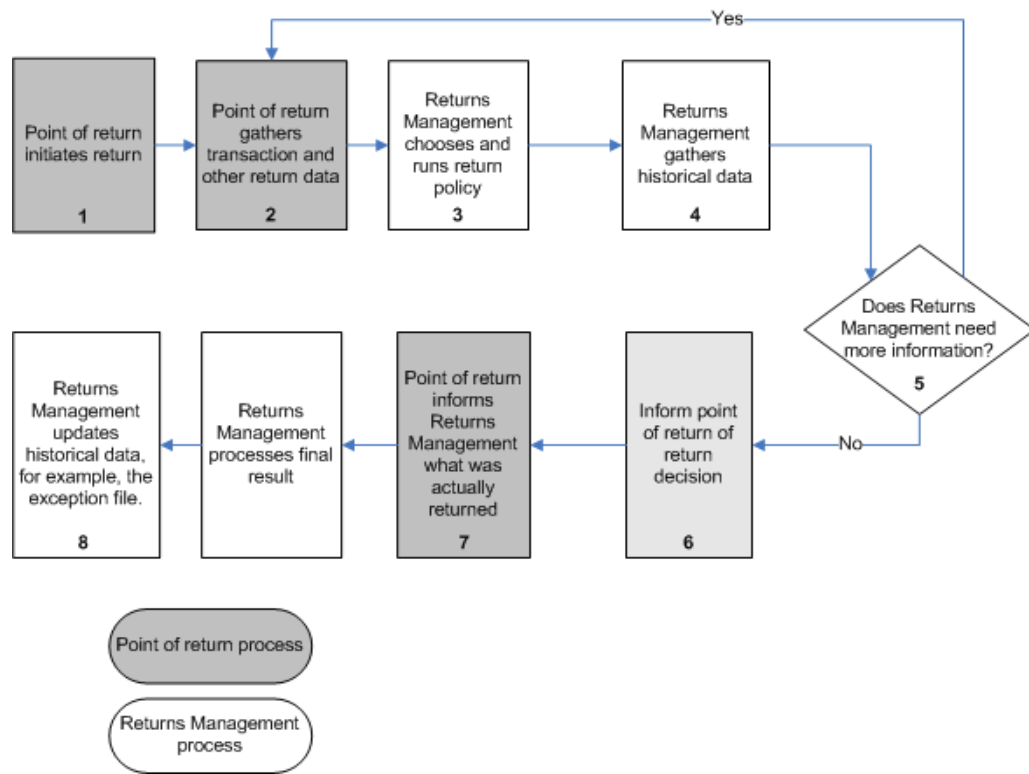
This chapter addresses the functional aspects of Oracle Retail Returns Management. This chapter provides the following:

- Conceptual Service Flow
- Conceptual Data Flow
- Historical Data
- Engine Data: Policies, Rules, And Return Activities
- Functional Assumptions
- Functional Overviews

Conceptual Service Flow

The following flowchart illustrates the steps in a typical Oracle Retail Returns Management session, including which steps are initiated by Oracle Retail Returns Management and which steps are initiated by the point of return (the point at which a return is initiated, for example, a cashier at a retailer).

Note: This flowchart is a simplified representation of the service flow and does not attempt to explain the technologies used to implement Oracle Retail Returns Management.

Figure 4–1 Oracle Retail Returns Management Conceptual Service Flow

The following sequence is a typical Oracle Retail Returns Management round-trip session:

1. A point of return initiates a merchandise return.
2. A message is sent from the point of return to the Oracle Retail Returns Management system indicating the item to be returned, if the customer has a receipt, and possibly other data. See “Point Of Return To Returns Management — Initial Return Request” on page 2 of [Chapter 5, "Integration Methods And Communication Flow"](#).
3. Oracle Retail Returns Management chooses which policy to initiate. A policy is comprised of one or more rules, and each policy has associated metadata that enables the service layer to choose the most appropriate policy for the current item in question. See "Determining Return Policies" in *Oracle Retail Returns Management User Guide*.
4. Oracle Retail Returns Management gathers together relevant server-side historical information, such as entries related to the customer in the exception file.
5. The policy might require additional data from the point of return, such as a positive ID from the customer. In this case, a message is sent back to the point of return asking for the additional data. See “Returns Management To Point Of Return — Return Response: Need Positive ID” on page 6 of [Chapter 5, "Integration Methods And Communication Flow"](#).
6. The policy decides if the item is returnable or not. Oracle Retail Returns Management informs the point of return of its decision, and provides a tender recommendation.

The point of return ultimately decides to accept the return or not. For example, Oracle Retail Returns Management might say that an item is non-returnable, but a local manager might override that decision. A local manager can also ignore a tender recommendation.

7. The point of return informs Oracle Retail Returns Management of its decision. See "Point Of Return To Returns Management — Return Result From Second Response" on page 12 of [Chapter 5, "Integration Methods And Communication Flow"](#).
8. Oracle Retail Returns Management uses information from the point of return to update its historical records such as the exception file. The exception file acts as a constantly evolving knowledge base that can help the analytic engine decide which customers, items, cashiers, or stores are at higher risk for return fraud.

Conceptual Data Flow

To understand how the various modules relate to each other at run time, imagine the flow of data through the system.

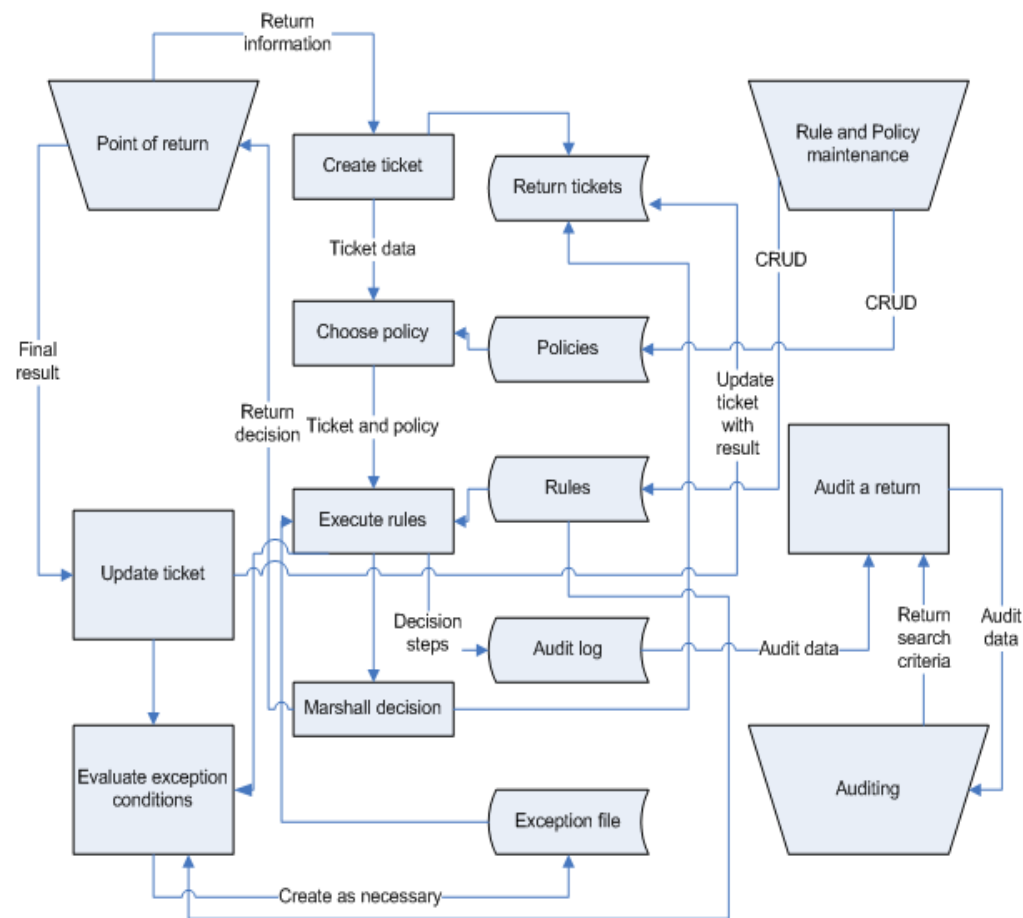
- The four main processes (return request, return results, policy administration, and auditing) operate on an intersecting set of data.
- Return requests are sent to Oracle Retail Returns Management and cause return tickets to be created and rules to be read.
- Rule initiation creates entries in the audit log.
- Return responses are sent back to the point of return.
- Return results update existing return tickets, and create entries in the exception file.

Policy administration enables the creation and maintenance of both rules and policies. See "Determining Return Policies" in the *Oracle Retail Returns Management User Guide*.

Auditing applications read the audit entries created during the evaluation phase.

The following data flow diagram lists the four main processes and the flow of the data that they create and consume.

Note: The Policy Rule and Maintenance interface is shown as directly updating the policy rules using Create, Read, Update, Delete (CRUD).

Figure 4–2 Oracle Retail Returns Management Conceptual Data Flow

Historical Data

Oracle Retail Returns Management maintains a set of historical data. To record a particular decision during the evaluation phase, Oracle Retail Returns Management creates a return ticket that records what item is being returned, who wants to return the item, and the Oracle Retail Returns Management decision about the returnability of an item. This return ticket is later updated during the return result process to reflect what was actually returned at the point of return.

An exception file that counts up the total number of times an instance of a tracked behavior occurs, for example, a customer with a non-receipted return or a customer with a tender override, is maintained by Oracle Retail Returns Management during the return result phase.

The following are grouped in the exception file:

- Customer exceptions, such as line items that reflect customer return activities being violated.
- Customer exception counts and freeze dates.
- Customer Service Overrides, that is, a count of overrides, per day, per customer.

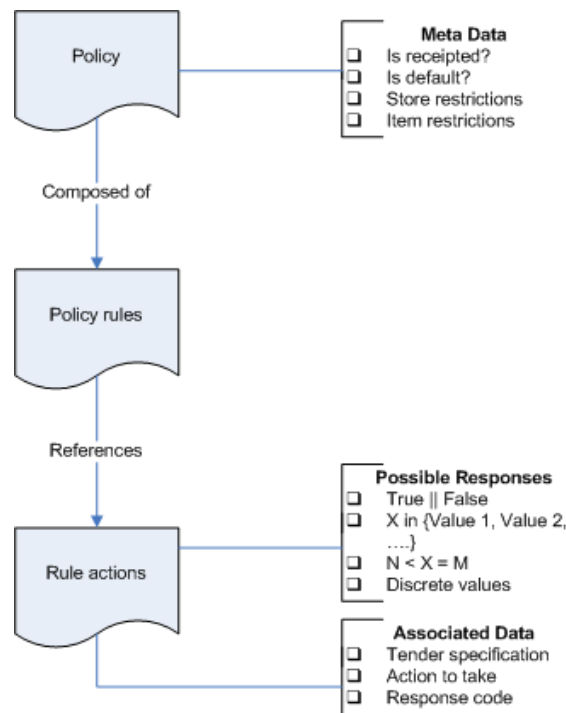
For more information, see the following:

- "Exceptions" in the *Oracle Retail Returns Management User Guide*.
- "Selecting Customer Exceptions to Track" in the *Oracle Retail Returns Management User Guide*.
- See [Chapter 6, "Exception Files"](#).

Engine Data: Policies, Rules, And Return Activities

Oracle Retail Returns Management uses a decision engine to codify and enforce returns policies. The decision engine operates on sets of rules, which are collected into policies. The rules operate on a set of facts that the engine supplies at run time. These facts are evaluated by the rules, which eventually return an answer back from the engine. See "Determining Return Policies" in the *Oracle Retail Returns Management User Guide*.

Figure 4–3 Oracle Retail Returns Management Policies and Rules



This diagram shows how policies, rules, and rule actions are conceptually related.

A **policy** is composed of one or more rules. Each policy has associated metadata that enables the service layer to choose the most appropriate policy for the current item in question.

A **rule** is configured to ask a certain question about the current item or customer in question. Each rule references several possible rule actions.

A **rule action** has two distinct functions:

- Identify an answer to a question. Each rule action corresponds to a value returned from the rule's question.

- Tell the analytic engine (for example, the service layer) what action to take in response to this answer, for example, whether to continue evaluation and determine what response to return to the client.

Functional Assumptions

- Though Oracle Retail Returns Management needs to be informed of returns performed, it is not a requirement that the point of return itself informs Oracle Retail Returns Management. This means that this information can be conveyed by a separate process, such as a scheduled transaction parsing routine. This also means that the point of return does not need to have direct access to the process final result API of Oracle Retail Returns Management.
- The historical data read by Oracle Retail Returns Management at the beginning of a return request is the data that is updated by the return information delivered after a successful return.
- The historical data recorded by Oracle Retail Returns Management is not the same as purely transactional data, for example, POSLog. The historical data is data that reflects inferred customer behavior, such as too many returns over a specified amount of time.
- The transactional data is read by the point of return, not by Oracle Retail Returns Management.

Functional Overviews

Return Tickets Functional Overview

Return tickets enable an operator to inquire about the particulars of a specific return approval or denial. A return ticket is any attempt by a customer to return one or more items, from one or more originating transactions or from no identifiable transaction. The return ticket carries a unique identifier that can consist of the store number and workstation ID from which the return attempt occurs, an eight-digit date in *MMDD-YYYY* format, and sequence number. The operator can search for a return ticket by the unique identifier or other information, such as cashier, customer, or item information.

The operator can be a loss prevention operator researching potentially fraudulent return activity, or a customer service person researching why a particular customer's return was denied.

Exception Files Functional Overview

The Oracle Retail Returns Management exception file is created and maintained by Oracle Retail Returns Management for use in detecting and preventing fraud at the point of return. The exception file acts as a constantly evolving knowledge base that can help the authorization engine decide which customers or cashiers are at higher risk for return fraud.

The exception file holds an exception counter for a customer; the exception counter is incremented based on suspicious return activity. If a return activity is selected for inclusion in the exception counter, the system increments the exception count for each suspicious shopping activity. Likewise, return activities can be configured for cashiers.

Exception counts are based on real-time refund attempt activities occurring at the point of return, using the return result message that is sent by the point of return to

Oracle Retail Returns Management at the conclusion of a transaction with an attempted refund. Return activities include activities that increment counters such as a return transaction by the customer without a receipt and with no retrieval of the original transaction, five same-day returns as purchases within the last three days, and three returns today. In turn, normal activity levels might be exceeded and counting generated based on those counters.

Messages And Responses Functional Overview

The message and response component of Oracle Retail Returns Management includes the messages sent from the point of return that might trigger action in Oracle Retail Returns Management and an appropriate response message. Oracle Retail Returns Management communicates with brick and mortar, e-commerce, and call center point of return environments using a messaging interface to do the following:

- Receive return authorization requests, use retailer-defined return policies to determine authorization or denial of items and valid return tenders, and respond with the applicable approval or denial code.

Policies And Exceptions Functional Overview

A return policy consists of multiple rules that ask a question about an attempted return. The retailer sets the order in which the rules are evaluated upon a return. The retailer determines the action to take based on the answer to the question. The action taken based on the answer to the question is:

- Continue
- Continue At Rule Number
- Stop Processing

Exceptions are instances of a behavior that a retailer has selected to track for a customer or cashier. Exceptions hold an exception counter for a customer or cashier that is incremented based on return activity. If an exception (behavior) is selected for inclusion in the exception counter, the system increments the exception count for each return activity. An exception is any activity that can be discerned from return ticket data, such as a non receipted return, a return of an item contained in the Return Pattern Watch file, or a particular type of refund transaction such as a price adjustment. When an exception occurs, a record is written to the exception file and the activity is available for research on that customer or cashier using the exception inquiry search and display screens. All exceptions are based on return ticket data.

Analytic Engine Functional Overview

Configuration

A return policy consists of multiple rules that ask a question about an attempted return, such as some of the following:

- Does the customer have a receipt?
- Is the item serialized?
- Does the serial number on the item being returned match the serial number of the item as originally purchased?
- What is the customer's cumulative exception count?
- What is the condition of the item?

The retailer sets the order in which the rules are evaluated upon a return.

The retailer determines through the front end the action to take based on the answers to the questions.

The answer can be **Yes** or **No** (Boolean), a certain numeric or currency number (Range), or one possible response from a valid list of responses (Discrete), based on the type of question being asked. For example, "Does the customer have a receipt?" has a **Yes** or **No** response. "What is the customer's cumulative exception count?" has a numeric response that would fall within a range configured by the retailer. "What is the condition of the item?" maps to a response chosen by the point of return operator, such as one of the following:

- Excellent
- Good
- Fair
- Poor
- Open Box
- Damaged
- Used

The analytic engine uses one of these items to decide returnability.

The action taken based on the answer to the question is one of the following:

- Continue — Check the next rule within the policy.
- Continue At Rule Number — Check a particular rule within the policy and then continue.
- Stop Processing — Do not continue checking rules. Processing complete.

Response Codes

The retailer can set a configurable response code to be returned with every action. Response codes consist of a required positive numeric code, response type, response priority within that type, short description, and an optional long description that can be used for scripting customer service responses to customer inquiries. The response type for each response code is selected from the following, which are listed in priority order:

- Denial.
- Manager Overridden Denial — The engine has denied the item but the denial can be overridden at the point of return by a properly authorized user.
- Contingent Authorization — The engine has approved the item contingent upon capture of an override at the point of return by a properly authorized user.
- Authorization.

Response codes are prioritized within response types. No two response codes of the same response type can have the same priority. As the analytic engine evaluates policy rules, the system holds the highest priority response code within that response type as the response, until a rule resulting in a higher response type, with a higher priority, supersedes it, thus the retailer can control whether the most favorable or least favorable response is returned to the point of return.

Tender Determination

The retailer also determines the tenders that are enabled for a return. When the response is Continue or Continue At Rule Number, the tenders set for the rule carry forward until they are superseded by the response to a following rule. If there is no following rule that must be evaluated, then the tenders collected as a response to that rule are the available tenders that are returned to the point of return in the response message.

Collection of Customer Demographics

An indicator can be set on a policy rule response that indicates positive ID is required in order to check this rule, and the policy cannot be evaluated unless the positive ID is obtained and the exception file checked. In this case, an additional call to Oracle Retail Returns Management is made, for another evaluation once customer positive ID is obtained.

Determination of the Policy for Use on a Return Attempt

The collection of rules (policy) is assigned to a combination of location (node of the store reporting hierarchy, ad hoc store groups, or individual stores) and items, which can be designated by item or merchandise hierarchy. When a return is attempted at a point of return, the system determines the appropriate policy to apply based on the item being returned and the store where the return is being performed. The item designation supersedes the store designation in the case where two policies might otherwise be equivalent.

Two default policies must be defined for the analytic engine to use:

- Receipted items
- Non receipted items

Exception policies can then be set to cover specific items, such as serialized items that include warranties, or articles of clothing that cannot be returned under any condition. When the system does not find a policy applying specifically to the line item being returned, the system falls back to the appropriate default receipted or non receipted policy to evaluate returnability.

When the returnability has been determined based on the appropriate policy, the system checks for any other items that the customer is attempting to return at that time. When the responses have been determined for all items in the attempted return, the system sends the return response message with the evaluation results for the attempted return. Response for an attempted line item return includes a response code and description that are determined by the retailer.

The point of return can then use the response information to control flow to complete the return, such as prompting for a manager override, presenting the enabled tenders, or displaying information for why a return is not allowed.

Penalty Box Concept

When a customer exceeds a particular threshold that can be configured on appropriate rules, the customer can be placed in the Penalty Box for the particular rule. Being placed in the penalty box indicates that the behavior has reached a point of violation, and cannot be tolerated again for a specified period of time. The penalty box is a separate threshold, usually set above the normal threshold used for counting, with its own specific duration (number of days).

The length of time spent in the penalty box can be configured for each rule. The system process that calculates exceptions for rule determination also determines those

entities that are new to the penalty box, or emerging from the penalty box. The system captures the customer positive IDs that go into the penalty box and the exceptions for which they enter the penalty box so that inquiries can be performed.

Customer Service Overrides

Customer service overrides are granted to a customer using the Customer Exception Details screen. The presence of a customer service override for a particular positive ID is checked at the end of return engine evaluation if any line item evaluates to a Contingent Authorization, Manager Overridden Denial, or Denial. Customer service overrides are associated and used with a return ticket. If the return ticket is subsequently voided, the customer service override is considered unused and might be used with a subsequent return authorization. If more than one customer service override exists, the system applies them to the return in order from oldest to newest, by date.

Customer service overrides can consist of more than one allowed return within an override. The Max Customer Service Overrides parameter limits the number of allowed returns within the override and the total number of overrides granted to a customer.

Integration Methods And Communication Flow

The main integration point of Oracle Retail Returns Management is with an external point of return. To communicate between the systems, Oracle Retail Returns Management provides methods which accept and return messages in a predefined format. This chapter discusses the methods and the messages. This section also discusses some of the implications of the chosen implementations.

Methods Of Contact

Oracle Retail Returns Management has two primary methods of contact with the point of return:

- The point of return requests return authorization from Oracle Retail Returns Management (evaluation).
- The point of return notifies Oracle Retail Returns Management of what was actually returned (exception tracking).

Both of these methods use XML messages. The call to evaluation is a synchronous call that returns a separate XML message. The call to scoring is an asynchronous call.

Oracle Retail Returns Management Messages

The three messages defined by Oracle Retail Returns Management are:

- Return Request
- Return Response
- Return Result

The return request is passed from the point of return to Oracle Retail Returns Management when evaluation is invoked. The Return Response is returned by Oracle Retail Returns Management to show the result of evaluation. The return result is passed by the point of return to Oracle Retail Returns Management to initiate scoring.

This chapter describes the integration of Oracle Retail Returns Management with an external point of return, using an example transaction and sample XML messages that are sent between the point of return and Oracle Retail Returns Management. To more clearly illustrate the XML messages, this chapter provides a scenario of a customer returning items under different situations. Each situation has a sample of the XML message with details around each element in the XML. There is a sample XML file for each of the three basic messages:

- Return Request

- Return Response
- Return Result

Each of these messages has a corresponding XSD that defines the valid XML for each message.

Sample XML For Return Transaction Scenarios

John Smith wants to return some sporting equipment he has purchased. We will examine the message sent from the point of return to Oracle Retail Returns Management when he firsts wants to return the items. Oracle Retail Returns Management will respond asking for positive ID. A second return request is made from the point of return to Oracle Retail Returns Management with the additional ID information. Oracle Retail Returns Management then responds with its decision.

The customer decides he wants to return the items. Then, for whatever reason, the return is voided. Finally, the customer decides to re-return the items when Oracle Retail Returns Management is offline.

All XSD's referenced are provided in the Oracle Retail Returns Management installation material:

Table 5–1 XSD Locations

Document Name	Location
Return Request	returnsMgmtInstall.jar/returnsMessaging.zip/CO-RM-ReturnRequest.xsd
Return Response	returnsMgmtInstall.jar/returnsMessaging.zip/CO-RM-ReturnResponse.xsd
Return Result	returnsMgmtInstall.jar/returnsMessaging.zip/CO-RM-ReturnResult.xsd

Point Of Return To Returns Management — Initial Return Request

In this scenario, John Smith has decided he wants to return some baseballs. He goes to the point of return which emits the following message:

Example 5–1 Initial Return Request

```
<ReturnRequest>
  <itemReturnInfo>
    <itemTransactionInfo>
      <receipted>true</receipted>
      <transactionID>
        <storeID>12345</storeID>
        <workstationID>124</workstationID>
        <sequenceNumber>2</sequenceNumber>
        <businessDate>2005-12-31</businessDate>
      </transactionID>
      <found>true</found>
      <validAtPointOfReturn>true</validAtPointOfReturn>
      <giftReceipt>false</giftReceipt>
      <purchaseDate>2005-12-31</purchaseDate>
      <deliveryDate>2006-01-01</deliveryDate>
      <validationAmount>40.00</validationAmount>
      <originalTender>
        <tenderType>
          <type>CASH</type>
        </tenderType>
      </originalTender>
    </itemTransactionInfo>
  </itemReturnInfo>
</ReturnRequest>
```



```

        <amount>12.00</amount>
      </tenderType>
      <tenderType>
        <type>CRDT</type>
        <amount>38.00</amount>
        <cardNumber>1234567812345678</cardNumber>
      </tenderType>
    </originalTender>
    <saleQuantity>10</saleQuantity>
  </itemTransactionInfo>
  <itemIdentifier>
    <itemID>40020002</itemID>
    <itemDescription>MLB Baseball</itemDescription>
    <itemType>Sporting Good</itemType>
  </itemIdentifier>
  <returnReason>Customer Satisfaction</returnReason>
  <quantity>10.00</quantity>
  <amountPaidPerUnit>4.00</amountPaidPerUnit>
  <requestedAdjustedPrice>4.00</requestedAdjustedPrice>
  <itemCondition>New</itemCondition>
  <manuallyEntered>false</manuallyEntered>
</itemReturnInfo>
<returnStoreID>04241</returnStoreID>
<returnWorkstationID>123</returnWorkstationID>
<employeeID>20051</employeeID>
<customerInfo>
  <customerID>80012</customerID>
</customerInfo>
  <transactionType>Return</transactionType>
</ReturnRequest>

```

The entire request has a root element of **<ReturnRequest>**.

The following sub- elements, unless specified otherwise, are of type String and are required.

<itemReturnInfo> complex type

Each return request is based around returning a discrete number of items. Each different type of item has a corresponding itemReturnInfo element. This means that if a customer is returning ten baseballs and two bats, there are two itemReturnInfo elements, not twelve.

<itemTransactionInfo> complex type, sub-element of <itemReturnInfo>

This complex type describes the transaction during which the item was originally purchased.

<receipted> boolean, sub-element of <itemTransactionInfo>

This boolean element tells Oracle Retail Returns Management whether the customer has a receipt for this item. Non-receipted returns are allowed, but can trigger a different return policy.

<transactionID>, optional, complex type, sub-element of <itemTransactionInfo>

This complex type identifies the ARTS-compliant transaction of the original purchase, if any.

<storeID>, <workstationID>, <sequenceNumber>, <businessDate>, sub-elements of <transactionID>

These elements correspond to the parts of the ARTS-compliant transaction ID.

Note:

- The sequence number is a positive integer and the business date is a date.
 - The store ID and workstation ID of these elements do not need to match the store ID and workstation ID of <returnStoreID> and <returnWorkstationID>. Therefore, the item can be purchased at one store and returned at another.
-

<found> boolean, sub-element of <itemTransactionInfo>

This boolean element tells Oracle Retail Returns Management if the transaction ID from the <transactionID> element was found. This element exists because it is possible to have a transaction number, for example, from a receipt, that is not found by the point of return when it queries existing transaction data. This element is required, but is only relevant if there is a transaction ID. If there is no <transactionID> element, this value should be set to **false**.

<validAtPointOfReturn> boolean, sub-element of <itemTransactionInfo>

This boolean element tells Oracle Retail Returns Management if the transaction ID from the <transactionID> element is considered valid by the point of return. Any transaction ID that is found should set this value to **true**. If the ID is not found, the point of return should decide if the transaction ID appears to be legitimate and set this value accordingly.

<giftReceipt> optional, boolean, sub-element of <itemTransactionInfo>

This boolean element tells Oracle Retail Returns Management if the receipt presented at the point of return is a gift receipt. This element should not be included in the message if the <receipted> element is **false**.

<purchaseDate>, <deliveryDate>, optional, date, sub-element of <itemTransactionInfo>

These date elements refer to the purchase and delivery dates of the item being returned, respectively. If this data cannot be determined at the point of return, these elements should be omitted.

<validationAmount>, optional, decimal, sub-element of <itemTransactionInfo>

This optional element represents the dollar amount of the items being returned. This is only included when there is no found transaction but there is a valid transaction ID. This could happen if, for example, a customer has a receipt with a transaction number and amount on it, but the point of return cannot find the transaction in storage.

<originalTender>, complex type, sub-element of <itemTransactionInfo>

This complex type represents the original tenders used to purchase these items. Though this type is required, the list of original tenders can be empty, for example, for a non-receipted, non-transaction ID return.

<tenderType>, optional, complex type, sub-element of <originalTender>

For each original tender that the point of return knows about, there is a tender type entry.

Note: There might be no tender type entries (for example, for non-receipted returns), one entry, or many entries (for a split-tender scenario, such as an item that was bought partially with a gift card and partially with cash).

<type>, <amount>, <cardNumber>, sub-elements of <tenderType>

These three elements describe the tender used. The <type> element is the only required element and is expected to match the standard four letter Oracle Retail tender types, for example, CASH. The <amount> element is an optional decimal value. The <cardNumber> element is listed as optional, but should be filled in with the appropriate card number if the tender type is CRDT.

<saleQuantity>, optional, decimal, sub-element of <itemTransactionInfo>

This element represents the original quantity of items sold to the customer. A customer might want to return only one out of ten items they have bought. This original sale quantity is compared to previous returns Oracle Retail Returns Management knows about. If the sum of items from previous returns plus the items from this return is greater than the sale quantity, Oracle Retail Returns Management can flag and deny this return attempt.

<itemIdentifier> sub-element of <itemReturnInfo>

This complex type identifies which item is being returned.

<itemID>, <itemDescription> sub-elements of <itemIdentifier>

Oracle Retail Returns Management relies on the <itemID> when referring to an item that it can look up in the AS_ITM table of the Oracle Retail data model. Though the XSD enables <itemDescription> to be included, it is unused by the code.

<itemType> optional, sub-element of <itemIdentifier>

The <itemType> element describes the type of the item as evaluated by ItemTypeEvaluator. Like the <returnReason> element, the point of return and Oracle Retail Returns Management must agree on valid values for this element. The user interface uses the ItemTypes parameter.

<returnReason>, sub-element of <itemReturnInfo>

The reason for which the item is being returned. This required element is used by the class ReturnReasonEvaluator. Based on the text provided here, the evaluator can choose various responses during policy initiation. The user interface uses the ReturnReasons parameter.

Note: This is free-form text. It is up to the client and the server to agree on this message.

<quantity>, decimal, sub-element of <itemReturnInfo>

This is the quantity being returned. Non-unitary units of measure, for example, feet, should be expressed in a decimal format, such as 1.5 feet for 18 inches.

<amountPaidPerUnit>, decimal, optional, sub-element of <itemReturnInfo>

The amount paid per item on this return.

<serialNumber>, optional, sub-element of <itemReturnInfo>

This element is currently unused by Oracle Retail Returns Management.

<requestedAdjustedPrice>, optional, decimal, sub-element of <itemReturnInfo>

This element is set to the price at which the point of return wants to return the item. For instance, the point of return might request to return the item for less than the original sales price. This value is compared to the original price in PriceAdjustmentAmountEvaluator class. That rule initiates different actions depending on the ratio of the adjusted price to the original price per unit.

<itemCondition>, optional, sub-element of <itemReturnInfo>

This element reflects the condition of the item. This value is used by the ItemConditionEvaluator class. Like the <returnReason> element, the legal values for this element need to be agreed upon by the point of return and Oracle Retail Returns Management. The user interface uses the ItemConditions parameter.

<manuallyEntered>, boolean, sub-element of <itemReturnInfo>

This required boolean element denotes if the information in the <transactionID> element was manually entered at the point of return. This element should be **false** if there is no transaction ID.

<returnStoreID>, <returnWorkstationID>, <employeeID>

These are the IDs of the store, workstation, and employee that are initiating the return, respectively. The employee ID is used for tracking cashier exceptions and is expected to correspond to an entry into the Oracle Retail employee table.

<customerInfo>, <moreCustomerInfo>

These complex element types represent information about the customer returning the items.

Note: Only one of these info types can be present in the return request.

<customerID>, <customerType> optional, sub-elements of <customerInfo>

The <customerID> element corresponds to the Oracle Retail customer ID.

Note: This element is different than the Returns Customer ID in the Oracle Retail Returns Management customer table, which is keyed off of positive ID.

The <customerType> element is unused and is not displayed in the sample XML.

<transactionType>

The <transactionType> element corresponds to the type of return requested by the point of return. Valid values are defined by the parameter RefundTypes. This parameter is defined in returnsmgmt.xml. Default values are:

- Return
- Layaway_Cancellation
- Order_Cancellation
- Price_Adjustment

Returns Management To Point Of Return — Return Response: Need Positive ID

After the initial return request has been submitted, Oracle Retail Returns Management determines that it needs a positive ID from the customer. Oracle Retail Returns Management responds, indicating that the point of return should obtain the ID from Mr. Smith.

Example 5–2 Return Response Requesting Positive ID

```
<ReturnResponse>
  <returnTicketID>04241-123-1025-2006-005021791</returnTicketID>
```

```

<responseApproveDenyCode>Denial</responseApproveDenyCode>
<languagePreference>en</languagePreference>
<itemReturnResponse>
  <itemIdentifier>
    <itemID>40020002</itemID>
  </itemIdentifier>
  <responseCode>10</responseCode>
  <approveDenyCode>Denial</approveDenyCode>
  <responseDescription>Insufficient quantity</responseDescription>
  <refundTenders/>
  <customerInfoRequired>true</customerInfoRequired>
</itemReturnResponse>
</ReturnResponse>

```

The entire request has a root element of **<ReturnResponse>**.

The following sub- elements, unless specified otherwise, are of type String and are required.

<returnTicketID>

This element refers to the return ticket created by Oracle Retail Returns Management. The point of return needs to use this element in future communications with Oracle Retail Returns Management about this return (for example, in response to a request for positive ID or when sending a final result).

<responseApproveDenyCode>

If there are multiple item responses, then the most cautious <approveDenyCode> value is used.

There are exceptions to the behavior of this element. If a current entry is found in the customer service override table (RM_CT_SV_ORD) that matches this Oracle Retail Returns Management customer and is active for the same date as the return, and the response would be a denial, then this value is set to **Approved** and the optional <availableCustomerServiceOverride> element is set.

<languagePreference>

This element informs the client in which language the <responseDescription>, <receiptMessage>, and other elements are sent. Currently this is always **en** for English.

<itemReturnResponse>, complex type

This element contains the detailed information about each of the items to which Oracle Retail Returns Management is responding. There can be many of these elements in a transaction.

<itemIdentifier>, complex type, sub-element of <itemReturnResponse>

This complex type identifies which item is being returned.

Note: Oracle Retail Returns Management sets the <itemID> sub-element only.

<responseCode>, <approveDenyCode>, <responseDescription> sub-elements of <itemReturnResponse>

Oracle Retail Returns Management has a response associated with each item. These response codes are configured by the rule actions of the policy that Oracle Retail Returns Management chose to execute. The codes are contained in the table RM_RSPS_RC.

The three values here correspond to the ID_RPSS_RC, TY_RSPSP, and DE_RSPSP respectively. The <responseCode> element itself is an integer that corresponds to an ID of a response code. The <approveDenyCode> is one of **Denial**, **Mgr Overridable Denial**, **Contingent Authorization**, or **Authorization**. The <responseDescription> is the short description of the response. Though these fields are required, in the message they can be ignored by the point of return since the <customerInfoRequired> element is **true**.

<customerInfoRequired>, optional, boolean, sub-element of <itemReturnResponse>

If this value is set to **true**, Oracle Retail Returns Management is asking the point of return to prompt for positive ID. If this value is not present the point of return assumes that it is **false**.

Point Of Return To Returns Management — Second Return Request

Once the point of return has gotten a positive ID for Mr. Smith, it returns that information to Oracle Retail Returns Management along with the data from the original return request.

Note: The <itemreturnInfo> content is the same as in Example 5-1 and has been left out for brevity.

Example 5-3 Second Return Request

```
<ReturnRequest>
  <itemReturnInfo>
    ...
  </itemReturnInfo>
  <returnStoreID>04241</returnStoreID>
  <returnWorkstationID>123</returnWorkstationID>
  <employeeID>20051</employeeID>
  <moreCustomerInfo>
    <lastName>Smith</lastName>
    <firstName>Carlos</firstName>
    <middleName>Juan</middleName>
    <gender>Male</gender>
    <birthDate>1972-06-25</birthDate>
    <address1>1234 Example Blvd</address1>
    <address2/>
    <city>Miami</city>
    <state>FL</state>
    <postalCode>33056</postalCode>
    <country>United States</country>
    <countryCode>US</countryCode>
    <telephoneAreaCode>888</telephoneAreaCode>
    <telephoneLocalNumber>5551212</telephoneLocalNumber>
  </moreCustomerInfo>
  <positiveID>
    <number>12345678</number>
    <type>DriversLicense</type>
    <issuer>US_MN</issuer>
    <issued>2004-01-01</issued>
    <expiration>2007-01-01</expiration>
  </positiveID>
  <transactionType>Return</transactionType>
  <returnTicketID>04241-123-1025-2006-005021791</returnTicketID>
</ReturnRequest>
```

The entire request has a root element of **<ReturnRequest>**.

Oracle Retail Returns Management works with two different customer data stores. The first data store is the Oracle Retail Returns Management customer data store, which is keyed off of positive IDs. This is the set of customers used for exception tracking. The second data store is the standard Oracle Retail customer data store. Oracle Retail Returns Management will attempt to link customers for which it has a positive ID to customers in the Oracle Retail customer data store, creating customers if necessary.

For purposes of the Oracle Retail Returns Management customer, only the **<positiveID>** element is relevant. Oracle Retail Returns Management either looks up or creates the customer corresponding to the positive ID. For the Oracle Retail customer, there are two elements which matter: **<customerID>** (underneath **<customerInfo>**) and **<moreCustomerInfo>**. If the **<customerID>** is passed in, Oracle Retail Returns Management assumes that this is the Oracle Retail customer relevant to the message. If the **<customerID>** element is absent, and both the **<positiveID>** and the **<moreCustomerInfo>** elements are present, then Oracle Retail Returns Management not only looks up or creates the Oracle Retail Returns Management customer, but it also creates a new Oracle Retail customer and associates it with the Oracle Retail Returns Management customer.

These examples are contrived to display the **<customerInfo>** and **<moreCustomerInfo>** elements. In the first message, the point of return had a valid customer ID. In this message, the XML is constructed to imply that Oracle Retail Returns Management should create a new customer matching the positive ID. In a real usage scenario, if the point of return knew the Oracle Retail customer but just needed to collect positive ID, the point of return would send the **<customerInfo>** element again rather than the **<moreCustomerInfo>** element.

<moreCustomerInfo>, optional, complex type

This element contains the information necessary to create an Oracle Retail customer.

<lastName>, <firstName>, sub-elements of <moreCustomerInfo>

These elements contain the last and first names (respectively) of the new Oracle Retail customer.

<middleName>, <gender>, <birthDate>, optional, sub-elements of <moreCustomerInfo>

These optional elements contain the middle name, the gender, and the birth date of the new Oracle Retail customer. Notice that these elements are all strings, including birth date. Also notice that the gender element is constrained to either male or female.

<address1>, <address2> sub-elements of <moreCustomerInfo>

These elements correspond to the usual two address lines of a customer. In this example, though **<address2>** is present, it is blank.

<city>, <state>, <postalCode>, sub-elements of <moreCustomerInfo>

These elements are further parts of the customer address. In the US, the state and postal code would correspond to the state and zip code. In Canada, they would correspond to the province and postal code.

<country>, <countryCode>, optional, sub-elements of <moreCustomerInfo>

These elements correspond to the country in which the customer resides. The country code is a two-letter ISO-compliant country code.

<telephoneAreaCode>, <telephoneLocalNumber>, optional, sub-elements of

<moreCustomerInfo>

These two optional elements reflect the telephone number and area code. For a number such as "888-555-1212", the "888" would be in the <telephoneAreaCode> element while the "5551212" would be in the in the <telephoneLocalNumber> element.

<positiveID>, complex type

Positive ID refers to a customer presenting credentials (such as a driver's license) to authenticate their identity. The positive ID is used to reference the Oracle Retail Returns Management customer. This element encodes information about the type of positive ID gathered by the point of return. See above for a discussion of the Oracle Retail Returns Management customer versus the Oracle Retail customer.

<number>, sub-element of <positiveID>

The unique identifier on the positive ID.

<type>, sub-element of <positiveID>

The type of identification presented. Valid types are DriversLicense, MilitaryID, Passport, and StateCard.

<issuer>, sub-element of <positiveID>

This is the issuing authority of the identification. For StateCard, this is the state which issued the card. For Passport, this is the country which issued the passport.

<issued>, <expiration>, optional, date, sub-elements of <positiveID>

These two optional date elements reflect the date of issue and the date of expiration, respectively, of the positive ID.

<transactionType>

This element is the same as the element in the initial return request.

<returnTicketID>, optional

By setting the <returnTicketID> element, the point of return lets Oracle Retail Returns Management know that it is responding to a request for more information. The element should be set to ticket ID sent from Oracle Retail Returns Management in the corresponding return result. In this case, the element has been set to 04241-123-1025-2006-005021791 to match our previous message.

Returns Management To Point Of Return — Second Return Response

Now that Oracle Retail Returns Management has obtained a positive ID, it can tell the point of return about its decision of whether to allow the return of the items.

In this scenario, Mr. Smith has been returning a lot of items lately and has had an entry put into the exception file. This would normally result in a denial. However, Mr. Smith's agent has called the customer service center and asked them to accept the return. They have entered an entry into the customer service override table for Mr. Smith. Oracle Retail Returns Management checks for these entries while it creates the final result. In this case, since it has found one, Oracle Retail Returns Management authorizes the return but marks it as using a customer override.

Example 5-4 Second Return Response

```
<ReturnResponse>
  <returnTicketID>04241-123-1025-2006-016085229</returnTicketID>
  <responseApproveDenyCode>Authorization</responseApproveDenyCode>
  <availableCustomerServiceOverride>true</availableCustomerServiceOverride>
  <receiptMessageNumber>1</receiptMessageNumber>
```



```

<receiptMessageDescription>Thank you for shopping!</receiptMessageDescription>
<languagePreference>en</languagePreference>
<itemReturnResponse>
  <itemIdentifier>
    <itemID>40020002</itemID>
  </itemIdentifier>
  <responseCode>150</responseCode>
  <approveDenyCode>Mgr Overridable Denial</approveDenyCode>
  <responseDescription>Exception file match</responseDescription>
  <receiptMessageNumber>1</receiptMessageNumber>
  <receiptMessageDescription>Thank you for
shopping!</receiptMessageDescription>
  <refundTenders/>
</itemReturnResponse>
</ReturnResponse>

```

Note: The return ticket ID in this response is different from the one sent in the first response. Each new return request generates a new return ticket ID.

<availableCustomerServiceOverride>, optional, boolean

This optional boolean element is only set when:

- The overall approve or denial code is a denial.
- This Oracle Retail Returns Management customer has an active entry in the customer service override table.

When this item is present, it is always **true**. For details about the approve and deny code process, refer to the “<responseApproveDenyCode>” on page 7 under the initial return response.

<receiptMessageNumber>, positive integer

Oracle Retail Returns Management has a number of receipt messages that it can send back to the point of return. These messages are intended to be printed on the receipt, but obviously the point of return can do what it would like with them. Each message has both a number and a description. The number is provided for internationalization purposes and the message is provided to make the XML more human readable.

Note that there is both a receipt message associated with both the overall return response as well as with each item on the response. The overall message is determined in the same manner as the overall response code. That is, the most cautious individual response code determines both the overall response as well as the overall receipt message.

<receiptMessageDescription>

This is the text associated with the receipt message number. This text will be in the language of the <languagePreference> element. Currently, this text is always in English.

<languagePreference>

This is the same element as detailed in the initial return response.

<itemReturnResponse>, complex type

This is the same element as returned in the first return response. In this case, the sub-elements of this complex type contain detailed information about the decision of Oracle Retail Returns Management regarding the returnability of this item.

<approvedQuantity>, decimal, optional sub-element of <itemReturnResponse>

This element is currently unused.

<receiptMessageNumber>, sub-element of <itemReturnResponse>

This is the receipt message number associated with the individual item. See "[<receiptMessageNumber>, positive integer](#)" on previous page.

<itemDispositionCode>, optional, sub-element of <itemReturnResponse>

This element corresponds to the disposition of the item after it has been returned, for example, "keep frozen". It corresponds to the table in ID_DPSN_CD in the ARTS schema. However, this element is currently not set.

<receiptMessageDescription>, sub-element of <itemReturnResponse>

This is the receipt message text associated with the individual item. See "[<receiptMessageDescription>](#)" on previous page.

<restockingFee>, optional, decimal, sub-element of <itemReturnResponse>

This element is currently unused.

Point Of Return To Returns Management — Return Result From Second Response

Once the positive ID has been collected and Oracle Retail Returns Management has told the point of return about the returnability of the item, the point of return processes the return as necessary. Once the return has been completed, the point of return sends Oracle Retail Returns Management a return result message.

Example 5-5 Return Result

```
<ReturnResult>
  <returnTicketID>04241-123-1025-2006-016085229</returnTicketID>
  <returnTransactionID>
    <storeID>04241</storeID>
    <workstationID>123</workstationID>
    <sequenceNumber>250</sequenceNumber>
    <businessDate>2006-10-25</businessDate>
  </returnTransactionID>
  <itemReturnResult>
    <itemIdentifier>
      <itemID>40020002</itemID>
    </itemIdentifier>
    <quantityReturned>10</quantityReturned>
    <finalResultCode>Authorized</finalResultCode>
    <overrideInfo>
      <managerID>20008</managerID>
      <overrideObtained>true</overrideObtained>
      <tenderOverride>false</tenderOverride>
    </overrideInfo>
    <originalTransactionID>
      <storeID>12345</storeID>
      <workstationID>124</workstationID>
      <sequenceNumber>2</sequenceNumber>
      <businessDate>2005-12-31</businessDate>
    </originalTransactionID>
    <returnTender>
      <tenderType>
        <type>CASH</type>
        <amount>40.00</amount>
      </tenderType>
    </returnTender>
  </itemReturnResult>
</ReturnResult>
```

```

    </itemReturnResult>
  </ReturnResult>

```

Once again, note the return ticket ID. It is the ticket ID referring to the second return response.

<returnTransactionID>, optional, complex type

This element refers to the ARTS-compliant return transaction generated by the point of return. It has the same format as the <transactionID> element of the return request.

<itemReturnResult>, complex type

This complex element represents detailed information about each item returned.

<quantityReturned>, sub-element of <itemReturnResult>

This number reflects the quantity actually returned by the point of return.

<finalResultCode>, sub-element of <itemReturnResult>

This element has one of two values, **Authorized** or **Denial**. For items that are returned, the value is set to **Authorized**.

<overrideInfo>, optional, complex type, sub-element of <itemReturnResult>

This complex type is included in the response if the point of return decided to override a return decision rendered either by Oracle Retail Returns Management or locally.

<managerID>, sub-element of <overrideInfo>

The ID (from the ARTS-compliant table PA_EM) that corresponds to the employee ID of the manager who overrode the return decision.

<overrideObtained>, boolean, sub-element of <overrideInfo>

This element is set to **true** if the override was about the returnability of the item.

<tenderOverride>, boolean, sub-element of <overrideInfo>

This element is set to **true** if the override was about which tenders to return money on.

<originalTransactionID>, optional, complex type, sub-element of <itemReturnResult>

This element refers to the ARTS-compliant transaction associated with the original sale. This element has the same format as the <transactionID> element of the return request. This element is currently unused by Oracle Retail Returns Management.

<returnTender>, complex type

This element is a list of tenders. It describes the number of tenders, and the amount of each one, used by the point of return to return money to the client. It has the same format as the <originalTender> element of the return request.

Point Of Return To Returns Management — Void Return

Mr. Smith has proven to be indecisive and decides that he wants to have his baseballs after all. He wants to void the return and receive the items back. To accommodate this, the point of return voids the previous return and informs Oracle Retail Returns Management of the fact.

Example 5–6 Void Return Result

```

<ReturnResult>
  <returnTicketID>04241-123-1025-2006-016085229</returnTicketID>
  <returnTransactionID>
    <storeID>04241</storeID>
    <workstationID>123</workstationID>
  </returnTransactionID>
</ReturnResult>

```

```
<sequenceNumber>250</sequenceNumber>
<businessDate>2006-10-25</businessDate>
</returnTransactionID>
<returnVoided>true</returnVoided>
</ReturnResult>
```

In this message, we see the same return ticket ID and the same return transaction ID of the return response above. This is used to let Oracle Retail Returns Management know which return is being voided. The only new element is the `<returnVoided>` element.

`<returnVoided>`, optional, boolean

This element shows that the return referenced by the `<returnTicketID>` element has been voided. Though this element is optional, exactly one of either `<returnVoided>` or `<itemReturnResult>` must appear in the return result. Thus, when this element is present, it must always be **true**.

Offline Return Result

Finally, Mr. Smith decides that he does, in fact, want to return the baseballs. When he returns to do so, the point of return is offline from Oracle Retail Returns Management. The point of return makes its own decisions about the returnability of the items. For the sake of illustration, the point of return makes exactly the same decisions that Oracle Retail Returns Management did previously, though there is no requirement for this.

Note: The `<itemreturnInfo>` content is the same as in Example 5-1 and has been left out for brevity.

Example 5-7 Offline Return Result

```
<ReturnResult>
  <offlineDate>2006-05-16</offlineDate>
  <offlineRequest>
    <itemReturnInfo>
      ...
    </itemReturnInfo>
    <returnStoreID>04241</returnStoreID>
    <returnWorkstationID>123</returnWorkstationID>
    <employeeID>20051</employeeID>
    <customerInfo>
      <customerID>8885551212</customerID>
    </customerInfo>
    <positiveID>
      <number>12345678</number>
      <type>DriversLicense</type>
      <issuer>US_MN</issuer>
      <issued>2004-01-01</issued>
      <expiration>2007-01-01</expiration>
    </positiveID>
    <transactionType>Return</transactionType>
  </offlineRequest>
  <returnTransactionID>
    <storeID>04241</storeID>
    <workstationID>123</workstationID>
    <sequenceNumber>263</sequenceNumber>
    <businessDate>2006-10-25</businessDate>
  </returnTransactionID>
  <itemReturnResult>
    <itemIdentifier>
```

```

        <itemID>40020002</itemID>
      </itemIdentifier>
      <quantityReturned>10</quantityReturned>
      <finalResultCode>Authorized</finalResultCode>
      <overrideInfo>
        <managerID>20008</managerID>
        <overrideObtained>true</overrideObtained>
        <tenderOverride>false</tenderOverride>
      </overrideInfo>
      <originalTransactionID>
        <storeID>12345</storeID>
        <workstationID>124</workstationID>
        <sequenceNumber>2</sequenceNumber>
        <businessDate>2005-12-31</businessDate>
      </originalTransactionID>
      <returnTender>
        <tenderType>
          <type>CASH</type>
          <amount>40.00</amount>
        </tenderType>
      </returnTender>
    </itemReturnResult>
  </ReturnResult>

```

The offline result is effectively a return result with two additional elements. The first element is the date of the offline return. The second element is a return request message encapsulated in the `<offlineRequest>` element.

<offlineDate>, optional, date

This element is the original date of the offline return. Though this element is optional in the XSD, it is required when processing an offline return.

<offlineRequest>, complex type

The `<offlineRequest>` element is an embedded return request. It has the exact same format as a normal return request message except that the root `<ReturnRequest>` element is replaced by the `<offlineRequest>` element.

Note: The XSD specifies that the result message has either an `<offlineRequest>` element or a `<returnTicketID>` element, but not both.

<returnTransactionID>, optional, complex type

This element refers to the ARTS-compliant return transaction generated by the point of return. It has the same format as the `<transactionID>` element of the return request.

<itemReturnResult>, complex type

This element contains the detailed data about the items returned in this offline request. It has the same format as the original return result described above.

Implementation Decisions

Asynchronous Versus Synchronous Communication

Synchronous communication involves a client sending a message to a server and then pausing while it waits for a response. Asynchronous communication enables a client to send a message to the server and then immediately resume operations.

Synchronous communication is usually more straightforward than asynchronous communication, but increases binding between a client and a server and can degrade concurrency. Asynchronous communication has the opposite problems: it is usually more complicated, but encourages decoupling and throughput.

Additionally, synchronous communication is necessary when a client needs a time-sensitive response to a message.

Oracle Retail Returns Management prefers to use asynchronous communication. This is mainly due to concurrency. When communication is asynchronous, it can be off-loaded onto a lightly loaded machine or scheduled to run at a time when there is relatively little system activity. Also, the communication can be more easily chained, for example, by inserting an arbitrary number of message forwarders between the client and server, or by inserting a message broadcaster. This enables greater flexibility in future system growth. However, asynchronous messaging is poorly suited for real time responses to messages.

Because asynchronous messaging provides greater latitude at installation and higher concurrency, the result message is implemented as an asynchronous call. Evaluation, however, has a strong requirement for a rapid response, for example, a customer is physically waiting at the point of return for a return approval. This evaluation is implemented as a synchronous call.

XML Versus JavaBean Messages

Extensible markup language (XML) is a text format that is language independent and human legible. It has wide support in a variety of programming languages and a robust description language, XML Schema Definition (XSD). Being a text format, XML is capable only of encoding data.

JavaBeans are Java language constructs that mainly encapsulate data in an object class. Being objects, however, JavaBeans can optionally contain behavior as well as data. Also because they are objects, both the originator and the receiver of a JavaBean must have access to compatible versions of the .class file.

JavaBeans are a well known Java idiom and have a great deal of support in the Java Development Kit (JDK). However, they are a Java-specific solution. Furthermore, XML is more accessible to a non-technical audience than either Java source or Java runtime debugging environments. Also, XML is the standard of Web Service communication. Therefore, the messages that Oracle Retail Returns Management passes are XML based rather than JavaBean (or Java Object) based. The XML is eventually transformed into JavaBeans. This transformation to and from JavaBeans is facilitated by Java XML Binding (JAXB) code.

Web Service Versus Enterprise JavaBeans And Remote Method Invocation Call

Web Services are language neutral, similar to XML. Web Services also provide a well-defined publishing and discovery mechanism: Universal Description, Discovery and Integration (UDDI).

Elements

The previous sections have discussed details of the XML messages. This section will provide more information about important elements.

Return Request

In the following table, the marks (x) indicate that the element is required and needs to have an appropriate value for that scenario. A value **true** or **false** indicates that this element should be set to this explicit value.

The **Element** column represents the various elements; the other columns represent the different scenarios. The XPath expressions clarify where the elements are in relationship to the entire message.

Note: In subsequent Positive ID messages, all original scenario elements should be sent.

Table 5–2 Required Elements By Return Request

			Receipted		
Element	Non-receipted	No transaction data	Has data, no transaction found	Has data, transaction found	Positive ID
/ReturnRequest/itemReturnInfo/itemTransactionInfo					
receipted	false	true	true	true	
transactionID			X	X	
found	false	false	false	true	
validAtPointOfReturn	false	false	X	true	
validationAmount			X		
originalTender	X	X	X	X	
/ReturnRequest/itemReturninfo					
itemIdentifier	X	X	X	X	
returnReason	X	X	X	X	
quantity	X	X	X	X	
manuallyEntered	false	false	X	false	
/ReturnRequest					
returnStoreID	X	X	X	X	
returnWorkStationID	X	X	X	X	
employeeID	X	X	X	X	
customerInfo or moreCustomerInfo	X	X	X	X	
positiveID					X
transactionType	X	X	X	X	
returnTicketID					X

Return Response

In the following table, the **Element** column refers to the elements in the return response. The remaining columns describe the various use cases. A mark (x) means that the point of return should be concerned with this data point when encountering it. A **true** or **false** value means that the point of return will examine this data point for this value to determine the use case. The XPath expressions help the reader orient themselves with the elements.

This is the minimum amount of data a point of return needs to implement. Additionally, the point of return must decide how to interpret the approval or denial of the <responseApproveDenyCode> element.

Keep in mind that some items might be approved while others are denied.

Table 5–3 Required Elements By Return Response

Element	Approval or Denial	Positive ID Required	CS Override
/ReturnResponse			
returnTicketID	X	X	X
responseApproveDenyCode	X		X
availableCustomerServiceOverride			true
/ReturnResponse/itemReturnResponse			
itemIdentifier	X	X	X
approveDenyCode	X		
refundTenders	X		X
customerInfoRequired		true	

Return Result

In the following table, the **Element** column is the list of elements the point of return needs to send in the ReturnResult message. A mark (x) indicates that this element should be present when sending this type of response to Oracle Retail Returns Management. A **true** or **false** value indicates that this element should be set explicitly to this value when sending this type of message. The XPath expressions help orient the reader.

Table 5–4 Required Elements By Return Result Use Case

Element	Standard Result	Offline Return	Voided Return
/ReturnResult			
returnticketID	X		X
offlineDate		X	
offlineRequest		X	
returnTransactionID	X	X	X
returnVoided			true
/ReturnResult/itemReturnResult			
itemIdentifier	X	X	
quantityReturned	X	X	

Table 5–4 Required Elements By Return Result Use Case

Element	Standard Result	Offline Return	Voided Return
finalResultCode	X	X	
overrideInfo ¹	X	X	
returnTender	X	X	

¹ The <overrideInfo> element should be included only if there was an override.

Adding A New Rule

Oracle Retail Returns Management is shipped with a wide range of rules already built in. To use these existing rules, it is a simple matter to build policies and add rules using the web user interface. See "Determining Return Policies" in the *Oracle Retail Returns Management User Guide*.

At some point, you might want to develop a custom rule and deploy it into the system. This section discusses the steps necessary to create and deploy a new rule.

This discussion assumes the following:

- Some familiarity with the rules engine
- The rules being added also track a return activity.

Note: Rules that do not track return activities are simpler to implement. These rules require only steps 2, 3, and 6 be performed.

1. Write the new calculator class.

Implement a new instance of `KPICalculatorIfc`. Extend the abstract class `BaseKPICalculator`. The most important methods to implement are `hasMatchingBehavior()` and `calculate()`. The first method is invoked during the result processing to determine if this behavior is present in the return, that is, it returns **true** if the behavior deals with overrides and this return was overridden. The second method returns a number that is relevant to what behavior is being measured. That is, if it is counting the number of times an override occurs.

2. Write a new evaluator class, if necessary.

Depending on the complexity of the calculation, you need to write a new instance of `EvaluatorIfc`. Simple rules can use existing evaluators (such as the `DiscreteRuleEvaluator`, `RangeRuleEvaluator`, or the `KPIRangeEvaluator`). Even if implementing a new evaluator, it is likely that the new class extends one of the existing classes (again, the `DiscreteRuleEvaluator` and the `RangeRuleEvaluator` are the most obvious base classes).

3. Compile and deploy the classes.

This usually means adding it to the ear file.

4. Add a new entry into the KPI template table (RM_KPI).

This entry provides the friendly name of the return activity to the system as well as specifying the calculator class.

5. Create entries for the return activity parameters passed into the system (RM_KPI_PRMR).

Create an entry for each of the parameters that is passed into this return activity. For instance, you might want to have the return activity break on some user-defined threshold. This table lets you specify the name and data type of the parameter.

- 6. Create an entry in the rules table to tie the evaluator and the calculator (RM_RU) together.

This table lets you specify the name of the rule, the type of rule, and the associated evaluator and calculator.

Environment Entries In ejb-jar.xml

This section describes the <env-entry> section in the Oracle Retail Returns Management ejb-jar.xml file. These entries enable manipulation of some aspects of the system.

Return Ticket Formatting Entries

The return ticket table is indexed using a composite key. This key is comprised of store number, workstation, business date, and a sequence number. To make this key end-user legible, it is formatted using the returnTicketIdPattern rather than passed as discrete data elements. The default pattern is *sssss-www-MMdd-yyyy-nnnnnnnnnn*:

- *sssss* marks the store ID.
- *www* marks the workstation ID.
- *nnnnnnnnnn* marks the sequence number
- *MMdd-yyyy* marks the business date

This <env-entry> element must be in sync with the other <env-entry> elements as follows:

- Note the ID divider. If you want to use a different divider, then the value returnTicketIdDivider must be changed to reflect the new divider used.
- Note the date format. If this format is changed, other than the divider character, the value returnTicketIdDatePattern must be changed to reflect this.
- Note the store pattern. If the store retrieved from the database is shorter than the value in returnTicketStoreIdPersistPattern, it is padded on the left hand side with the value in returnTicketPersistPad (default is 0).
- Note the sequence pattern. If the sequence number is smaller than the length of returnTicketSeqNumberPersistPattern, then it is padded with the value from returnTicketIdPad (default is 0).
- returnTicketMaxSequenceValue has no effect.
- returnTicketBusinessDate is used in an unused method in the ReturnTicketKeyFormatter class and can be safely ignored.

Table 5–5 Return Ticket Format <env-entry>

Entry	Default
returnTicketIdPattern	sssss-www-MMdd-yyyy-nnnnnnnnnn
returnTicketIdDivider	-
returnTicketIdDatePattern	MMddyyyy

Table 5–5 Return Ticket Format <env-entry>

Entry	Default
returnTicketStoreIdPersistPattern	sssss
returnTicketWorkstationIdPattern	www
returnTicketSeqNumberPersistPattern	nnnnnnnnnn
returnTicketIdPad	0
returnTicketPersistPad	0
returnTicketMaxSequenceValue	999999999
returnTicketBusinessDate	yyyy-MM-dd

Auditing Entries

Table 5–6 Audit Target <env-entry>

Entry	Default
journalDataPath	1

This integer value tells Oracle Retail Returns Management where to send audit log messages. Valid values are:

- 0 – no audit log
- 1 – send log messages to a JMS queue (found using JNDI lookup at `java:comp/env/jms/JournalingMessage`)
- 2 – send log messages directly to the EJB interface of the Journaling Service.

Any other value results in no audit log being created and an error message logged.

Web Service Interface

The web service exposes two methods. These methods, with their parameters, are as follows:

Table 5–7 Web Service Methods

Method Name	Input	Output
evaluateReturnRequest	String (ReturnRequest)	String (ReturnResponse)
processFinalResult	String (ReturnResult)	None

Note that though the web services expect to produce and consume XML, the XML is passed and returned as a simple string rather than a DOM object.

By default, the web service will be accessed at:

`http://hostname:port/retwebsvc/services/ReturnsManager`

Where *hostname:port* is replaced with the host and port to which the web service .ear file is deployed. The retwebsvc context can be modified by changing the context-root element of the WebModule_Returns_WebServices module in the application.xml of the deployed ear.

Note: Although the web service exposes the `processFinalResult` method, all return results should be sent asynchronously using the JMS queue monitored by the `FinalResultMDB` as a best practice.

Relationship Of Oracle Retail Returns Management Data To ARTS Transaction Data

The Association for Retail Technology Standards (ARTS) is an international membership organization dedicated to reducing the costs of technology through standards. ARTS has four standards:

- The Standard Relational Data Model
- UnifiedPOS
- IXRetail
- Standard RFPs

For more information about ARTS, go to:

<http://www.nrf-arts.org/>

One of the design goals of Oracle Retail Returns Management is to reduce its dependency on external systems. At the same time, customers will need traceability of Oracle Retail Returns Management data back to original transaction data.

To account for this, the return request and return result messages sent to Oracle Retail Returns Management contain the ARTS-compliant transaction IDs for the relevant transactions. Therefore, when a return request is made, Oracle Retail Returns Management is told of the transaction ID, if any, of the original sale. When a return result is sent, Oracle Retail Returns Management is told of the transaction ID of the return transaction. This ID is stored with the return ticket.

Exception Files

The Oracle Retail Returns Management exception file is created and maintained by Oracle Retail Returns Management for use in detecting and preventing fraud at the point of return. The exception file acts as a constantly evolving knowledge base that can help the Authorization Engine decide which customers, items, cashiers, or stores are at higher risk for return fraud.

The exception file holds an exception counter for a customer that is incremented based on suspicious return activity. If an activity is selected for inclusion in the exception counter, the system adds 1 to the exception count for each suspicious shopping activity. Likewise, activities can be configured for cashiers.

Exceptions and counting are based on real-time refund attempt activities occurring at the point of sale or return using the return result message that is sent by the point of sale or return to Oracle Retail Returns Management at the conclusion of a transaction with an attempted refund. Return activities include activities that increment counters such as a return transaction by the customer without a receipt and with no retrieval of the original transaction, five same day returns as purchases within the last three days, and three returns today. In turn, activity thresholds might be breached and counting generated based on those thresholds.

The exception file holds an entry for each factor that triggers a count addition.

Exception File And Count Calculation

This section describes exception file count calculation for the following:

Customer

- Customer positive ID consisting of ID type, number, and issuer
- Exception count

Cashier

- Cashier ID
- Exception count

Exceptions triggered

- Exception (the return activity that was breached)
- Target of the return activity (the customer ID, cashier ID)
- Date/Time of the exception

Note: Oracle Retail Returns Management requires the use of a unique cashier ID for scoring.

Definition Of Return, For Calculation

A return, for purposes of calculation, refers to an attempted return of a line item quantity.

Note: Returns count by type at the transaction level (unique line item level). This accommodates variations between points of sale that allow mixed situations or inherently disallow mixed situations. Counting at the quantity level could abnormally inflate exceptions, for example, returning a quantity of 8 china plates. Counting at the transaction level could exclude appropriate return or non return counts due to the ability to mix returns from multiple original receipted transactions, or no receipt, within one Oracle Retail Returns Management Point of Sale transaction.

Counting unique exceptions at the transaction level is conducted so that the customer is not penalized twice for the same situation within one transaction. If an exception occurs multiple times in a single transaction, that is counted as a single exception. For example, if a customer returns three different items without a receipt in a single return transaction, only one exception is generated.

Table 6–1 Exception Counting Examples

Scenario	Exceptions Counted
Return attempt for 5 different items:	1 count for without a receipt
■ 1111 quantity 1 without receipt (no original transaction retrieved)	1 count for with a receipt
■ 2222 quantity 1 without receipt	
■ 3333 quantity 1 without receipt	
■ 4444 quantity 1 without receipt	
■ 5555 quantity 1 with receipt (original transaction retrieved)	
Return attempt for 4 different items, total quantity 5:	1 count for without a receipt
■ 1111 quantity 1 without receipt	1 count for with a receipt
■ 2222 quantity 2 without receipt	
■ 3333 quantity 1 without receipt	
■ 5555 quantity 1 with receipt	

Table 6–1 Exception Counting Examples

Scenario	Exceptions Counted
Return attempt for 2 different items:	1 count for without a receipt
<ul style="list-style-type: none"> 1234 attempted return quantity of total 4 with two different receipts. Quantity 2 comes from original transaction 042419999999 and another quantity 2 comes from 042418888888. Split as two separate lines in Oracle Retail Returns Management's Point of Sale because they would be selected from two different original transactions. 5555 quantity 1 without a receipt 	1 count for with a receipt
Two renter line items and one non-renter line item on the same receipted return attempt:	1 renter return
<ul style="list-style-type: none"> One item from the renter file is returned in the renter time frame, resulting in potential authorization, override, or denial. A second item from the renter file is returned in the renter time frame, resulting in potential authorization, override, or denial. Another item, not listed in the renter file, is being returned and is sent to Oracle Retail Returns Management for evaluation, resulting in authorization, override, or denial. 	1 expired receipted return
The receipt is older than a parameterized number of days old (hence an expired receipt).	
Can mix returns both with and without receipts in the same transaction, as well as returns received from multiple transactions.	1 count for without a receipt
In one return transaction, six line items could consist of:	1 count for with a receipt
<ul style="list-style-type: none"> Two returns without receipt — sample: resolve this one to authorized Two receipted — quantity available — sample: resolve this one to authorized 	

Exceptions

Customer Exceptions

Customer Exceptions can be flagged as behaviors that are tracked in the application, for use in Return Policies, using the Customer Exceptions to Track screen.

An exception is any activity that can be discerned from Return Ticket data, such as a non receipted return, a return of an item contained in the Return Pattern Watch file, or a particular type of refund transaction such as a Price Adjustment.

When a customer exception occurs, a record is written to the exception file and the activity is available for research on that customer using the Customer Exception Search and Customer Exception Search Results screens.

The total number of exceptions that have occurred can be checked using a rule that can be included in return policies.

All of the exceptions are based on return ticket data.

Cashier Exceptions

Cashier Exceptions can be flagged as behaviors that are tracked in the application, for use in Return Policies, using the Cashier Exceptions to Track screen.

When a cashier exception occurs, a record is written to the exception file and the activity is available for research on that cashier using the Cashier Exception Search and Cashier Exception Search Results screens.

The total number of exceptions that have occurred can be checked using a rule that can be included in return policies.

All of the exceptions are based on return ticket data.

Cashier in this case is considered to be anyone captured as the employee on the return ticket, regardless of whether they have a cashier, associate manager, manager, or other store role.

Extension Guidelines

In order to pass arbitrary data, the schemas used for return request and return result contain an optional element called MessageExtension. This optional element contains zero or more sub elements known as ExtensionElements that are simple name/value pairs of string data.

Element Location And Schema Definition

The MessageExtension element is attached to multiple parent elements in the two schemas. These elements are summarized in the following table.

Table A–1 *MessageExtension Locations*

Schema	Type
CO-RM-ReturnRequest.xsd	ReturnRequestType
	ItemReturnInfo
	ItemTransactionInfo
CO-RM-ReturnResult.xsd	ReturnResultType
	ItemReturnResult

The following example is a schema definition for the MessageExtension and ExtensionElement. For reference, we can also see the relevant section of the ReturnRequestType schema.

Note: The names and values are strings.

Example A–1 *Message Extension*

```
<xsd:complexType name="MessageExtension">
  <xsd:sequence>
    <xsd:element name="extensionEntry" type="ExtensionEntry"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExtensionEntry">
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ReturnRequestType">
```

```
<xsd:sequence>
... lines omitted ...
  <xsd:element name="messageExtension" type="MessageExtension" minOccurs="0"
maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
```

Element Usage And Retrieval

To use this optional element, the message sent to Oracle Retail Returns Management needs to include a MessageExtension with as many ExtensionEntry elements as necessary. For example, an XML message using the extension to pass custom data might look similar to the following example:

Example A-2 XML Message Using The MessageExtension

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ReturnRequest>
... lines omitted ...
  <transactionType>Return</transactionType>
  <messageExtension>
    <extensionEntry name="LegacyID" value="sun"/>
    <extensionEntry name="LegacyTransaction" value="moon"/>
  </messageExtension>
</ReturnRequest>
```

Here we see that the message contains two values for "LegacyID" and "LegacyTransaction."

Once the values have been added to the message, the message is then sent to Oracle Retail Returns Management using the desired transport (such as through a web service or a direct API call). The XML is placed by JAXB into a list accessible through the appropriate element. In this case, the element is the ReturnRequest object. The user can then search through this list to find the values the user wants. The following example demonstrates searching the MessageExtension elements.

Example A-3 Searching The MessageExtension Elements

```
MessageExtension ext = returnRequest.getMessageExtension();

if (ext != null) {
    for (Iterator it = ext.getExtensionEntry().iterator(); it.hasNext();) {
        ExtensionEntry entry = (ExtensionEntry) it.next();

        if (entry.getName().equals("LegacyID")) {
            // do something
        }
    }
}
```

Note: If the MessageExtension element is not included with the ReturnRequest or ReturnResult, the getMessageExtension() method returns a null. If the element exists but does not have any child elements, then the list returned by getExtensionEntry() is zero length.
