Oracle® Retail POS Suite

Implementation Guide – Oracle Retail POS Suite Implementation Solutions, Volume 1

Release 13.3

E15733-01

January 2011

ORACLE®

Oracle Retail POS Suite Implementation Guide – Oracle Retail Strategic Store Solutions Implementation Solutions, Volume 1, Release 13.3

E15733-01

Primary Author: Graham Fredrickson

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

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

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

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

Contents

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle Retail POS Suite Implementation Guide – Oracle Retail Strategic Store Solutions Implementation Solutions, Volume 1, Release 13.3

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

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

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

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

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

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

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

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

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

Preface

Audience

The Implementation Guide is intended for the Oracle Retail Point-of-Service integrators and implementation staff, as well as the retailer's IT personnel.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail Release 13.2 documentation set:

- Oracle Retail POS Suite Licensing Information
- Oracle Retail Back Office documentation set
- Oracle Retail Labels and Tags documentation set
- Oracle Retail Central Office documentation set
- Oracle Retail Point-of-Service documentation set

Customer Support

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

- https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.2) or a later patch release (for example, 13.2.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code):

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

The following text conventions are used in this document:

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

1
Introduction

Oracle Retail POS Suite Implementation Guide – Oracle Retail POS Suite Implementation Solutions, Volume 1 includes information that is useful for using and configuring specific features in the Point-of-Service application.

Some of these features include:

- Centralized Customer: Overview of the Centralized Customer feature for POS Suite.
- Changing and Configuring a New Base Currency: Steps for changing an existing base currency, or adding a new base currency.
- Returns Authorization: Overview of the integration with Returns Management that enables Point-of-Service to collect positive ID during the return transactions, to form and send the Return Request messages to Returns Management, to interpret and present the Returns Management Return Response messages, and to form and send Final Result messages to Returns Management.
- Bill Pay: Overview of the Bill pay feature, which enables retailers to accept bill payments from their customers and interface with their billing system to record the payments.
- Oracle Retail Point-of-Service to Oracle Retail Store Inventory Management Architecture: Overview of the Oracle Retail Point-of-Service-to-Store Inventory Management integration.
- Configuring Multiple Printers for Labels and Tags: Provides configuration information for using multiple printers to print labels and tags.
- Appendix: Serial Numbers: Overview of how serial numbers are used in POS Suite, and how to configure to use serial numbers.

2

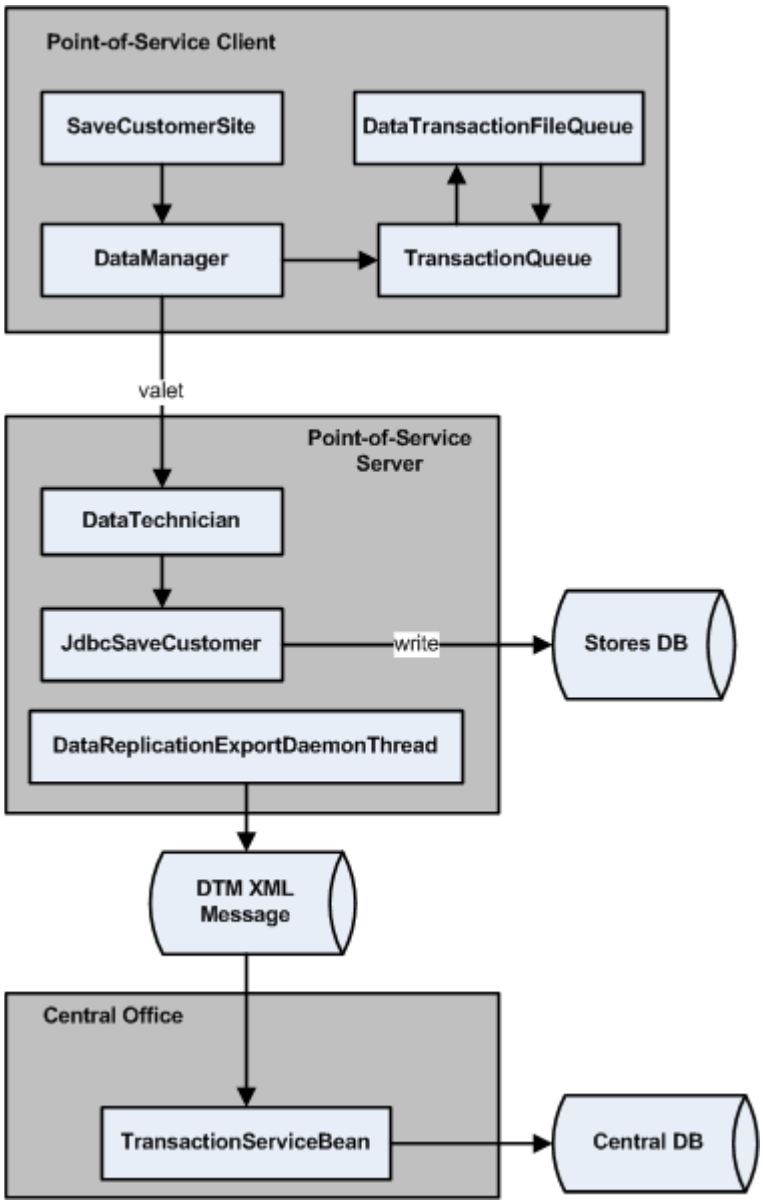
Centralized Customer

Centralized Customer allows an Oracle Retail Central Office operator to enter and manage customer data. Centralized Customer also provides Oracle Retail Point-of-Service the ability to retrieve customer information from a central database. This functionality enables Point-of-Service to support customer-specific pricing. Other Point-of-Service features support the need for Centralized Customer such as assisting in pickup and delivery orders, and obtaining Tax ID numbers for customers required to manage specific tax forms. Retail stores and cashiers also benefit from this functionality. Since customer information can be retrieved from a central database, customer information does not have to be re-entered at different stores.

The Centralized Customer package enables the operator to manage existing customer information and add new customers to the central database. The operator has the ability to search for a customer, modify existing customer information, or mark a customer's record for deletion from the database. The operator can also assign a Pricing Group to the customer which allows retailers the ability to offer customer specific pricing. Pricing Groups can be assigned to a Price Promotion or Discount Rule.

There are two types of customers: Individual and Business. Business customers require slightly different data than individual customers such as tax certificate numbers.

Figure 2–1 Centralized Customer Object Model



3

Changing and Configuring a New Base Currency

Changing Currency

In order to switch to another base and alternate currency, perform the following steps:

1. Set the base currency flag in the primary currency of the currency table. For example, if EUR is the base currency:

   ```
   update co_cny set FL_CNY_BASE='1' where DE_CNY='EUR'
   ```

2. Remove the base currency flag from any other currencies in that table. For example:

   ```
   update co_cny set FL_CNY_BASE = '0' where DE_CNY <> 'EUR'
   ```

3. Enforce ordering so that the primary currency is first and the alternate currency is second for the AI_CNY_PRI column in the currency table. Other rows should be ordered, but the specific order isn't important. For example if EUR is base currency and GBP is the alternate:

   ```
   update co_cny set AI_CNY_PRI=0 where DE_CNY='EUR'
   update co_cny set AI_CNY_PRI=1 where DE_CNY='GBP'
   update co_cny set AI_CNY_PRI=2 where DE_CNY='USD'
   update co_cny set AI_CNY_PRI=3 where DE_CNY='CAD'
   update co_cny set AI_CNY_PRI=4 where DE_CNY='MXN'
   update co_cny set AI_CNY_PRI=5 where DE_CNY='JPY'
   ```

4. Add store safe tenders supported for the new base/alternate currency. For example, if EUR is the new base currency, add money order tender support for EUR:

   ```
   insert into le_tnd_str_sf
   (ID_RPSTY_TND, TY_TND, TY_SB_TND, LU_CNY_ISSG_CY, TS_CRT_RCRD, TS_MDF_RCRD, ID_
   CNY_ICD )
   VALUES ('1','MNYO', ' ', 'EU', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 5);
   ```

   Remove store safe tenders no longer supported for the old base/alternate currency. For example, if USD if the old base currency, remove money order tender support for USD:

   ```
   delete from le_tnd_str_sf where LU_CNY_ISSG_CY = 'US' and TY_TND = 'MNYO';
   ```

5. Add exchange rate records for alternate and base currencies into the CO_RT_EXC table based on the new base currency. Delete all exchange rate records based on any previous base currency.

There are some application parameters that must be changed as well:

- Tender Group:
  - `CashAccepted`: For example, if EUR is base and GBP is alternate, make sure that the `CashAccepted` parameter is changed so that EUR and GBP are selected.
  - `TravelersChecksAccepted`: For EUR as base and GBP as alternate, the values for the `TravelersChecksAccepted` parameter should be EURCHK and GBPCHK.
  - `ChecksAccepted`: For EUR as base and GBP as alternate, the values for the `ChecksAccepted` parameter should be EURCHK and GBPCHK.
  - `GiftCertificateAccepted`: Change the values to reflect all the currencies accepted (base and alternate). For example the values may be EUR and GBP, or EUR, GBP and USD.
  - `StoreCreditAccepted`: Change the values to reflect all the currencies accepted (base and alternate). For example the values may be EUR and GBP, or EUR, GBP and USD.
- Reconciliation Group:
  - `TendersToCountAtTillReconcile`: For EUR as base and GBP as alternate, the values for the `TendersToCountAtTillReconcile` parameter should be:
    * Cash
    * Check
    * ECheck
    * Credit
    * Debit
    * TravelCheck
    * GiftCert
    * Coupon
    * GiftCard
    * StoreCredit
    * MallCert
    * PurchaseOrder
    * MoneyOrder
    * GBPCash
    * GBPTravelCheck
    * GBPCheck
    * GBPGiftCert
    * GBPStoreCredit

Configuring a New Base Currency

Throughout this section, Krona is used as the example new base currency that is being configured. The Krona currency code is SEK, and the issuing country code is SE.

Currency SQL Configuration

The following SQL configurations for Currency are available.

Currency Table CO_CNY

A new record describing the new currency information such as its currency code, issuing country code and so forth, must be inserted into this table.

In the base currency flag column **FL_CNY_BASE**, the new currency must be set to **1** indicating that it is the base. The flag for other currencies must be set to **0**, indicating that they are alternate currencies.

> **Note:** Point-of-Service supports base-plus-one alternate currency. The priority column AI_CNY_PRI must be set to 0 for the new base currency. It must be set to 1 for the supported alternate currency. For other alternate currencies, they must be ordered and greater than 1, but the specific order isn't important.

Example 3–1 Add Krona as Base to Currency Table CO_CNY

```
INSERT INTO CO_CNY
(ID_CNY_ICD, LU_CNY_ISSG_CY, CD_CNY_ISO, DE_CNY, DE_CNY_ISSG_NAT, FL_CNY_BASE, QU_
CNY_SCLE, AI_CNY_PRI)
VALUES (7,'SE', 'SEK', 'SEK', 'Sweden', '1', 2, 0);

UPDATE CO_CNY
SET FL_CNY_BASE = '0'
WHERE CD_CNY_ISO <> 'SEK';

UPDATE CO_CNY
SET AI_CNY_PRI = AI_CNY_PRI + 1
WHERE CD_CNY_ISO <> 'SEK';
```

Currency Denomination Table CO_CNY_DNM and I8 table CO_CNY_DNM_I8

Denominations for the new base currency must be added to the CO_CNY_DNM and CO_CNY_DNM_I8 table. For example:

Example 3–2 Add Krona Denominations to Denomination Table CO_CNY_DNM

```
INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 1, 'SE_50Ores', '0.50', 1);

INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 2, 'SE_1Kronas', '1.00', 2);

INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 3, 'SE_5Kronas', '5.00', 3);

INSERT INTO CO_CNY_DNM
```

```
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 4, 'SE_10Kronas', '10.00', 4);

INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 5, 'SE_20Kronas', '20.00', 5);

INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 6, 'SE_50Kronas', '50.00', 6);

INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 7, 'SE_100Kronas', '100.00', 7);

INSERT INTO CO_CNY_DNM
(ID_CNY_ICD, ID_CNY_DNM, NM_DNM, VL_DNM, CD_DNM_DPLY_PRI)
VALUES (7, 8, 'SE_1000Kronas', '1000.00', 8);
```

Example 3–3 Add Krona Denominations to I8 Table CO_CNY_DNM_I8

```
INSERT INTO CO_CNY_DNM_I8
(ID_CNY_ICD, ID_CNY_DNM, LCL, NM_DNM)
VALUES (7, 2,'en','1 Kronas');

INSERT INTO CO_CNY_DNM_I8
(ID_CNY_ICD, ID_CNY_DNM, LCL, NM_DNM)
VALUES (7, 2,'fr','1 couronne');
```

> **Note:** For each denomination record in the CON_CNY_DNM table, there are I8 records in CO_CNY_DNM_I8 table, one for each supported language.

Exchange Rate Table CO_RT_EXC

Add exchange rate records for alternate and base currencies into the CO_RT_EXC table based on the new base currency. Delete all exchange rate records based on any previous base currency. For example:

Example 3–4 Add Alternate Currency Exchange Rates to Krona

```
-- Delete all the existing records
Delete from CO_RT_EXC;

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 1, 6.3337, 6.3362, 0.00);

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 2, 6.2849, 6.2898, 0.00);

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
```

```
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 3, 0.5799, 0.5816, 0.00);

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 4, 12.434, 12.441, 0.00);

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 5, 9.3739, 9.3796, 0.00);

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 6, 0.05782, 0.05786, 0.00);

INSERT INTO CO_RT_EXC
(LL_CNY_EXC, DC_RT_EXC_EF, DC_RT_EXC_EP, ID_CNY_ICD, MO_RT_TO_BUY, MO_RT_TO_SL,
MO_FE_SV_EXC)
VALUES(0.00, TO_DATE('1990-01-01', 'YYYY-MM-DD'), TO_DATE('2099-12-31',
'YYYY-MM-DD'), 7, 1.0, 1.0, 0.00);
```

Store Safe Tender Table LE_TND_STR_SF

Add the store safe tenders supported for the new base currency. For example:

Example 3–5 Add Store Safe Tenders for Krona

```
INSERT INTO LE_TND_STR_SF
    ( ID_RPSTY_TND, TY_TND, TY_SB_TND, LU_CNY_ISSG_CY, TS_CRT_RCRD, TS_MDF_RCRD,
ID_CNY_ICD )
    VALUES('1','CASH', ' ', 'SE', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 7);
INSERT INTO LE_TND_STR_SF
    ( ID_RPSTY_TND, TY_TND, TY_SB_TND, LU_CNY_ISSG_CY, TS_CRT_RCRD, TS_MDF_RCRD,
ID_CNY_ICD )
    VALUES('1','CHCK', ' ', 'SE', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 7);
INSERT INTO LE_TND_STR_SF
    ( ID_RPSTY_TND, TY_TND, TY_SB_TND, LU_CNY_ISSG_CY, TS_CRT_RCRD, TS_MDF_RCRD,
ID_CNY_ICD )
    VALUES('1','TRAV', ' ', 'SE', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 7);

-- MoneyOrderSafeTender

INSERT INTO LE_TND_STR_SF
(ID_RPSTY_TND, TY_TND, TY_SB_TND, LU_CNY_ISSG_CY, TS_CRT_RCRD, TS_MDF_RCRD, ID_
CNY_ICD )
VALUES ('1','MNYO', ' ', 'SE', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 7);
```

Money Order Tenders are only accepted for base currency, therefore before inserting records for the new base currency, delete any money order tenders for the other currencies:

```
DELETE * from LE_TND_STR_SF where ty_tnd='MNYO'
```

Parameter Configuration

The following tender parameters must be enhanced to include the new base currency:

- StoreCreditsAccepted
- ChecksAccepted
- CashAccepted
- GiftCertificatesAccepted
- TravelersChecksAccepted

The reconciliation parameter **TendersToCountAtTillReconcile** parameter must include all the tenders to count for both base and alternate currencies during till reconciliation. For example:

Example 3–6 Parameters to support Krona as the base and USD as the alternate currency

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SOURCE PUBLIC "SOURCE"
"classpath://com/extendyourstore/foundation/tour/dtd/paramsourcescript.dtd">
<SOURCE name="register">
<GROUP hidden="N" name="Tender">
<PARAMETER final="N" hidden="N" name="StoreCreditsAccepted" type="LIST">
<VALIDATOR class="EnumeratedListValidator"
package="oracle.retail.stores.foundation.manager.parameter">
<PROPERTY propname="member" propvalue="None"/>
<PROPERTY propname="member" propvalue="USD"/>
<PROPERTY PROPNAME="MEMBER" PROPVALUE="SEK"/>
<PROPERTY PROPNAME="MEMBER" PROPVALUE="EUR"/>
</VALIDATOR>
<VALUE value="SEK"/>
<VALUE value="USD"/>
<VALUE value="EUR"/>
</PARAMETER>
<PARAMETER final="N" hidden="N" name="ChecksAccepted" type="LIST">
<VALIDATOR class="EnumeratedListValidator"
package="oracle.retail.stores.foundation.manager.parameter">
<PROPERTY propname="member" propvalue="None"/>
<PROPERTY propname="member" propvalue="USDCHK"/>
<PROPERTY propname="member" propvalue="SEKCHK"/>
<PROPERTY propname="member" propvalue="EURCHK"/>
</VALIDATOR>
<VALUE value="SEKCHK"/>
<VALUE value="USDCHK"/>
</PARAMETER>
<PARAMETER final="N" hidden="N" name="CashAccepted" type="LIST">
<VALIDATOR class="EnumeratedListValidator"
package="oracle.retail.stores.foundation.manager.parameter">
<PROPERTY propname="member" propvalue="None"/>
<PROPERTY propname="member" propvalue="USD"/>
<PROPERTY propname="member" propvalue="SEK"/>
<PROPERTY propname="member" propvalue="EUR"/>
</VALIDATOR>
<VALUE value="SEK"/>
<VALUE value="USD"/>
</PARAMETER>
<PARAMETER final="N" hidden="N" name="GiftCertificatesAccepted" type="LIST">
<VALIDATOR class="EnumeratedListValidator"
package="oracle.retail.stores.foundation.manager.parameter">
```

```
<PROPERTY propname="member" propvalue="None"/>
<PROPERTY propname="member" propvalue="USD"/>
<PROPERTY propname="member" propvalue="SEK"/>
<PROPERTY propname="member" propvalue="EUR"/>
</VALIDATOR>
<VALUE value="SEK"/>
</PARAMETER>
<PARAMETER final="N" hidden="N" name="TravelersChecksAccepted" type="LIST">
<VALIDATOR class="EnumeratedListValidator"
package="oracle.retail.stores.foundation.manager.parameter">
<PROPERTY propname="member" propvalue="None"/>
<PROPERTY propname="member" propvalue="USDCHK"/>
<PROPERTY propname="member" propvalue="SEKCHK"/>
<PROPERTY propname="member" propvalue="EURCHK"/>
</VALIDATOR>
<VALUE value="SEKCHK"/>
<VALUE value="USDCHK"/>
Configuring a New Base Currency
Appendix: Changing and Configuring a New Base Currency D-7
</PARAMETER>
</GROUP>
<GROUP hidden="N" name="Reconciliation">
<PARAMETER final="N" hidden="N" name="TendersToCountAtTillReconcile" type="LIST">
<VALIDATOR class="EnumeratedListValidator"
package="oracle.retail.stores.foundation.manager.parameter">
<PROPERTY propname="member" propvalue="Cash"/>
<PROPERTY propname="member" propvalue="Check"/>
<PROPERTY propname="member" propvalue="ECheck"/>
<PROPERTY propname="member" propvalue="Credit"/>
<PROPERTY propname="member" propvalue="Debit"/>
<PROPERTY propname="member" propvalue="TravelCheck"/>
<PROPERTY propname="member" propvalue="GiftCert"/>
<PROPERTY propname="member" propvalue="Coupon"/>
<PROPERTY propname="member" propvalue="GiftCard"/>
<PROPERTY propname="member" propvalue="StoreCredit"/>
<PROPERTY propname="member" propvalue="MallCert"/>
<PROPERTY propname="member" propvalue="PurchaseOrder"/>
<PROPERTY propname="member" propvalue="MoneyOrder"/>
<PROPERTY propname="member" propvalue="USDCash"/>
<PROPERTY propname="member" propvalue="USDTravelCheck"/>
<PROPERTY propname="member" propvalue="USDCheck"/>
<PROPERTY propname="member" propvalue="USDGiftCert"/>
<PROPERTY propname="member" propvalue="USDStoreCredit"/>
</VALIDATOR>
<VALUE value="Cash"/>
<VALUE value="Check"/>
<VALUE value="ECheck"/>
<VALUE value="Credit"/>
<VALUE value="Debit"/>
<VALUE value="TravelCheck"/>
<VALUE value="GiftCert"/>
<VALUE value="Coupon"/>
<VALUE value="GiftCard"/>
<VALUE value="StoreCredit"/>
<VALUE value="MallCert"/>
<VALUE value="PurchaseOrder"/>
<VALUE value="MoneyOrder"/>
<VALUE value="USDCash"/>
<VALUE value="USDTravelCheck"/>
<VALUE value="USDCheck"/>
```

```
<VALUE value="USDGiftCert"/>
<VALUE value="USDStoreCredit"/>
</PARAMETER>
</GROUP>
</SOURCE>
```

Resource Bundle Configuration

New resource bundle keys describing the new currency, its issuing country must be added to the Point-of-Service resource bundle commonText, ejournalText, tillText, dailyOperationsText, and parameterText. For example:

Example 3–7 New commonText Resource Bundle Keys

```
#
# Supported Nationalities
Common.SE_Nationality=Swedish

#
# Supported Currencies
Common.SEK=Swedish Krona

#
# Supported Checks
Common.SEKCHK=Swedish Krona

#
#  Tender Types
#
Common.SEKCash=SEK Cash
Common.SEKCheck=SEK Check
Common.SEKTravCheck=SEK Trav. Check
```

Example 3–8 New ejournalText Resource Bundle Keys

```
JournalEntry.SEK=SEK
```

Example 3–9 tillText Resource Bundle Keys

```
SelectTenderSpec.SelectSEK=SEK

Add example for dailyOperations Resource Bundle Keys:

FinancialTotalsSummaryEntrySpec.CURRCODE_SE=SEK

Add example for parameterText Resource Bundle Keys:
Common.SEKCash=SEK Cash
Common.SEK TravelCheck=SEK  Traveler's Check
Common.SEK Check=SEK  Check
Common.SEK GiftCert=SEK  Gift Certificate
Common.SEK StoreCredit=SEK  Store Credit
Common.SEKGiftCard=SEK Gift Card
```

4

Returns Authorization

The integration allows for Point-of-Service to collect positive ID during the return transactions, to form and send the Return Request messages to Returns Management, to interpret and present the Returns Management Return Response messages, and to form and send Final Result messages to Returns Management right before the return transaction is completed. Point-of-Service has support/flow for all Returns Management Return Response types. Point-of-Service has support/flow for accepting and managing Returns Management recommended tenders.

Returns Management provides the ability to deliver an accept/deny response for attempted refunds on line items of return transactions as well as non-receipted return attempts through standard XML messages. The retailer can configure enterprise-wide, down to store- and item-specific, receipted and non-receipted policies that are applied to line items on transactions occurring at a point of sale or point of return. The policy definition as well as accept/deny logic is contained within the enterprise and therefore is abstracted from the point of sale or return such that Returns Management can work with any point of sale or return application, including web or phone order systems. Returns Management provides the ability to count instances of behavior for customers and cashiers based on negativity activity and deny returns based on frequent suspicious activity. Included are inquiry screens to research an attempted refund or a particular score and its history.

Exception Flow

Communication with Returns Management is available only when Point-of-Service Server is in ONLINE mode. If Point-of-Service Server goes offline at any time during authorization or sending Final Result, the authorization request and final result information will be saved in Point-of-Service as offline return information, the message in E-Journal is logged and the offline return information will be sent to Returns Management when it is available.

Error Handling

Error handling is limited to logging errors during the return authorization. The exceptions such as IOException and invalidItem that occur during WSService communication are re-thrown as WSException, as well as logged for error tracking and resolution.

Logging

RPI uses Log4J for Logging. The following logging levels can be used:

- **Info**: For logging information messages.
- **Debug**: For logging all the debug messages.
- **Error**: For logging application errors.
- **Warn**: For logging warning messages.

The logging level can be configured with log4J.xml.

5

Bill Pay

The Oracle Retail Point-of-Service Bill Pay feature enables retailers to accept telecom bill payments from their customers and interface with their billing system to record the payments. This solution is primarily intended for the telecom service providers who run their outlet stores primarily in developing markets. The new Oracle Retail Point-of-Service Bill Pay feature provides an ability for the cashier at the store to accept telecom bill payments and provides an extendible and generic framework to support integration of Point-of-Service with different billing systems, such as Oracle Billing and Revenue Management (BRM), Amdoc and so forth.

Point-of-Service Bill Pay provides the telecom retailer with the following functionality:

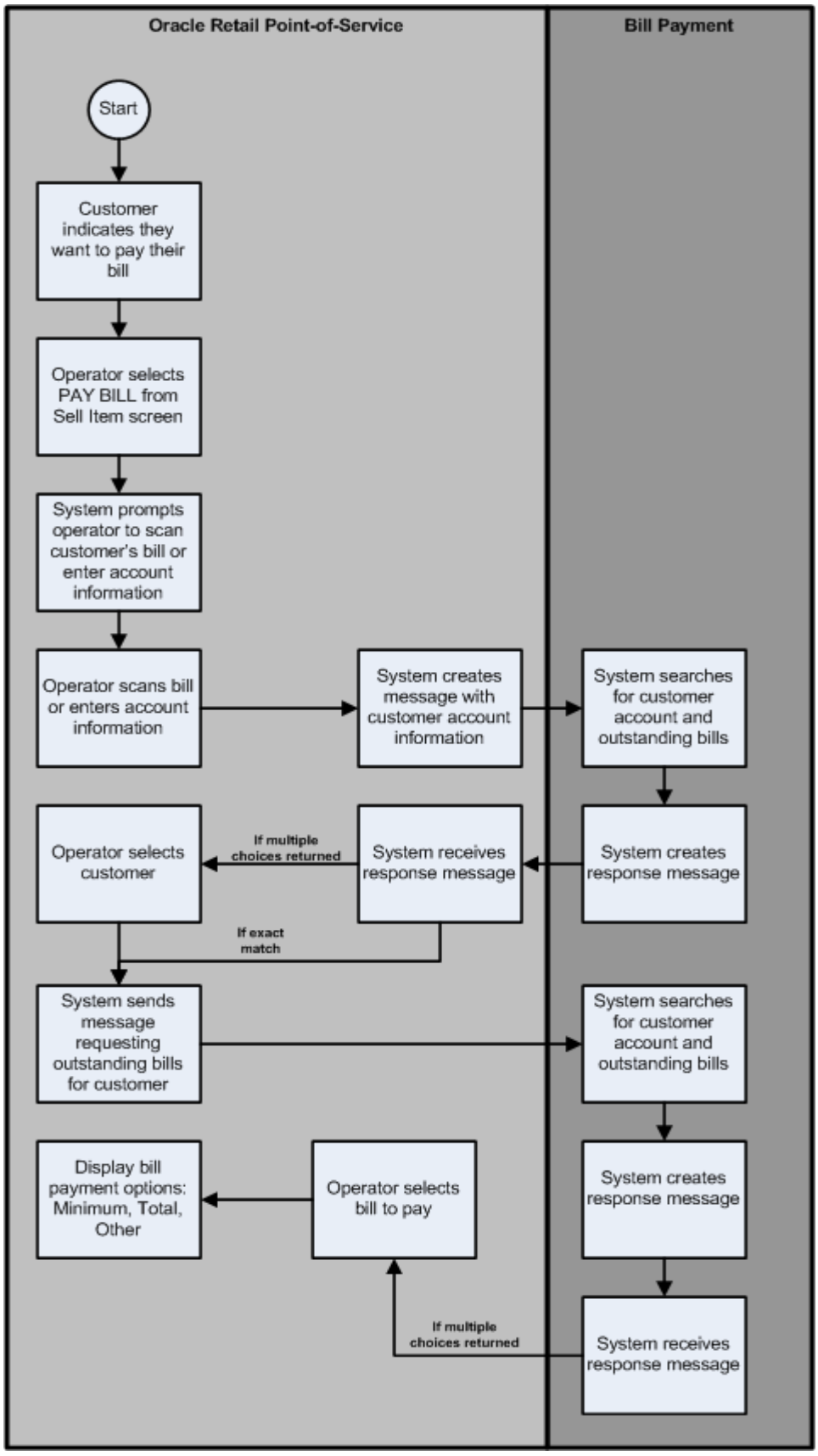
- Bill Search and Pay: The operator can scan the bill number to get the bill information and options to pay the bill using different tender types. The operator can also look up the bill details on a third-party billing system by providing customer information.
- Offline Bill Pay: When the third-party billing system is offline, Point-of-Service can take the payment by capturing the minimum information required for that bill payment and later sending this detail to the billing system when the system is online.
- Integration Framework: Enables the service implementers (SI) to integrate Point-of-Service with different third-party billing systems.

Process Flow

The Bill Pay process flow includes following:

- Bill Lookup Process: A blocking (synchronous) call to the billing system to retrieve bills.
- Transaction Update Process: A non-blocking (asynchronous) call to the billing system to update the bill payment details.

Figure 5–1 Bill Pay Process Flow



The following sections detail the process flow in Point-of-Service Client and Point-of-Service Server. The Bill Lookup Process flow diagram depicts the UI tour process to prompt and get the bill information. The Transaction Update process flow depicts the process to send the transaction.

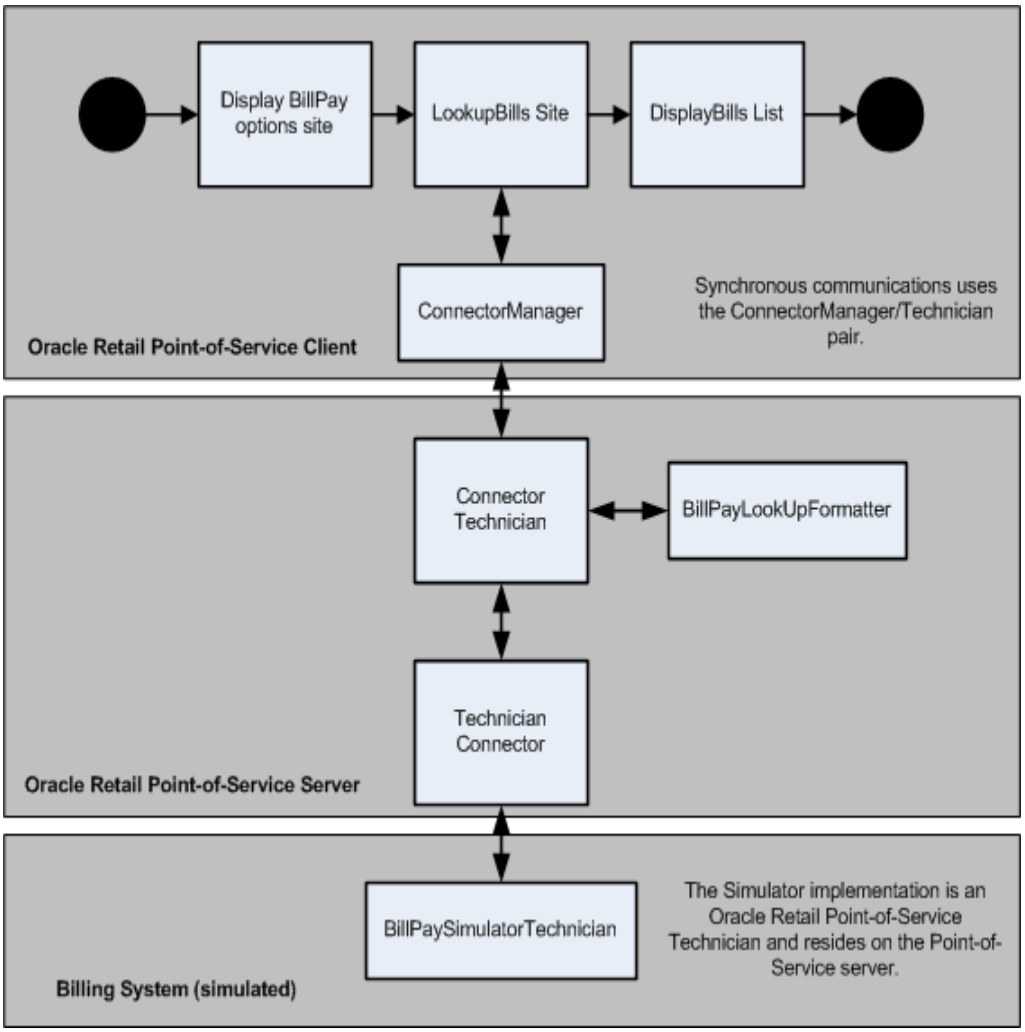
The external BRM system can expose APIs through a WebService or an EJB, or any other API. The retailer can add whatever means of contacting the Bill Pay system using the Connector framework of Point-of-Service. The system currently uses an API call, invoking the simulated Bill Pay system in Oracle Retail Point-of-Service.

Bill Lookup Process

The Bill Pay lookup process requires the Point-of-Service Client to connect to the connector framework using the Bill Pay Connector service.

The following diagram shows the overall process flow for bill lookup in a bill payment process:

Figure 5–2 Bill Pay – Lookup Bill Information Process



The ConnectorManager in Point-of-Service Client delegates the bill lookup request to the ConnectorTechnician (entry point to the connector framework) that integrates with the Simulated Bill Pay system.

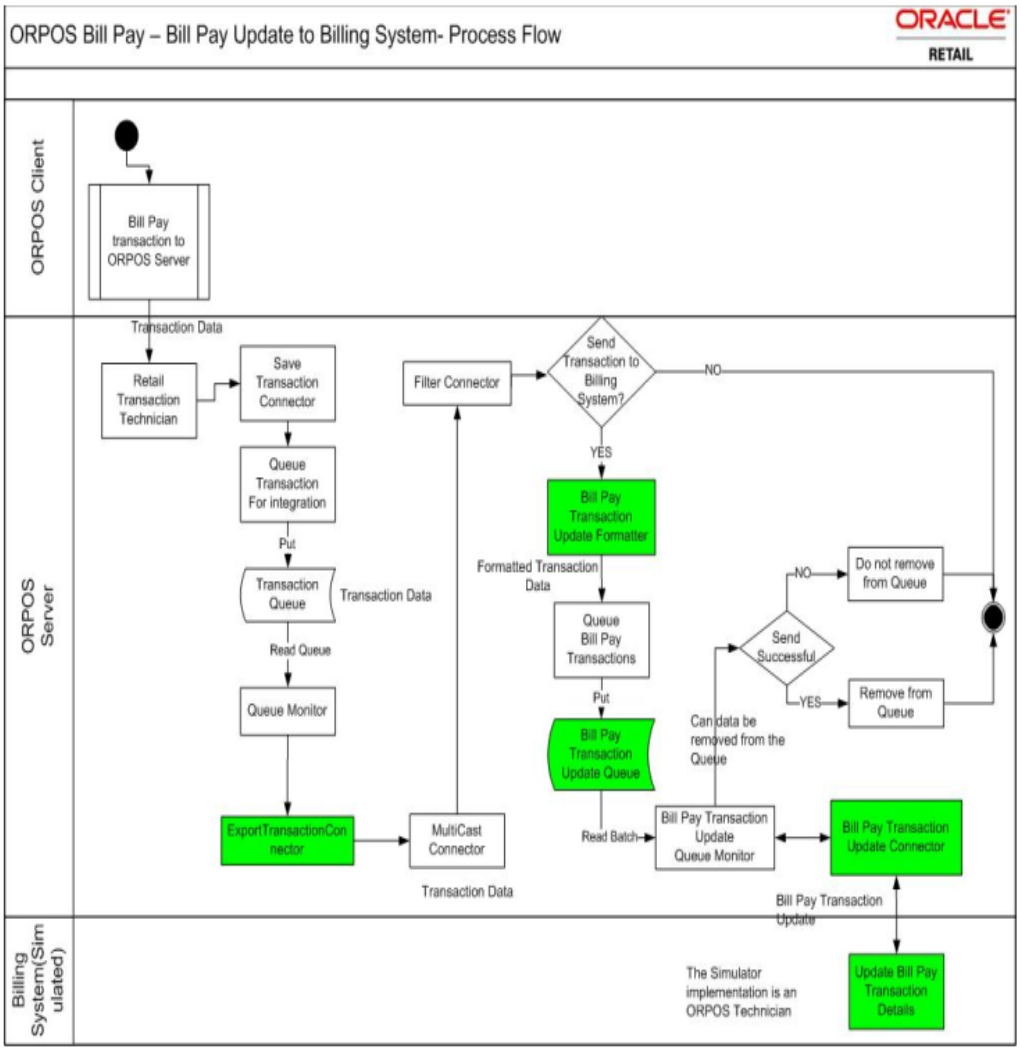
- The above diagram shows the details of the bill lookup process when the operator scans/enters bill details.
- The bill pay tour invokes the ConnectorManager. The connector framework is used for integration with billing systems and getting the results back to the tour.

Transaction Update Process

The Bill Pay transaction update process is a background process. The process runs once the transaction is tendered and persisted to the stores database. The process requires the Point-of-Service Server to connect to the connector framework through the RetailTransactionTechnician (entry point to the connector framework for transaction object).

The following diagram shows the overall process flow for the Bill Pay transaction update process:

Figure 5–3 Bill Pay Transaction Update Process

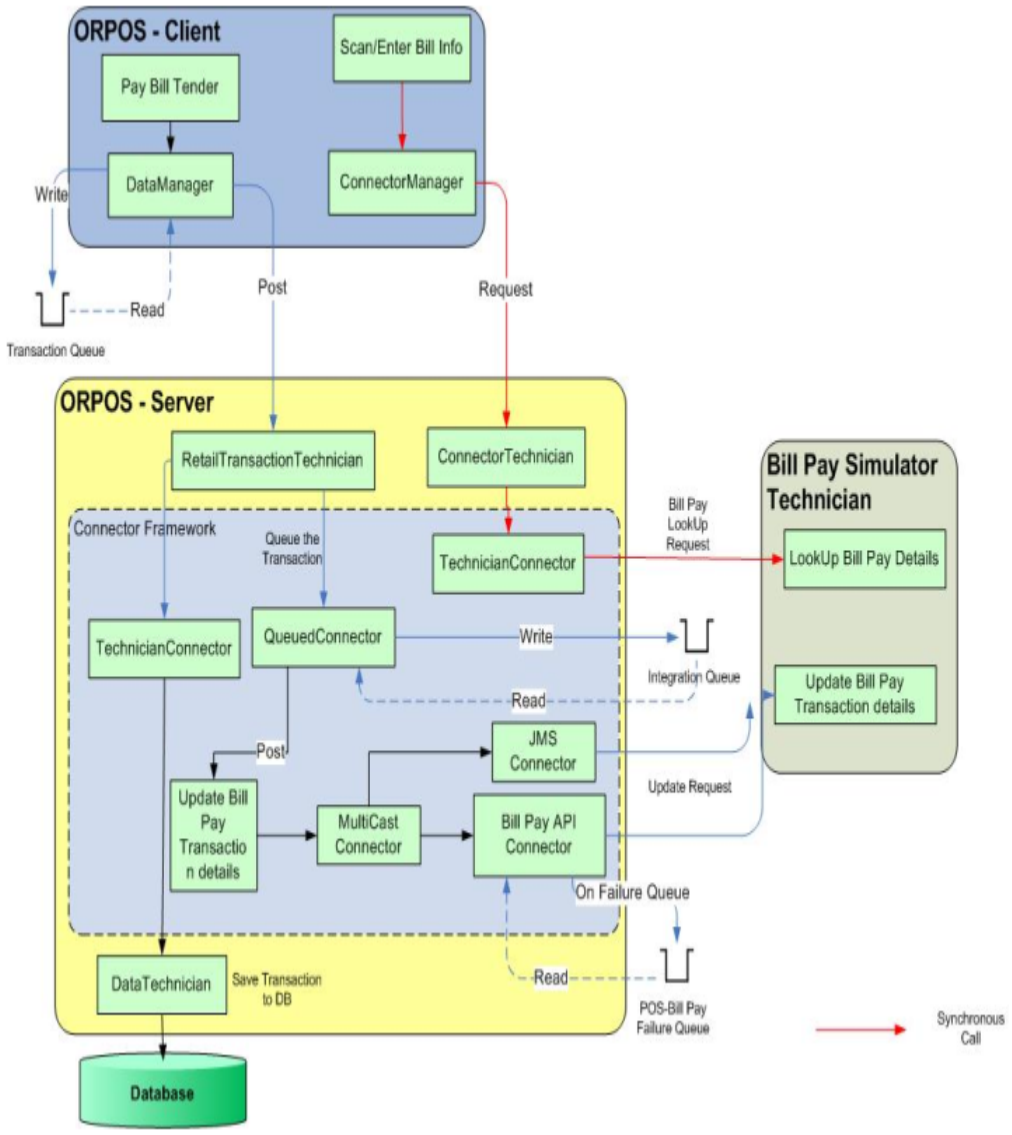


- The Bill Pay transaction update process uses the existing Point-of-Service connector framework to export transactions to Billing systems.
- Connector framework can be configured to support near real time or batch type transaction updates to the billing systems. The current implementation supports near real time transaction updates to the simulator.
- The transaction data is posted by the Point-of-Service Client to the Server for saving the data to the database. The transaction data is received by the RetailTransactionTechnician in the Point-of-Service Server. The RetailTransactionTechnician forms the entry point into the connector framework (CommExt).
- The RetailTransactionTechnician persists the data to the database and then queues the data to send to the billing system. Once the data is read from the queue it is then picked up by the Export Transaction Connector service.
- The Export Transaction Connector service is a new connector developed for exporting transactions to different systems such as Oracle BRM, Seibel and so forth. The transaction export connector service forwards transactions to different message routers using the existing multicast connector based on a configuration xml.

Logical Architecture

Bill Pay uses Point-of-Service connector integration framework for integration with billing systems:

Figure 5–4 Bill Pay Logical Architecture



The Bill Pay system is broken into four main sub-systems:

- Point-of-Service Client
- Point-of-Service Server
- Bill Pay Simulator Technician
- Database

Each of these sub-systems is described in the following table:

Table 5–1 Bill Pay Logical Architecture

Component	Description
Point-of-Service Client	Changes to the Point-of-Service Client to incorporate new BillPay tour and parameters to control tender types allowed for bill payments.
Point-of-Service Server	Following new components are added to connector framework,· Update BillPay Transaction details – This connector service is used to export the transaction to external billing systems.·BillPay API Connector – This connector is used invoke APIs of the billing system.·JMS Connector – This connector can be used to connect to billing systems which exposes JMS queues.
Bill Pay Simulator Technician	This is a Point-of-Service technician and will simulate a billing system.
Database	Point-of-Service database used to store the bill pay transaction

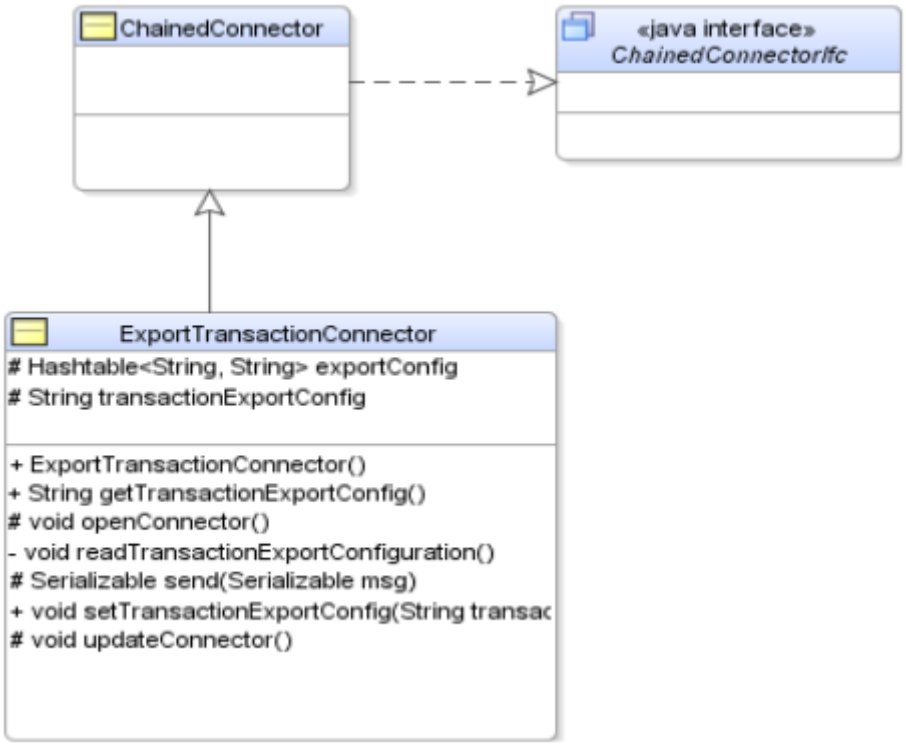
Connector Framework Changes

This section details changes to the Point-of-Service Connector framework to export transactions to billing systems.

Export Transaction Connector

This connector is added to RetailTransactionTechnician.xml to export transactions to external systems. This connector extends chained connectors and sends transactions to a list of MessageRouters defined for the transaction type given in the ExportTransactionConfig.xml configuration file. The connector uses the MultiCastConnector to broadcast the transactions to different message routers.

Figure 5–5 Export Transaction Connector Class



Example 5–1 ExportTransactionConnector Configuration

```
<!-- This Connector forwards the transactions to different message routers using
multicast connector -->
<CHAINED_CONNECTOR connector="MulticastQueuedTransactions"
                   javaclass="oracle.retail.stores.platform.connector.
                              ExportTransactionConnector"
                   name="ExportTransactionConnector">
     <PROPERTY propname="transactionExportConfig" proptype="STRING"
               propvalue="classpath://config/ExportTransactionConfig.xml" />
 </CHAINED_CONNECTOR>
```

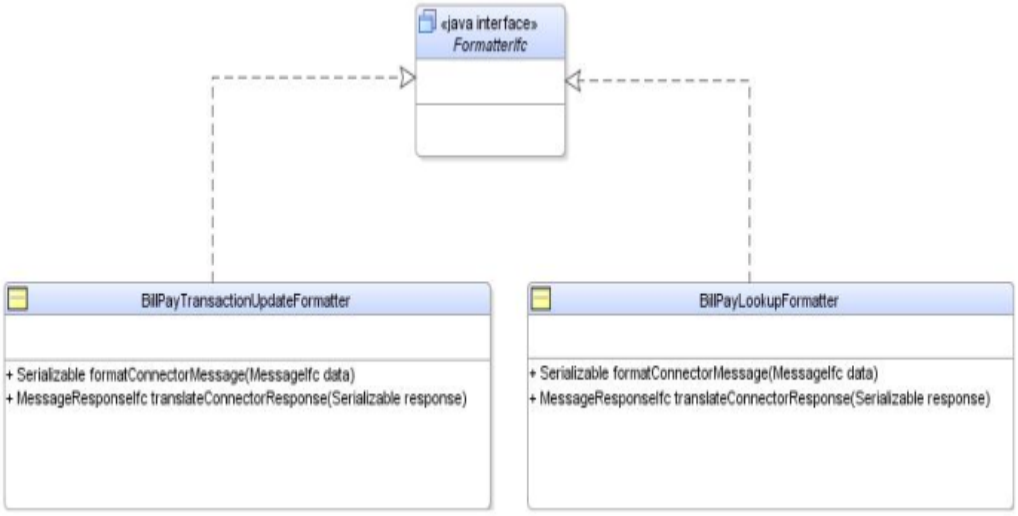
Example 5–2 Export Transaction Connector Configuration Script (ExportTransactionConfig.xml)

```
<EXPORT_TRANSACTION>
    <COMMENT>
Connector Configuration for Transaction to External systems, the   transactions
are sent to message routers using multicast connector
    </COMMENT>
<!-- The Transaction Names should be same as provided in   TransactionConstantsIfc
TYPE_DESCRIPTORS[] -->
    <TRANSACTION name="SALE" MSGROUTERS="FORMAT_PSI_TRANSACTIONS" />
    <TRANSACTION name="RETURN" MSGROUTERS="FORMAT_PSI_TRANSACTIONS" />
    <TRANSACTION name="VOID" MSGROUTERS="FORMAT_PSI_TRANSACTIONS" />
    <TRANSACTION name="EXCHANGE" MSGROUTERS="FORMAT_PSI_TRANSACTIONS" />
    <TRANSACTION name="BILLPAY"
MSGROUTERS="BILL_PAY_TRANSACTIONS, SEIBEL_TRANSCTION" />
……
</EXPORT_TRANSACTION>
```

BillPay Connector Formatters

Two connector formatter classes are added to format bill lookup and transaction update request/responses.

Figure 5–6 Bill Pay Formatters Class Diagram



The following formatter configuration is added to ConnectorTechnician's configuration file DefaultConnectorTechnician.xml for bill lookups:

```
<FORMATTER name="BillPayTransactionUpdateDataFormatter"
javaclass="oracle.retail.stores.pos.formatter.BillPayTransactionDataFormatter" />
```

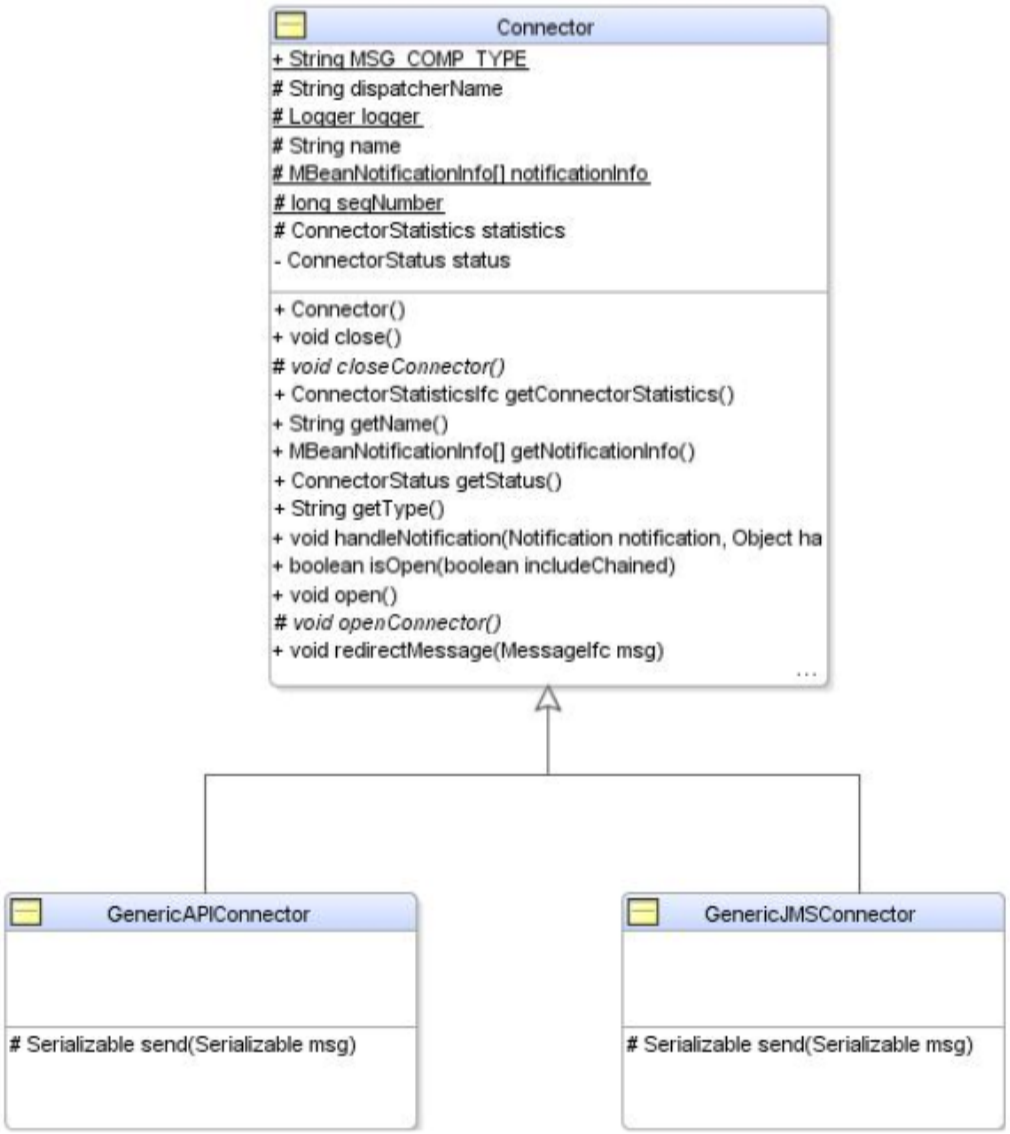
The following formatter configuration is added to RetailTransactionTechnicians's configuration file RetailTransactionTechnician.xml for Bill Pay transaction updates:

```
<FORMATTER name="BillPayLookupDataFormatter"
javaclass="oracle.retail.stores.pos.formatter.BillPayTransactionDataFormatter" />
```

Bill Pay API/JMS Connector

Two new generic connector classes are developed to invoke APIs/JMS exposed by external systems:

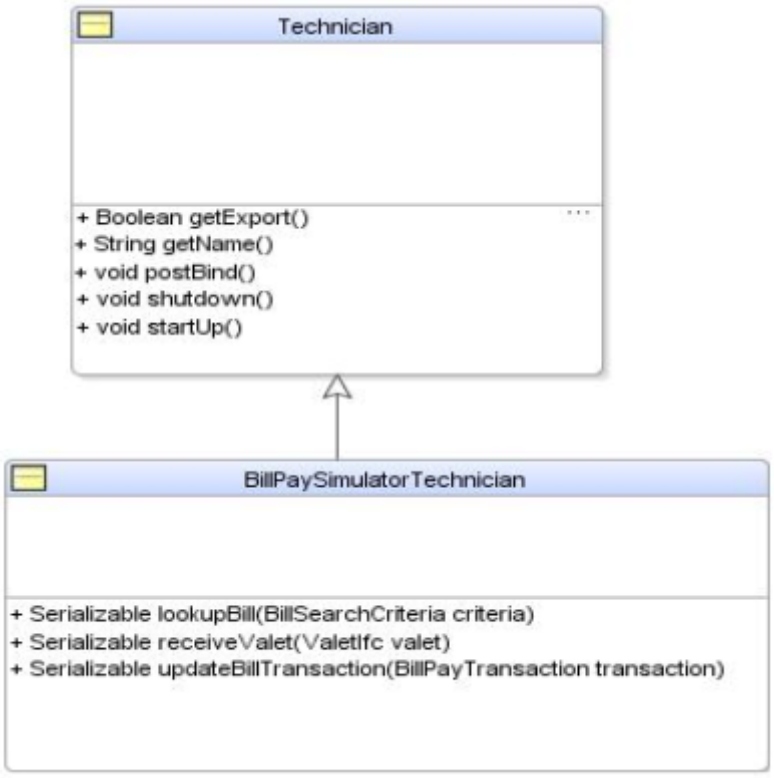
Figure 5–7 Generic Connectors Class Diagram



Bill Pay Simulator Technician

A new technician is added to Point-of-Service Server. This technician gives simulated responses to the bill pay lookup/ transaction update requests.

Figure 5–8 Bill Pay Simulator Class Diagram



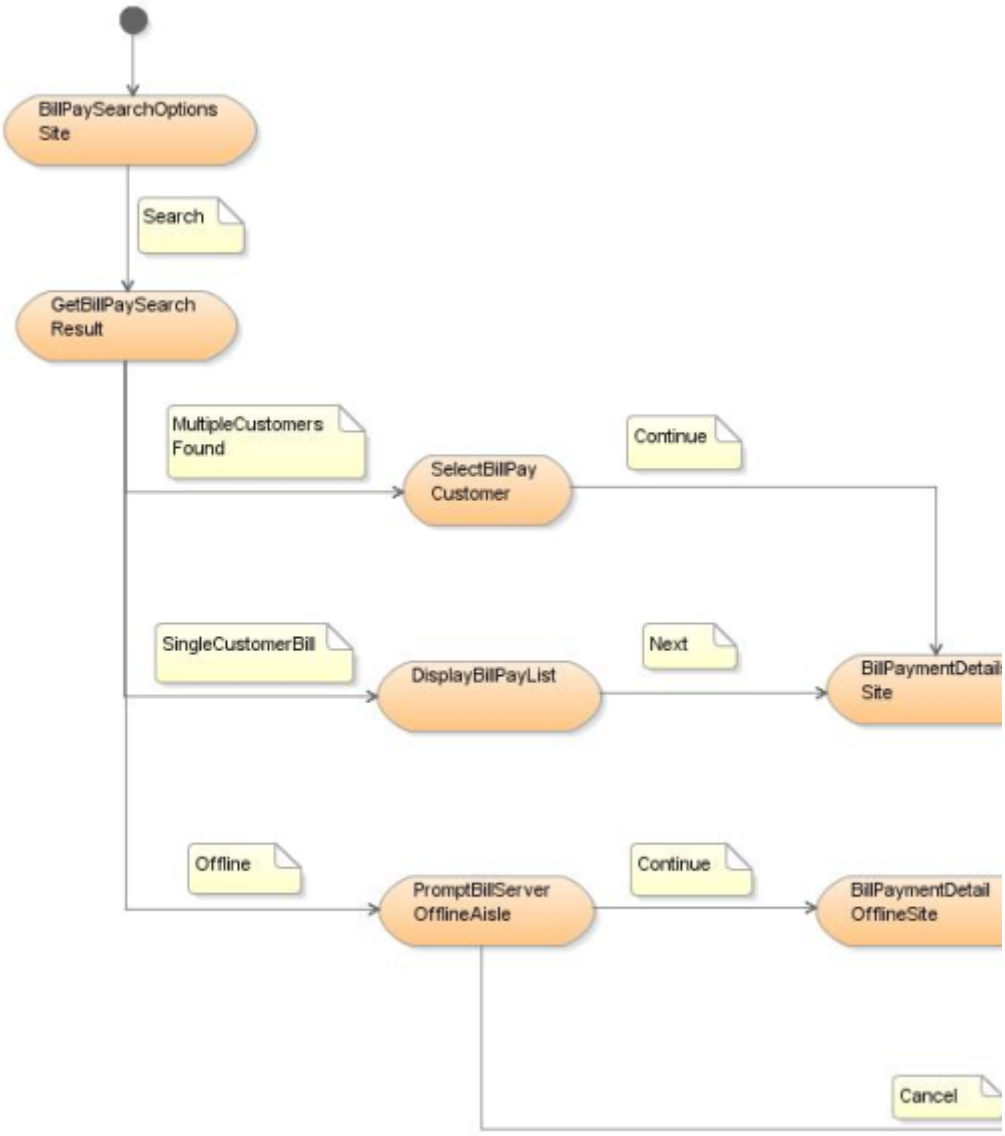
Tour Changes

This section details the changes to Point-of-Service Client tours and workflows.

Bill Pay Tour

The new bill pay Tour (billpay.xml) is introduced to cater to the lookup bills from external billing system. The design of the tour workflow is as follows:

Figure 5–9 Bill Pay Tour



The start site of this new tour is the BillPaySearchOptionsSite which is responsible to display bill search options. The operator can either scan the bill number or alternatively use the other search criteria to find the customer and the bills related to the customer.

The next site, GetBillPaySearchResult, invokes the ConnectorManager to integrate with external billing system to get the billing details. If the billing system is offline, the control moves to PromptBillServerOfflineAisle. If the call returns single or multiple bills for a customer, the control moves to DisplayBillPayList Site which displays the bill information. When the call returns multiple customers along with bill details, the control moves to SelectBillPayCustomer site to select the customer against which payment needs to be done.

Addition of a new Transaction Type BILL_PAY

A new transaction type BILL_PAY is added in Point-of-Service.

Following are the list of classes which are added or modified:

- TransactionConstantsIfc – A new transaction type TYPE_BILL_PAY is added along with TYPE_DESCRIPTORS and IXRETAIL_TYPE_DESCRIPTORS.
- BillPayTransactionIfc and BillPayTransaction which extends TransactionIfc and implements BillPayTransactionIfc are added in domain project.
- DomainObjectFactoryIfc – Adds two overloaded methods getBillPayTransactionInstance with locale object to get the instance of BillPayTransactionIfc.
- I18NDomainObjectFactory – Implements the overloaded method defined above with Locale object.
- TransactionWriteDataTransaction – This class performs the persistent operation on the POS transaction object. This class checks for the Transaction type and based on type, creates data actions and makes a call to server to save the transaction to the database.
- DefaultDataTechnician.xml – Add the operation class for the newly created dataAction.
- JdbcSaveBillPayTransaction - This is the new jdbc class which saves the bill pay transaction into the database.

Synchronous Interfaces (Reply/Request)

Bill Pay Retrieval Connector Service

The Technician Connector Service is the integration component to talk to BillPaySimulatorTechnician for the bill lookups. The class diagram for the new components is as shown below:

Figure 5–10 Static Diagram – Bill Pay Connector Service

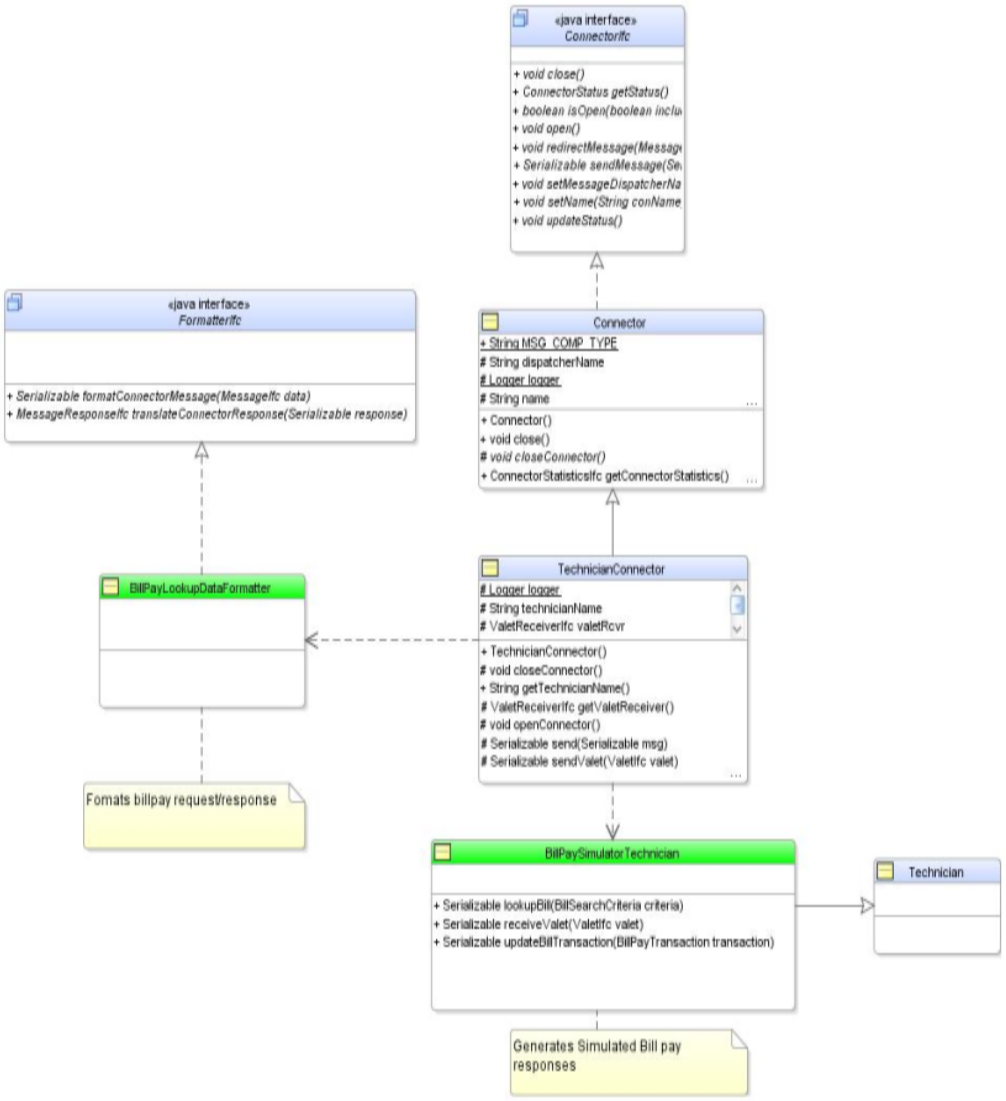
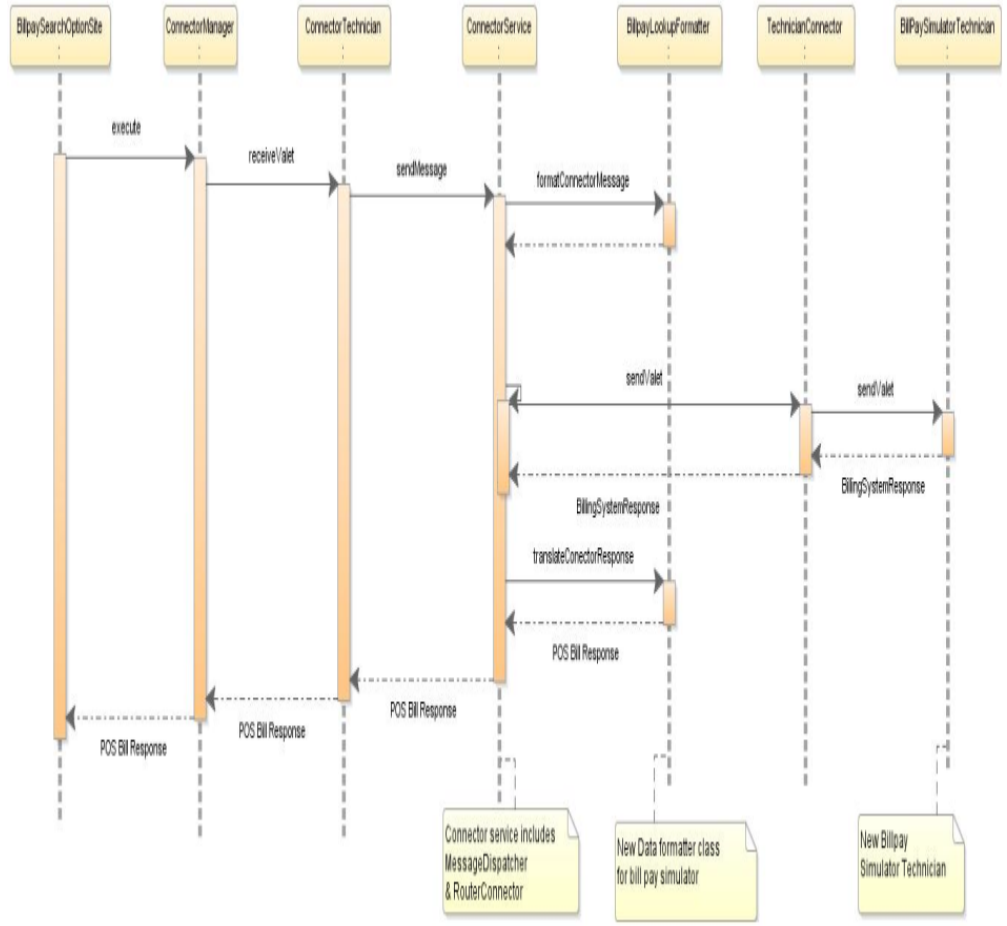


Table 5–2 Bill Pay Connector Service

Element	Description
BillPayLookupFormatter	The formatter class implements the FormatterIfc interface. The class is responsible to transform the request message to BillPayValet for bill lookups. This valet object is sent to the BillPaySimulatorTechnician for bill lookup. The class is also responsible to format the response from BillPaySimulatorTechnician to a POS format.
BillPaySimulatorTechnician	This is a Point-of-Service technician to simulate a billing system.

The bill pay lookup service interaction diagram is as shown below:

Figure 5–11 Interaction Diagram – Bill Lookup Connector Service



The ConnectorManager gets invoked from the BillPaySearchOptionsSite in the billpay tour. The request to ConnectorManager is sent with a requestType. The request from the ConnectorManager gets delegated to the ConnectorTechnician.

The ConnectorTechnician invokes the MessageDispatcher with the requestType that identifies which connector services the request from the connector configuration XML. The MessageDispatcher retrieves the connector from the list of configured connectors based on the requestType and invokes RouterConnector with the configuration.

The connector for bill pay is configured with the formatter as BillPayLookupFormatter and the connector as BillPayConnector. These classes are invoked by the RouterConnector in the sequence shown in the interaction diagram.

Bill Pay Transaction Update Connector Service

The Technician Connector Service is the integration component to talk to BillPaySimulatorTechnician for the bill pay transaction updates. The class diagram for the new components is as shown below:

Figure 5–12 Static Diagram-Bill Pay Transaction Update Connector Service

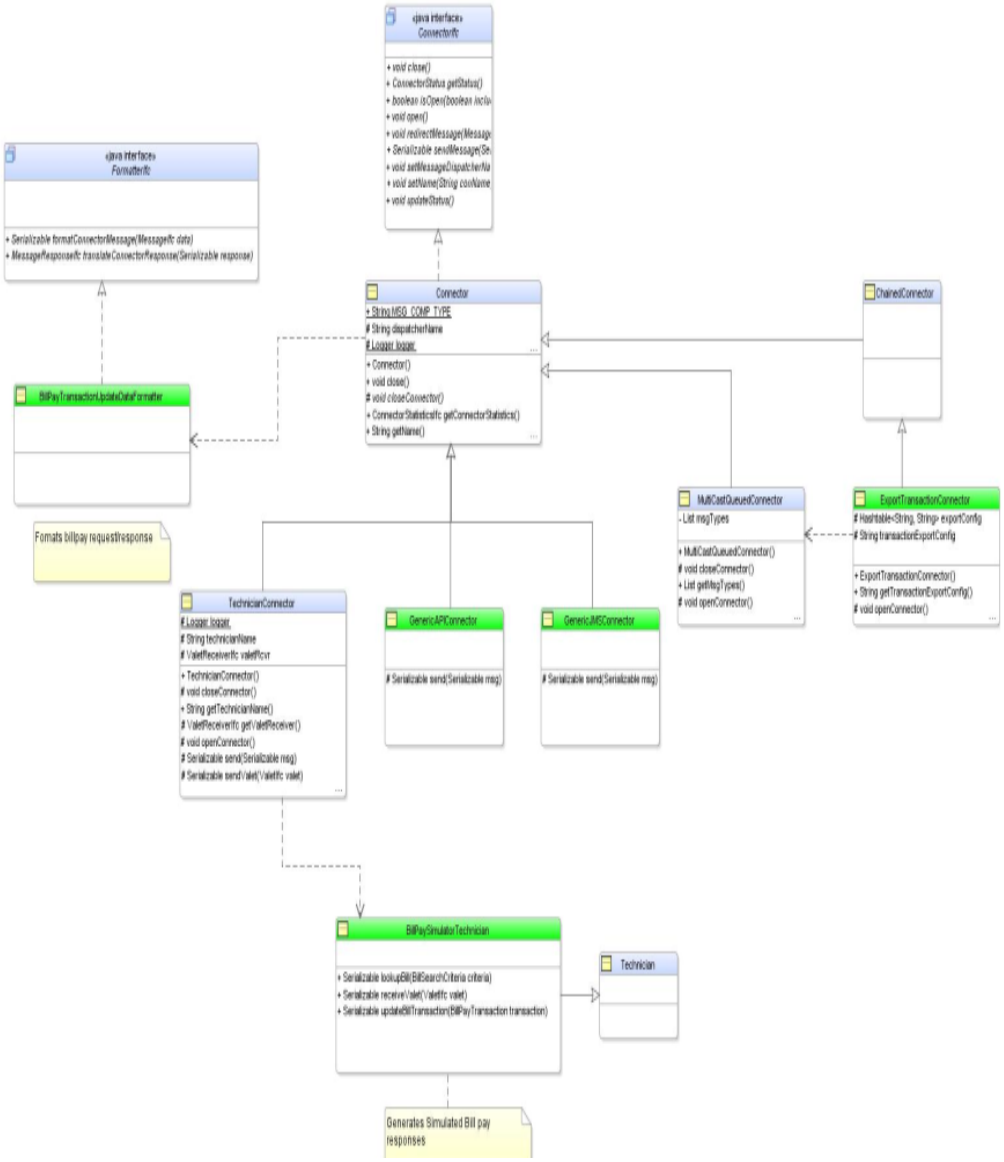


Table 5–3 Bill Pay Transaction Update Connector Service

Element	Description
BillPayTransactionUpdateDataFormatter	The formatter class implements the FormatterIfc interface. The class is responsible to transform the request message to BillPayValet for bill lookups. This valet object is then sent to the BillPaySimulatorTechnician for bill lookup. The class is also responsible to format the response from BillPaySimulatorTechnician to a POS format.
BillPaySimulatorTechnician	This is a Point-of-Service technician to simulate a billing system.
ExportTransactionConnector	This connector extends ChainedConnector and uses Multicast Connector to broadcast messages to different message routers.
GenericAPIConnector	This connector extends the base connector class and contains logic to invoke APIs exposed by the external systems.
GenericJMSConnector	This connector extends the base connector and contains the logic to invoke JMS services exposed by external systems.

6

Oracle Retail Point-of-Service to Oracle Retail Store Inventory Management Architecture

The Oracle Retail Point-of-Service-to-Store Inventory Management integration is intended to provide integration for the Point-of-Service application to interact with Store Inventory Management for inventory information. The following features are supported for integration with an inventory management system:

- Inventory Inquiry: This feature is provided to enable Point-of-Service to check the item inventory in Home Store, Buddy Store, Specific Store and Transfer zone. The Item Inventory feature is available to Point-of-Service Client only when the Point-of-Service Client is in the ONLINE mode.
- Item Basket: This feature is provided for line busting using the Store Inventory Management handheld. The items in a customer basket are scanned using the Store Inventory Management handheld and staged in the Store Inventory Management database. Point-Of-Service can then look up the basket details and add the line items to the sell item screen.
- Serial Number Validation and Update: Point-Of-Service supports serialized items. The operator is prompted to enter/scan the serial number of the serialized item on the Point-of-Service Client. The serial number that is entered is then validated by interfacing with Store Inventory Management. Once the transaction is tendered, the serialized items along with the captured serial number will be sent to Store Inventory Management for updating the status of the particular serial number.
- Inventory Reservation: Point-of-Service interfaces with Store Inventory Management to send the order transactions so that the items can be marked as reserved in Store Inventory Management. Also, once the items are picked up or delivered to the customer, the status needs to be updated in Store Inventory Management.
- Real Time Inventory Status Update: This interface sends Point-of-Service transactions to Store Inventory Management to update the inventory status based on the transactions.

The following outlines the Point-of-Service-to-Store Inventory Management integration approach:

1. Expose the inventory features from Store Inventory Management in the form of Web service.
2. Provide pluggable inventory Web service interface to integrate Point-of-Service-to-Store Inventory Management.

3. Point-of-Service Client interacts with Point-of-Service Server over RMI as in the existing Point-of-Service architecture. Point-of-Service Server interacts with inventory Web service interface to interact with Store Inventory Management.
4. Point-of-Service uses the connector framework to achieve a pluggable and extendable integration with Store Inventory Management.

The Point-of-Service-to-Store Inventory Management integration system is broken into five main sub-systems:

ORPOS Client

The various functionalities are incorporated in Point-of-Service Client by having new tours and new components, namely the ConnectorManager for interaction with the ConnectorTechnician.

Oracle Retail Point-of-Service Server

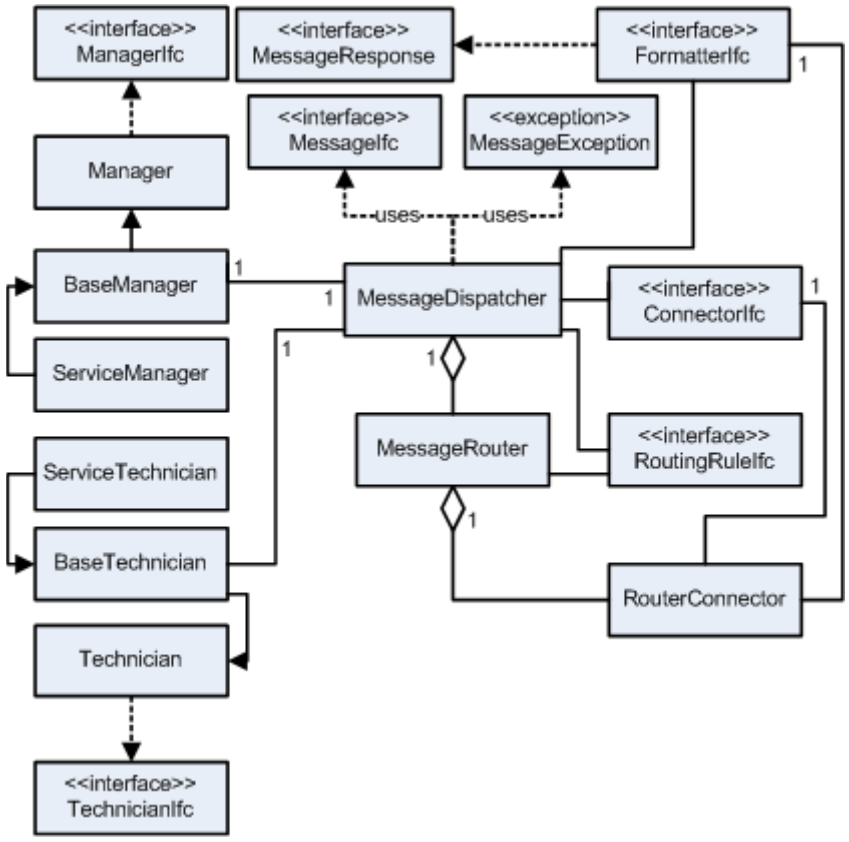
The Point-of-Service Server contains the connector framework which embeds the integration details. The connector framework is exposed through the new ConnectorManager and RetailTransactionTechnician. The connector framework consists of pluggable Formatters (request-response formatting) and Connectors (ORSIMWebServiceConnector) to abstract the connection-specific logic.

New Component ORSIMWebServiceConnector is added in Point-of-Service Server. PSITechnician interacts with PSIInventoryWS_Stub to call InventoryWS over intranet using HTTP/SOAP protocol.

Point-of-Service Connector Framework

Connector is an out of box integration framework. It provides a very extendable approach to the integrations both online and offline. The Point-of-Service Connector Framework model is as shown in Figure 6–1. The separation of concerns between data structure manipulation or transformation, and handling connectivity to a service is separated between the two components–the formatter and the connector.

Figure 6–1 Point-of-Service Connector Framework Model



The MessageDispatcher is the core of the communication framework. Its primary function is to dispatch messages to mapped routers. In addition, MessageDispatcher performs administrative and control operations on the associated connectors. When invoked, the MessageDispatcher delegates the message handling to a specific MessageRouter.

The MessageRouter coordinates the processing of a message using the associated routing rule and the RouterConnectors.

A RouterConnector provides an association between a message type, connector, and formatter. This decouples the formatting of the message from the chosen connector.

ConnectorIfc handles the communication between the application and the external service. It is responsible for locating the service, establishing a connection, and interacting with the service using appropriate protocols.

FormatterIfc translates the raw data from the message into the format expected by the external service. It also translates the response from the remote service into the format expected by the application.

Once a message has been sent with a request type to the MessageDispatcher it will get the instance of MessageRouter that is configured for that request type from the instantiated list. The processing is then delegated to the MessageRouter. The MessageRouter will route the request message to the list of connectors that are configured for that request. There can be multiple connectors that can be defined to process the same request message.

The connector framework provides all the building blocks to realize any integration requirement with a combination of connectors, formatters, ChainedConnectors, RoutingRules and JMX notifications. The XML configuration ties up the various blocks to implement any integration requirement.

Store Inventory Management Server
Inventory web service component deployed in Store Inventory Management server provides the entry point into the application for the various functionalities.

Store Inventory Management DB
Store Inventory Management inventory database.

Error Handling

Error handling is limited to logging errors during the inventory lookup. The exceptions such as IOException and invalidItem that occur during WSService communication are re-thrown as WSException, as well as logged for error tracking and resolution.

Logging

Point-of-Service-to-Store Inventory Management uses Log4J for logging. The following logging levels can be used:

- Info: For logging information messages.
- Debug: For logging all the debug messages.
- Error: For logging application errors.

The logging level can be configured with log4J.xml.

7

Configuring Multiple Printers for Labels and Tags

Oracle Retail Labels and Tags supports multiple printers that can be used for printing labels and tags. Users can select a printer from a list of printers on the Add Batch and Batch Detail screens.

To use multiple printers for printing labels and tags:

1. To enable users to select from a list of printers on the Add Batch and Batch Detail screens, set up the Allow Multiple Printers parameter. For information on the parameter, see the *Oracle Retail POS Suite Configuration Guide*.
2. Change the list of available printers in the `printers.properties` file. The instructions for adding printers are included in the file.
   - For Oracle WebLogic, the `printers.properties` file is located in the `<WebLogic_INSTALL_DIR>\user_projects\domains\<orlat-domain-name>\lib\properties.jar` file.
   - For IBM WebSphere, update the `<WebSphere_INSTALL_DIR>/profiles/<profile-name>/properties/printers.properties` file.

A

Appendix: Serial Numbers

Serial numbering is a system used by manufacturers to be able to trace the history of any finished good that reaches the customers. When customer complains of defective goods, knowing the serial number enables the manufacturer to find out where the raw materials were purchased, who was involved in each production step, as well as which distributors the goods were channeled by.

Retailers that sell such high-valued or high-risk items have to track unique numbers or attributes for a single item or a group of items. This enables the retailer to have a tight control over every unit of every item in the inventory. The sale/return process needs to capture the serial number of the items, reserve/reverse status of item in Oracle Retail Store Inventory Management and transmit the serial number to mark the item as sold to Store Inventory Management. The serial number of the sold item will also need to be transmitted with the transaction data to all the downstream applications that require Point-of-Service transaction data.

Point-of-Service will need to support sale of serial controlled items. The overall processing of a serial controlled item is broken into the following two parts:

- Serial Number Validation: When an item is scanned, if the UIN-required flag is set to **Yes**, the user will be prompted for the serial number. If the UIN capture time is set to **StoreReceiving**, then the serial number will need to be validated from Store Inventory Management. This process will need to be followed for the following transactions:
  - Normal Sale
  - Non-Receipted return transaction
  - Layaway Initiate
  - Sale initiated Pickup and Delivery
  - Special Order Pickup
  - Layaway Pickup if UIN was not captured at the initiation
  - Transaction Re-entry mode for all the above listed transaction
- Serial number status update: Serial number status will need to be updated in Store Inventory Management based on the stock movement. All the transactions listed in the validation step will need to be sent to Store Inventory Management for update. The following transactions must be sent as well:
  - Post void of the transactions listed in the validation step
  - Layaway delete
  - Pickup/Delivery order cancel

  - Post void of partial pickup of Layaway
  - Return with receipt

The impacts of the requirement are as listed below:

- The item scan process in the transaction listed for the serial number validation process will need to prompt the user for the serial number if the item is a serial controlled item. The serial number will be validated and upon the completion of the transaction, the inventory reserved in Store Inventory Management. Simply validating the serial number at item entry will not reserve that item in SIM.
- On completing the transactions listed in serial number update process, the serial number will need to be transmitted as part of the transaction information to all downstream applications such as to Central Office, a sales audit application, Store Inventory Management, and so forth.
- Point-of-Service will need to handle the scenarios when Store Inventory Management is offline.

Configuration

Enabling or Disabling Serialization Functionality

The property **SerializationEnabled=false** in `application.properties` file controls enabling or disabling of the feature. The Point-of-Service Client installer sets the value **true** or **false** based on whether the user selects the serialization functionality.

Serial Number Validation Process

The serial number validation process requires the Point-of-Service Client to connect to the connector framework using the Point-of-Service-to-Store Inventory Management Serial Validation Connector service.

The ConnectorManager in the Point-of-Service Client will delegate the validation request to the ConnectorTechnician (Entry point to the connector framework) that will integrate with Store Inventory Management Web service through the Point-of-Service-to-Store Inventory Management Serial Validation Connector service to get the serial number validated.

- On a scan of an item that is serial controlled, the user is prompted to enter/scan the serial number.
- The new validation tour will invoke the ConnectorManager. The connector framework is used for integration with Store Inventory Management and getting the result back to the tour. If the validation is successful the item is added to the **Sell item** screen. The transaction is marked as having serial controlled items. If the validation fails, the exception flow is executed.

For Sale Reversal flow (Return/Post Void) there will be no validation of serial numbers even when there is modification to the serial number.

The response from Store Inventory Management will result in one of the following actions in Point-of-Service:

- Allowed to Sell
- Not Allowed to Sell
- Conditional Sell – driven from the item attribute **External System create UIN**. If the attribute is set to not allow the creation of UIN then the process falls back to Not Allowed to Sell.

The statuses and errors sent from Store Inventory Management will map to one of the above actions in Point-of-Service.

IMEI Scan

The user can also scan the serial number directly on the sell item/return item/item inventory inquiry screen. IMEI scan increases the complexity of the item lookup process:

- The Sale and Return without receipt process prompts the user to enter/scan the item ID.
- The Point-of-Service Client will have a configuration to enable/disable the IMEI scan.
- The user enters/scans the IMEI number on the Sell item/Returns without receipt screen. The IMEI number will be assumed as an Item ID and the normal lookup in stores database will be done. The item lookup will fail if the number entered/scanned is an IMEI number.
- Both the processes will now be modified to check if the IMEI scan is enabled. If it is not enabled then the existing unknown item process flow executes.
- If the IMEI scan is enabled the control then moves to the new serial validation tour.
- The serial validation tour will invoke Store Inventory Management with just the IMEI number and Store Inventory Management will return with either one specific item ID or return the list of item Ids if more than one item is found with the same IMEI.
- If Store Inventory Management is offline then the control moves to a display error to the user and instructing the user to scan the item ID and then the serial number. Else the control moves to check if there is even one item that has a Sellable status. If none of the items returned from Store Inventory Management have a Sellable status then the control moves to a display error to the user and instructing the user to scan the item ID and then the serial number.
- If there are items in the returned response that have Sellable status then those items are looked up from the stores database. If none of the items are found in database then the control moves to a display error to the user and instructing the user to scan the item ID and then the serial number.
- If multiple items are returned a screen with the various items will be displayed to the user and the user will need to select the specific item.
- If only one item is returned or if the user chooses a specific item from the list the process will again execute the item lookup process.

The IMEI scan is also enabled for the item inventory inquiry process. The process to enable IMEI scan is as shown below:

- The current process prompts the user for item details to be entered and then does an item lookup in stores database. If the item is not found then the user can choose to search for the item in the Buddy Store or the Transfer Zone.
- The change to the process will be done to check if IMEI scan is enabled when the item lookup fails. If IMEI scan is enabled then the serial validation tour is invoked to get the items corresponding to IMEI from Store Inventory Management. The rest of the process is as already mentioned for the sale and returns without receipt process.

Enabling or Disabling IMEI Functionality

The property **IMEIEnabled=false** in `application.properties` file controls enabling or disabling of the feature. This feature is not set by the installer and needs to be configured post-installation.

Serial Number Update Process

The serial number update process is a background process. The process runs once the transaction is tendered and persisted to the stores database. The process requires the Point-of-Service Server to connect to the connector framework through the RetailTransactionTechnician (entry point to the connector framework for transaction object).

- The transaction data is posted by the Point-of-Service Client to the Server for saving the data to the database. The transaction data is received by the RetailTransactionTechnician (RTT) in the Point-of-Service Server. The RTT forms the entry point into the connector framework (CommExt).
- The RTT persists the data to the database and then queues the data for later delivery to Store Inventory Management. Once the data is read from the queue it is then picked up by the Point-of-Service-to-Store Inventory Management Transaction Update connector service. The formatter associated with the service creates the transaction post request in the format Store Inventory Management requires. If a serial number was captured by Point-of-Service for any item within the transaction, those items are sent along with the serial number.
- Store Inventory Management updates the status of the serial numbers once the data is received at Store Inventory Management.

Glossary

Batch

A collection of data operations that are processed during import at one time. The size is determined by a configurable parameter.

Bundle

A collection of import files, one file per data type, stored as a compressed archive containing a manifest. It is expected that the retailer or implementation team is responsible for delivering to the Store the bundle along with manifest for all data feeds to the Store. MOM applications can package the bundle but do not provide delivery functions.

Corporate

Used interchangeably with *enterprise*. The enterprise environment of the retailer where enterprise applications are deployed. Oracle Retail Central Office is deployed in the enterprise.

Data Access Object (DAO)

A Java class that can retrieve and persist data to and from a data source. DAO is well-known JEE development pattern.

Data Distribution Infrastructure (DDI)

The infrastructure and application components that are responsible for distributing seed data from enterprise applications to Store applications, ODS at Corporate (or enterprise), and Store Database at the stores.

Data Transfer Object (DTO)

A class that contains data records from a received payload. The DTO's attributes are populated with the parsed data.

DIMP

Data Import

Incremental

There are two types of update operation, full incremental and delta incremental. Full incremental assumes that all the fields for a data type are supplied in the XML. A delta incremental import contains only the fields that are being changed.

ISP

In-Store-Processor

JEE/J2EE

Java Enterprise Edition (formerly Java 2 Enterprise Edition) is a set of APIs designed to support tier 1 type business models.

Java Database Connectivity (JDBC)

An API used to communicate with relational databases.

Kill And Fill

Kill And Fill refers to a data operation where all the existing data in a table is deleted (kill) and then replaced with new data (fill).

Limit (discount rule)

The maximum price allowed for a source or target to be part of a deal. Used most often when the source or target is a classification or department where many different priced items exist.

Manifest

A file within a bundle that lists the data files in the bundle and their interdependencies.

Minimum Data

Minimum Data is defined as the minimum set of data necessary to support the deployment of Stores applications only.

If the user attempts to select any function or log in, an error may occur in the application without Sample Data loaded. See Sample Data.

Operational Data Store (ODS)

The corporate data repository that services Oracle Retail Central Office.

ORBO

Oracle Retail Back Office

ORCO

Oracle Retail Central Office

ORLT

Oracle Retail Labels and Tags

ORPOS

Oracle Retail Point of Service

ORRM

Oracle Retail Returns Management

ORSIM

Oracle Retail Store Inventory Management

POS Suite

The Oracle Retail business unit that assumes responsibility for applications running in the Store environment.

ReSA

Oracle Retail Sales Audit

RMS

Oracle Retail Merchandising System

RPM

Oracle Retail Price Management

RTLog

Retail Transaction Log

Sample Data

A set of data used to demonstrate application features.

Store Applications

Oracle Retail applications that run in the store environment. This includes:

- Oracle Retail Back Office
- Oracle Retail Point-of-Service
- Oracle Retail POS Suite
- Oracle Retail Labels and Tags
- Oracle Retail Store Inventory Management
- Oracle Retail Central Office
- Oracle Retail Returns Management.

It must be noted that even though Oracle Retail Central Office runs in the corporate environment, it is classified as a store application.

Store Database (SDB)

The data repository for store applications.

Threshold (discount rule)

The minimum price allowed for a source or target to be part of a deal. Used most often when the source or target is a classification or department where many different priced items exist.