Oracle® Retail XBR*i* Cloud Services

Implementation Guide
Release 16.0
E81383-05

June 2018

ORACLE

Oracle® Retail XBRi Cloud Services Implementation Guide, Release 16.0

E81383-04

Primary Author: Barbara Clemmer-Dunn

Contributors: Nishant Sethi, John Bonin, Karen Bagdasarian, Kim Payson

Contents

Send Us Your Comments

Oracle Retail XBRi Cloud Services, Implementation Guide, Release 16.0

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 Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

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

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

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

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

Preface

This Implementation Guide describes the requirements and procedures to complete manual post-installation and configuration of this Oracle Retail XBR^{i} Loss Prevention Cloud Services release.

Audience

This guide is for the following audiences:

- System administrators and operations personnel
- Integrators and implementation staff personnel

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Retail XBRi Cloud Services Release 16.0 documentation set:

- Oracle Retail XBRi Cloud Services Administration Guide
- Oracle Retail XBRi Cloud Services Release Notes
- Oracle Retail XBRi Cloud Services Web User Guide
- Oracle Retail XBRi Cloud Services Administrator User Guide

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times **not** be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site,

or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

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

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

Oracle Retail Documentation on the Oracle Technology Network

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

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

(Data Model documents are not available through Oracle Technology Network. You can obtain them through My Oracle Support.)

1
Introduction

Oracle Retail XBR^i Cloud Services offer Business Intelligence (BI) reporting and analysis modules in the areas of Loss Prevention (LP) and Sales and Productivity (SP). Retailers can purchase just the LP Module, or have both LP and SP bundled into a single application.

The XBR^i LP module is the world's most widely used loss prevention and store data analysis tool. It uses exception based reporting methods to easily identify, track, and respond to store events. The intent is to detect, investigate, and reduce losses from fraud and noncompliance. Using advanced exception based reporting (EBR) techniques. Oracle's loss prevention solution analyzes transaction data from many aspects and identifies patterns that may indicate fraud or theft. The EBR solution issues alerts, and recommends further exploration based on the likely severity of the wrong doing.

The XBR^i SP module offers robust and highly configurable reporting across all levels of the retail organization hierarchy (Salesperson, Store, District, Region, and so on), merchandise hierarchy (item, class, dept., and so on) and/or by geographic attributes. Through a comprehensive set of grid and graph reports, documents and interactive dashboards, users can compare same store sales to past performance and custom goals, measure sales members' productivity, and evaluate the impact of merchandise characteristics on productivity.

Scope of this Document

This document applies to an implementation environment only. Any actions pertaining to staging should be handed to the Cloud team.

Cloud Components

There are several key components included in an XBR^i installation. The business intelligence server, an application server, an Oracle database, a Secure File Transfer Protocol (SFTP) server and an Extract, Load, Transform (ELT) server hosing Oracle Data Integrator (ODI) or XBRLoader web service or both. The XBRLoader Web Service is only used for XSTORE- XBR^i customers. (XSTORE 7.0 and higher). In a Cloud implementation, the necessary components are pre-installed.

Cloud Environments

When XBR^i is implemented in the Cloud, there are three environments: Implementation, Staging and Production. All environments are provisioned by the Cloud Team.

> **Note**: The Implementation and Staging environments will only be available as long as needed for their respective tasks.

Implementation Environment

In the Implementation environment, partners code and configure the ELT component to bring in both Point of Sale (POS) and supplemental data feeds. Partners also configure the database specific to the customer's business requirements and configure the business intelligence front end. Unit testing is performed in the Implementation environment before promoting configurations to the Staging Environment.

Proposed Environment	
XBRi Implementation Server	- Oracle Linux 7.0 - hardened - Oracle XBRi App Version: -16.0 -Tomcat -Apache - ODI 12.2 - JDK - Version required by applications - Oracle DBMS client -core DTV structure - XBR Loader webservice

Non-Production Environment

In the Non-Production Environment the customer is engaged to perform User Acceptance testing. In this environment access is limited to the XBR^{i} application front end. More information on the deployment process and how to address customer issues in this environment is covered in Chapter 11: Troubleshooting.

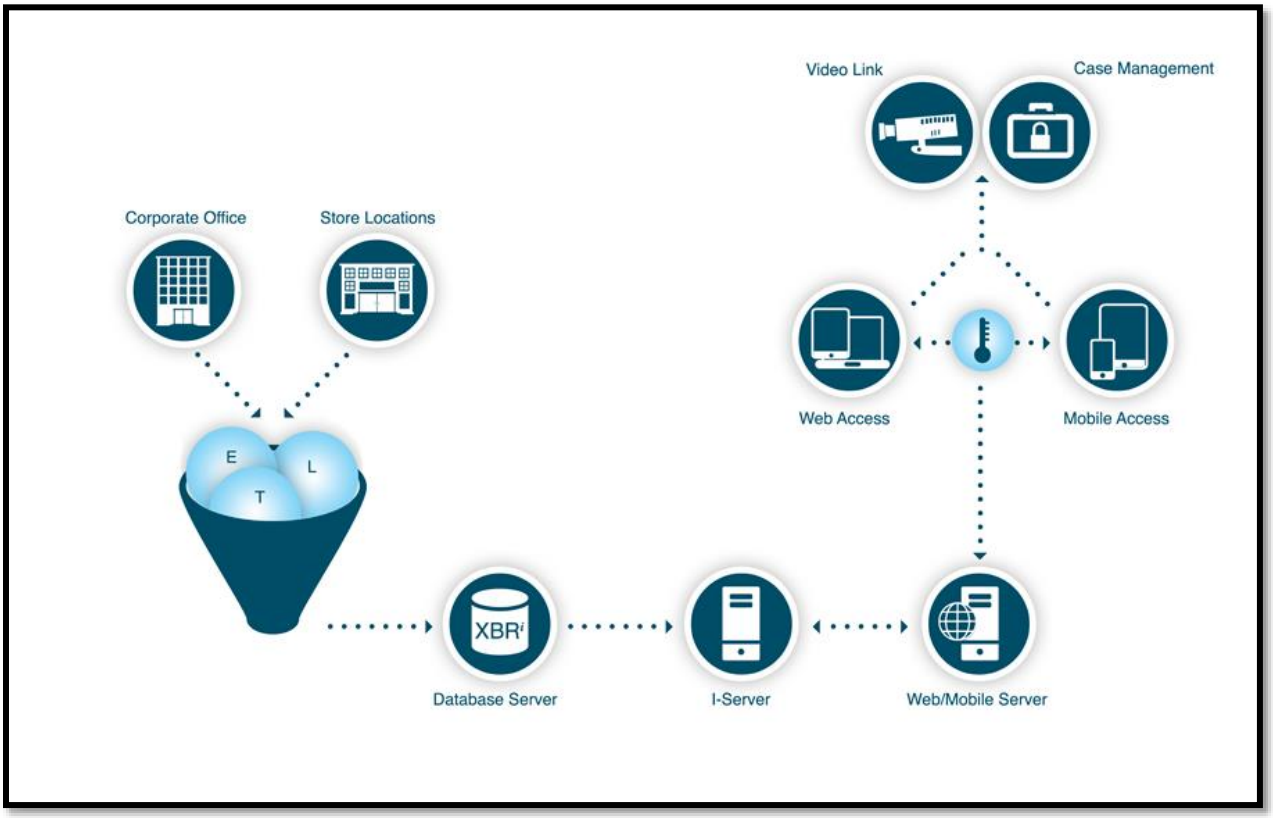
Production Environment

Production is the customer's go live environment.

Data Flow

At a high level, data flows into XBR^i from the POS system and other corporate data warehouse repositories. The customer is responsible for delivering data feeds, both transactional logs and supplemental files, to a SFTP site. The ELT process, built on Oracle Data Integrator (ODI), extracts and transforms the files and loads the data to staging tables on the XBR^i database server. A set of stored procedures moves transactional data to historical tables. Supplemental data feeds are loaded to temporary tables in the XBR^i database by the ELT and then moved to the core master file tables through additional stored procedures. Some stored procedures are responsible for populating the lookup tables as well. Stored procedures are covered in more detail in Chapter 2, The XBRi Data Model. The workflow for all this movement is controlled by the ODI ELT framework.

Once data has been loaded to the XBR^i database, it is available for viewing by the XBR^i application.

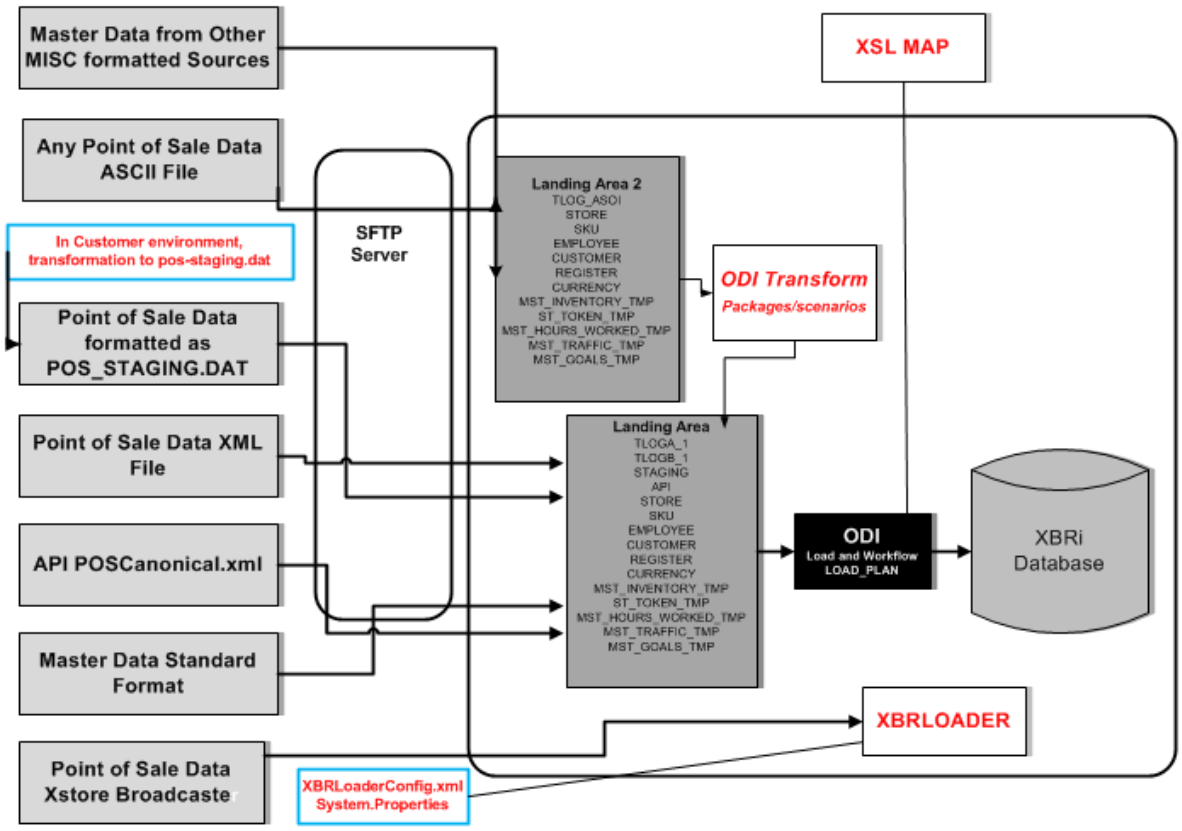


XBR^i Data Flow

Integration Components

For the Data Flow to succeed and accurately present data in XBR^i, several components must be configured by the person responsible for the integration. Configuration is not limited to settings through a front end user interface. Most of the XBR^i integration is performed through creating database scripts, command tool scripts, and one of the following: ODI maps, an XSL map, or an XPATH map. Before focusing on any of these tasks, you should complete a thorough review of the customer's business, data feeds, and transactional logs.

In the diagram below, the integration components are displayed in white boxes with dark font. Note that only one transactional data mapping component may be required.



XBR^i Integration Components

Transactional Log Delivery Method	Required Mapping
ASCII File – any POS	ODI Map
POS_STAGING.DAT	N/A (Done outside Cloud)
API POSCanonical.xml	N/A (Done outside Cloud)
Ant XML format	XSL Map
XStore - Broadcaster	XBRLOADERConfig.xml

Creating Database Scripts

PRO_VIEW_SYNTAX

Create a script to modify the fields in this table as required to achieve the required aggregation. This is covered in more detail in Chapter 4, XBR^i Statistics Aggregation.

PRO_SP_VARIABLES

Create a script to set the database variables appropriately for the customer's implementation. For example, cashier uniqueness is defined in this variable table. The variables in this table are used by the stored procedures that are executed in the database. The fields in this table are defined in the XBR^i Cloud Services Core Field Mapping Guide.

PRO_REQUESTOR

Create a script to activate/deactivate the specific requestor. This table must be in agreement with the variable settings in sp_pro_variables. More details will be provided in Chapter 2, the XBR^i Data Model.

POS_STAGING_HDR_LOAD

Create a script to modify this view if required. This view is responsible for setting various flags, and occasionally needs to be adjusted to accommodate a customer's business requirements. Typically, custom reccodes on tenders will necessitate a change to this view. Custom reccodes are discussed in Chapter 2, the XBR^i Data Model.

Creating Command Tool Scripts

A set of sample Command tool scripts are available for implementing configurations. They can be modified as required by the customer and allowed by Oracle. The allowed customizations are described in Chapter 9: Customer Configuration.

Creating ODI Map for Supplemental Data Feed

If a supplemental data feed is delivered in a non-standard Oracle format, you must create an ODI package/scenario that transforms the data into the standard format required by the ODI ELT black box framework. This scenario must be scheduled through ODI. The unformatted supplemental data feed must be processed and then delivered to the core ELT INCOMING_FILES directory structure. It cannot be loaded to the database directly because this separate process will not have visibility in the workflow, and loading the master file data while the database stored procedures are running would be problematic.

> **Note**: This is not required if supplemental data feeds are in standard Oracle format.

2

The XBR^i Data Model

Master File Data

Master File data is used to supplement data that may or may not be present in a transactional log file. The front-end reporting uses the data in the associated tables to enhance the reports and to provide lookups. The core Cloud implementation requires master file data to be delivered in a standard format as defined by Oracle. This standard format is an ASCII Pipe-Delimited file. The fields in the file are defined by the layout of the corresponding table in the XBR^i database. See the XBR^i Cloud Services Core Field Mapping Guide, available on My Oracle Support (MOS) for the required layout.

The master file must be delivered with a trigger file, which is a file with the identical name with the .trg extension added. Master files are first loaded to a _TMP version of their destination table in the database. A stored procedure moves data from _TMP to _TAB. In addition, the stored procedure responsible for moving data from _TMP to _TAB back fills the tables based on data in the POS_STAGING table.

For example, if a new store (44) opens and the transactional data has data from store 44, but the last time the store master was delivered it did not contain store 44, Store 44 will be added to MST_STORE_TAB by the backfill process. The name of the store will be "not on file." This is how you can identify in the database that the back fill process was used.

Points of concern are:

- Is the key field unique? The framework does not account for duplicates. The ODI ELT framework will load the duplicates to the _TMP table and then during the move from _TMP to _TAB one of the records will be moved.
- The master SKU tables requires SKUs to be unique across the chain.
- The master file data provided by the customer must agree with the data in the transactional log. For example, if the cashier number in the transactional log is eight characters long with leading zeros, the cashier number in the employee master file must be the same.
- If the customer cannot provide data in the Oracle standard format, as the consultant, you will need to transform the master file data using a tool of your choice in the customer environment or manipulate the data in a separate ODI scheduled scenario.

Transactional File Data

Transactional file data is the output of the POS system. These files are referred to as TLOGs or POSLogs depending on the POS vendor. They can be delivered as XML, ASCII or binary files. In the case of a binary file, the translation to an ASCII file must be done outside of the Cloud. The format can vary widely and is never a one-to-one relation with the XBR^i data model. The terminology used to describe functionality can also vary from vendor to vendor.

The standard format of transactional data can be found in the XBR^i Cloud Services Core Field Mapping Guide. When transactional data is delivered in the standard format as a pos_staging.dat file, the fields should have been populated as described in this chapter. The POS Staging Table Layout section in Appendix A of the XBR^i Cloud Services Core Field Mapping Guide, which is available on MOS, basically mimics the format of the pos_staging table in the database. If mapping XML files, the POSCanonical.xsd file should be used as the target when creating the appropriate XSL mapping. The purpose of the POSCanonical.xsd file is to normalize the XML transactional log to an XBR^i data model XML file. POSCanonical.xsd will be provided.

Transaction Types

Transaction Type (TRANSTYPE) describes the overall activity that is taking place within a transaction.

SALE

Defined as at least one non-voided sales line item (merchandise or non merchandise) with no net returned line items (a return item that has been line voided results in no net returns).

This includes a sale of just a gift card.

RETURN

Defined as at least one non-voided returned line item (merchandise or non merchandise) with no net sales line items (a sales line item that has been line voided results in no net sales).

If the only thing returned is a gift card, this is usually considered to be a tender exchange rather than a return.

If merchandise is returned and a gift card is returned but nothing is sold, the trans type is RETURN, but the gift card sold should be flagged in attribute_code3 as a gift card cashed.

EXCHANGE

Defined as at least one non-voided sales line item (merchandise or non merchandise) and one non-voided returned line item.

Exchange In: defined as an exchange where the net tender is positive (that is, in to the store). Or money in (Exchange MI)

Exchange Out: defined as an exchange where the net tender is negative (that is, out to the customer). Or money out (Exchange MO)

Exchange Even: defined as an exchange where the net tender is 0.

NOSALE

This corresponds to the function key on most registers that allows a cashier to open the register's drawer without processing a transaction. This is often used to make change for a customer.

POSTVOID

This refers specifically to a transaction that is voiding a previously completed transaction.

It could have a TRANSSTAT of COMPLETE, SUSPEND, CANCEL or AUDITVOID. The transaction that is being voided by the POSTVOID is indicated with the TRANSSTAT of POSTVOID.

PAYMENT (House payments only)

Defined as a transaction in which a customer is making a payment to reduce the balance on their House (store) charge / credit card account.

PETTYCASH

Defined as transaction used for paying small charges, as for minor office supplies or deliveries, often referred to as PAYIN or PAYOUT.

TNDEXCH

This refers to a transaction in which the customer is exchanging one tender (for example, check, gift certificate, merchandise credit, or gift card) for cash. For example, a customer tenders a check and cash is given back. No items are purchased.

ADMIN

For an XBR^i implementation, this is used for enter training, exit training, price inquiry, clock in, clock out and time card maintenance transactions only.

UNKNOWN

If it is not possible to determine the transaction type of a transaction that would normally be processed, the transaction type will be defaulted to UNKNOWN.

Transaction Functionality

Points of sale are highly customized. Based on the business, certain functionalities are enabled. A grocery point of sale for instance has more functionality than a retail point of sale. Grocery points of sale usually allow for bottle slips, food stamps and WIC as tenders, for example. Coupon functionality is more complex at a grocery store.

Voiding is a functionality shared by points of sale. In some points of sale the fact that a tender, item, or discount is voided does not show up in the tlog. In these cases, the original item is not present either. In XBR^i, a record that is voided and visible in the tlog requires two records: a voided record and a voiding record. These records contain the same information, with the exception that the void code is different and the signage is different. The purpose of the voiding record is to ensure the transaction balances out.

Discounts are an area where the functionality varies even within the same point of sale. Work closely with the customer to understand their use of discounts, price overrides and coupons. One thing to note is that the discount amount field is populated at detail in both a LDS/TDS record and the actual item (SKU) record. Pay attention to selecting the proper field/tag for the reason_type field. In certain points of sale there is a reason_type and a discount number. For these customers the discount number should be posted to the accountnum field.

Layaways and Special Order transactions, although they are both Sale transactions, typically involve multiple transactions to accomplish one sale. There will be a transaction when the Layaway is initiated "initial," there may or may not be a fee or a deposit, or there could be multiple transactions each making partial payments. Eventually the items will be picked up. It is important to understand exactly when the customer recognizes the revenue on a Layaway transaction.

E-commerce functionality must also be understood. E-commerce transactions commonly have an invoice number associated with them in addition to a transaction number. E-commerce transactions are best posted to XBR^i at the point the item has shipped. Returns in an e-commerce transaction, if not coded for properly, will often appear to be Exchanges, since a return in e-commerce has both an item returned and a shipping fee, which will appear to be a "sold" non-merchandise item. You must consider the special requirements of each type of business.

> **Note:** It is important to understand the capabilities of the Point of Sale and how to translate this information into XBR^i properly.

Table 2.1: Required TLOG Data Elements

Data Element	Description	Functionality Provided
Trans Number	POS transaction number	Reporting of Transaction details
Trans Date	Date of the POS transaction	Reporting by date and date range
Trans Time	Time of the POS transaction	Reporting by time of day
Store	Store number where the transaction occurred	Reporting by Store
Register	Register number that logged the transaction	Reporting by Register
Cashier	Cashier number that created the transaction	Reporting by Employee

Trans type	Identifies the type of activity within the transaction*	Provide transaction definition
Trans Status	Identifies if transaction was completed: Complete, Canceled, Suspended, Post Voided	Transaction Status
Record Type	Indicates type of record: header, merchandise, non-merchandise, line discount, transaction discount, tax, tender, and so on	

TLOG Record and Data Elements

A record consists of one or more fields. An example of a record would be an Item record. An item record may contain fields such as selling price, list price, quantity, and so on

Within these records you can derive various data elements. Related data elements need not necessarily reside in the same record. For instance, if a credit card number resides in the next or previous record to the record containing the amount, you can properly associate that information to the correct dollar value. However, if there is no logical relationship, either by a cross-reference to line sequence number or a positional (next record/ previous record) relationship, it would not be possible to make that connection. In many POS systems there are records that combine disparate data elements. For example, Tax, Tender, or both might appear on the same total record.

If the value of an amount field is captured as an absolute value, there must be some indicator within the record or transaction that would define whether it is positive or negative. Otherwise, the sign of the amount is assumed to be correct as logged.

Mapping Transactional Data to POS_STAGING Table

All transactional data is mapped to the POS_STAGING table in the XBR^i Database. A set of stored procedures (see: XBR^i Database Stored Procedures) move the data to historical tables for viewing in the XBR^i application. Detailed mapping instructions are found in the XBR^i Cloud Services Core Field Mapping Guide.

Header Records

Header records contain summary data about a single transaction or contain information and attributes that apply to the transaction as a whole.

- The data mapped to a Header record will vary by Transaction Type.
- The header record type (RecType) for any transaction is always HDR.

Detail Records

The detail level indicates the specific parts or elements within a transaction.

The data mapped to a Detail record will vary by Record Type (RecType) and Transaction Type.

The detail record types are identified as the following:

- Line Item (SKU, LAYSKU, SOSKU)
- Non Merchandise (NM, LAYNM, SONM)
- Line Discount (LDS, LAYLDS, SOLDS)
- Transaction Discount (TDS, LAYTDS, SOTDS)
- Sales Tax (TAX, LAYTAX, SOTAX)
- Tender (TND)
- Petty Cash (PTC)

Mapping Detail Records Items

Line Item (SKU/NM)

The SKU is the most basic element of the POS transaction. PLU (price lookup) Price, Final Price, Price Override Amount and Discount Amount are not always explicitly stated in the transaction log (TLOG). It is often possible to derive the value of one or more of these fields from other information present in the TLOG. For example, if a Discount or Price override amount is available, it is sometimes possible to derive the PLU price by working backwards. That is, PLU Price = Final + Discount + Override.

> **Note:** PLU Price is captured in the original amount field.

The Non Merchandise record, NM, is used to represent the purchase of non-tangible items. This may include activity such as:

Fees: for example, shipping fees, layaway fees, check cashing fees

Services: for example, alterations, warranties, subscriptions

Other non-tangible charges or activities: for example, bottle deposits, house card payments, and so on.

When these are not present as separate records in the TLOG, you must create non-merchandise records to represent layaway deposit, layaway payment and house payment records in order to have a record to balance out the tender paid within these transactions.

DETAIL RECCODES	
SKU: Item Type	
00	Undefined
01	Regular
07	Layaway Item
08	Special Order Item
NM: Non Merch Type	
02	Non Merch (Non Fee)
03	Fee
04	Bottle Deposit
05	Layaway Deposit
06	Layaway Fee
09	Spec Order Fee
10	Layaway Payment
11	Spec Order Payment
12	Rounding record
13	Spec Order Deposit
14	House Card Payment

Items and Reccode Assignment

ATTRIBUTE_CODE2	
SKU / NM:	Activity Attributes
1	Raincheck Item
2	Dept Key
3	Price Override

Items and Attribute Code2

ATTRIBUTE_CODE3	
SKU, NM:	Item Detail Attributes
1	Phone Card
2	Phone Card Activation
3	Alcohol (primarily used for Grocery)
4	Tobacco (primarily used for Grocery)
5	Western Union sold
6	Lottery (primarily used for Grocery)
7	Money Order sold
8	Tender Cashing Fee
9	
10	
11	Misc
12	
13	NSF Fees (if treated as Non Merch)
14	Mall Card
15	SPIFF (SPIFF PTC records only)
20	Session Summary[1]
21	Store Summary[1]

Items and Attribute Code3

Mapping Detail Records Discount

Line Item Discount

In some POS systems, line item discounts are reported in separate records apart from the SKU. Typically, a discount record is written out immediately before or after the SKU line.

Transaction Discount

Transaction discounts differ from Line discounts in that they are not associated with a particular SKU but apply to the entire transaction.

If a transaction discount is pro-rated back to the line item, the discount should be treated as a line discount and will be posted to the configuration results as an LDS record type. However, if there is an indicator in the tlog that the line discount is due to a pro-rated transaction discount, the reccode is posted starting with a T rather than a D.

Coupons

Store Coupons (that is, coupons that are applied prior to tax) are posted as discount records.

Manufacturer Coupons (that is, coupons that are applied after tax) are posted as tender records. This information is in the Tender record section of the SALE, RETURN, EXCHANGE information chapter in the XBR*i* Cloud Services Core Field Mapping Guide.

Promotions/Deals

Mapping Detail Records Tender

Tender (TND)

Every tender record in a tlog is associated with a tender ID that identifies the type of tender, for example, cash, check, debit card, and so on. In order to build a standardized query library, the tender IDs from the tlog are translated into standard tender types for the data model.

If a POS has tenders that are outside of the XBR[i] standard list, custom tender types are assigned within the code range of 500 – 599.

DETAIL RECCODES			
TND: Tender Type			
400	Unknown	435	JCB
401	Cash	436	EBT Cash
402	Foreign Currency (not in use)	439	Unknown Credit Card
409	Debit	440	EBT Foodstamps
410	Check	442	Paper Foodstamps
411	Traveler's Check	443	WIC
412	Merch Credit (paper)	444	Pharmacy
413	Store or Corp Check	445	Bottle Slip
415	MFR Coupon	446	Coin Slip
416	Gift Certificate (paper)	447	Payroll Check
417	Mall Gift Certificate	448	Lottery Ticket
418	Gift Card	449	Western Union
419	(Reserved for LW restocks)[4]	450	Enroute Card
420	House Card	451	Government Voucher
421	(Reserved for SO restocks)[4]	452	Account Receivable[4]
422	Merchandise Credit (plastic) card	453	Float
430	Visa	454	Foreign Currency Exchange
431	Master Card	455	Money Order
432	AMEX	456	Mobile Phone
433	Discover	457	PayPal
434	Diner's Club		
	[4] May be used to create a tender if one does not exist in tlog.		
All Code Fields - Codes 500-599 are Reserved for Customization			

Tenders and Reccode

Attribute Code Fields

The Attribute Code fields in the POS_STAGING table are used to identify the type of usage of a particular tender. They are emphasized here because the proper population of these fields affects the aggregation of statistics.

ATTRIBUTE_CODE1		
TND: Tender Code1 Attributes		
301	Exchange In	
302	Exchange Out	
303	Exchange Even (only if TND records post to tlog)	

Tenders and Attribute Code1

ATTRIBUTE_CODE2	
TND: Tender Activity Attributes	
100	Coupon
101	Double Coupon
102	Triple Coupon
103	Don't use
104	Quadruple Coupon

Tenders and Attribute Code2 (Manufacturer's coupons are treated as tender)

TND: Tender Detail Attributes Attribute Code 3			
20	Session Summary[1]	217	Merch Credit Issued
21	Store Summary[1]	218	Merch Credit Cashed
201	Regular Deposit	219	Merch Credit Redeemed
202	Other Deposit	220	Misc Tender Cashed
203	Noncorr Coupon[6]	221	Gift Card Issued
204	Correlated Coupon[6]	222	Gift Cert Issued
205	Gift Card Sold	223	Lottery Redeemed
206	Gift Card Cashed	224	Lottery Cashed
207	Gift Card Redeemed	225	Cashback on Debit Card[3]
208	Gift Card Reloaded	226	Cashback on Checks[3]
209	Gift Cert Sold	227	Cashback on Credit Card[3]
210	Gift Cert Cashed	228	Begin Count in TNDCNT only
211	Gift Cert Redeemed	229	End Count in TNDCNT only
212	Bottle Slip Issued	230	Tender Redeemed
213	Bottle Slip Cashed	231	Tender Issued
214	Bottle Slip Redeemd	232	Tender Cashed
215	Coin Slip Cashed	234	Tender Reloaded
216	Coin Slip Redeemed	235	Till Audit in TNDCNT

Tenders and Attribute Code3

Gift Card and Gift Certificate Guidelines

In the XBR^i data model, gift cards are represented as tenders. The reasoning is based on the liability associated with gift cards. The same is true for gift certificates and merchandise credits. Attribute Code3 further defines the activity involved with this tender.

A gift certificate/card redeemed (either in a tender record or as a returned gift card in a line item or non merch record) in a transaction in which items are sold is considered a certificate/card redeemed.

A gift certificate/card redeemed (either in a tender record or as a returned gift certificate/card in a line item or non merch record) either:

- By itself with no other line items or non merch items is considered a gift certificate/card cashed.
- With returned line items or returned non merchandise only (that is, no line items or non merchandise sold) is considered a gift certificate/card cashed.

Mapping Detail Records Tax

Depending on the POS, a sales tax record may contain the sales tax for the entire transaction or it may contain the tax for a single line item.

Typically, in the United States, a single tax record is posted for the entire transaction except in the case of layaways and special orders where a tax line for each line item is preferred so that individual taxes can be recognized along with the associated line item.

Canadian POS's usually post the individual taxes per line item and generally prefer for taxes to be broken out by federal versus province (for example GST, PST, HST, QST).

For countries with a Value Added Tax (VAT), the tax may already be incorporated into the price of the item. If this is the case, do not post a tax record, as this will throw the transaction out of balance. An exception to this might be a transaction keyed in a country with VAT tax but to be delivered to a country that does not have VAT tax. In that instance you might need to post a tax record for a negative tax amount that represents the tax being deducted from the transaction total.

The RECTYPE field is populated with TAX, LAYTAX, and SOTAX to identify it as a tax record.

The RECCODE field is populated to identify the type of tax.

TAX: Tax Type populated in RECCODE field	
300	Regular Tax
301	Canadian Federal - GST (VAT)
302	Canadian Provincial - PST
303	Canadian Harmonized - HST (VAT)
304	Layaway Tax (*)
305	Special Order Tax (*)
306	Canadian QST/TVQ - Quebec
307	VAT tax when extended amount
	of item has Vat Tax removed.

Tax Type in Reccode Field

The extended amount field is populated with the tax amount associated with the record, whether it be a transaction tax or a line item tax. Pay particular attention to the signage of the amount of this tax. For example on a return the tax is leaving the store, so it should post as a negative amount.

Tax exempt is not handled on a tax record but at the header or detail item record when the local, state or federal tax authority issues a tax exempt number to an individual, an organization or a business. A Tax Exempt transaction or item is a transaction containing one or more qualifying taxable items that is sold to a customer without tax when the customer has a tax exempt status. In XBR^i the Tax Exempt ID provided by the shopper is posted in the ATTRIBUTE_CHAR2 field of POS_STAGING for the individual item detail record or at header if the entire transaction is tax exempt.

A tax can be overridden and this information should be mapped as detailed in the XBR^i Cloud Services Core Field Mapping Guide.

Mapping Detail Records Petty Cash

Petty Cash (PTC)

This record represents the petty cash activity. It notes whether the money was moving in or out of the register and the reason for the petty cash activity. Typically, money is taken from the till for paying small charges, such as minor office supplies or deliveries, for example, Repairs, Employee Incentive, Postage, In Store Service, Money Found In Store, Petty Cash Replenishment.

Petty Cash transactions do not contain a purchase or return. When a petty cash activity occurs within a sale/return transaction the SALE takes precedence.

The Recode field is used to indicate if the money went in out of the register.

The Reason_type field is used to post the reason for the petty cash activity.

A Petty Cash transaction is typically limited to a Header record, a Petty Cash record, and a tender record (TND).

XBR*i* Database Settings

PRO SP Variables:

You must create a script to update the variables specific to your customer's business.

> **Important**: Remember to update both the VAR and VAR2 variables.

Additional information on these customizations is covered in Chapter 8, Application Configuration, Update PRO_SP_VARIABLES.

XBR*i* Database Stored Procedures

All data is loaded to staging and temp tables in the database. There are stored procedures that control the movement of this data to the appropriate today and historical tables. An explanation of these stored procedures is shown in the following table.

> **Note**: These stored procedures cannot be modified for customers.

Table 2.2 Stored Procedures

Stored Procedure	Description
SP_ETL_XSTART_BATCH	This procedure clears the pos_staging table and runs the partitioning procedure if the ETL Partitioning variable is set to 'Y'. This procedure runs once per day by the XBRI_LOADPLAN_BATCH.
SP_ETL_XSTART_REAL	This procedure clears the pos_staging table and runs the partitioning procedure if the ETL Partitioning variable is set to 'Y'. This procedure runs every 15 minutes in the beginning of XBRI_LOADPLAN_REAL
SP_ETL_XFINISH_BATCH	This procedure will read a variable in PRO_SP_VARIABLES which will determine if the customer has selected the LP Module, or both LP and SP Module. Based on the variable's value, This procedure will execute the appropriate procedures based on whether they have, LP or both. This procedure runs once per day by the XBRI_LOADPLAN_BATCH
SP_ETL_XFINISH_REAL	This procedure will read a variable in PRO_SP_VARIABLES which will determine if the customer has selected the LP Module or both LP and SP Module. Based on the variable's value, this procedure will execute the appropriate procedures based on whether they have, LP, SP or both. This procedure runs every 15 minutes in the end of XBRI_LOADPLAN_REAL
SP_ETL_XFINISH_EOD	This procedure calls the stored procedures responsible for moving data from today tables to historical tables. This procedure will run at the end of XBRI_LOADPLAN_EOD
SP_PRO_CLEAR_STAGE:	Clears the staging table of transactions that have successfully moved to downstream systems
SP_PRO _SET_BATCHNO:	Creates a batch number for everything in pro_staging that does not have one; calls sp_pro_sequencer to get the next batch number
SP_PRO _DUP_CHK:	Removes duplicate transactions from the staging table and puts them in the pos_staging_dups table

SP_PRO _LOAD_HIST:	Copies transactions from the staging table to the Analytics history tables. Calls sp_pro_nomatch_pvcancel and sp_pro_nomatch_returnexch to populate no match data.
SP_PRO _LOAD_STATS:	Loads the pos_statistics_tab table for use in Analytics
SP_PRO _NOMATCH	Matches refunds and exchanges to their original transactions
SP_PRO _TRANSDATE_PURGE:	Purges POS historical tables of old entries
SP_PRO_INVENTORY	Updates/Inserts inventory data from the temp table
SP_PRO_LOAD_SPO_STATS	Aggregates data for Sales and Productivity summary tables
SP_SPO_COMP_POLL	Populate comp, no poll flag and also creates no poll estimates
SP_SPO_HRS_WORKED	Reads the clock out and time card adjustment transactions from staging table and populates the hours worked table
SP_SPO_UPD_GOALS	Updates/Inserts goal data from the temp table
SP_SPO_UPD_TRAFFIC	Updates/Inserts traffic count data from the temp table
SP_MST_UPD_STORE;	Updates/Inserts store data from the temp table. Adds stores for new stores in the pos_staging table
SP_MST_UPD_EMP	Updates/Inserts employee data from the temp table. Adds employees for new employees in the pos_staging table
SP_MST_UPD_SKU;	Updates/Inserts sku data from the temp table. Adds skus for new skus in the pos_staging table
SP_MST_UPD_REGNUM;	Updates/Inserts register data from the temp table. Adds registers for new registers in the pos_staging table
SP_MST_UPD_CUSTOMER;	Updates/Inserts customer data from the temp table. Adds customers for new customers in the pos_staging table
SP_MST_UPD_LOOKUPS;	Updates and inserts data in the lookup tables
SP_MST_UPD_TOKEN	Updates pos_tnd_tab with token values which were delivered separately from the pos data

3

Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. XBR^i has been internationalized to support multiple languages. This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following:

- Graphical user interface (GUI)
- Error messages
- Reports

The following components are not translated:

- Documentation (online help, release notes, implementation guide, administration guide)
- Batch programs and messages
- Log files
- Configuration tools
- Demonstration data
- Training materials

The user interface for XBRi has been translated into:

- English (US)
- French (France)
- German
- Italian
- Portuguese (Brasil)
- Spanish

Multi-Language Setup

XBR^i data is supported in two languages. This section provides details of various scenarios that you may come across during implementation. Since multi-language data support in XBR^i is dependent on the availability of the multi-language data in the source system, it is important to understand various scenarios the user may encounter. Before proceeding review the following facts about multi-language support:

- XBR^i programs extract multi-language data from source systems.
- A list of languages for multi-language data support can be chosen during the installation process.
- Depending on the implementation, the source system may or may not have data for particular supported languages. For example, XBR^i supports Item Descriptions in multiple languages but the item's description may not be available in the translated languages.
- For source system released languages, please refer to source system operations guides.
- You must select XBR^i primary language for data purposes to be supported within the source system.

Scenario 1

All the supported languages are implemented in XBR^{i} and the same set of languages are supported in the source system as well.

Multi-lingual data sets are enabled in both XBR^{i} and the source system.

Data Scenario 1a

Translated data exists for all records in the source system: This is an ideal scenario where the source system supports data for the same set of languages as XBR^{i}, and data for the required columns exists in all the languages in the source system. In this scenario the attributes that are supported for multi-languages will get all the multi-language data in XBR^{i}.

Data Scenario 1b

Translated data does not exist for some of the records in the source system.

For the attributes for which data is not available in the source system, XBR^{i} will display the attribute in source system's primary language. For example, XBR^{i} requests data in German and English languages. In XBR^{i} the Item attribute description is not available in the German language but is available in English language.

Scenario 2

All or a subset of languages are implemented in XBR^{i} and some of these are not supported in the source system:

Data Scenario 2a

Translated data does not exist for some of the languages in the source system. In this case, the data is displayed in the XBR^{i} primary language.

Scenario 3

Source system supports more languages than are supported for XBR^{i}. In this case XBR^{i} filters out the additional languages' data. This data will not be loaded into XBR^{i} tables and cannot be used for reporting.

Localizing Currency in XBR^{i}

If multi-currency reporting is required for an XBR^{i} implementation, the customer needs to provide a multi-currency exchange rate feed to the XBR^{i} database. The schedule for updating the database with the customer exchange rate data is implemented by the Oracle Enablement team. The Customer Administrator can set currency metric defaults in the XBR^{i} Admin interface, as described in the *Oracle Retail XBR^{i} Cloud Services Administration Guide.*

Implementing Currency Exchange Rates

You can schedule XBR^{ii} to be updated with currency exchange rates on a regular basis. This can be daily, weekly, or any other interval, depending on how often the currency exchange rate data is provided by the customer. The customer data feed populates the MST_CURRENCY_RATE_TMP table. The updated currency data is transferred to the MST_CURRENCY_RATE_TAB table using the SP_MST_UPD_CURRENCY_RATE update procedure.

The SP_MST_UPD_CURRENCY_RATE procedure begins by inserting all the rows that are new in MST_CURRENCY_RATE_TMP into MST_CURRENCY_RATE_TAB. Then it inserts any new rows from MST_CURRENCY_RATE_TMP into MST_CURRENCY_RATE_TAB where TRANSDATE is greater than the minimum TRANSDATE in MST_CURRENCY_RATE_TAB, filling in any gaps in the transdate range.

It adds these rows with a 0.0 exchange rate. It then finds each of these rows and updates the exchange rate with the most recent exchange rate for that currency code it can find.

When inserting missing currency rates, it will follow these rules:

If there is no history default rate to 0.

Any rate that comes in tmp table should overwrite what is in mst_currency_rate_tab.

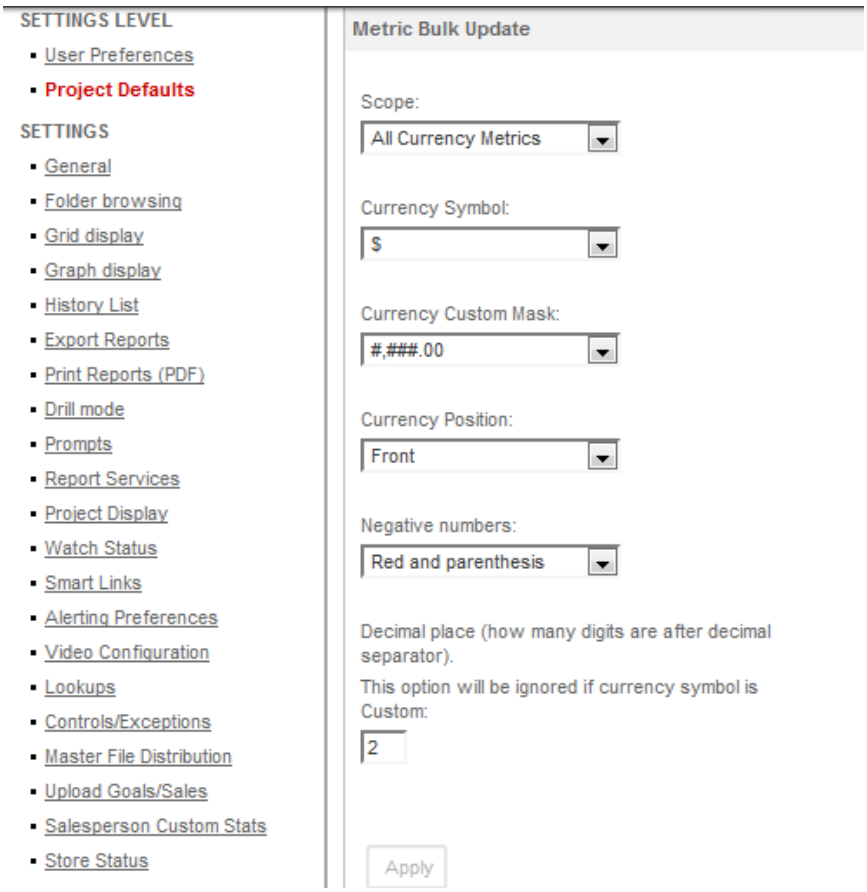
When local = base then set rate to 1.

Applying Updates for Currency Metrics

If you are using multiple currencies in your project, set the defaults applied to currency metrics in the Project Defaults, Metric Bulk Update page, which lets you set default formats for local, common or all currency metrics. This includes defaults for symbol, custom mask, position, negative numbers, and decimal place. The changes are displayed wherever currency is shown in the application, such as in reports, documents, dashboards, and control points.

> **Note:** XBR*i* is updated with currency exchange rates on a schedule determined by the customer. The customer data feed is provided by the customer. The schedule for updating the database with the customer exchange rate data is implemented by the Oracle Enablement team.

1. Log in to XBR*i* as the Customer Administrator.
2. From the Admin menu, choose **Project Defaults.**
3. Under Settings, choose Metric Bulk Update.



Project Defaults – Metric Bulk Update

4. Make selections for the following options:

Scope

Select the group of metrics to which you want to apply bulk updates from the drop-down list.

Available options are:

All Currency Metrics - Applies to both Local and Common Currency Metrics.

Local Metrics only - Applies to the metrics based on amounts that will display a currency except for those in the Common Currency folder.

Common Metrics only - Applies to the metrics based on amounts that will display a currency only for those in the Common Currency folder.

Currency Symbol

From the drop-down list, select the symbol associated with the metric currency. The list of symbols is determined by the languages available for your project.

- If you choose **No Currency Symbol** from the list, the currency symbol present in the metric is removed, and the currency amount is displayed without a symbol.
- If you choose Custom as the currency symbol, the symbol is based on the selection for Number and Date Format in User Preferences. For example, if the Number and Date Format is Italian, the Custom Currency symbol will be for the Euro. See: General Preferences in the XBR^i Administrator online help for more information on setting the Number and Date Format preferences.

Currency Custom Mask

From the drop-down list, select the characters to use to mask currency amounts. Available options are:

#,###.## - The currency amount is displayed with thousands separated by a comma, and with trailing zeroes suppressed in the decimal value. For example: 123456.78 is displayed as 123456.78, but 1234.50 is displayed as 1234.5

#,###.00 - The currency amount is displayed with thousands separated by a comma, and with trailing zeroes displayed in the decimal value. For example: 1234.50 is displayed as 1234.50

#.###,## - The currency amount is displayed with thousands separated by a period, with the decimal value separated with a comma, and with trailing zeroes suppressed in the decimal value. For example: 123.456;78 is displayed as 123.456,78, but 1.234,50 is displayed as 1.234,5

Currency Position

From the drop-down list, select the currency position to apply to the metric currency. Available options are:

Front - For example, $123.45

Back - For example, 123.45$

Front and space - For example, $ 123.45

Back and space - For example, 123.45 $

Negative Numbers

From the drop-down list, select a format for displaying negative numbers. Available options are:

Minus/Black

Red

Black and parentheses

Red and parentheses

Decimal Place

In the empty box below the Decimal Place label, enter the number of digits to display after the decimal separator.

> **Note:** This option is ignored if the Currency Symbol is Custom.

Click **Apply** to apply the settings.

4

XBR*i* Statistics Aggregation

Introduction

Statistics records contain summary data for a particular store on a particular day. The statistics data is calculated and summed through a stored procedure. This data is then posted to the POS_STATISTICS_TMP table and used to insert new records into POS_STATISTICS_TAB.

Statistics

Statistics can be summed at several different levels depending on the key fields invoked. The four possible levels are:

- By Register
- By Cashier
- By Store (hierarchy, such as region, district, and so on)
- By Date

The table layouts for the POS_STATISTICS_TAB can be found in the Oracle Retail XBR*i* Cloud Services Core Field Mapping Guide.

PRO_VIEW_SYNTAX

PRO_VIEW_SYNTAX is a table in the XBRi database that dictates the aggregation of data that is posted to the POS_STATISTICS_TAB table. The SYSTEM field is populated with STAT for all aggregations that are associated with the POS_STATISTICS_TAB table.

Fields in PRO_VIEW_SYNTAX table

SYSTEM

POSTING_SOURCE

TARGET_FIELD

SOURCE_FIELD

VIEW_NAME

FROM_CLS

WHERE_CLS

GROUPBY_CLS

ORDERBY_CLS

COMMENTS

ACTIVE_FLAG

CUSTOM_FLAG

END_OF_DAY_FLAG

Table 4.1 SALES_REF_EXCH_AMOUNT statistics bucket as defined in PRO_VIEW_SYNTAX

Field Name	Value
SYSTEM	STAT
POSTING_SOURCE	ST
TARGET_FIELD	SALES_REF_EXCH_AMOUNT
SOURCE_FIELD	sum(case when TRAINING_FLAG = 'N' AND TRANSTYPE IN ('SALE', 'RETURN', 'EXCHANGE') AND STATUS IN ('OK', 'AO', 'AR') AND TRANSSTAT = 'COMPLETE' AND (RECCODE = '600' OR (RECCODE IN ('601', '602', '603') AND ATTRIBUTE_CODE1 <> 12)) then case when tender_amount is null then 0 else tender_amount end else 0 end)
VIEW_NAME	SALES_REF_EXCH_AMOUNT
FROM_CLS	POS_STAGING
WHERE_CLS	RECTYPE = 'HDR' AND VOID_CODE < 3
GROUPBY_CLS	TRANSDATE, STORENUM, REGNUM, CASHIERNUM, DIVISION, BATCHNO, ORGID, CURRENCY_CODE, BUSINESS_DATE
ORDERBY_CLS	
COMMENTS	
ACTIVE_FLAG	N
CUSTOM_FLAG	0
END_OF_DAY_FLAG	N

You can add or update core statistics buckets in the PRO_VIEW_SYNTAX table. By default all statistic buckets are deactivated. You must create a script to activate the buckets that correspond to functionality included in the customer's implementation. The ACTIVE_FLAG should be set to Y.

```
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'REF_EXCH_MO_NOMATCH_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'REF_EXCH_MO_NOMATCH_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SALES_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SALES_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SALES_EXCH_MI_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SALES_EXCH_MI_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SALES_REF_EXCH_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SALES_REF_EXCH_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SCAN_GUARANTEE_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SCAN_GUARANTEE_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'SKU_LINE_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_ACTIVITY_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_ACTIVITY_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_CANCEL_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_CANCEL_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_DEPOSIT_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_DEPOSIT_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_FEE_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_FEE_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_FINAL_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_FINAL_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_INITIAL_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_INITIAL_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_PAYMENT_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_PAYMENT_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SPECORDER_SKU_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'STORE_COUPON_LINE_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'STORE_COUPON_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'STORE_COUPON_KEYED_LINE_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'STORE_COUPON_KEYED_AMOUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SUSPEND_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'N' WHERE TARGET_FIELD = 'SUSPEND_FB_NOSALE_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'TAX_EXEMPT_COUNT'  AND SYSTEM = 'STAT';
UPDATE PRO_VIEW_SYNTAX  SET ACTIVE_FLAG = 'Y' WHERE TARGET_FIELD = 'TAX_EXEMPT_AMOUNT'  AND SYSTEM = 'STAT';
```

Activating Statistic Buckets

You use the CUSTOM_FLAG column in conjunction with the ACTIVE_FLAG column to customize a core statistics bucket. There may be a need to customize a statistics bucket. Quite often this is necessary when a customer has a custom tender such a special credit card. The credit card activity should post to the credit card statistic buckets. Since the core code is being changed, you need to set the CUSTOM_FLAG. The WHERE clause would need to change to include reference to the new custom reccode.

Example: Modifying an Existing Core Statistic

```
UPDATE XBRADMIN.PRO_VIEW_SYNTAX
SET
FROM_CLS = '(SELECT transdate, storenum, regnum, cashiernum, division, batchno, orgid, COUNT
(DISTINCT transnum) AS cc_keyed_count FROM (SELECT transdate, storenum, regnum, cashiernum,
division, batchno, orgid, transnum, sum(CASE WHEN void_Code = 2 THEN -1 ELSE 1 END) void_Code
FROM  POS_STAGING WHERE STATUS IN (''OK'', ''AO'', ''AR'') AND transstat = ''COMPLETE'' AND
reccode IN (''430'', ''431'', ''432'', ''433'', ''434'', ''435'', ''501'', ''502'') AND rectype =
''TND'' AND transtype IN (''SALE'', ''EXCHANGE'', ''RETURN'', ''TNDEXCH'', ''PAYMENT'') AND
training_flag = ''N'' AND manual_keyed_code = 1 AND VOID_CODE < 3 group by transdate, storenum,
regnum, cashiernum, division, batchno, orgid, transnum) Z GROUP BY transdate, storenum, regnum,
cashiernum, division, batchno, orgid HAVING SUM(void_Code) <> 0)',
COMMENTS = 'Add TND 501 (HOTL), 502 (LDSC)',
CUSTOM_FLAG=''Y'',
ACTIVE_FLAG= 'Y'
where system = 'STAT' and POSTING_SOURCE = 'ST' and target_field LIKE 'CC_KEYED_C%';
```

There are twenty custom statistic buckets available for aggregating different subsets of data by count and amount. They are typically created in pairs. You would add an entry to the PRO_VIEW_SYNTAX table that would ultimately populate the pos_statasitcs_tab.customX_count and the pos_statasitcs_tab.customX_amount fields. The TARGET_FIELD in PRO_VIEW_SYNTAX would contain the name of the field in pos_statistics_tab that you plan on populating.

Sales and Productivity Tables

The Sales and Productivity (SP) module is divided into the following four business areas:

Comparative Sales

The purpose of the Comparative Store Sales category is to report on sales by store by day. These sales include This Year vs. Last Year (TY/LY) transformation metrics, based on retailers' predefined fiscal calendars. A second set of transformation metrics track comparative (Comp), or same store sales. These metrics add further logic to the TY/LY comparison, by comparing store sales only to when the store was open at the same time this year and last; with additional business logic. The Comp settings are maintained within the application using a new Store Status page in Administration, Project Defaults. In addition, the customer can define and load sales goals or budgets to compare a store's sales performance to as many as three different sales goals. This is accomplished within the application, using a new Upload Goals/Sales page in Administration, Project Defaults. The goal file can also be loaded from a customer-provided file feed.

Salesperson/Employee Productivity

Salesperson Productivity reporting tracks key sales and productivity KPI's. These KPI's are attuned to a true selling environment in which employees are held accountable for sales and the quality and focus of what they sell. Another component of salesperson productivity is measuring sales against hours worked. Hours worked can be classified as selling or non-selling hours. These metrics are used to create and monitor sales/labor hour comparatives. A comprehensive set of new reports is provided in the Productivity category, as well as a new Salesperson Productivity dashboard. Using the new Salesperson Custom Stats page in Administration, Project Defaults, users can identify up to 10 custom count, transaction count, and amount statistics to use in reporting.

Merchandise Productivity

Merchandise Productivity provides retailers with sales analysis across their merchandising hierarchy. A merchandise (SKU Master) table has been incorporated, which provides the levels of summarization needed within these hierarchies along with added attributes associated with the item such as vendor or manufacturer. This allows retailers to analyze their merchandise sales by categories and by items across their merchandise and operational hierarchies. It also allows for Margin analysis as well as, Return, Discount, and Voids summary information through these same hierarchies.

Store Sales Flow by Period

The Store Flow by Period components of this module provide customers with added business analytics focusing on the flow of sales transactions throughout business days, by hour, or by day part custom categorizations as well as by traffic counts and conversion rates.

The tables that contain the aggregated data are:

SPO_STATS_PERIOD_TAB	Contains aggregated data by Day, Store, Fixed Period
SPO_STATS_SKU_TAB	Contains aggregated data by Day, Store, Item
SPO_STATS_SLSPSN_TAB	Contains aggregated data by Day, Store, Salesperson
SPO_STATS_STORE_TAB	Contains aggregated data by Day, Store

SP Stats in PRO_VIEW_SYNTAX

The PRO_VIEW_SYNTAX table is also used to formulate the aggregation of data for the sales and productivity tables. The system field is populated with SPOXXXX. In the example below SPOSTORE indicates the statistic is part of **SPO_STATS_STORE_TAB**

Example: SALE_EXTENDED_AMOUNT statistics bucket as defined in PRO_VIEW_SYNTAX

Field_Name	Value
SYSTEM	SPOSTORE
POSTING_SOURCE	ST
TARGET_FIELD	SALE_EXTENDED_AMOUNT
SOURCE_FIELD	SUM(extended_amount)
VIEW_NAME	SALE_EXTENDED_AMOUNT
FROM_CLS	POS_STAGING
WHERE_CLS	STATUS IN ('OK', 'AO', 'AR') AND TRANSSTAT = 'COMPLETE' AND (RECTYPE = 'SKU' OR (RECTYPE IN ('LAYSKU','SOSKU') AND ATTRIBUTE_CODE1 IN (9,10,11)) OR (RECTYPE = 'NM' AND RECCODE = '02') OR (RECTYPE IN ('LAYNM','SONM') AND RECCODE = '02' AND ATTRIBUTE_CODE1 IN (9,10,11))) AND TRANSTYPE IN ('SALE','EXCHANGE') AND TRAINING_FLAG = 'N' AND VOID_CODE < 3 AND RETURN_FLAG = 'N'
GROUPBY_CLS	BUSINESS_DATE, STORENUM, DIVISION, BATCHNO, ORGID, CURRENCY_CODE
ORDERBY_CLS	
COMMENTS	
ACTIVE_FLAG	N
CUSTOM_FLAG	0
END_OF_DAY_FLAG	N

5

Data File Delivery

Introduction

This section describes how the core data files are delivered to the XBR^i database.

Core Data Files

File Names:

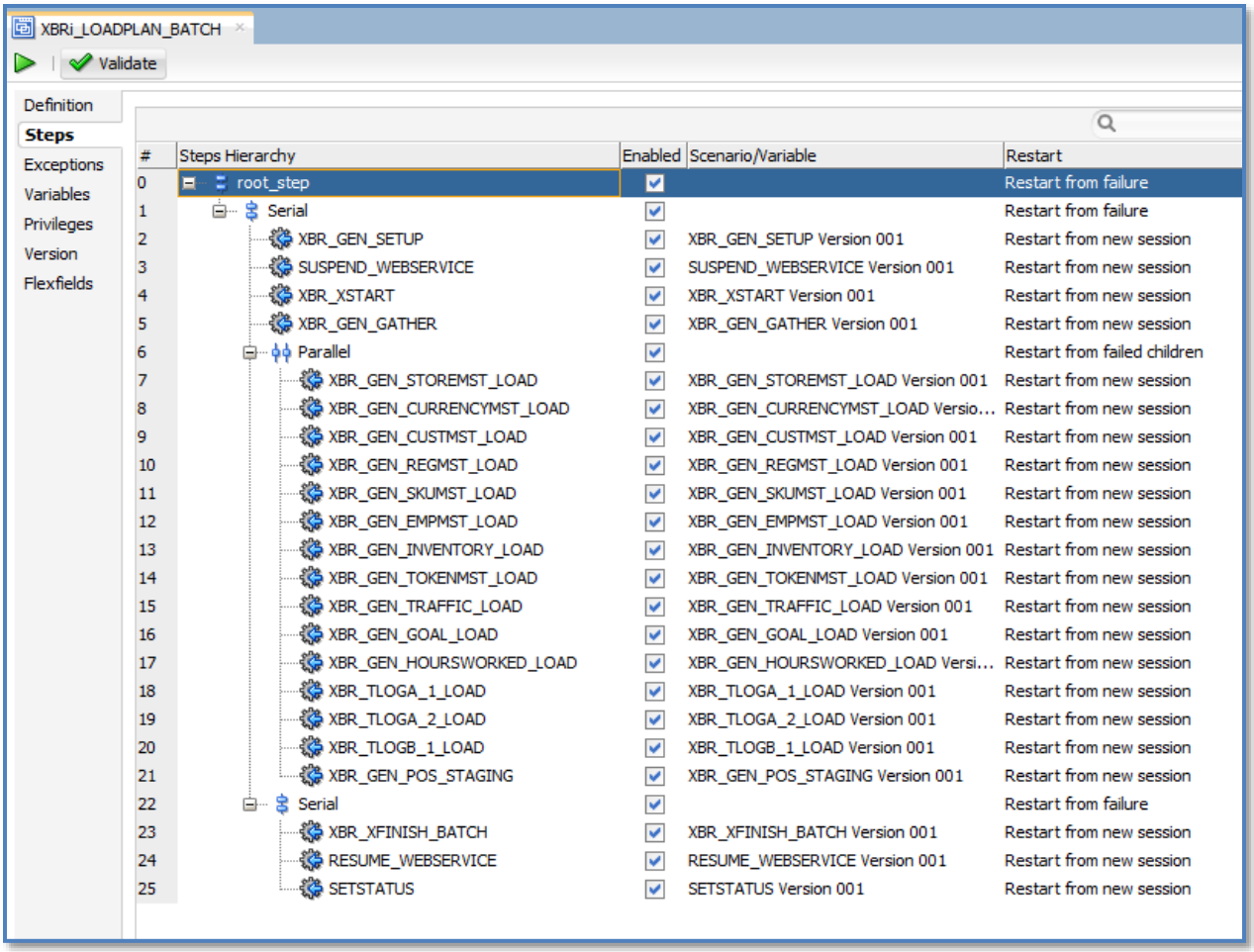
- TRANSACTIONAL LOGS
- STORE
- SKU
- EMPLOYEE
- CUSTOMER
- REGISTER
- CURRENCY
- INVENTORY
- TOKEN
- HOURS_WORKED
- TRAFFIC
- GOALS

Core ELT and Associated Directory Structure

The purpose of the ELT is to appropriately load the data files, as discussed in Chapter 2 The XBR^i Data Model, to their corresponding tables in the XBR^i database. The ELT is built using Oracle Data Integrator (ODI). ODI is set up to use the ODI Studio and a standalone agent configuration. In the Oracle XBR^i Cloud, the XBR^i ELT is a black box capable of loading Point of Sale (POS) data and supplemental data feeds into the XBR^i database for viewing by the application. All workflows are also handled through ODI: file movement, stored procedure calls, event logs, and alerting. The original provisioning of ODI in the XBR^i's Cloud implementation environment requires setting multiple ODI Global Variables. The information should be provided to the Cloud team. Scheduling is also handled by the ODI studio. ODI Studio is used during development to test integrations. The ELT black box load plan is called XBRi_LOADPLAN

ODI XBRI_LOADPLAN

The ODI XBRi_LOADPLAN is the ODI component that is scheduled to run in order to load data to the XBR^i database. It is comprised of several packages/scenarios as shown in the following image. XBR_GEN_SETUP checks prior status and confirms the ELT is ready to run. XBR_GEN_GATHER moves files from the landing area, INCOMING_FILES/FILENAME, to the staging TRANSFORMS/FILENAME/tmp directory for processing. XBR_XSTART executes the xstart stored procedures in the database. All of the data feed packages/scenarios are run in parallel. XBR_XFINISH executes the xfinish stored procedure in the database. SETSTATUS sets the status flag.



ODI XBRi_LOADPLAN_BATCH

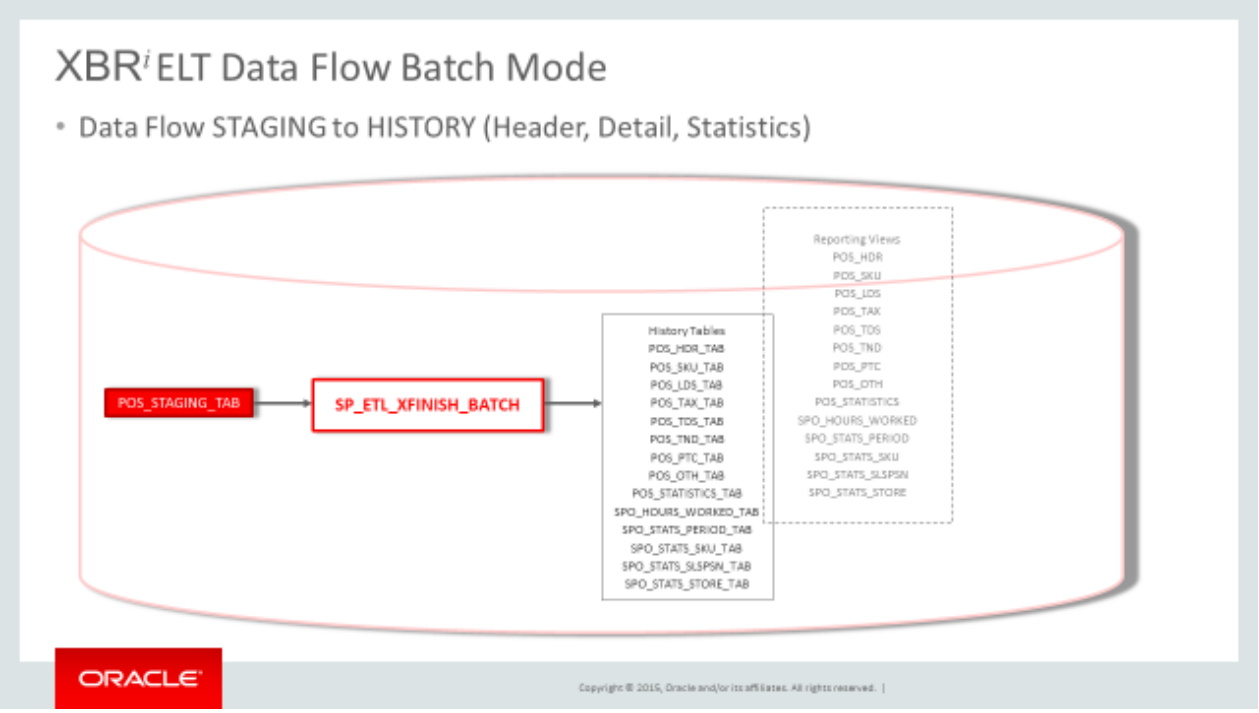
High Level ELT Description

The ELT is a four stage process:

- The first stage involves Extracting or Gathering data from Retailer. Transforming Point of Sale data to adhere to prescribed XBR^i's point or sale staging table format as well as transforming core master file data to prescribed XBR^i format.
- The second stage is to deliver transformed data in stage one to Oracle Cloud through SFTP.
- The third stage is to load that data to the XBR^i database staging table and master tables.
- The last stage of executing stored procedures in the database to load history tables, that is, header detail and summary tables, varies based on whether the integration type is Batch or Real-time.

Batch Mode

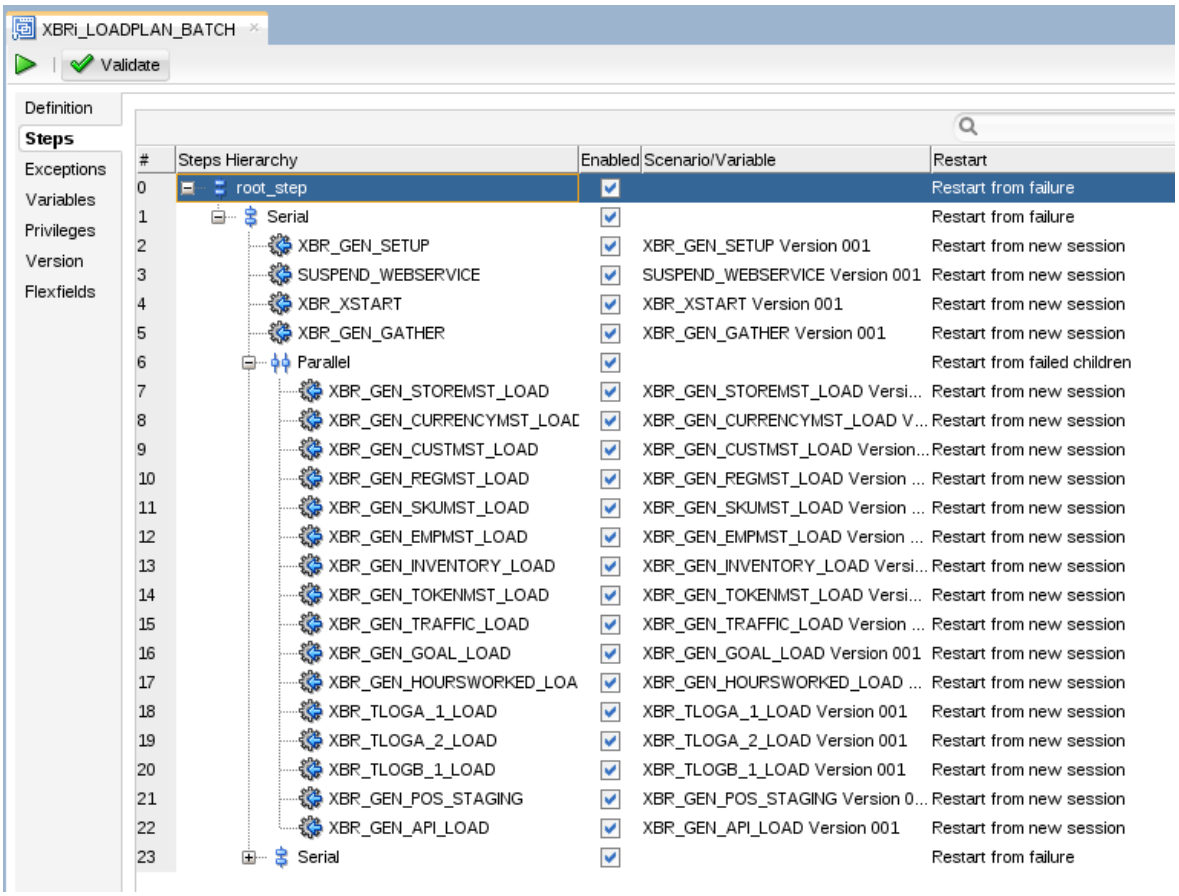
In Batch Mode, the SP ETL XFINISH BATCH stored procedure loads data from Point of Sale Staging table to History tables and is responsible for aggregating statistical data by day, store, cashier, salesperson, item, and so on.



SP_ETL_XFINISH_BATCH diagram

Batch Processing

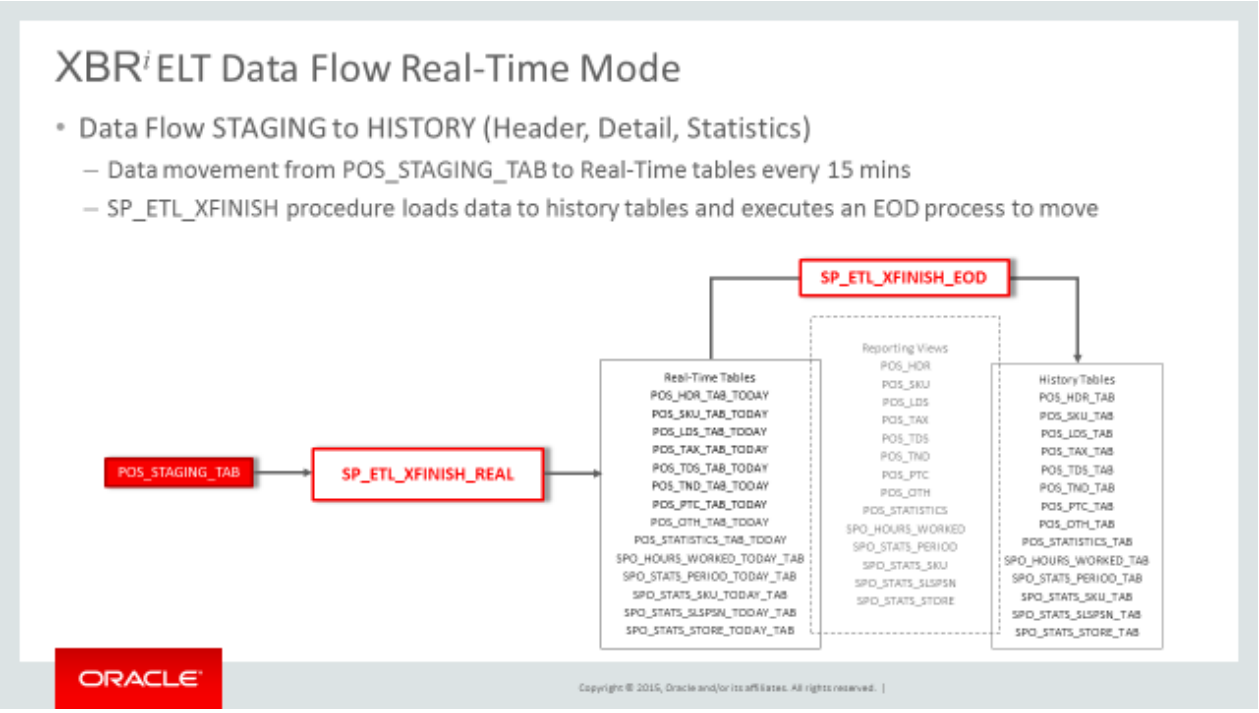
For customers that deliver data once a day, all data will be moved from the POS_STAGING table to the history tables at one time. There is an ODI Load Plan to handle this process, XBRI_LOADPLAN_BATCH. XBRI_LOADPLAN_BATCH will handle the workflow for loading all master and transactional data in addition to calling sp_xfinish_batch for moving data to the history tables and for data aggregation. The customer is responsible for selecting a time of day when they feel it is best to do EOD processing. In general, this is normally done overnight at 3:00 AM of the time zone where their corporate headquarters is located. See an example of the XBRI_LOADPLAN_BATCH in the image below:



XBRI_LOADPLAN_BATCH

Real Time Mode

Real-Time Mode uses the SP ETL XFINISH REAL stored procedure to load data from Point of Sale Staging table to a new set of tables called TODAY tables. These tables have the same structure as the HISTORY tables, however they contain current data while History tables have older data. SP ETL XFINISH REAL stored procedure constantly feeds data from staging to TODAY tables in regular intervals. Once a day SP ETL X FINISH End of Day stored procedure moves data from today tables to history tables while performing additional adjustments to data for elements like post voids, followed by no sale flags, No match and so on. ELT and stored procedures rely on the configuration of variable "PROCESSING TYPE" in "PRO SP VARIBALES" table to process data in near Real-time.

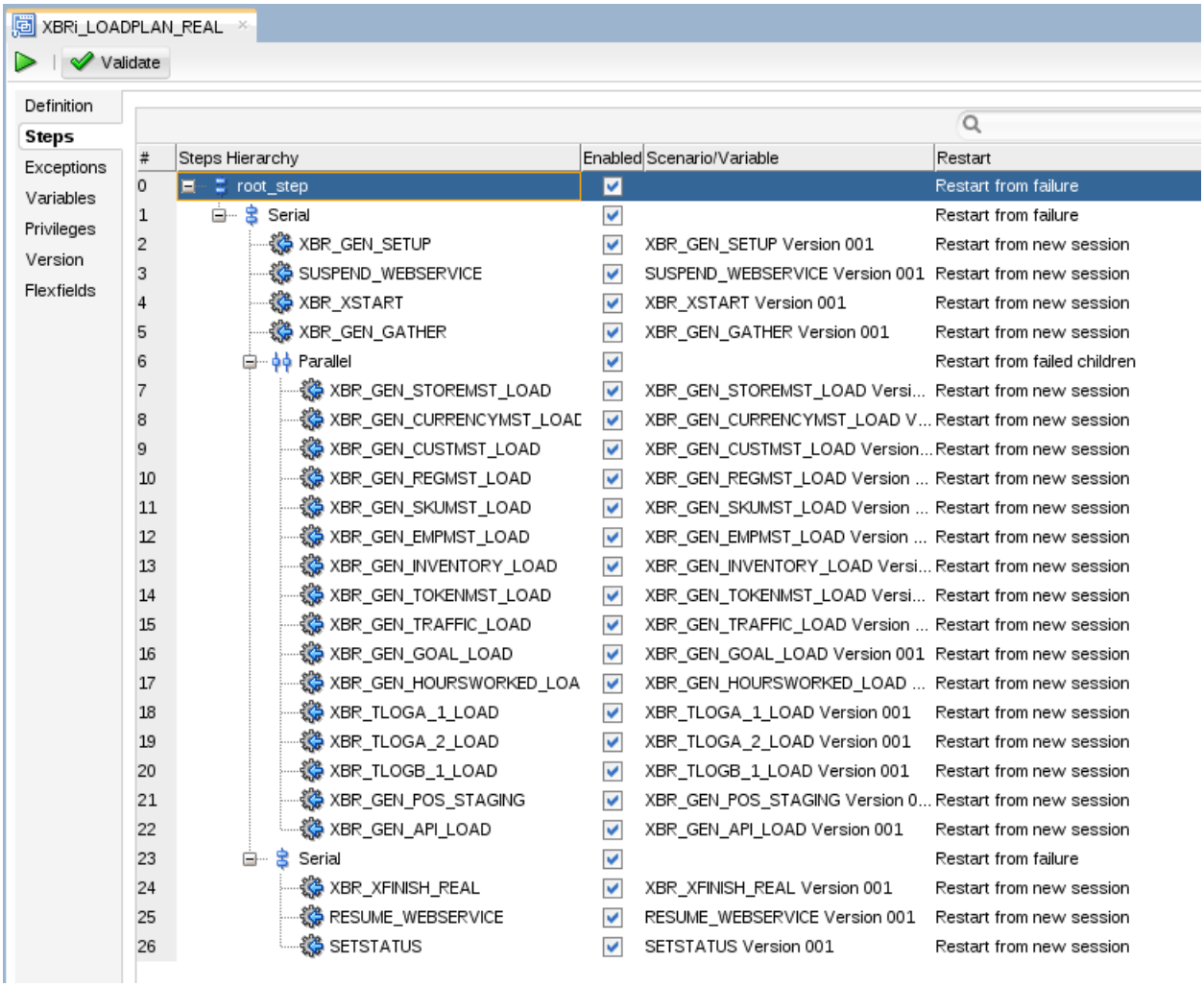


SP_ETL_XFINISH_REAL diagram

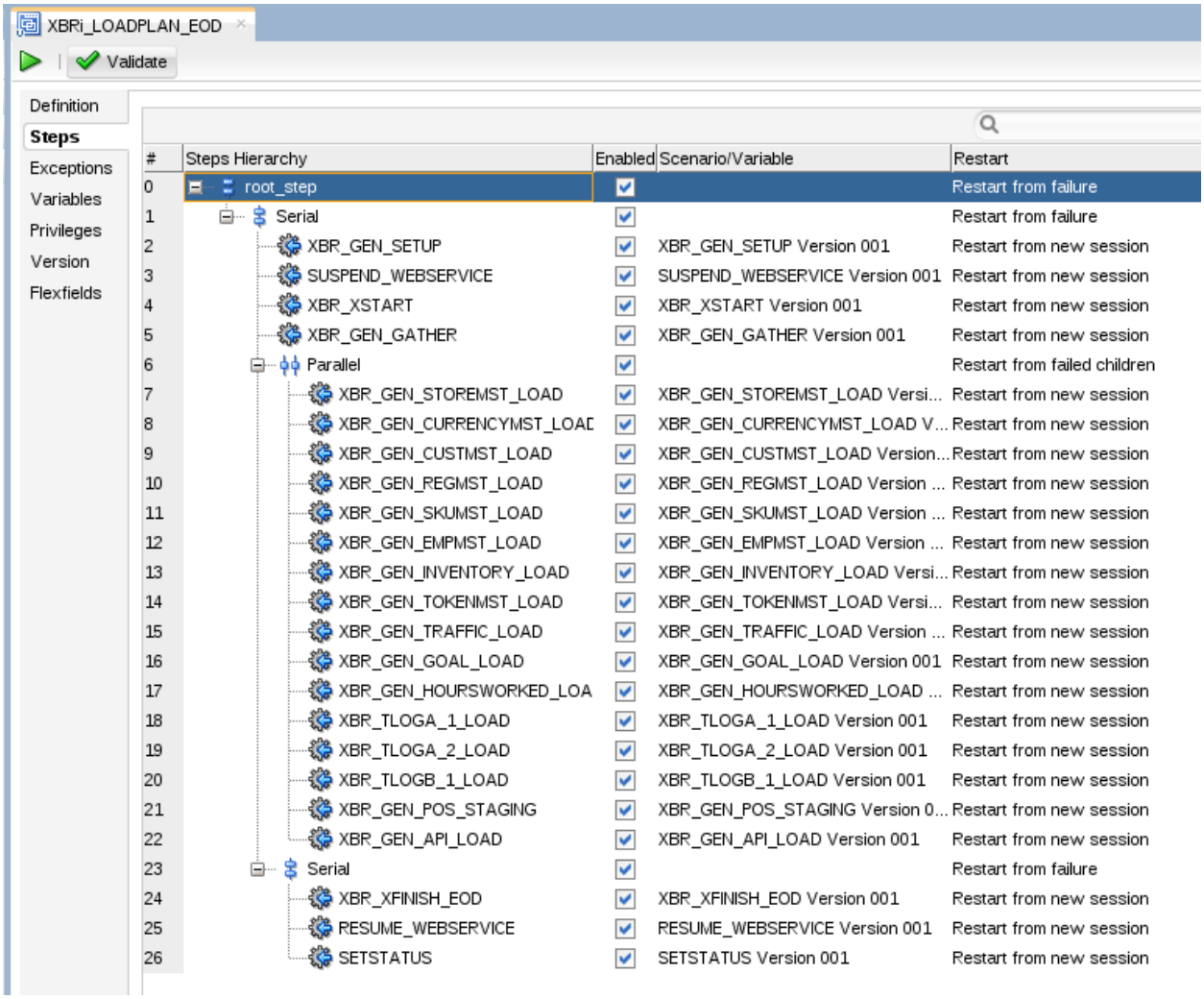
Real Time Processing

XBR^i provides real-time processing to support intraday sales flow reporting in the new Sales and Productivity module. This is enabled by processing data in specified time increments throughout the day rather than once at end of day. Additional business logic supports the inclusion of post voids, no sale, and no match transactions in real-time processing.

For a customer that is able to deliver data throughout the day, there are two ODI Load Plans. XBRI_LOADPLAN_REAL and XBRI_LOADPLAN_EOD. XBRI_LOADPLAN_REAL should be scheduled to run throughout the day. This Load Plan calls the sp_xfinish_real stored procedure in the database. At a high level, this moves data from the POS_STAGING table to the TODAY_XXXXXX tables. The customer is responsible for selecting a time of day where they feel it is best to do EOD processing. In general this is normally in the overnight time frame 3:00AM of the time zone where their corporate headquarters is located. The second XBRI_LOADPLAN_EOD is executed at this time, and after processing, any lingering data will run the sp_xfinish_eod stored procedure. At this time data will move from the today tables to the history tables and process any end of day adjustments. Both today and history tables are visible to the XBR^i front end.



XBRI_LOADPLAN_REAL

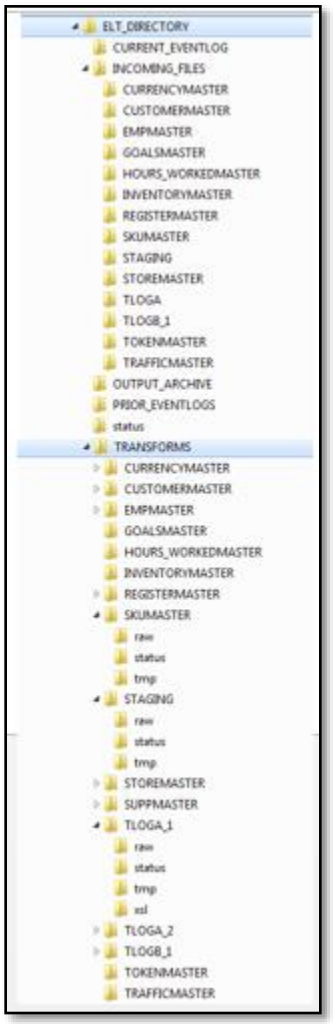


#	Steps Hierarchy	Enabled	Scenario/Variable	Restart
0	root_step	☑		Restart from failure
1	Serial	☑		Restart from failure
2	XBR_GEN_SETUP	☑	XBR_GEN_SETUP Version 001	Restart from new session
3	SUSPEND_WEBSERVICE	☑	SUSPEND_WEBSERVICE Version 001	Restart from new session
4	XBR_XSTART	☑	XBR_XSTART Version 001	Restart from new session
5	XBR_GEN_GATHER	☑	XBR_GEN_GATHER Version 001	Restart from new session
6	Parallel	☑		Restart from failed children
7	XBR_GEN_STOREMST_LOAD	☑	XBR_GEN_STOREMST_LOAD Versi...	Restart from new session
8	XBR_GEN_CURRENCYMST_LOAD	☑	XBR_GEN_CURRENCYMST_LOAD V...	Restart from new session
9	XBR_GEN_CUSTMST_LOAD	☑	XBR_GEN_CUSTMST_LOAD Version...	Restart from new session
10	XBR_GEN_REGMST_LOAD	☑	XBR_GEN_REGMST_LOAD Version ...	Restart from new session
11	XBR_GEN_SKUMST_LOAD	☑	XBR_GEN_SKUMST_LOAD Version ...	Restart from new session
12	XBR_GEN_EMPMST_LOAD	☑	XBR_GEN_EMPMST_LOAD Version ...	Restart from new session
13	XBR_GEN_INVENTORY_LOAD	☑	XBR_GEN_INVENTORY_LOAD Versi...	Restart from new session
14	XBR_GEN_TOKENMST_LOAD	☑	XBR_GEN_TOKENMST_LOAD Versi...	Restart from new session
15	XBR_GEN_TRAFFIC_LOAD	☑	XBR_GEN_TRAFFIC_LOAD Version ...	Restart from new session
16	XBR_GEN_GOAL_LOAD	☑	XBR_GEN_GOAL_LOAD Version 001	Restart from new session
17	XBR_GEN_HOURSWORKED_LOA	☑	XBR_GEN_HOURSWORKED_LOAD ...	Restart from new session
18	XBR_TLOGA_1_LOAD	☑	XBR_TLOGA_1_LOAD Version 001	Restart from new session
19	XBR_TLOGA_2_LOAD	☑	XBR_TLOGA_2_LOAD Version 001	Restart from new session
20	XBR_TLOGB_1_LOAD	☑	XBR_TLOGB_1_LOAD Version 001	Restart from new session
21	XBR_GEN_POS_STAGING	☑	XBR_GEN_POS_STAGING Version 0...	Restart from new session
22	XBR_GEN_API_LOAD	☑	XBR_GEN_API_LOAD Version 001	Restart from new session
23	Serial	☑		Restart from failure
24	XBR_XFINISH_EOD	☑	XBR_XFINISH_EOD Version 001	Restart from new session
25	RESUME_WEBSERVICE	☑	RESUME_WEBSERVICE Version 001	Restart from new session
26	SETSTATUS	☑	SETSTATUS Version 001	Restart from new session

XBRI_LOADPLAN_EOD

ELT Directory Structure

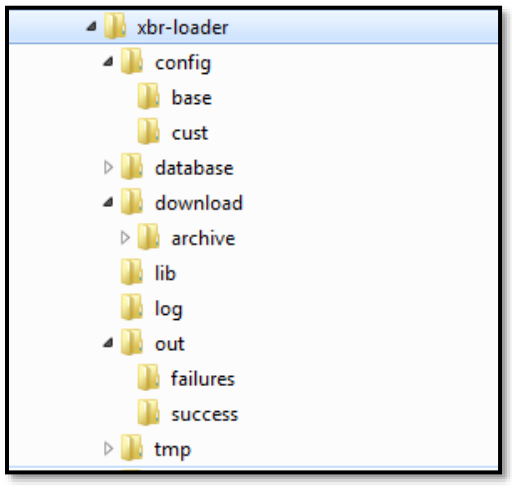
The workflow of the ODI ELT black box is dependent on the directory structure show in the image that follows. INCOMING_FILES is the Landing area. There is a subdirectory for each core data feed to be loaded to the database. The CURRENT_EVENTLOG directory holds the log files generated by the ODI black box for the latest execution of the XBRi_LOADPLAN. It does not contain ODI generated logs. PRIOR_EVENTLOGS contains the log files from prior runs of the XBRi_LOADPLAN as the name implies. The status directory indicates if ODI black box is in an error, running or finished state. Failures of individual file loads do not affect this status. The TRANSFORMS directory has a subdirectory for each type of data feed. The subdirectories have subdirectories within them for file movement and any related configuration files for the particular feed type. Each TRANSFORMS data feed subdirectory also includes a status directory. This directory reports the status of that particular files load.



ELT Directory Structure

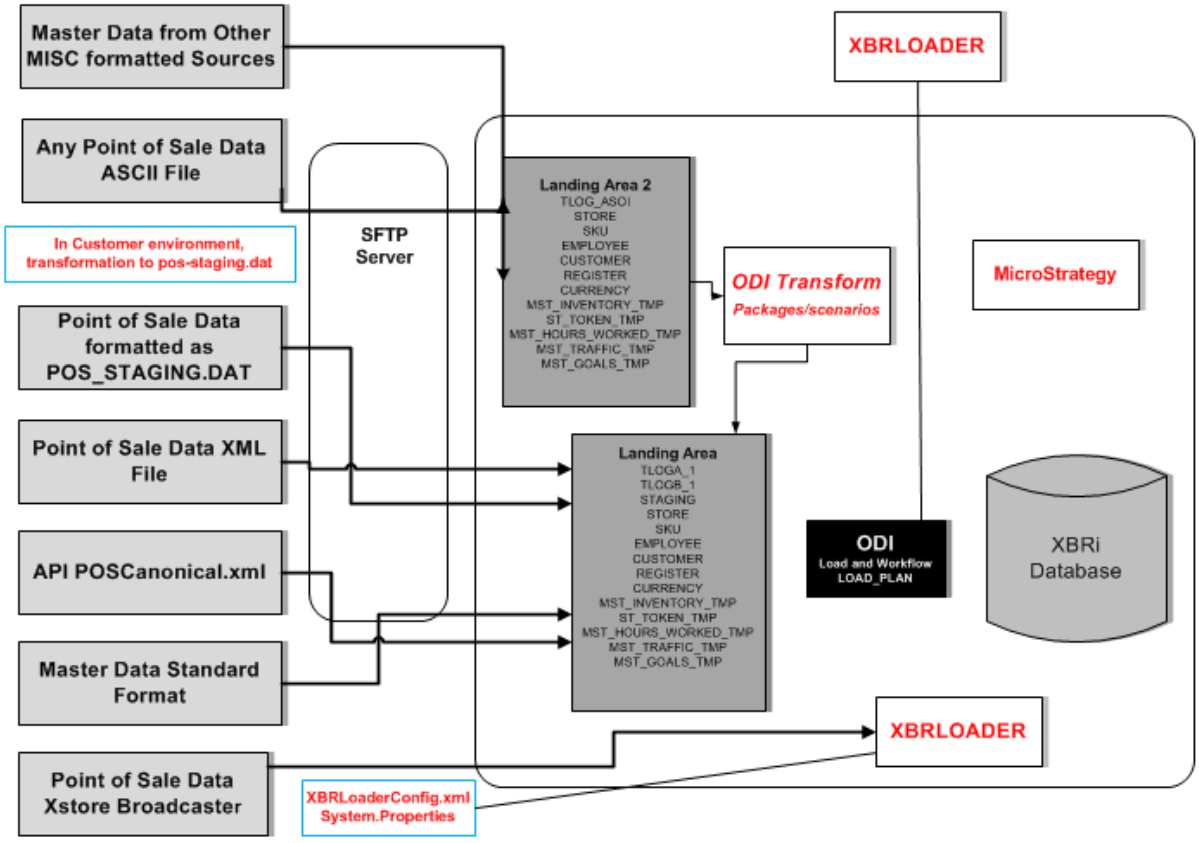
XBRLOADER Directory Structure

The following directory structure is only used for XSTORE-XBR^{i} web service implementations. This type of implementation is described in the Data File Delivery Options section and in Chapter 6: Xstore/XBR^{i} Integration.

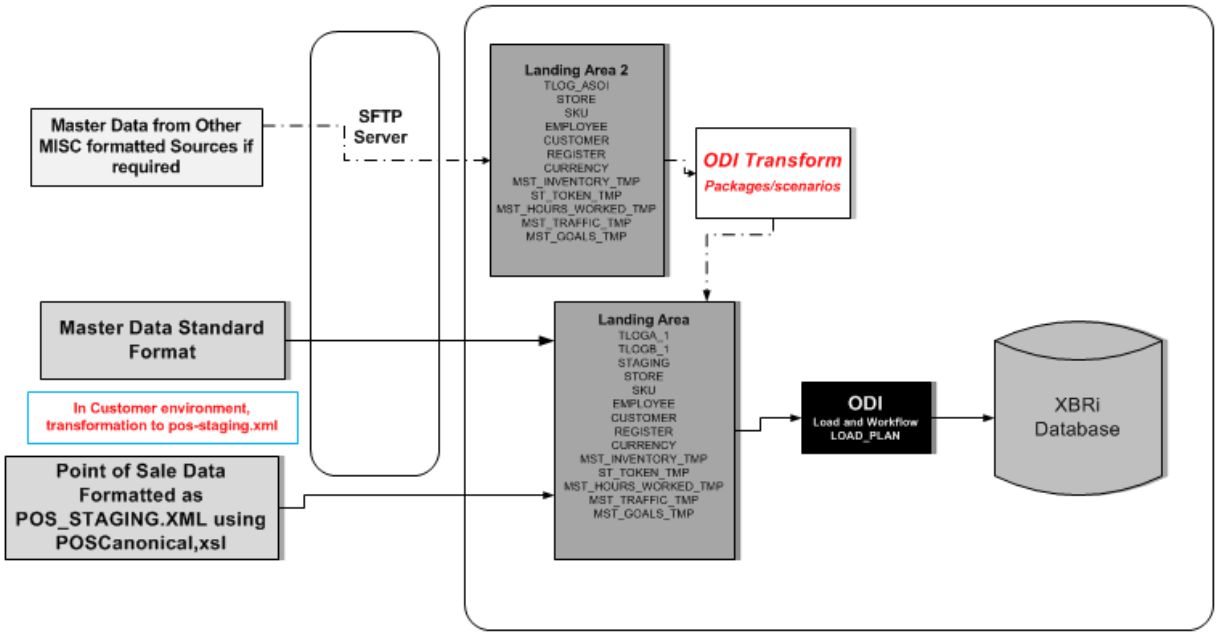


xbr-loader Directory Structure

Data File Delivery Options



Data File Delivery Options Diagram



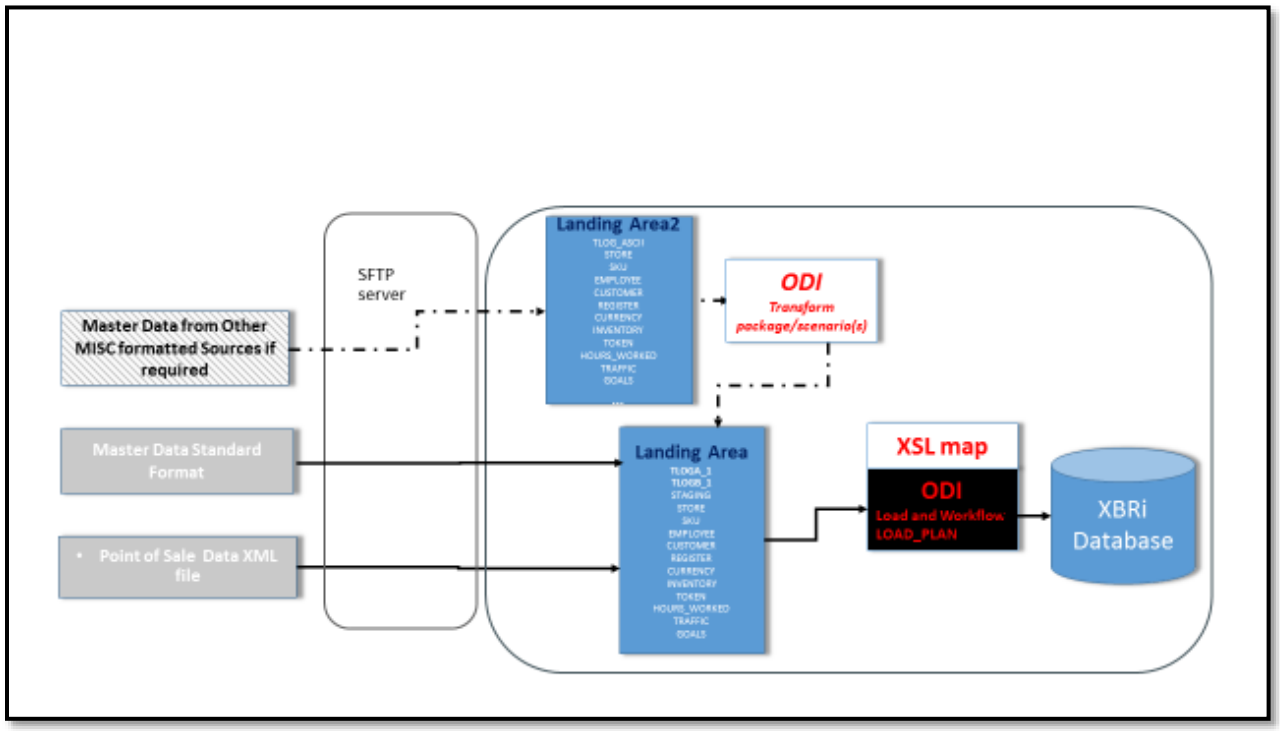
POSCanonical.xml API File Delivery Diagram

POSCanonical.xml API

The ODI black box ELT framework is designed to consume through an API POS transactions. The XBRi API interface will include an .XSD file (XML Schema Definition). The XML schema definition is used as a set of rules to which the XML document must conform, thus aiding the customer in validating their XML file design for the XBRi API.

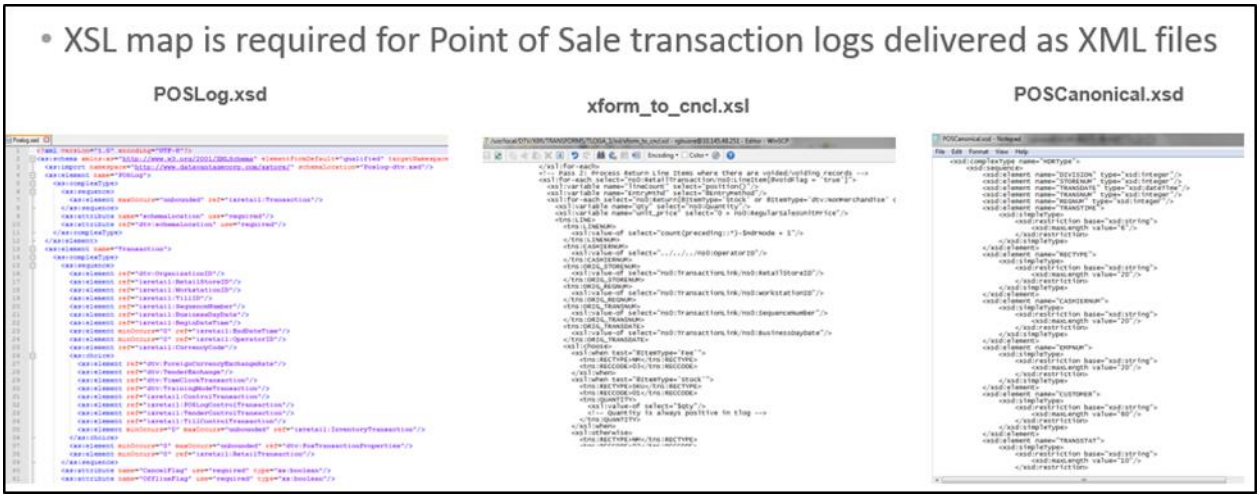
XML poslogs

The ODI black box ELT framework is designed to consume up to two types of XML files. Files that are delivered to INCOMING_FILES/TLOGA or INCOMING_FILES/TLOGB are assumed to be XML.



XML poslogs File Delivery Diagram

For point of sale transactional logs that are delivered as XML files (excluding XSTORE poslogs), you must create an XSL map that converts the customer's XML file to the XBR*i* Canonical version defined by POSCanonical.xsd. The XSL map must be named: xform_to_cncl.xsl. Chapter 4: Data File Delivery provides additional information on where this file should be placed and the dataflow in this data delivery scenario. Once this XSL map has been added to the ELT directory structure, XML transactional logs delivered to the INCOMING_FILES landing area will be processed by the ODI ELT black box framework.



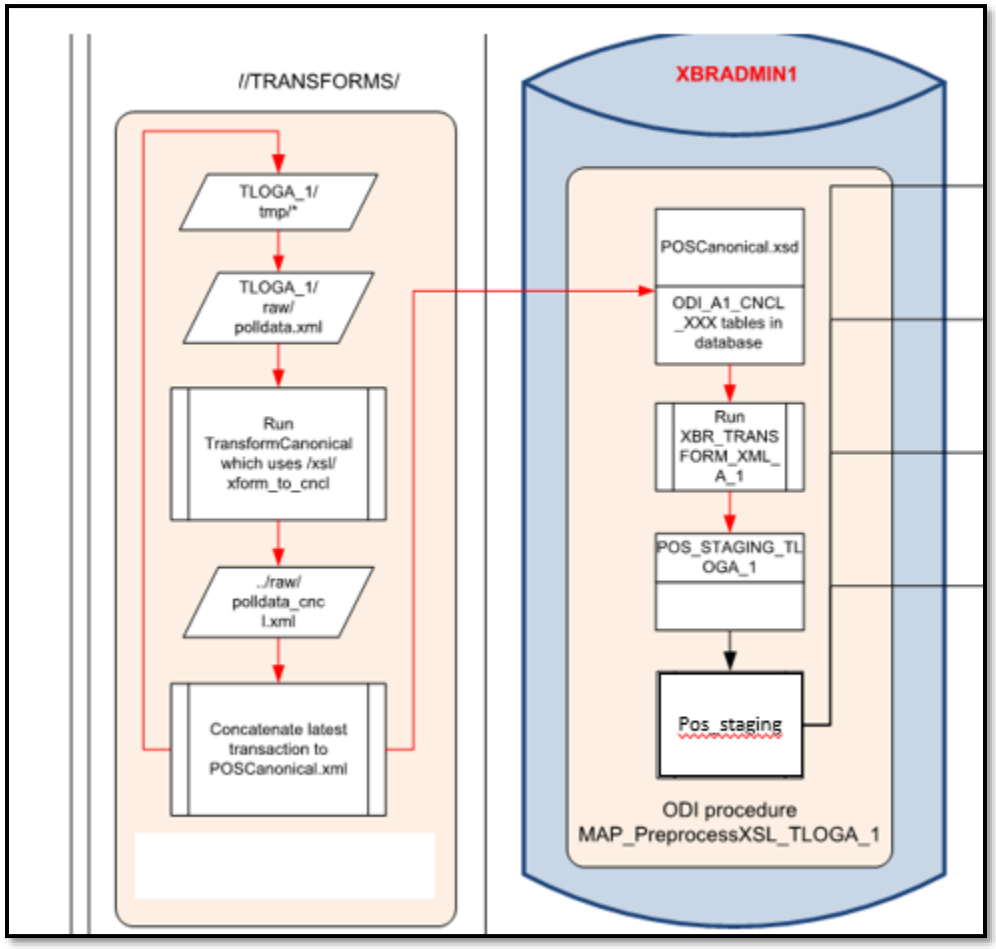
POS Logs for Incoming Files Landing Area

There are fields in POS_STAGING that are difficult to populate using an XSL map. These fields are aggregated/set in an ODI procedure.

These are set: MANUAL_KEYED_CODE, MANUAL_AUTH_CODE.

These are summed: QUANTITY, EXTENDED_AMOUNT, TENDER_AMOUNT,TAX_AMOUNT, LINE_DISCOUNT_AMOUNT, TRANS_DISCOUNT_AMOUNT, COUPON_AMOUNT, FEE_AMOUNT, OVERRIDE_AMOUNT, OTHER_AMOUNT, DEPOSIT_AMOUNT

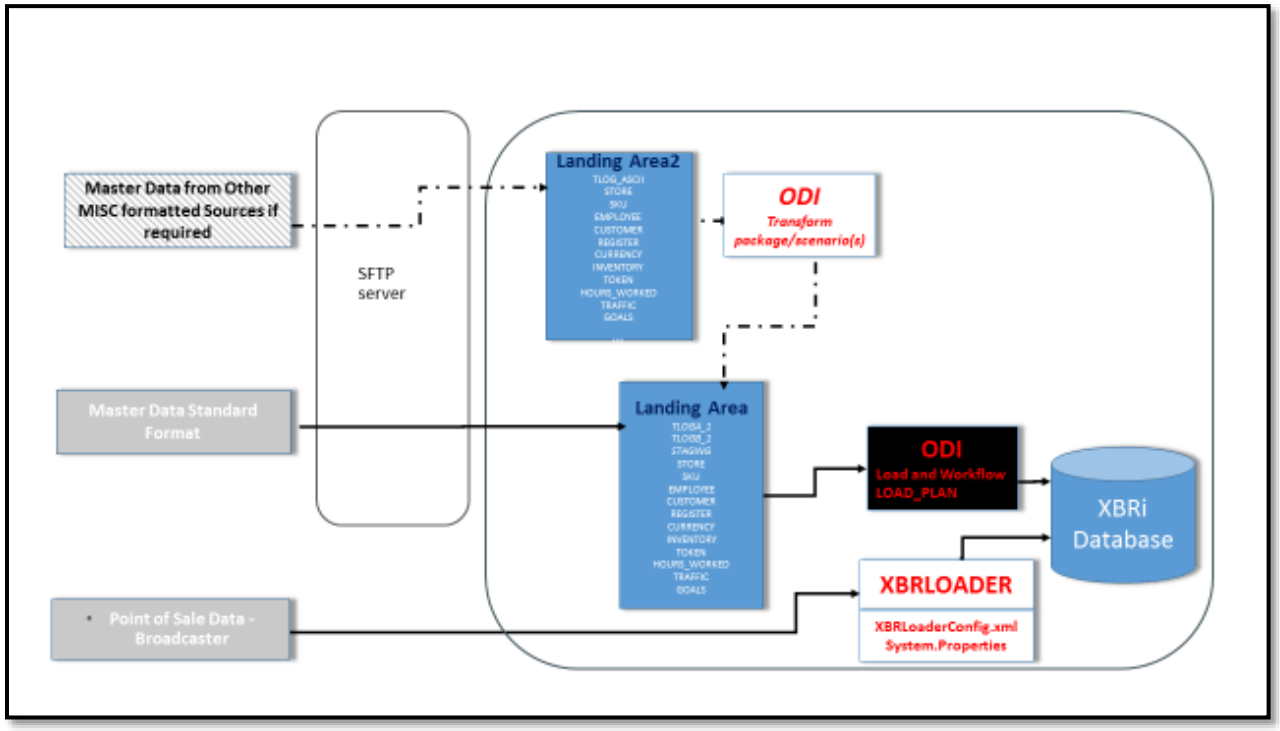
The following figure shows the data flow of XML poslogs from the tmp directory to the pos_staging table in the database. The ODI ELT black box handles file movement from the landing area to the /tmp directory and runs the stored procedures to move, and aggregate data from the staging table to the historical tables.



Data Flow: XML poslogs from tmp Directory to POS_STAGING Table.

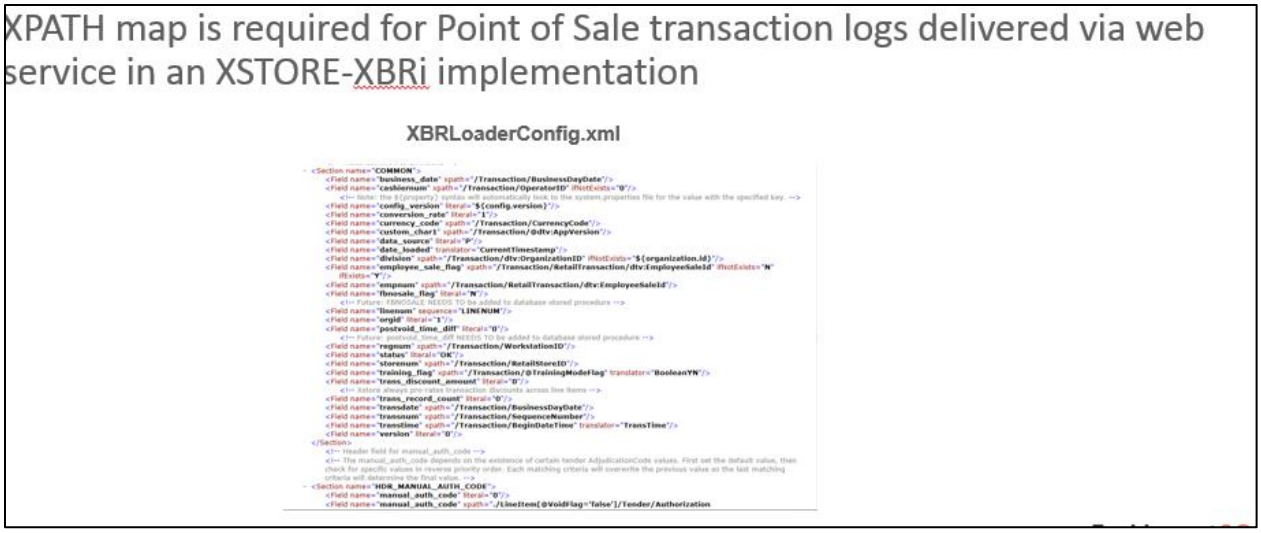
XSTORE poslogs through Web Service

XSTORE Poslogs delivered by the XCENTER Broadcaster to the XBRLOADER Web Service are transformed using a configuration. The coding is based on XPATH. Additional information on coding using XPATH is covered in Chapter 6, Xstore/XBR^{i} Integration. The ODI black box framework handles the workflow. For Xstore - XBR^{i} implementations it is important that the ODI global variable WEBSERVICE_PATH is set. The value should be DRIVE:/Xbrloader. The purpose of this variable is to control suspending loading data to the pos_staging table in the database while the database stored procedures are running.



Xstore poslogs to XBRLOADER Web Service File Delivery Diagram

As part of the Cloud provisioning, if the environment is to be used for an Xstore-XBR^{i} implementation, the XBRLOADER directory is created. It contains a BASE mapping for XSTORE poslogs. You must modify this file to account for modifications to the XSTORE poslog. See example below:



XBRLoaderConfig.xml

ASCII File any format

Many Point of Sale systems generate ASCII files containing transaction data, some are delimited some are fixed length. The format of the data varies by vendor and even with the same vendor there are variations.

If the point of sale transaction log is delivered as an ASCII file and not in standard Oracle format (POS_STAGING.DAT file that has been mapped to conform with the XBR*i* Data Model), you must create an ODI package/scenario that transforms the data into the standard format, POS_STAGING.DAT, which can be consumed by the ODI ELT black box framework.

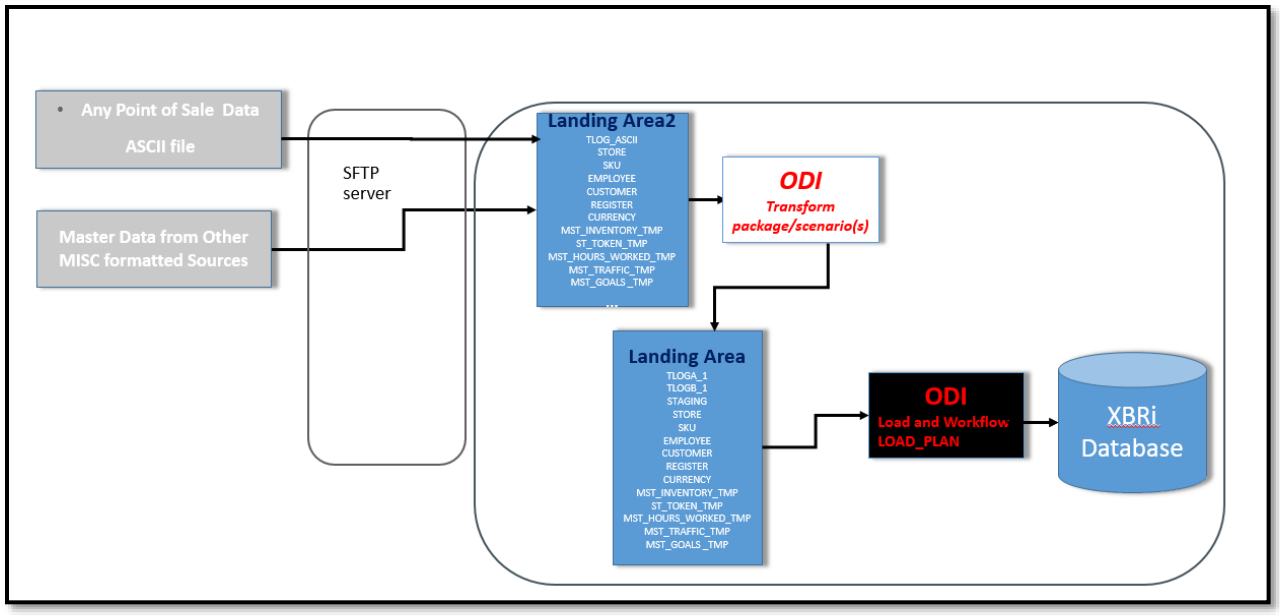
This scenario must be scheduled through ODI. The unformatted transaction log must be processed and then delivered to the core ELT. It cannot be loaded to the database directly because this separate process will not have visibility in the workflow, and loading the master file data while the database stored procedures are running would be problematic

It involves the following steps:

1. Consuming the transactional logs from the sftp site,
2. Transforming the data to the pos_staging.dat file format,
3. Placing the pos_staging.dat file and a trigger file (pos_staging.dat.trg) in the INCOMING_FILES/STAGING directory.

This package/scenario can run multiple times during the day – the file name can be "anything" but a date timestamp_pos_staging.dat is recommended. This package/scenario exists outside of the black box ELT framework. All of the MMDDYYYY_pos_staging.dat files that are place in the INCOMING_FILES/STAGING directory are processed by the next ELT run.

The following figure shows the flow when transformation for both transactional log data and supplemental data feeds is required through an ODI instance in the Cloud before the ODI black box can load the data to the XBR*i* database.



ASCII File Delivery Diagram

Notes:

If the point of sale transaction log is a binary file, before delivering it to the Cloud SFTP server, it should be converted in the customer's environment to an ASCII file.

If ODI is being used, the sftp script is responsible for file movement from the sftp site to a second landing area. INCOMING_FILES_2 directory needs to be created for processing this data.

When promoting from the implementation environment to the staging environment, this scenario and related objects must be exported and handed off to the Cloud team. In addition to the new landing area and working directory, the structure must be communicated to the Cloud team.

Binary File any format

The party responsible for transformation would be responsible for creating a pre-process which transforms the Binary transactional log file to an ASCII file. This pre-process would have to be performed prior to delivering the ASCII file to the Cloud sftp site. In these cases it may be advisable to not only translate the binary data but to transform it as well so that the pos_staging.dat file is delivered ready to load immediately.

ODI ELT – Steps

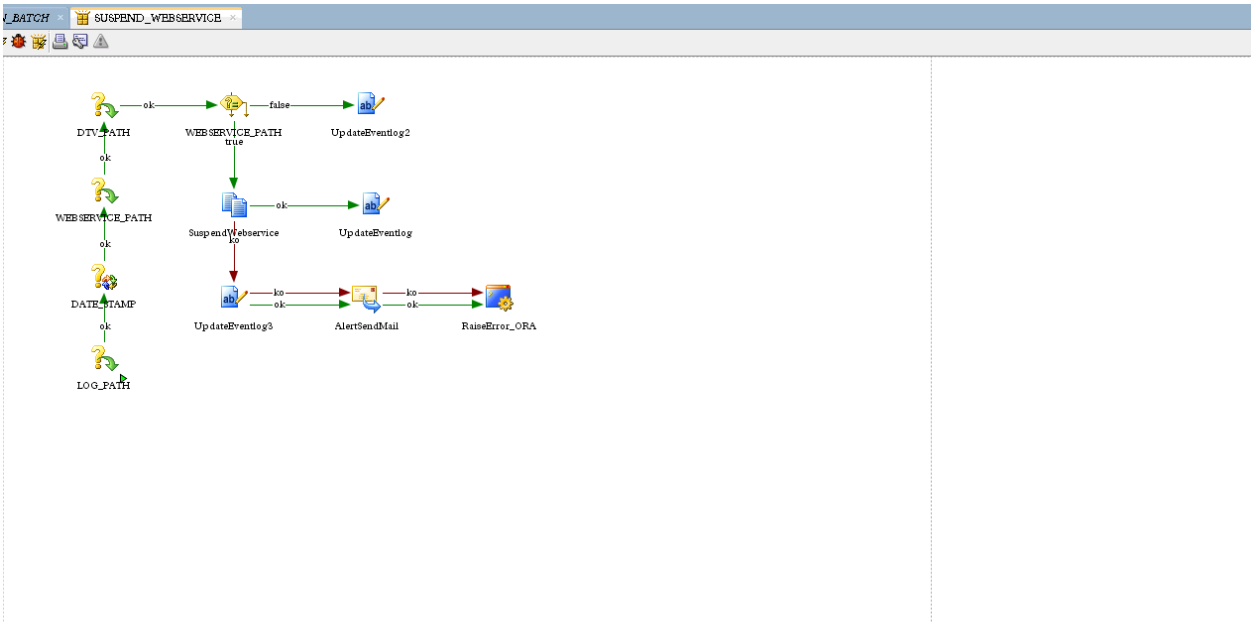
XBR_GEN_SETUP

ODI package that checks the status from the prior run, purges old log files, zips up the log files from the last run and purges old data files.

XBR_GEN_SETUP

SUSPEND_WEBSERVICE

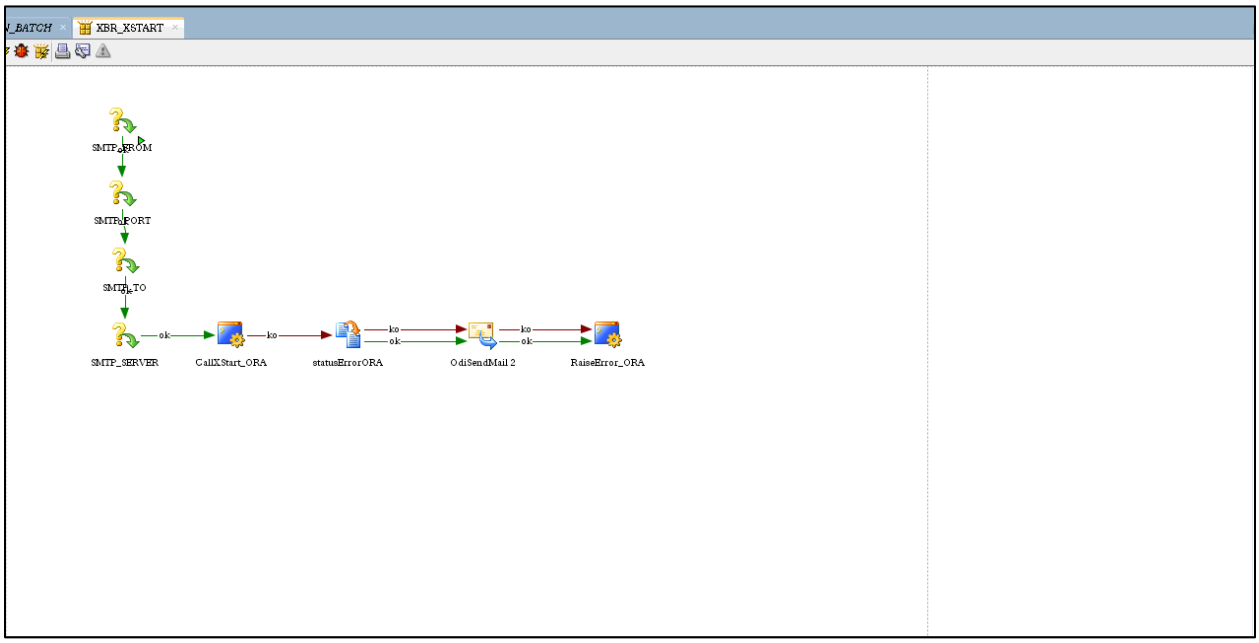
Suspends the XSTORE_XBRi webservice from loading data to the pos_staging table.



SUSPEND_WESERVICE

XBR_XSTART

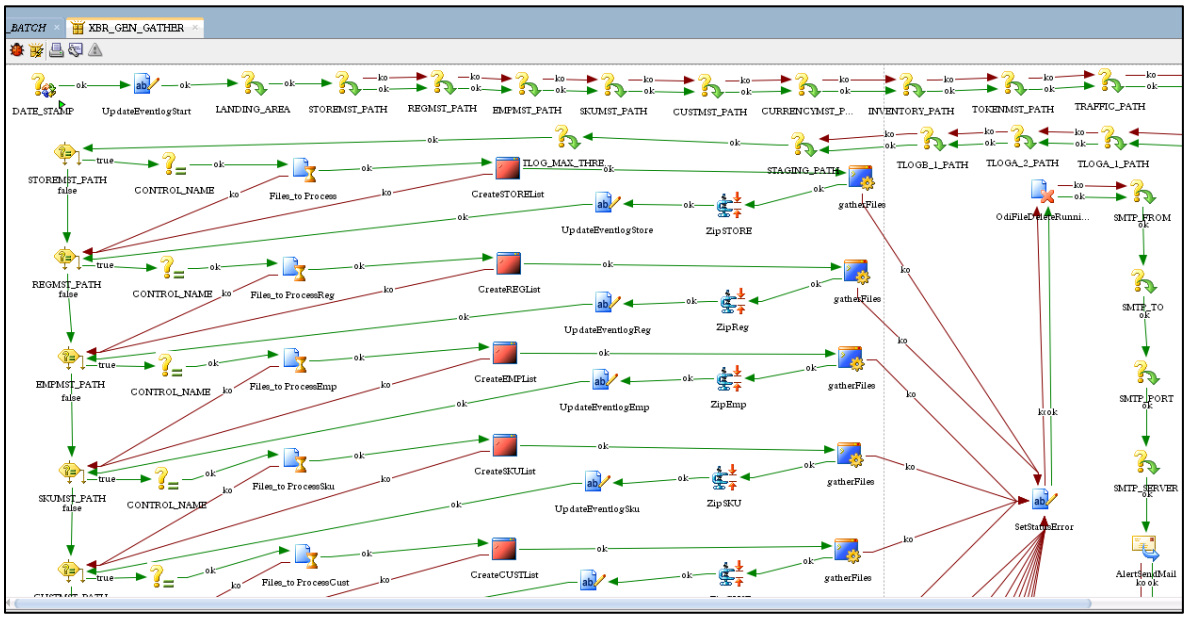
Executes the sp_xstart stored procedure in the database.



XBR_START

XBR_GEN_GATHER

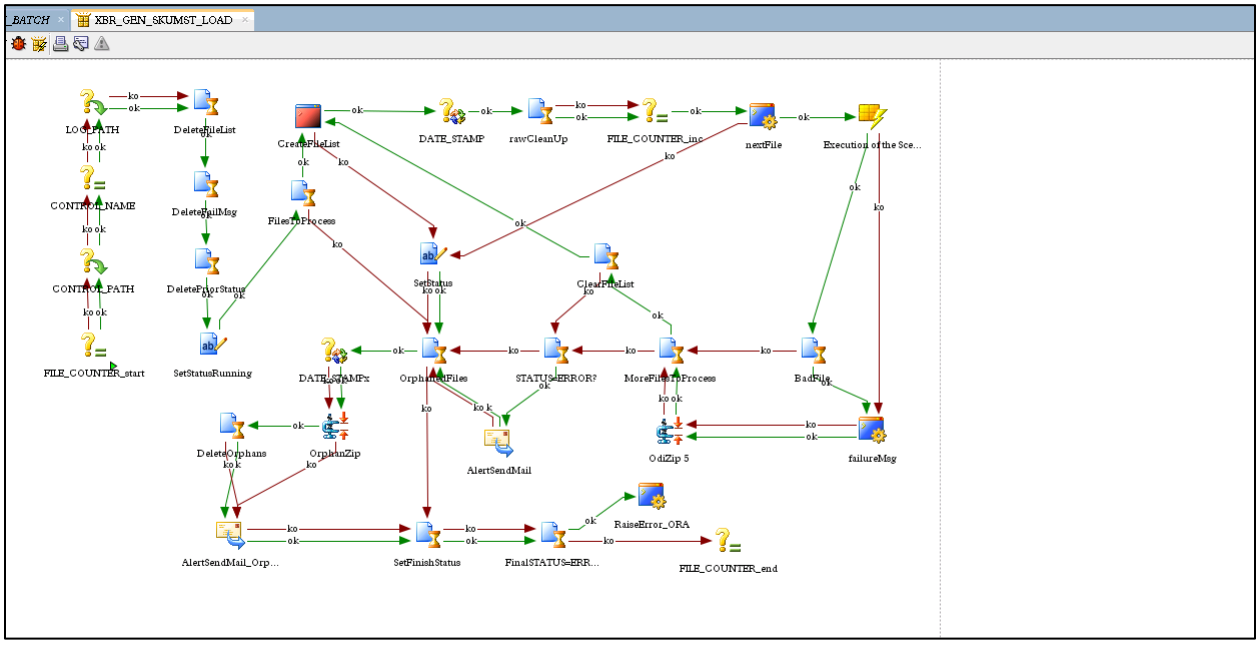
Gathers files from the landing area (INCOMING_FILES) and delivers them the appropriate TRANSFORMS/xxxxx/tmp directory. Additionally all data files are archived to OUTPUT_ARCHIVE.



XBR_GEN_GATHER

XBR_GEN_SKUMST_LOAD, XBR_GEN_CUSTOMERMST_LOAD….

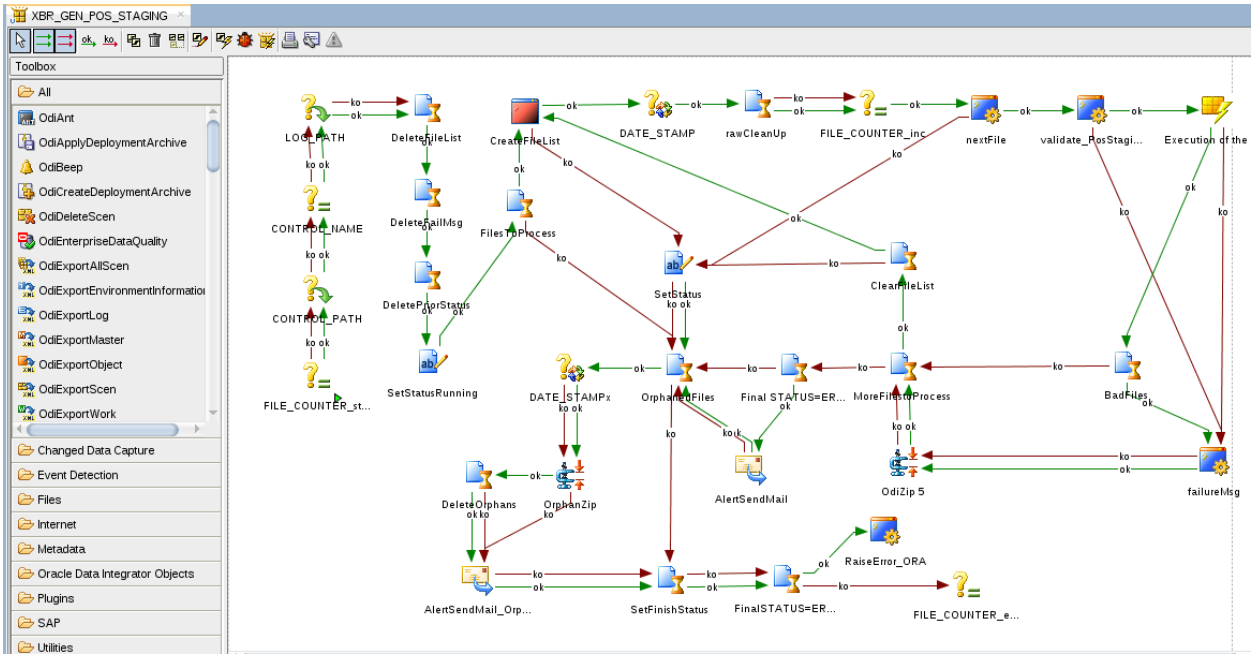
All master files follow the same process to map data to the _TMP version of the table in the database.



XBR_GEN_SKUMST_LOAD, XBR_GEN_CUSTOMERMST_LOAD….

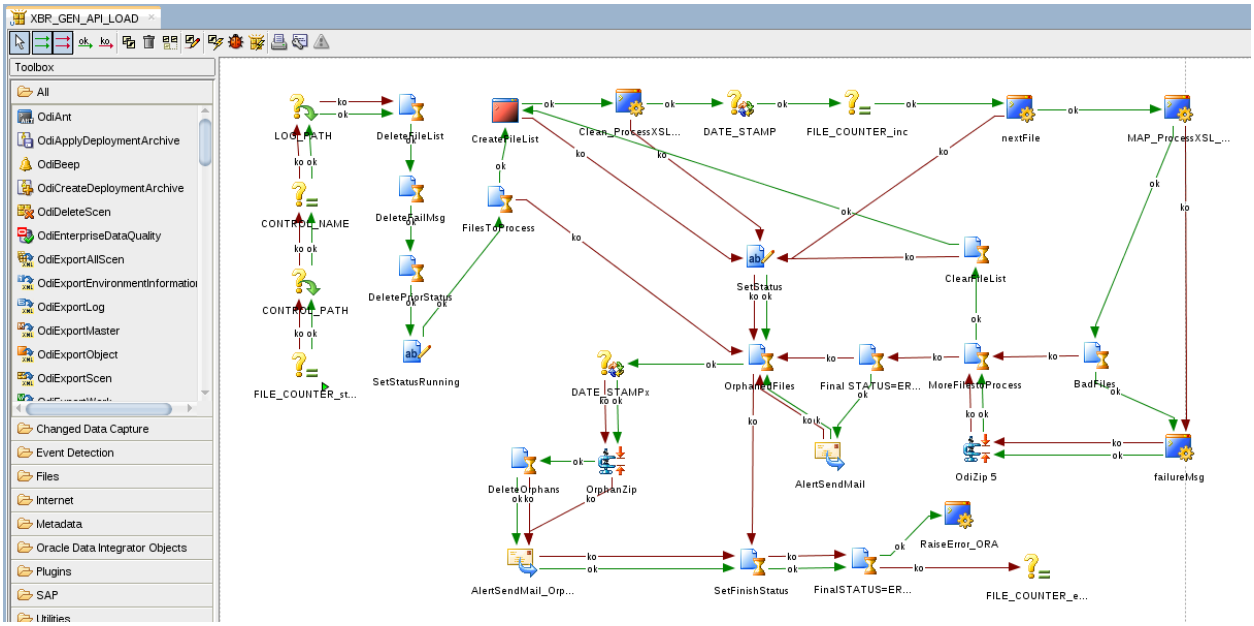
XBR_GEN_POS_STAGING

Maps a POS_STAGING.DAT file to the POS_STAGING table in the database.



XBR_GEN_POS_STAGING

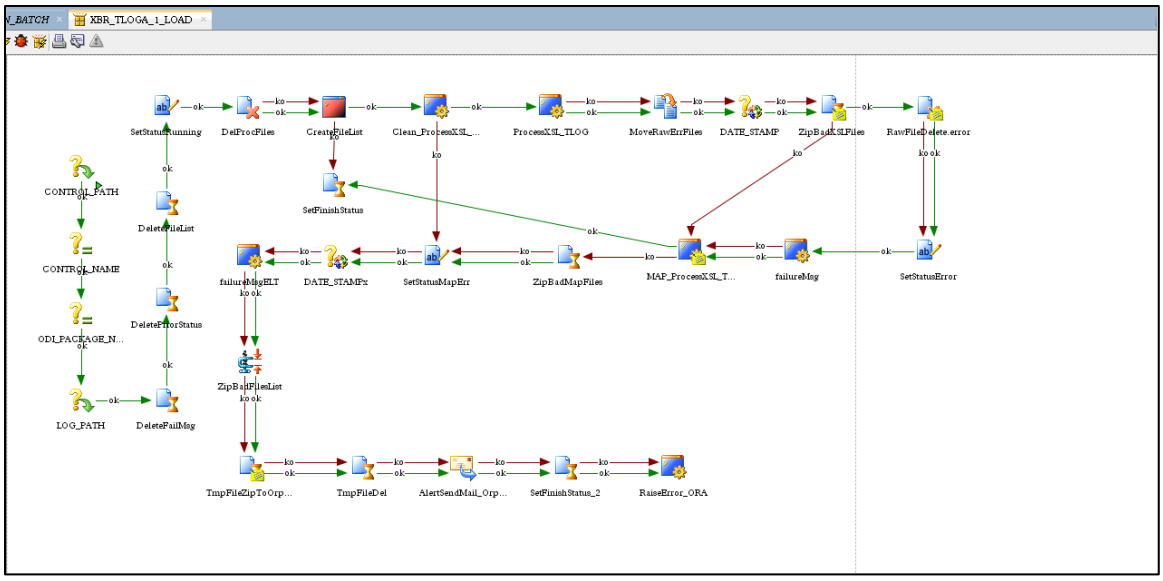
XBR_GEN_API_LOAD



XBR_GEN_API_LOAD

XBR_GEN_TLOGA_1, XBR_GEN_TLOGA_2, XBR_GEN_TLOGB_1

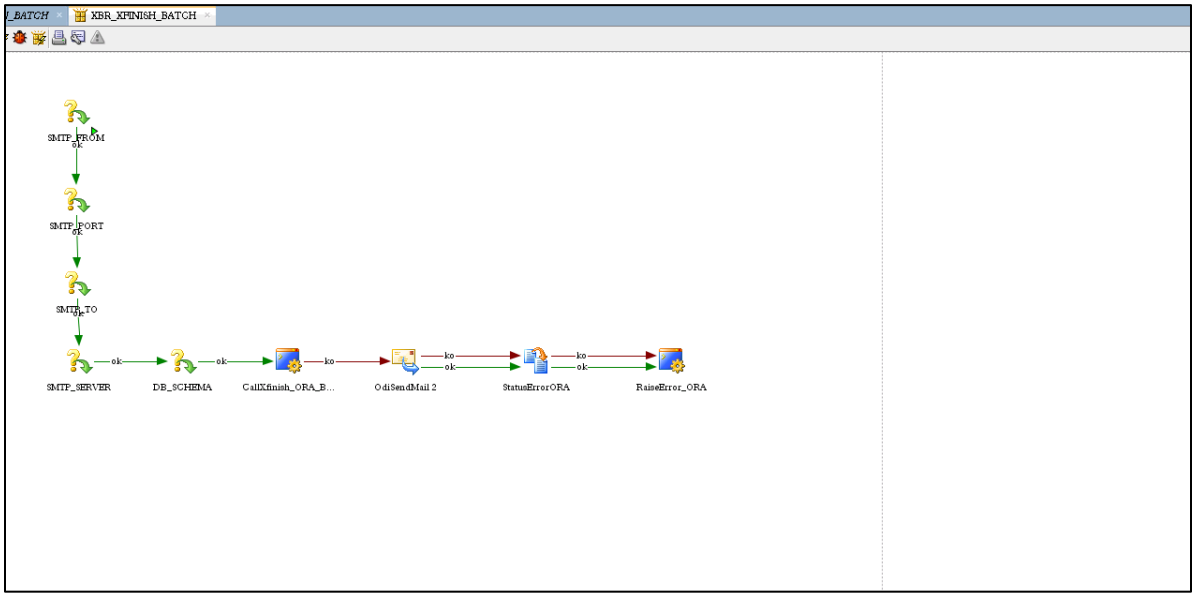
The package that handles the flow when transforming XML tlogs to a standard canonical format and ultimately loading to the POS_STAGING table in the database.



XBR_GEN_TLOGA_1, XBR_GEN_TLOGA_2, XBR_GEN_TLOGB_1

XBR_GEN_XFINISH_BATCH, XBR_GEN_XFINISH_REAL, XBR_GEN_XFINISH_EOD

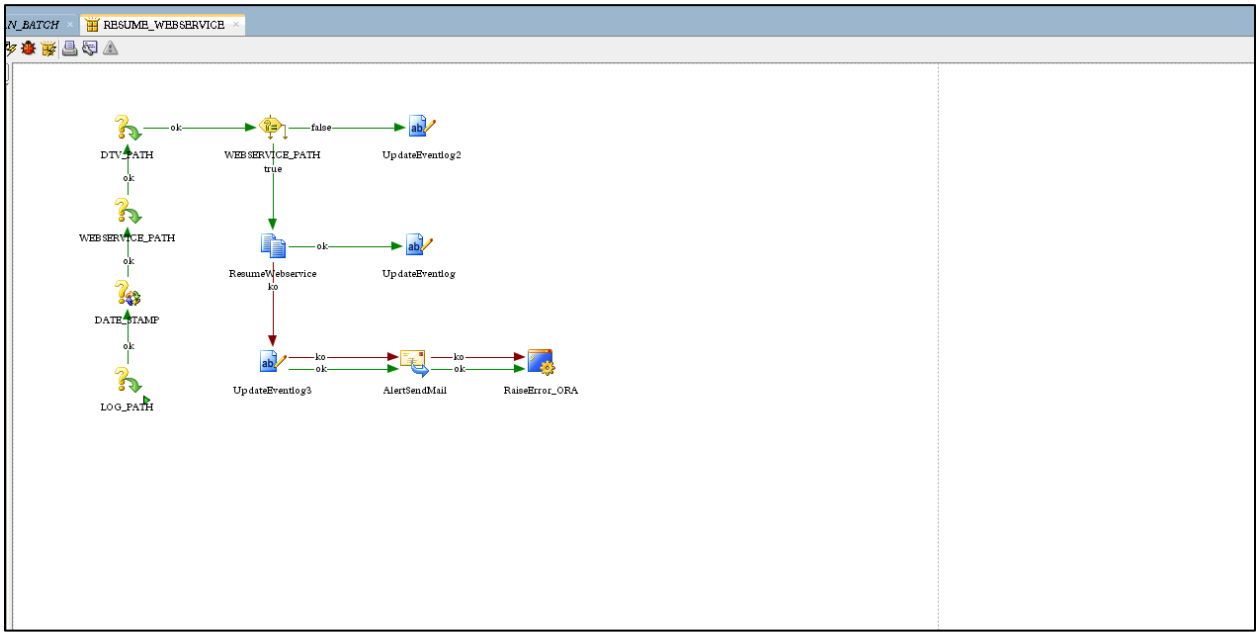
Package that calls the corresponding sp_XFINISH stored procedure in the database.



XBR_GEN_XFINISH_BATCH, XBR_GEN_XFINISH_REAL, XBR_GEN_XFINISH_EOD

RESUME_WEBSERVICE

Package that resumes the XSTORE-XBRi webservice load to pos_staging.



RESUME_WEBSERVICE

SETSTATUS

Procedure that sets the DTV/XBR/status to finished.

SETSTATUS

6

Xstore/ XBR*i* Integration

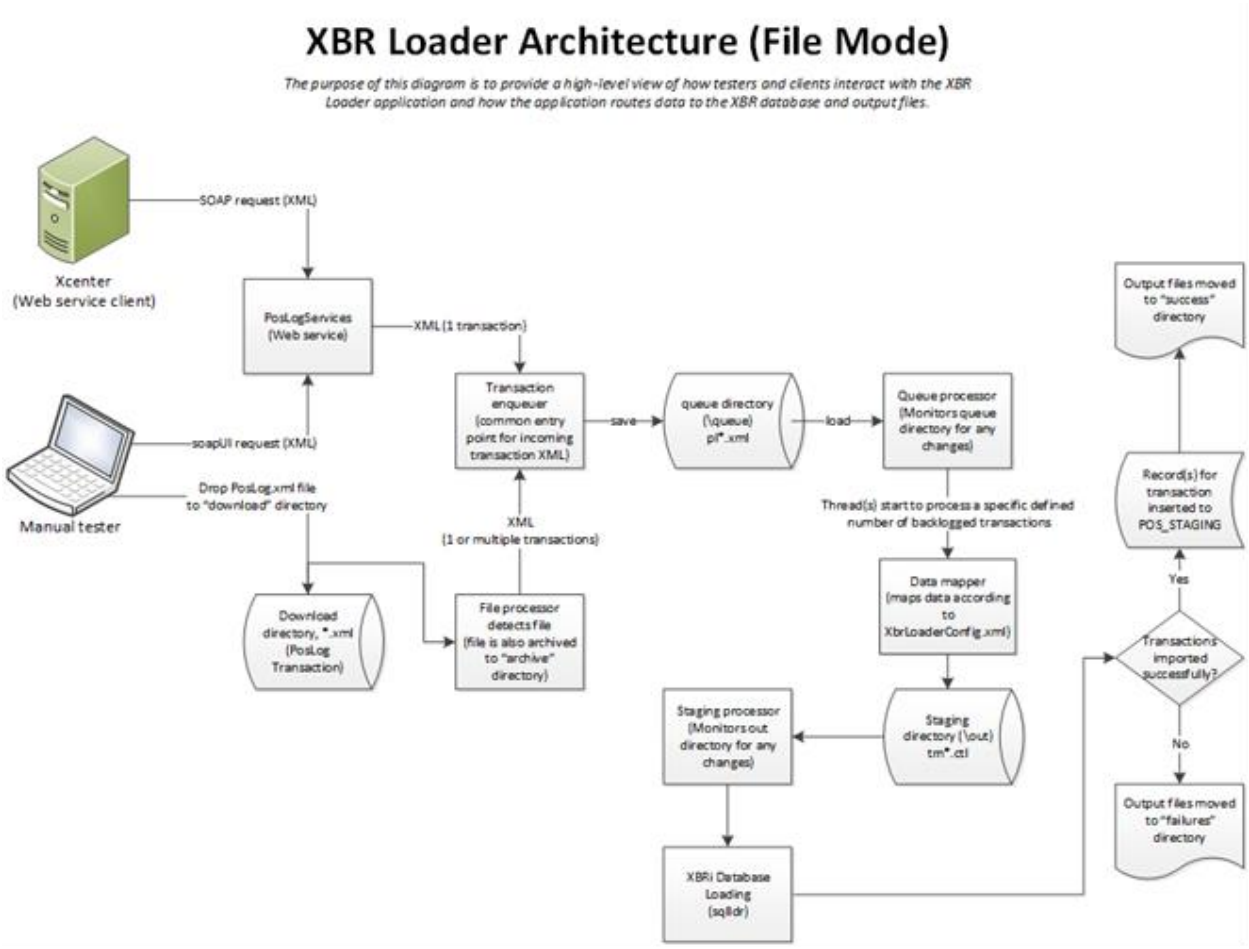
Introduction

The XBR Loader is a web application that uses the Broadcaster web service to extract transaction data from the Xstore/Xcenter database and transfer it to the XBR*i* database where it is processed for use in an XBR*i* data model. Since a web service is being used, this process occurs automatically once the integration is complete. The database tables and transaction records of XBR*i* and Xstore are dissimilar in structure thus necessitating a configuration file based on XPath to perform the transformation between the structures.

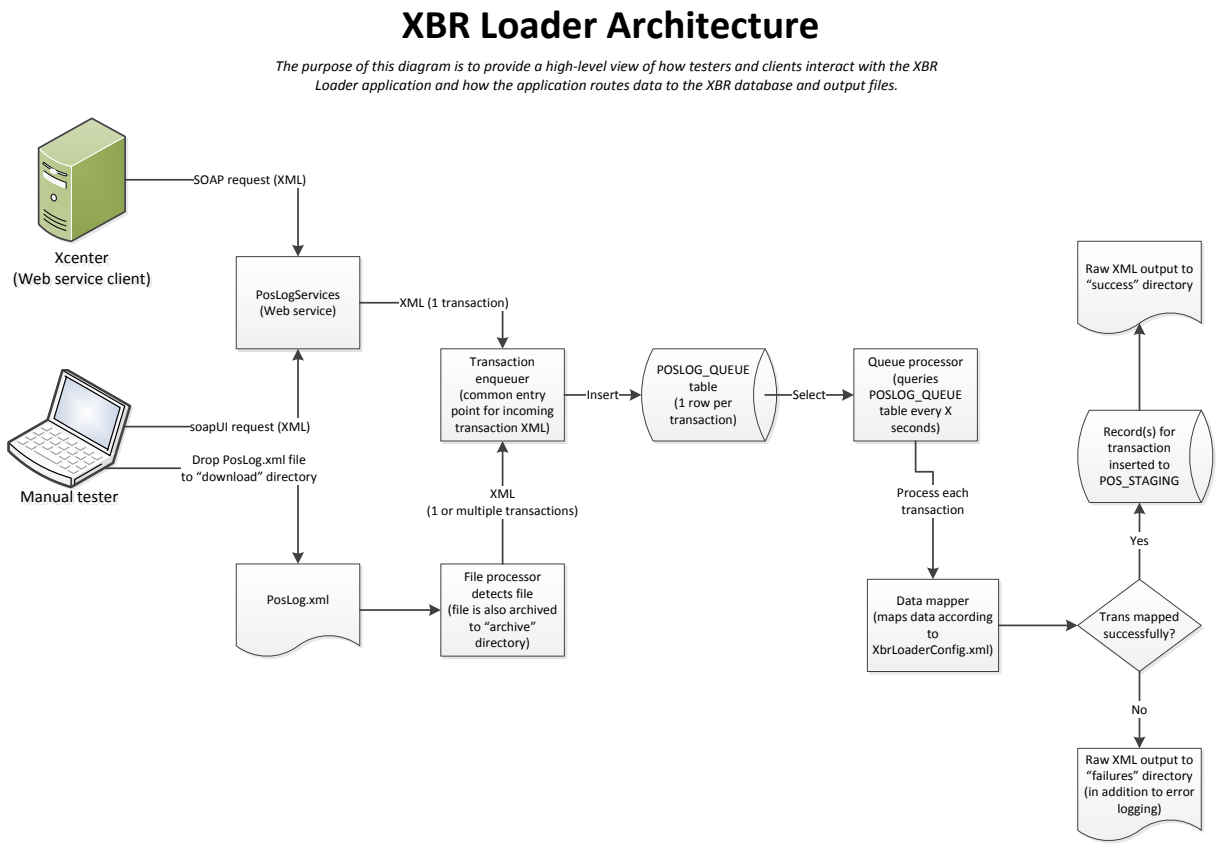
XBR Loader Architecture

The Following diagrams show the two different modes of XBR Loader architecture.

File Mode (Default)



Database Mode



Components of an Xstore- XBR*i* Web Service Integration

Xstore/Xcenter 6.5 or Later

This implementation option is only available to customers implementing XBR*i* who have Xstore 6.5 or later as their point of sale.

XBR*i* Broadcaster Enabled in Xcenter

The broadcaster system in Xcenter is a pluggable component that will allow posting to any third-party system (such as Relate, Serenade, XBR*i*, and so on) through platform-independent technologies (such as Web services), as transactions flow through existing Xcenter servlet for persistence. The XBR*i* system requires raw string poslog data, which is exactly the same as what is sent from Xstore to Xcenter. The partner will need to coordinate with the XSTORE team and the Cloud team to insure the broadcaster is enabled in Xcenter.

XBR*i* Database

There are two tables in the XBR*i* database that are unique to a web service integration that is in Database Mode. If the web service was installed in File Mode, the tables are not used.

POS_POSLOG_QUEUE

Table to hold the individual transactions before they are transformed and loaded to the POS_STAGING table in XBR[i].

POS_QUEUE_STATUS

Table to maintain the processing status of the queue.

XBRLoader Directory Structure and Associated Files

The file structure for XBR Loader is as follows:

- /usr/local/xbr-loader
  - Top level directory structure for the xbrloader application
- /usr/local/xbr-loader/config
  - Directory containing all configuration files
  - Some of the files in this directory should be preconfigured based on the values entered during the installation performed by the Cloud team. Any file which is manually changed will need to be delivered to the Cloud team prior to promoting to the Staging or Production environments .properties files for configuring the application should be properly formatted by the installation. Email.list contains the e-mail address of the point of contact for alerting when web service errors occur
  - Base/XbrloaderConfig.xml

    The XbrLoaderConfig.xml configuration file governs how specific POSLog XML elements map to the POS_STAGING database table. The file is designed and structured to promote maintainability and flexibility for users that need to make changes to the base data mappings and override specific mapping components for client implementations. Most Base XSTORE tags/data elements that are required to be loaded to XBRi are included in the default XbrloaderConfig.xml. Since Xstore is highly customizable work will be required to validate the required information is mapped to XBRi properly.
- /usr/local/xbr-loader/database
  - Scripts for web service database objects.
- /usr/local/xbr-loader/download
  - Directory used for manually loading poslogs
- /usr/local/xbr-loader/lib
  - XBRLoader jar files
- /usr/local/xbr-loader/log
  - xbrloader-fileprocess.log
  - xbrloader-webapp.log
  - In the Cloud a symbolic link will be set up to the following common log directory /usr/local/xbr-loader/log/*.log.
- /usr/local/xbr-loader/out
  - Failures and success subdirectories for archiving individual poslogs as they are moved from the poslog_queue to the pos_staging table
- /usr/local/xbr-loader/tmp

The xbr-loader.war is the WAR file (Web application Archive) that has been deployed to the Tomcat application server by the Cloud team during the initial installation of the XBR-Loader application.

Tomcat Services Configured to Run XBR Loader Application

A Tomcat service for running the XBR Loader is provisioned by the Cloud team.

Each change to any of the XBR Loader configuration files requires the Tomcat service to be restarted. The Tomcat service can be controlled as follows:

- To show status of the tomcat02 service:

  sudo systemctl status tomcat02.service
- To see if service is active or inactive:

  sudo systemctl is-active tomcat02.service
- To stop the service:

  sudo systemctl stop tomcat02.service
- To start the service:

  sudo systemctl start tomcat02.service

Submitting Transactions

Once The XBR Loader is installed, you must submit transactions, in the form of well formatted XML poslogs, to the loader from the Xcenter database. You can do this automatically, using a web service or manually, by copying them to a download directory. The location of this directory is configurable in the system.properties file but should not be changed in the Cloud.

Web Services Submission (File Processor)

The primary way to submit transactions to the XBR Loader is by invoking a Web service method.

The service responds with a SOAP message indicating the number of successfully queued transactions (which through the Web service typically will be 0 or 1): The ***file processor*** is only responsible for loading records into the pos_poslog_queue table in the XBR database.

If the producer, XCENTER Broadcaster, is unsuccessful the poslog remains on the producer side and is reprocessed.

In the event that invalid XML is sent to the service, the service reply contains an exception message which should be logged by the XCENTER broadcaster. If this occurs, the XML payload is also written to a file in the configured "failures" directory. The file name is:

/usr/local/bxbr-loader/out /failures/**bad.malformed.tlog.{Date/Time}.xml**

Manual File Submission

In addition to submitting transactions through the Web service, PosLog.xml files generated at the POS can be manually copied to a configurable download directory. The default download directory is:

/usr/local/xbr-loader/download

XML files placed in this folder may contain any number of transactions, however file names must end in ".xml" to be recognized by the processor. To process XML files in the download directory execute the xbr-loader.sh batch script:

/usr/local/xbr-loader/xbr-loader**.**sh

XML files are archived after processing. The default archive directory is:

/usr/local/xbr-loader/archive

In the event that invalid XML is entered, the XML poslog is written to a file in the configured "failures" directory. The file name is:

/usr/local/xbr-loader/out /failures/bad.malformed.tlog.{Date/Time}.xml

POS_STAGING Data Load (Queue Processor)

This section describes the two modes of data loading.

File Mode

By default, the web service is installed in File Mode. When in file mode, the queue processing is controlled by threads. Depending on the number configured, a spare thread picks up a pending queued transaction file from the staging directory:

/usr/local/xbr-loader/staging/)

And transforms them to be processed by the bulk loading tool.

Database Mode

In Database mode, the loader processes records from the queue table at a timed interval. A number of transactions up to a configurable maximum per cycle are read from POSLOG_QUEUE and mapped into POS_STAGING. Transactions are processed in the order received.

In both modes, the XBR Loader data mapping engine relies heavily on XPath to map POSLog fields into POS_STAGING. The format of the XbrLoaderConfig.xml is dictated by its corresponding XML schema (XSD) file:

/usr/local/xbr-loader/config/base/**XbrLoaderConfig.xsd**

> **Note:** Any changes to data mapping configuration files are not recognized by the application until the Tomcat service is restarted.

If mapped successfully, the raw XML for a transaction is written to a file in a configurable "success" directory.

The default success directory is:

/usr/local/xbr-loader/out/success/success.tlog.{Store#}-{Register#}-{Trans#}.{Date/Time}.xml

If mapping is unsuccessful, the raw XML for a transaction is written to a file in a configurable failures directory.

The default failures directory is:

/usr/local/xbr-loader/out/failures/bad.transform.tlog.{Store#}-{Register#}-{Trans#}.{Date/Time}.xml

If the XBR Loader is unable to map the transaction to POS_STAGING due to a "soft failure" (database offline, and so on.), the transaction XML remains in the POSLOG_QUEUE table/Staging directory, the RETRY_COUNT column is incremented and the FAILURE_REASON column is updated. Note that transactions with lower retry counts have precedence when the queue processor determines which transactions to process.

XBR Loader should not be loading transactions to POS_STAGING during the execution of database stored procedures XSTART and XFINISH. The “CommandProcessor” application within the web service handles suspending and resuming execution as described in the following sections. The ODI ELT framework is responsible for creating the suspend.command and resume.command files at the proper points in the workflow. Whether the customer is operating in a “Real-Time” or Batch mode model as described in Chapter 2, The XBRi Data Model, population of pos_staging will be suspended and transactions will be queued in the poslog_queue table during each ODI ELT execution.

Suspending and Resuming Poslog Dataload (CommandProcessor)

This section describes suspending and resuming the data loading process.

Suspending

To suspend the queue processor and prevent new data from being inserted to POS_STAGING (in the event that the table contents need to be frozen for XBRi), place a file named "xbr.command" into the download directory with "suspend" as the file contents.

Resuming

To resume processing of records from POSLOG_QUEUE into POS_STAGING, place a file named "xbr.command" into the download directory with "resume" as the file contents.

> **Note:** Restarting the Tomcat service will automatically cause the queue processor to resume processing.

Purging (PurgeProcessor)

This job controls the interval at which the download and output file directories are purged of old files based on the "purge.*.days" value in system.properties. This is informational and will be controlled by the Cloud team. If a customer requires a change to this interval, reach out to the Cloud team.

Mapping Data through XPath

This section describes data mapping through XPath.

Introduction and Basic Syntax

XPath, the XML Path Language, is a query language for selecting nodes from an XML document. As such, it is designed around XML structure. Here are some basic operators:

self::node() refers to the XML node currently being operated on

parent::node() refers to parent of the current XML node

./xxxxxx refers to child xxxxxx of the current XML node

../xxxxxx refers to child xxxxxx of the parent of the current XML node

xxxxxx[] square brackets immediately after a node enclose a conditional statement which determines if that node is to be used

@xxxxxx refers to an attribute of an XML node

xpath="..." means a path to some XML node, usually starting with / (top node of an XML file), ./ (current node), or ../ (parent of the current node).

literal="..." means literal string to be assigned or otherwise used

${xxxxxx} means evaluate macro xxxxxx ; XPath term for a macro is “XpathAlias”

ifExists and **ifNotExists** are custom conditional operators used in our Webservice to evaluate expressions if some node exists (or does not exist)

negateIf="..." is a custom functions used in our Webservice to reverse the sign of a numerical value if condition within quotes is met

negateOnVoidRecord is a custom functions used in our Webservice to indicate that a value going into Pos_Staging must be included with reversed sign if a voiding record gets generated for the Pos_Staging detail row currently being constructed.

= sign is used for both assignment and checking equality. To check for inequality, one must **use !=** sign; <> is not valid in xpath.

Root XML Elements

These are the root XML elements for XPath:

XPathAlias

<XPathAlias>

The XPathAlias element is used to define XPath expressions that need to be referenced many times throughout the configuration file. XPath expressions defined as aliases can be utilized elsewhere in XbrLoaderConfig.xml using the ${expression} syntax. The intent is to eliminate the need for complex definitions throughout the file and minimize typing.

Attributes of <XPathAlias>:

name (required)

the name of the alias value (required)

the XPath expression that the alias defines

InsertSet

<InsertSet>

The InsertSet element defines a set of insert statements (table rows) for POS_STAGING. Typically, a single POSLog transaction will only qualify for one specific insert set, and that insert set will determine how many and what types of records to load into the XBR database.

Each <InsertSet> node is designed to identify in the tlog a particular type of transaction, and to convert it into Pos_Staging format. XbrLoaderConfig.xml has following <InsertSet>'s:

- <InsertSet name="RETAIL_TRANS">
- <InsertSet name="NO_SALE">
- <InsertSet name="PAID_IN">
- <InsertSet name="PAID_OUT">
- <InsertSet name="STORE_OPEN">
- <InsertSet name="STORE_CLOSE">
- <InsertSet name="TENDER_EXCHANGE">
- <InsertSet name="TENDER_EXCHANGE_GIFTCARD_RETURN">

Section

<Section>

The Section element defines set of <Field> tags that belong to a single, logical grouping. For example, a section might define all fields common to a non-merchandise item record or a line item discount record.

Translator

The Translator element defines a value translator that can be used within the data mapping sections of the file. Simple translators are used to map input data to POS_STAGING values on a one-to-one basis; complex translators are driven by custom Java classes and used for less straightforward mappings. Translators are XPath equivalents of SELECT statement – a function which returns one of many set choices depending on as many sets of conditions.

Custom Translator Classes

All custom translator classes reside in the "dtv.xbr.translator" Java package. Fully-qualified class names must be supplied in the <Translator> "class" attribute (for example "dtv.xbr.translator.CoalesceTranslator") when using custom translators.

CoalesceTranslator

This translator selects the first non-null value from a list of configurable XPath expressions.. Parameters for this class must begin with the string "xpath." to be evaluated by the translator, whose name is derived from the popular COALESCE SQL function.

DateTranslator

This date translator requires input data and formats the supplied input data (which must be a date value) according to the format string defined by the "format" parameter. The supported date format pattern characters are:

Letter	Date or Time Component	Presentation	Examples
G	Era Designator	Text	AD
y	Year	Year	1996;96
M	Month in Year	Month	July;Jul;07
w	Week in Year	Number	27
W	Week in Month	Number	2
D	Day in Year	Number	189
d	Day in Month	Number	10
F	Day of Week in Month	Number	2
E	Day in Week	Text	Tuesday; Tue
a	am/pm marker	Text	PM
H	Hour in Day (0-23)	Number	0
k	Hour In Day (0-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in Hour	Number	30
s	Second in Minute	Number	55
S	Millisecond	Number	978
z	Time Zone	General Time Zone	Pacific Standard Time;PST;GMT-0800
Z	Time Zone	RFC822 Time Zone	0800

DiscountRecCodeTranslator

This translator requires input data and resolves a supplied discount reason code to an XBR- appropriate discount rec code value based on the discount type. The supported parameters are:

- **lineDiscountExpr**

  XPath expression used to qualify the current root node as a line item discount.
- **transDiscountExpr**

  XPath expression used to qualify the current root node as a transaction discount lineDiscountPrefix

  Character string to prepend to the supplied discount reason code for line item discounts
- **transDiscountPrefix**

  character string to prepend to the supplied discount reason code for transaction discounts

DurationTranslator

This translator calculates the duration (in minutes) between a start time value ("beginTime" parameter) and corresponding end time value ("endTime" parameter). Parameters are configured as XPath expressions relative to the current root node.

PriceOverrideTranslator

This translator calculates the aggregate price override amount for a specific XML node set. The supported parameters are:

- nodeList
- the XPath expression defining the node set to aggregateFor example, header-level price override amounts aggregate all non-void price override modifiers for an entire transaction; item-level price override amounts aggregate non-void modifiers for a single line item

oldPrice

The XPath expression relative to each node in the node set used to determine the "old price" (pre-override) for the price modifier.

newPrice

The XPath expression relative to each node in the node set used to determine the "new price" (post-override) for the price modifier.

Multiplier

XPath expression indicating the factor by which to multiply the difference between the new and old price amounts in order to obtain the desired value for POS_STAGING.

SignedValueTranslator

This translator requires input data and returns a different literal value depending on whether the supplied input data (which must be numeric) is positive, negative, or zero. The supported parameters are:

- positive
- the literal value to use if the input data is positive
- negative
- the literal value to use if the input data is negative zero
- the literal value to use if the input data is zero

XpathTranslator

This translator evaluates a set of XPath expressions and returns a value based on the first expression that matches the transaction being processed. XPath expressions to evaluate are configured using parameter names beginning with "xpath." and each "xpath.[suffix]" parameter must have a matching "value.[suffix]" parameter. XPath expressions are evaluated in alphabetical order by suffix.

String Manipulation

Xpath has a function string-length(X) which returns the length of parameter X, and function substring(X,Y,Z) which returns the substring of X starting at Y, and Z characters long. If Y is 1, becomes equivalent to left() function; if Z is omitted, returns substring(X,Y) returns the rest of X starting at Y. There is no direct equivalent of right() function, but it can be done by combining string-length() and substring().

Example 1: To check whether last 4 characters of <TenderId> is “CARD”, you use this construction:

substring(TenderId, string-length(TenderId)-3, 4) = "CARD"

Example 2: To looking for a particular pattern in the string and do not care where exactly it occurs, there is function contains(), which returns true/false:

contains(TenderId, "CARD")

XPath Resources

XPath is a robust query language used to select elements and attribute values from XML files. The XBR Loader data mapping engine makes substantial use of XPath expressions in order to provide a flexible, extensible, and powerful mapping framework to its users. The following link is helpful for learning XPath syntax and applying it effectively:

http://www.w3.org/TR/xpath/

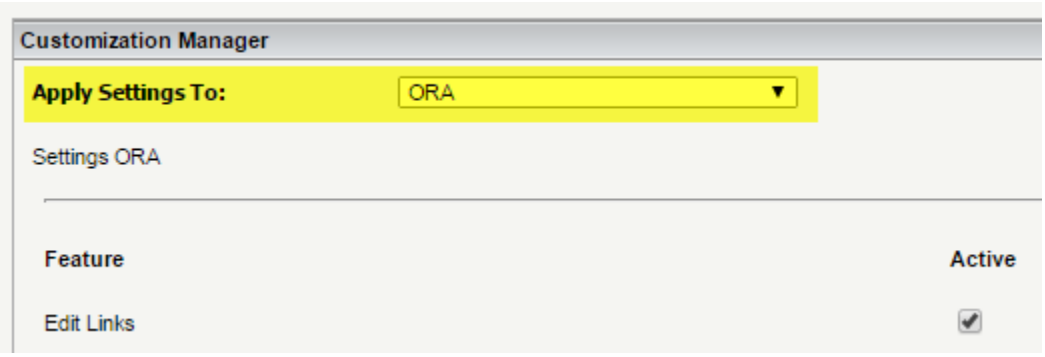
7

Manual Post Installation Steps

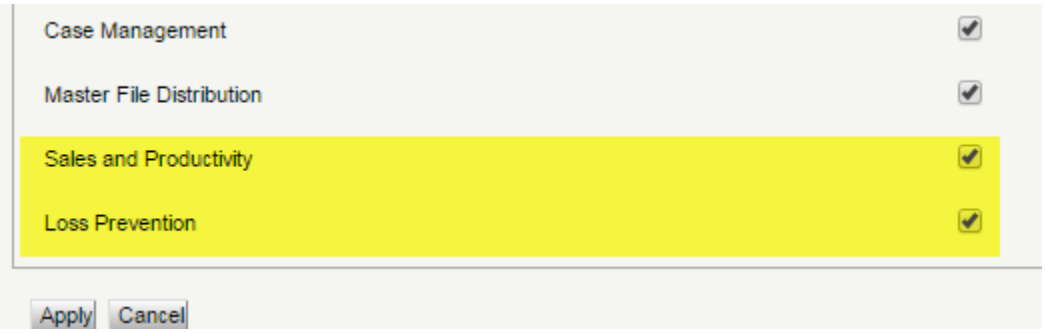
After XBRi is deployed on the Cloud server, there are a number of steps required to configure the installation for the customer. These are explained in this section.

Setting the Module Type for the Organization

1. Log in to XBR^i as Core XBR^i Administrator user.
2. From the Admin menu, choose Customization Manager.
3. From the Apply Settings to drop-down list, select the customer code.



4. In the Feature list, select the **Active** checkbox next to the module or modules to be used by the customer, **Sales and Productivity**, **Loss Prevention**, or both.
5. Click **Apply**.



The selected modules will be active by default for all new users.

Enabling Full Desc Data View Privileges for the Project

To ensure that the Full Desc Data View user privilege is available in the User Manager, you must select Full Desc Data View in the Customization Manager. Enabling this option gives Administrator users the privilege to see private data and to assign that privilege to other users.

1. Log in to XBR^i as the Core XBR^i Administrator.
2. From the Admin menu, choose **Customization Manager**.
3. In the Apply Settings To: drop-down list, if Global is selected, verify that in the Feature list the check box next to **Full Desc Data View** is selected.
4. If not already selected, select the project In the Apply Settings To: drop-down list, and verify that the check box next to **Full Desc Data View** is still selected.

5. Click **Apply**.

Setting the Default Start Page to the Home Page

1. Log in as the Core XBR^i Administrator user for the project.
2. From the Admin menu, choose **Project Defaults**.
3. Set the Default start page to Home.
4. **Apply** or **Save** the change and log out.

Creating the First Customer Administrator

After XBR^{i} is installed, you need to create a new Customer Administrator account. After you set up the account, an e-mail is mailed to the Customer Administrator with a temporary password with which to log in to the application.

To create the customer administrator:

1. Go to the Analytics website at http://WEBSERVERURL/analytics.
    a. Log into XBR^{i} using the following credentials:

    User name – the user name of the XBR^{i} administrator

    Password – default password
2. From the **Admin** menu, choose **User Manager**.
3. Click on the **Administrator** group.

    Note: If there is more than one Administrator group level, go to the lowest one.

4. Click the **Create New User** icon. This displays the User Editor.

    On the **General** tab, enter the following:

    Login name – The name the customer administrator will use when logging into the XBR^{i}.

    Full name – A description of the user.

    Set User Type – Choose Administrator from the drop-down list.
5. On the **Addresses** tab, under E-mail Addresses, click **Add a new address**, and enter the following:

    Address Name – the customer administrator's e-mail name.

    Physical Address – add the customer administrator's e-mail address.
6. Under Actions, click **Save**.
7. On the **Feature Security** tab, select the check boxes next to the Features that the customer wants enabled.
8. When finished, click **OK**.

    An e-mail is sent to the user with a temporary login password. During their first login, are prompted to specify security questions and answers.

Configuring for Master File Distribution

The Master File Distribution (MFD) feature allows customers to send subscription reports to recipients who are not defined users in XBR^i. This is done through Dynamic Address Lists that are derived from reports based on the core master tables for Store, District, and Region for Retail installations. When a dynamic address list is created, it becomes available on the Recipients list for creating e-mail subscriptions.

The reports used for creating dynamic address lists are stored in the Shared Reports > Master File Distribution folder. They include columns for Email, Device ID and Linked User ID. The Device ID column is initially populated when the application is installed or when new rows are added to the master files.

The Linked User ID column is updated in the master files when the nightly Master File automatic update process runs, or through the application for a limited number of new records. See Populating the Master File tables.

The following sections contain the steps for enabling master file distribution and manually populating the Master File tables:

Enabling Master File Distribution for the Project

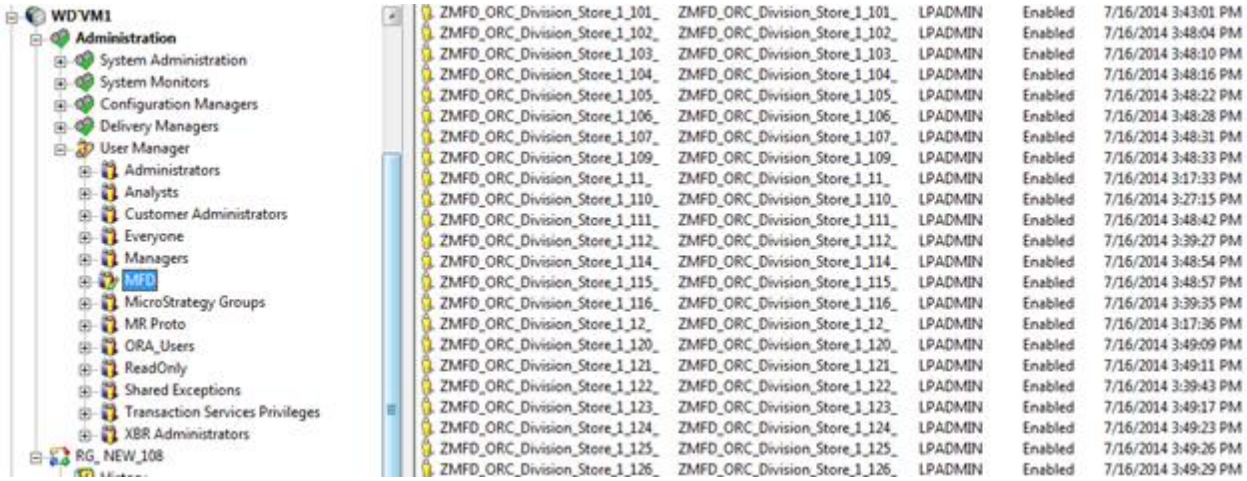
6. Log in to XBR^i as the Core XBR^i Administrator.
7. From the Admin menu, choose **Customization Manager**.
8. In the Feature list, select the check box next to **Master File Distribution**.
9. Click **Apply**.

Note: By default, this feature is active.

Populating the Master File Tables

In order for Dynamic address to work correctly, the master file tables must be populated with a linked user ID for each row in the master file. The master file tables are updated nightly through a process that is scheduled when the application is installed. For master files with fewer than 30 rows of data, customers can use XBR^i Project Defaults, Master File Distribution and run the **Refresh table** option.

The update process sets up a MFD 'dummy' user in the application, and this ID links it to the master table for each row. It also creates an associated security filter to limit the data in the emailed report to only the one store, one district, and so on.



MFD "Dummy" Users

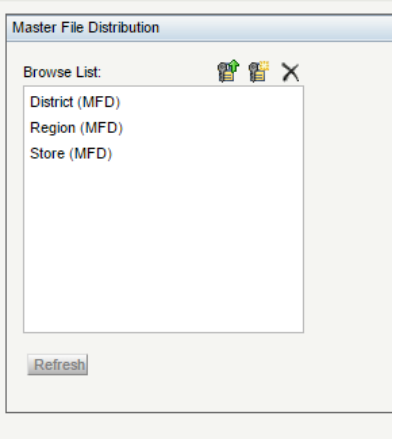
Setting Key Columns and Security Filter Key Attributes

Several attributes are used specifically for Master File Distribution. For Retail, they are Store MFD, District MFD and Region MFD. These attributes are used in the reports on which the dynamic address lists are based.

After XBR^i is installed and MFD is enabled for the project, you need to review the Master File Distribution settings in Project Defaults and verify that MFD mapping matches the key columns for each of the master tables. For example, if Store is not unique, Store and Division should be selected as keys.

Also, each of the mappings has a security filter associated with it. The security filter ensures the data sent is only for that one store or one district, and so on. These also need to be correct for the customer site.

> **Note:** If you change the security filter after an MFD update has already run, the new change will not take place until the previous MFD users and MFD filters have been deleted and the process is run again. If you need to delete MFD users or filters, use the Delete function in the Project Defaults – Master File Distribution page. See the following steps for more information.



Project Defaults – Master File Distribution Page

To verify key columns are correct:

1. Log in to XBR^i as the Core XBR^i Administrator.
2. From the Admin menu, choose **Project Defaults**.
3. Under Settings, choose **Master File Distribution**.
4. For each table in the Browse List, select the table and click the **Define key columns in database table** icon.
5. Verify that the key columns are correct.
6. If you need to add key columns, move them from the Available to the Selected list.
7. If you need to remove key columns, move them from the Selected to the Available list.
8. When finished, click **OK**.

To verify key attributes in the security filter are correct and modify if needed:

1. Log in to XBR^i as a the Core XBR^i Administrator.
2. From the Admin menu, choose **Project Defaults**.
3. Under Settings, choose **Master File Distribution**.
4. For each table in the Browse List, select the table and click the **Define key attributes for Security Filter** icon.
5. Verify that the key attributes are correct.

6. If you need to add key attributes, move them from the Available to the Selected list.
7. If you need to remove key attributes, move them from the Selected to the Available list.
8. When finished, click **OK**.

> **Note:** Only Oracle XBRAdmin Core Administrators can modify key columns and security filters.

If the key columns or security filters are modified, you must delete and refresh the data

To delete MFD users and Linked User IDs already set up by the program:

1. Log in to XBR^i as the Core XBR^i Administrator.
2. From the Admin menu, choose **Project Defaults**.
3. From Settings, choose **Master File Distribution**.
4. Select each attribute in the Browse List that you want to delete, and click the **Delete** icon.
5. When finished click **OK**.

> **Note:** If a file has more than 30 rows and you try to use the Refresh Table command in Project defaults, you will see a message that lets you know the file is too large to process. In that case, you must wait for the update process to run.

To populate the linked user ID in the Master File tables from Project Defaults:

1. Log in to XBR^i as a Core or Customer Administrator.
2. From the Admin menu, choose **Project Defaults**.
3. From Settings, choose **Master File Distribution**.
4. Select the master table you want to update in the Browse List and click the **Refresh table <Table name MFD>** button.

8

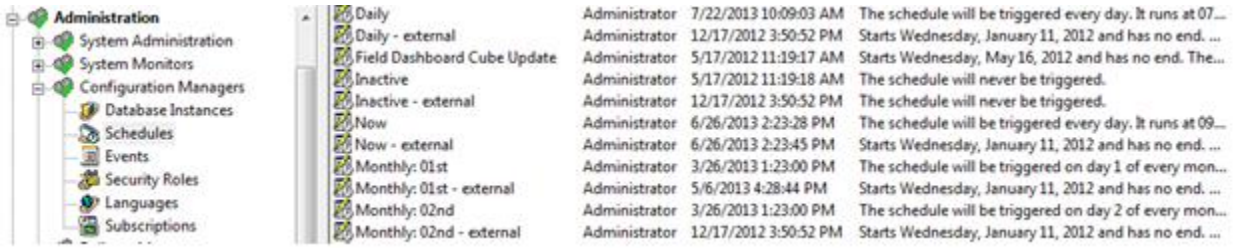
Application Configuration

This chapter provides instructions on how to do additional configuration that may be required after an installation, dependent on what each customer requires.

Adding and Modifying Schedules

The XBR^{I} OOTB schedules provided for reports, control points, and control groups are created and maintained using Command Manager scripts. In addition, schedule data must also be maintained in the data warehouse table, MD_LP_ES_SCHEDULE table. These schedules are used to process subscriptions at a future date and time such as monthly or weekly as opposed to Run Immediately or Send Now.

The schedule data maintained in the MD_LP_ES_SCHEDULE table is just for control points and groups and only used by the External Scheduler. When these schedules are selected in a control subscription, the External Scheduler is set to poll at 1 minute intervals to look for a time match. When a match is found, it will submit the job to the Iserver. The schedule data maintained in the Iserver is for all subscription types. In the Iserver, there is a duplicate of each schedule appended with **'- external** 'that is just for control points and groups. The – external schedules correspond to the schedule maintained in the MD_LP_ES_SCHEDULE table.



List of All Schedules

Adding New Schedules

If you want to create a new custom schedule that can be used for subscriptions in XBR^{i}, you need to create the schedule and a duplicate - external schedule. You also need to enter a new row for the schedule in the MD_LP_ES_SCHEDULE table in the data warehouse. These steps are executed using command manager scripts.

> **Note:** For all scripts referenced in the command line, the entire command must be in the script.

- See: Running Command Manager Scripts in this guide to review how to run the scripts.
- Use the following example scripts as templates.
- The text in red (lighter than surrounding text) in the example scripts should be modified to meet customer requirements.
- A schedule can be Event Triggered or Time Triggered.
- Event Triggered schedules can be based any event like Database load or Books Closed and so on.
- Time Triggered schedules can be set as per desired time like hourly, daily, weekly, monthly and so on.

Event Trigger Template

CREATE SCHEDULE “Schedule Name “DESCRIPTION "XXXX" STARTDATE MM/DD/YYYY ENDDATE NEVER TYPE EVENTTRIGGERED EVENTNAME "Books Closed";

Event Trigger Example

CREATE SCHEDULE "Database Load" DESCRIPTION “The schedule will be triggered by the event ‘Books Closed’" STARTDATE 09/10/2010 ENDDATE NEVER TYPE EVENTTRIGGERED EVENTNAME "Books Closed";

Time Trigger Template

CREATE SCHEDULE "Schedule Name" DESCRIPTION "XXXX" STARTDATE MM/DD/YYYY ENDDATE NEVER TYPE TIMETRIGGERED WEEKLY EVERY 1 WEEKS ON SUNDAY EXECUTE 00:00;

Time Trigger Example

CREATE SCHEDULE "Weekly: 1-Sunday" DESCRIPTION "The schedule will be triggered on Sunday of every week. It runs at 15:30:00 AM" STARTDATE 05/09/2012 ENDDATE NEVER TYPE TIMETRIGGERED WEEKLY EVERY 1 WEEKS ON SUNDAY EXECUTE 15:30;

CREATE SCHEDULE "Weekly: 1-Sunday" DESCRIPTION "The schedule will be triggered on Sunday of every week. It runs at 15:30:00 AM" STARTDATE 05/09/2012 ENDDATE 01/01/2013 TYPE TIMETRIGGERED WEEKLY EVERY 1 WEEKS ON SUNDAY EXECUTE 15:30;

Modifying Schedules

If you need to modify a schedule, for example to change the start time, you must make the same changes to the schedule time in the MD_LP_ES_SCHEDULE table in the data warehouse. This is accomplished using command manager scripts

To modify a schedule using command manager scripts:

> **IMPORTANT:** For all scripts referenced in the command line, the entire command must be in the script.

- See: Running Command Manager Scripts in this guide to review how to run the scripts.
- Use the following Example scripts as templates.
- The text in red in the example scripts should be modified to meet customer requirements.

The Command Manager Script template below can be used to alter or update the timing of existing schedules. The example also demonstrates specifics to be updated accordingly.

Template:

ALTER SCHEDULE "Schedule Name" DESCRIPTION "XXXX" STARTDATE MM/DD/YYYY ENDDATE NEVER TYPE TIMETRIGGERED WEEKLY EVERY 1 WEEKS ON SUNDAY EXECUTE 00:00;

Example:

ALTER SCHEDULE "Weekly: 1-Sunday" DESCRIPTION "The schedule will be triggered on Sunday of every week. It runs at 15:30:00 AM" STARTDATE 05/09/2012 ENDDATE NEVER TYPE TIMETRIGGERED WEEKLY EVERY 1 WEEKS ON SUNDAY EXECUTE 15:30;

List Properties of a Schedule:

LIST ALL PROPERTIES FOR SCHEDULE "Schedule1";

LIST ALL PROPERTIES FOR SCHEDULE " Weekly: 1-Sunday ";

Making Custom Fields Visible in XBR*i*

There are many custom fields provided to expand your data model, but they are delivered in the installation's hidden fields, not visible to the XBR*i* web application. You can use the use the Command Manager to make these custom fields visible to the application. The following ALTER ATTRIBUTE statement is an example of a command manager command that can make the custom fields visible. Here is a sample of the command to unhide SKU Custom Field 1 in the SKU Master:

ALTER ATTRIBUTE ATTRIBUTE "Custom SKU Char1" IN FOLDER "\Schema Objects\Attributes\Loss Prevention\Detail\SKU\SKU Master" HIDDEN FALSE FOR PROJECT "XBRI";

Running Command Manager Scripts

Command Manager lets you perform various administrative and application development tasks by using text commands that can be saved as scripts. Command Manager allows you to manage various configuration settings for project sources.

The following instructions show you how to enter Command Manager scripts from the Command Line of the XBRi Linux server through a Linux shell using a sudo account. Where command scripts are referred to in this guide, script templates and examples are provided.

How to Run a Command Manager Script in the Command Line

Note: You can only run one script at a time.

1. Navigate to the Linux sever using a Linux shell such as SSH or Putty and log into an account with sudo privileges.
2. From the command prompt, enter the command script following the template, and fill in the required information where there are placeholders enclosed in angle brackets:

   Template:

   sudo /var/opt/MicroStrategy/bin/mstrcmdmgr -n <Project Source Name> -u <User ID> -p <password> -f /usr/local/mstr/work/<Your CM Script>.scp -or /usr/local/mstr/work/<Your CM Script>ResultsFile.log -of /usr/local/mstr/work/<Your CM Script>FailFile.log -os /usr/local/mstr/work/<Your CM Script> SuccessFile.log –e

   Example

   sudo /var/opt/MicroStrategy/bin/mstrcmdmgr -n XBRi -u administrator -p xxxxxxxx -f /usr/local/mstr/work/create_script.scp.scp -or /usr/local/mstr/work/ create_script.scp ResultsFile.log -of /usr/local/mstr/work/ create_script.scp FailFile.log -os /usr/local/mstr/work/ create_script.scp SuccessFile.log –e
3. Press **Enter** to run the script.

Update PRO_SP_VARIABLES

Database Overview

Many of the database stored procedures use the variables to determine how to load the data. It is important to set the variables correctly so the data in the database loads properly. An overview of the database variables is provided in the following sections:

Note: Remember to set the variable for both VAR_VALUE and VAR_VALUE2

Note: Check and set all database variables in PRO_SP_VARIABLES table according to the installation questionnaire spreadsheet that should have been already completed.

Store and Cashier/Employee/Salesperson variables:

- STORE_UNIQUE - determines if store numbers are unique across the organization.
- CASHIER_UNIQUE – determines if cashier numbers are unique across the organization.
- EMPLOYEE_COPY - determines if cashier number is copied to employee number.
- SALESPERSON_COPY - determines if cashier number is copied to salesperson number.
- EMPNUM_USED - determines if employee number is used or always NULL.
- SALESPERSONNUM_USED - determines if salesperson number is used or always NULL.
- CASHIER_SIZE - determines the maximum size of CASHIERNUM, EMPNUM, and SALESPERSONNUM

SKU and Customer variables:

- SKU_STAGE_OVERRIDE - determines if the SKU data from staging should override SKU temp.
- CUST_STAGE_OVERRIDE - determines if the customer data from staging should override Customer temp.

General LP and SP Module Variables:

- PROCESSING_TYPE – determines the type of processing, either batch or real time.
- EOD_PROCESSING – used for real time processing when the PROCESSING_TYPE = 'real'.
- FN_PRO_VOID_SCHEME – indicates which void scheme the customer uses.
- LP_SP_OPTION – determines whether the customer purchased the Loss Prevention Module alone, or also purchased the Sales and Productivity Module.
- CAPTURE_PV_DETAILS – Indicates if the tlog captures the Post void details.
- POSTVOID_PROCESSING – Determines if the database or ETL handles the post void processing.
- FB_NOSALE – Determines if the database or ETL handles the Followed by No Sale processing.
- COMP_BACKFILL_DAYS – This is the number of days to go back to populate the spo_comp_poll table.
- POLLED_DAYS - The number of days sp_spo_comp_poll procedure will look back for late polls.
- SALES_THRESHOLD – Used for reporting sale transactions with a non-zero tender amount below a defined threshold value.

No Match Processing Variables:

- PROCESS_NM_RETURNEXCH – determines if we run the procedure that populates the return and exchange no match data.
- PROCESS_NM_PVCANCEL - determines if we run the procedure that populates the post void and cancel no match data.
- CAPTURE_ORIG_REGNUM – indicates if the tlog captures the original register number on returns.
- PV_MINS – indicates the number of minutes to look forward for post void no match processing.
- CANCEL_MINS – indicates the number of minutes to look forward for cancel no match processing.

Configuring Store & Cashier/Employee/Salesperson variables- Unique, Used, Size

Store Unique in Chain

These variables are indicated in the POS Questionnaire. Confirm them from the content of the customer's tlog and Store Master.

PRO_SP_VARIABLES.CASHIER_STORE.STORE_UNIQUE.VAR_VALUE & VAR_VALUE2

- If Store Number is unique in Chain, set to Y.
- If Store Number is not unique in Chain, set to N.
- If Store Number is not unique in Chain and is unique within division, set to N.

Cashier Unique in Chain (does not matter if cashiers float between stores)

These variables are indicated in the POS Questionnaire. Confirm them from the content of the customer's tlog and Employee Master

PRO_SP_VARIABLES.CASHIER_STORE.CASHIER_UNIQUE.VAR_VALUE & VAR_VALUE2

- If Cashier Number is unique in Store but not unique in Chain, set to N.
- If Cashier Number is unique in chain, set to Y.

Employee Copy from Cashier

These variables are indicated in the POS Questionnaire. Confirm them from the content of the customer's Employee Master.

Employee number refers to the identification of the employee that is posted to the tlog on transactions where the employee is the customer, not the employee number from the customer's HR system.

PRO_SP_VARIABLES.CASHIER_STORE.EMPLOYEE_COPY.VAR_VALUE & VAR_VALUE2

- If employee numbers on employee sales can be different from cashier numbers, set to N.
- When a record is added to the employee master as a 'NOF' (Not on File) from the tlog and the cashier number should be copied to the employee number field in the NOF, set to 'Y'.
- Info - This flag controls if EMPLOYEENUM and EMPLOYEEID are populated with the Cashiernum and CashierID respectively or left NULL by the Not On File procedure.

Salesperson Copy from Cashier

These variables are indicated in the POS Questionnaire. Confirm them from the content of the customer's Employee Master.

PRO_SP_VARIABLES.CASHIER_STORE.SALESPERSON_COPY.VAR_VALUE & VAR_VALUE2

- If salesperson numbers can be different from cashier numbers, set to N.
- If salesperson numbers are not used at all, set to N.
- If the cashier number should be copied to the salesperson number when a record is added to the employee master as a 'NOF' (Not on File) from the tlog, set to Y.
- If the salesperson field in the tlog would be populated with the same number as the cashier number of the employee, then set to Y.
- Info - This flag controls if SALESPERSONNUM and SALESPERSONID are populated with the Cashiernum and CashierID respectively or left NULL by the NOF procedure.

Employee Number Used in Tlog

This variable is indicated in the POS Questionnaire. Confirm them from the content of the customer's tlog and Employee Master.

This is true when an employee is the customer for a transaction and their employee number is captured and posted in the tlog

PRO_SP_VARIABLES.CASHIER_STORE.EMPNUM_USED.VAR_VALUE & VAR_VALUE2

- If employee number field in the tlog is not null for employee sales and returns, set to 'Y'.
- If employee number field is null in the tlog for employee sales & returns, set to 'N'.
- Info – the employee number in the employee master is expected to match the number in the employee number field in the tlog for employee sales and returns.

Salesperson Number Used in Tlog

This variable is indicated in the POS Questionnaire. Confirm them from the content of the customer's tlog and Employee Master

PRO_SP_VARIABLES.CASHIER_STORE.SALESPERSONNUM_USED.VAR_VALUE & VAR_VALUE2

- If salesperson number can be populated in the tlog for sales and returns, set to Y.
- If salesperson number is not in the tlog for sales & returns, set to N.

Cashier Size

This variable is indicated in the POS Questionnaire. Confirm them from the content of the customer's Employee Master

PRO_SP_VARIABLES.CASHIER_STORE.CASHIER_SIZE.VAR_VALUE & VAR_VALUE2

Determines the maximum size of the customers column which will be the source for CASHIERNUM, EMPNUM, SALESPERSONNUM – this value is used in the calculation of the CASHIERID, EMPLOYEEID and SALESPERSONID column values.

- The max size allowable is 20 for CASHIERNUM, EMPNUM, SALESPERSONNUM and will control the sizing of all three of the ID columns.
- The default is 10. If the length of the cashier number in tlog and Employee Master are shorter than or equal to the default, use the default.
- If the cashier number is longer than the default, change cashier_size to that length.

Configuring SKU and Customer Master Variables

SKU Stage Override

This variable determines if the SKU master update procedure will overwrite the values in the SKU master table with values from the staging table.

PRO_SP_VARIABLES.MASTERUPDATE.SKU_STAGE_OVERRIDE.VAR_VALUE & VAR_VALUE2

- If you want the sku master data to be overwritten by the pos_staging sku data then set this to Y.
- If you don't want the sku master data overwritten then set this to N.

Customer Stage Override

This variable determines if the customer master update procedure will overwrite the values in the customer master table with values from the staging table.

PRO_SP_VARIABLES.MASTERUPDATE.CUST_STAGE_OVERRIDE.VAR_VALUE & VAR_VALUE2

- If you want the customer master data to be overwritten by the pos_staging customer data then set this to 'Y.'
- If you do not want the customer master data overwritten then set this to 'N.'

Configuring General LP and SP Module variables

Processing Type

This variable determines the type of processing; either batch, where the ETL is run once per day, or real time, where the ETL is run every 15 minutes during the day and one end of day procedure at the end of the day.

PRO_SP_VARIABLES.XBRI.PROCESSING_TYPE.VAR_VALUE & VAR_VALUE2

- For batch processing set this to BATCH.
- For real time processing set this to REAL.

LP or SP Option

This variable determines whether an organization has purchased the both LP Module alone, or both LP and SP modules.

PRO_SP_VARIABLES.XBRI.LP_SP_OPTION.VAR_VALUE & VAR_VALUE2

- If the organization has purchased the Loss Prevention module only, set to LP.
- If the organization has purchased both modules, set to BOTH.

Capture Post Void Details in Tlog

This variable is indicated in the POS Questionnaire. Confirm them from the content of the customer's tlog.

If post void details are not captured on the post void transaction a database stored procedure will create the detail lines by looking at the original transaction.

PRO_SP_VARIABLES.PROACT.CAPTURE_PV_DETAILS.VAR_VALUE & VAR_VALUE2

- If Post Void transactions have the detail lines from the voided transaction, set to Y.
- If Post Void transactions do not have details, set to N.

Post Void Processing

This variable determines whether the database or ETL handles the post void processing.

This is for the transaction status and post void time difference columns at header level. If the database handles the processing it will mark the original transaction with transstat = 'POSTVOID' and calculate the time difference. In a real time processing environment this should be set to Y'

PRO_SP_VARIABLES.XBRI. POSTVOID_PROCESSING.VAR_VALUE & VAR_VALUE2

- If the database procedure handles the post void processing set to Y.
- If the ETL handles the post void processing, set to N.

Void Scheme

This variable is indicated in the POS questionnaire. Confirm from the content of the customer's tlog which void scheme they are using. This variable indicates if the tlog includes both the voided and voiding lines or just the voiding line.

Options are:

1. 1 - 0,1,2 scheme line voids include both the voiding and voided lines (includes only VOID_CODE = 0 items)
2. 2 - 0,2 scheme line voids include only the voiding line (includes voided and voiding items)

PRO_SP_VARIABLES.VOID_SCHEME.FN_PRO_VOID_SCHEME.VAR_VALUE & VAR_VALUE2

Followed by No Sale

This variable determines whether the database or ETL handles the Followed by No Sale processing. This is for the followed by no sale flag column at header level. If the database handles the processing it will

mark the transaction prior to a NOSALE with fbnosale_flag = Y. In a real time processing environment this should be set to Y.

PRO_SP_VARIABLES.XBRI. FB_NOSALE.VAR_VALUE & VAR_VALUE2

- If the database procedure handles the followed by no sale processing set to Y.
- If the ETL handles the followed by no sale processing, set to N.

Comp Backfill Days

This is the number of days to go back to populate the spo_comp_poll table.

PRO_SP_VARIABLES.SPO.COMP_BACKFILL_DAYS.VAR_VALUE & VAR_VALUE2

- Used in the SP Module only
- The default is 730 days

Polled Days

This is the number of days sp_spo_comp_poll procedure will look back for late polls.

PRO_SP_VARIABLES.SPO.POLLED_DAYS.VAR_VALUE & VAR_VALUE2

- Used in the SP Module only.
- The default is 14 days.

Sales Threshold

See "Configuring Sales Less than Threshold Variable, Setting and Managing Threshold Groups" for more information.

PRO_SP_VARIABLES.XBRSTATS.SALES_THRESHOLD

- The transaction total tender value for reporting 'sales less than threshold'. This value is can be obtained from the customer by the Oracle product manager.
- Core default is 5.00.

Process No Match Return Exchange

The No Match process is performed using the SP_PRO_NOMATCH_RETURNEXCH procedure, which is run from within the SP_PRO_LOAD_HIST procedure. This procedure looks for original purchase transactions related to refunds and exchanges. Based on the results of these lookups, Match Codes are assigned.

PRO_SP_VARIABLES.NOMATCH.PROCESS_NM_RETURNEXCH.VAR_VALUE and VAR_VALUE2

- If original transaction STORE, TRANSNUM, REGNUM & DATE for returns are in tlog, then set to Y, else N.
- If customer has more than one POS and one POS captures original transaction information for returns and the other POS does not, must be discussed with project manager. If we enable Return No Match, the system will report a lot of false positives for the POS that does not capture original transaction information.

Process No Match Post Void & Cancelled

The No Match process is performed using the SP_PRO_NOMATCH_PVCANCEL procedure, which is run from within the SP_PRO_LOAD_HIST procedure. These procedures look for subsequent re-ring transactions related to post voids and cancels. Based on the results of these lookups, Match Codes are assigned.

PRO_SP_VARIABLES.NOMATCH.PROCESS_NM_PVCANCEL.VAR_VALUE and VAR_VALUE2

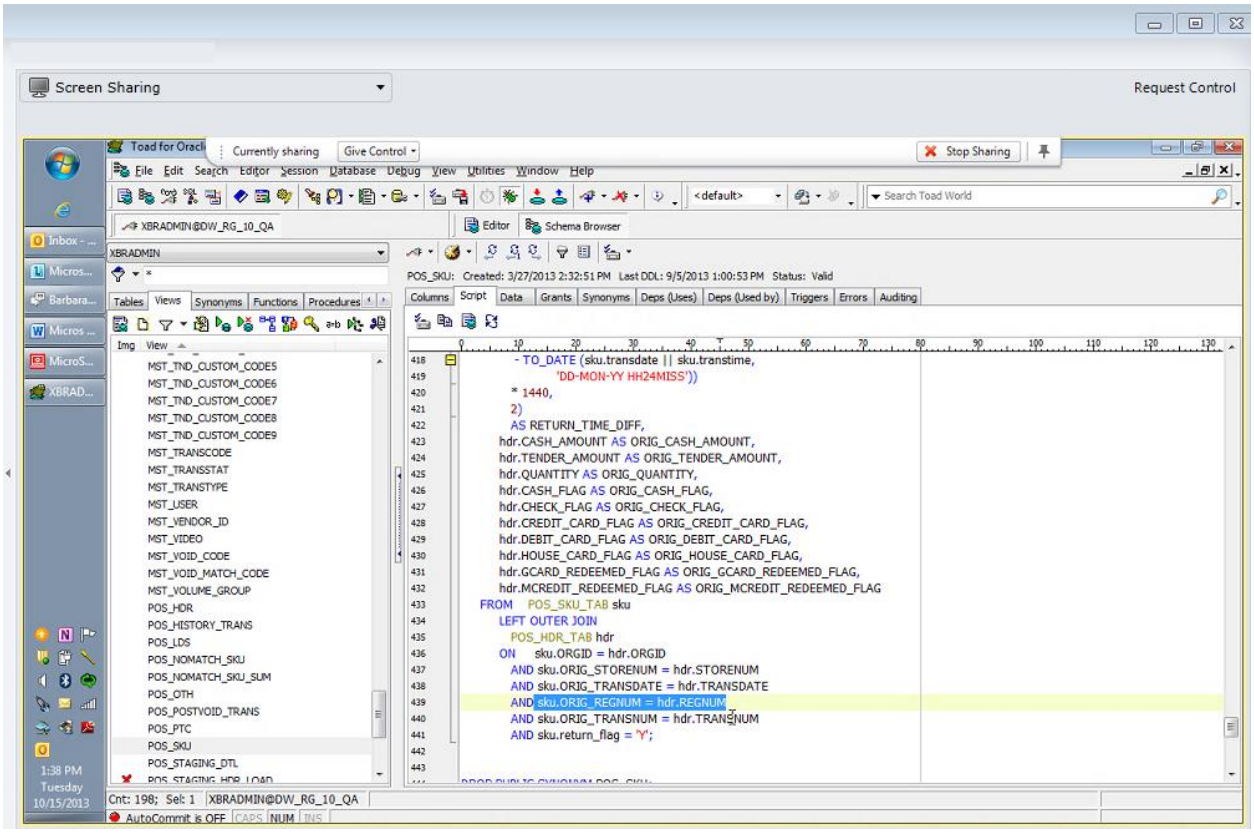
- If the tlog has Post Voids and/or Cancels, set to Y, else N.

Capture Original Regnum on Returns

This variable is indicated in the POS Questionnaire. Confirm them from the content of the customer's tlog.

PRO_SP_VARIABLES.NOMATCH.CAPTURE_ORIG_REGNUM.VAR_VALUE and VAR_VALUE2

- If the tlog captures the original register number for returns, set to Y, else N.
- If this is set to N, there is an additional step to update the POS_SKU view. The POS_SKU view is installed defaulting a join for the orig regnum. This must be removed from the view by creating a custom POS_SKU view. The steps for creating a custom view are in the View Extensibility section. Modify the view by removing the line in the JOIN "AND sku.ORIG_REGNUM = hdr.REGNUM", as highlighted below:



POS_SKU View

> **Note:** if this were a real time environment there would be two places to modify the view.

Post Void Minutes

The number of minutes to look forward to see if a SKU in a post-voided transaction was re-rung.

PRO_SP_VARIABLES.NOMATCH.PV_MINS

- Core default is 15.

Cancel Minutes

The number of minutes to look forward to see if a SKU in a cancelled transaction was re-rung.

PRO_SP_VARIABLES.NOMATCH.CANCEL_MINS

- Core default is 15.

Configuring Data Purge Process Variables

To enhance data minimization for personal data, a purge process is configurable for the deletion of inactive Customer, Employee and Store personal data. The application will delete data considered to be

personal data in the database, such as customer and ship to names, addresses, email addresses, etc. The purge routine is added to SP_ETL_XFINISH_BATCH and real time end of day SP_ETL_XFINISH_EOD procedures. Settings in pro_sp_variables enable this to be turned on when the active flag is set to Y and reaches the number of days defined. By default, the active flag is set to Y, and the default number of days setting is 370 days for each:

CUSTOMER_INACTIVE_DAYS - based on number of days since the transaction date that is associated with a customer number. Customer First / Last Name, Shipping Address, Email Address, Shipping First / Last Name, Phone Number, Zip code, State, Country will be deleted from transaction history.

EMPLOYEE_TERMINATED_DAYS - based on number of days since the termination date set in the employee master file. Employee first name, last name, federal ID, employee image, and salesperson image data will be deleted from the employee master file.

STORE_CLOSED_DAYS - based on number of days since the closed date in the store master file. Manager name and email address will be deleted from the store master file.

Table 6.1 Pro SP Variables Default Settings

Pro SP Variables: SYSTEM / VAR_NAME	Default Setting
CASHIER_STORE / STORE_UNIQUE	N
CASHIER_STORE / CASHIER_UNIQUE	N
CASHIER_STORE / EMPLOYEE_COPY	Y
CASHIER_STORE / SALESPERSON_COPY	Y
CASHIER_STORE / EMPNUM_USED	N
CASHIER_STORE / SALESPERSONNUM_USED	N
CASHIER_STORE / CASHIER_SIZE	10
MASTERUPDATE / SKU_STAGE_OVERRIDE	N
MASTERUPDATE / CUST_STAGE_OVERRIDE	N
XBRI / EOD_PROCESSING	N
XBRI / PROCESSING_TYPE	BATCH
XBRI / LP_SP_OPTION	LP
PROACT / CAPTURE_PV_DETAILS	Y
XBRI / POSTVOID_PROCESSING	Y
XBRI / FB_NOSALE	Y
SPO / COMP_BACKFILL_DAYS	730
SPO / POLLED DAYS	14
XBRSTATS / SALESTHRESHOLD	5
NOMATCH / PROCESS_NM_RETURNEXCH	N
NOMATCH / PROCESS_NM_PVCANCEL	N
NOMATCH / CAPTURE_ORIG_REGNUM	Y
NOMATCH / PV_MINS	15
NOMATCH / CANCEL_MINS	15
VOID_SCHEME / FN_PRO_VOID_SCHEME	1
CUSTOMER_INACTIVE_DAYS	370
EMPLOYEE_TERMINATED_DAYS	370
STORE_CLOSED_DAYS	370

Configuring Sales Less than Threshold Variable, Setting and Managing Threshold Groups

Sales Less Than Threshold is an edit criterion for detecting and reporting sale transactions with a non-zero tender amount below a defined threshold value.

The system default value for Sales Less Than Threshold is stored in PRO_SP_VARIABLES table with the following settings:

1. SYSTEM = 'XBRSTATS',
2. VAR_NAME = 'SALES_THRESHOLD'
3. threshold value in VAR_VALUE (initially set to 5.00).

This default value should be reviewed with the customer and, if necessary, changed early in the project.

About Threshold Basis and Threshold Groups:

In XBR*i*, sales threshold value can be set by country (threshold basis) and a multiple countries can be mapped to a single threshold value (threshold group). Threshold groups provide a single point for changing the threshold values of all the countries in the group. Countries with the same currency could logically be in the same threshold group. There is no reporting by threshold group. The threshold basis of country can be changed to another field in the Store Master, such as state or district. Multiple bases for threshold groups cannot be chosen; for example,: calculating Sales Less Than Threshold for both Region and District is not allowed. Only one (1) threshold group basis should be implemented.

Implementing the Country Threshold Basis:

XBR*i* offers the ability to have different thresholds for different countries. To set up threshold groups, you need to you need to add rows to the tables PRO_SP_THRESHOLD and PRO_SP_THRESHOLD_MAPPING, add the tables PRO_SP_THRESHOLD and PRO_SP_THRESHOLD_MAPPING to the XBR*i* Data Editor, be aware of the contents of the table MST_STORE_TAB and know the customer's countries and threshold value for each country.

These are the three tables involved in the threshold group implementation:

PRO_SP_THRESHOLD

ORGID	NUMBER (10),
THRESHOLD_ID	NUMBER (3), (maps to THRESHOLD_ID in PRO_SP_THRESHOLD_MAPPING)
THRESHOLD_NAME	VARCHAR2 (30 BYTE), (Name of Threshold Group)
THRESHOLD_VALUE	NUMBER (10,2) (threshold value for comparison to sale tender $)

PRO_SP_THRESHOLD_MAPPING

ORGID	NUMBER (10),
THRESHOLD_ID	NUMBER (3), (maps to THRESHOLD_ID in PRO_SP_THRESHOLD table)
SOURCE_VALUE	VARCHAR2 (10 BYTE) (Country code as in Store Master COUNTRY field)

MST_STORE_TAB

All other store master columns +	
COUNTRY	(maps to SOURCE_VALUE)
THRESHOLD_ID	NUMBER(3) (THRESHOLD_VALUE from PRO_SP_THRESHOLD)

This example shows you how to set up a threshold for COUNTRY. The threshold values will be added into the PRO_SP_THRESHOLD table with a threshold ID, a group name and a threshold value. The

countries will be added into the PRO_SP_THRESHOLD_MAPPING table with the country code and the corresponding threshold ID. The countries in the PRO_SP_THRESHOLD_MAPPING table must be the same country codes as in the COUNTRY field in the Store master.

In this example, we will define 2 threshold groups:

'USD', ID 100, threshold 4.00 and

'Euro', ID 200, threshold 3.76

and the countries that use those thresholds:

USA (United States) threshold 4.00

DEU (Germany) threshold 3.76

FRA (France) threshold 3.76

To set up a threshold group:

3. Edit the XBRI_Threshold.sql update template, and supply the values into the INSERT statement for PRO_SP_THRESHOLD to insert the two rows in the example.

   First row

   - For the value THRESHOLD ID enter 100, for THRESHOLD NAME enter 'USD' and for THRESHOLD VALUE, enter 4.00. Execute this script from an Oracle client tool supporting PL/SQL.

   Example:

```
`INSERT INTO XBRADMIN.PRO_SP_THRESHOLD (
 ORGID, THRESHOLD_ID, THRESHOLD_NAME,
`      THRESHOLD_VALUE)
VALUES ( 1, 100, 'USD', 4.00 );
COMMIT;
```

   Second row

   - For the value THRESHOLD ID enter 200, for THRESHOLD NAME enter 'EURO and for THRESHOLD VALUE, enter 3.76. Execute this script from an Oracle client tool supporting PL/SQL. See the previous example.

2. If there were additional threshold values to be entered, step 1 would be repeated as needed. When you are finished, now add rows to PRO_SP_THRESHOLD_MAPPING table.

   First row

   - For the value THRESHOLD ID enter 100, for SOURCE VALUE enter 'USA'. Execute this script from an Oracle client tool supporting PL/SQL.

   Example:

```
INSERT INTO XBRADMIN.PRO_SP_THRESHOLD_MAPPING (
ORGID, THRESHOLD_ID, SOURCE_VALUE)
VALUES ( 1, 100, USA );
COMMIT;
```

   Second row

   - For the value THRESHOLD ID enter 200, for SOURCE VALUE enter 'DEU'. Execute this script from an Oracle client tool supporting PL/SQL. See the previous example.

   Third row

   - For the value THRESHOLD ID enter 200, , for SOURCE VALUE enter 'FRA'. Execute this script from an Oracle client tool supporting PL/SQL. See the previous example.

3. If there were additional countries to be entered, Step 2 would be repeated as needed.
4. When the Store Master Update procedure (SP_MST_UPD_STORE) executes, the THRESHOLD_ID field in the Store Master will be updated by matching the COUNTRY field in the Store Master to the SOURCE_VALUE field in the PRO_SP_THRESHOLD_MAPPING table and posting the

THRESHOLD_ID from PRO_SP_THRESHOLD_MAPPING to the THRESHOLD_ID in the Store Master. Continuing the example:

   a. Store 1 in the USA would have a THRESHOLD_ID of 100.
   b. Store 2 in Germany would have a THRESHOLD_ID of 200.
   c. Store 3 in France would have a THRESHOLD_ID of 200.
   d. Stores in other countries would have a NULL THRESHOLD_ID.

5. When transactions are loaded to history, the store in the transaction record is used to look up the THRESHOLD_ID on the Store Master. The THRESHOLD_ID is used to look up the threshold value on the PRO_SP_THRESHOLD table for comparison to the tender amount of SALE transactions. If a store has a NULL THRESHOLD_ID, the system default from PRO_SP_VARIABLES is used for the comparison. The threshold values for the example stores would be:

   a. Store 1 – USA – THRESHOLD_ID 100 – value 4.00.
   b. Store 2 - Germany - THRESHOLD_ID 200 – value 3.76.
   c. Store 3 - France - THRESHOLD_ID of 200 – value 3.76.
   d. Stores - other countries - THRESHOLD_ID NULL – value 5.00 (system default).

Changing Threshold Basis:

If the Threshold Basis or a Threshold value is changed, the change is effective for future processing; history is not changed. If you wanted to set thresholds using a different basis, such as state instead of country, the variable in procedure SP_MST_UPD_STORE (code snippet below) would need to be changed from "@vs_source_map = 'COUNTRY'" to "@vs_source_map = ''STATE''", and you would follow the previous procedure to use the Data Editor to add all the distinct values for state into the threshold tables.

```
---- Update threshold IDs
--update MST_STORE_TAB t set threshold_id = (select threshold_id from
pro_sp_threshold_mapping where source_value=t.country);
set @vs_source_map = 'COUNTRY'
select @vs_sql = 'update MST_STORE_TAB set threshold_id = (select threshold_id from
pro_sp_threshold_mapping where source_value=MST_STORE_TAB.'+@vs_source_map+')'
```

PRO_REQUESTOR

The pro_requestor table is used by the pro_clear_stage procedure to determine if a particular batch can be cleared.

This table identifies critical processes. The pro_batch_control table has four columns for each record in this table. The pro_clear_stage procedure looks at the error code field in pro_batch_control to confirm all critical processes have successfully completed before clearing the batch.

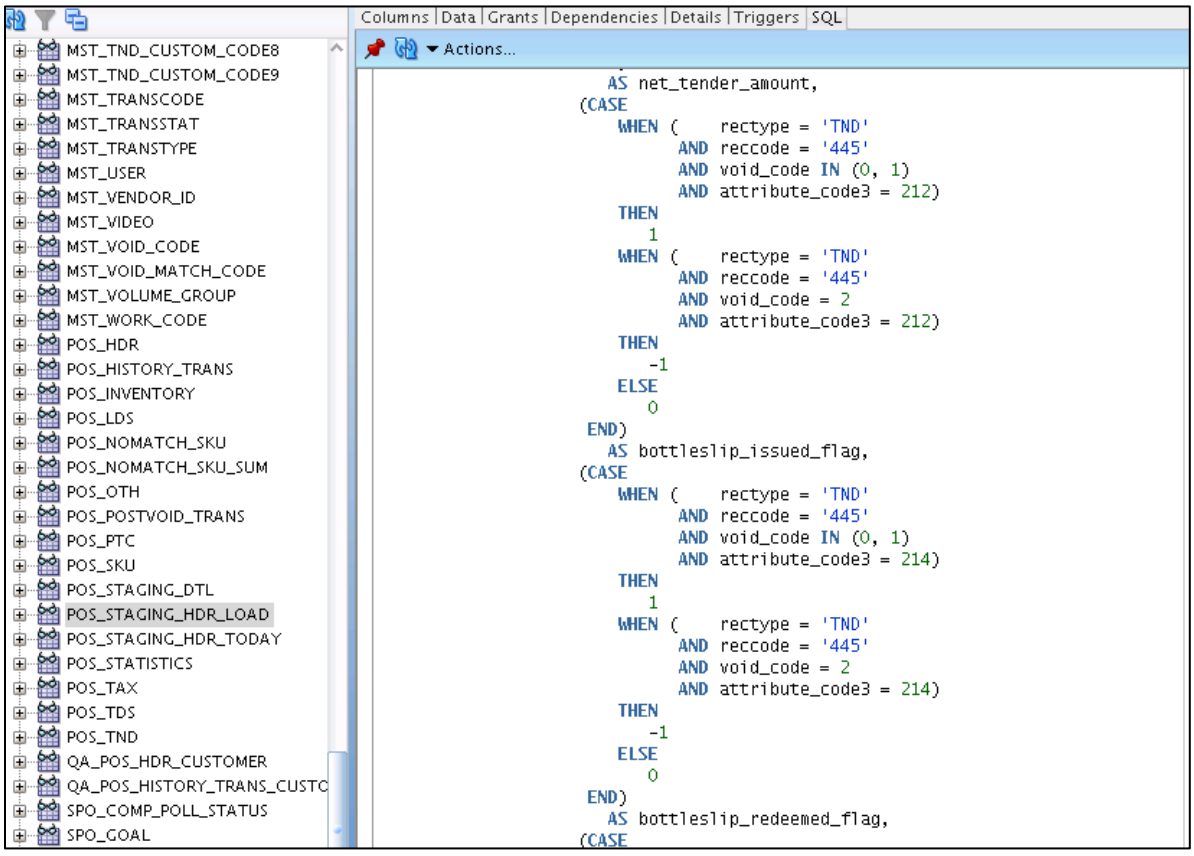
You must create a script to activate or deactivate the specific requestor. This table must be in agreement with the variable settings in sp_pro_variables.

DATA_REQUESTOR_NAME	REQUESTOR_ID	ACTIVE_FLAG
Delete records (SP_PRO_CLEAR_STAGE)	DEL	Y
Check For Duplicate Records (SP_PRO_DUP_CHK)	DUP	Y
Insert History Records (SP_PRO_LOAD_HIST)	HIST	Y
Insert Statistics Records (SP_PRO_LOAD_STATS)	STATS	Y
Insert EOD History Records (SP_PRO_LOAD_HIST)	EOD_HIST	Y
Insert EOD Statistics Records (SP_PRO_LOAD_STATS)	EOD_STATS	Y
Insert EOD Deleted Records (SP_PRO_CLEAR_STAGE)	EOD_DEL	Y
Insert SPO Statistics Records (SP_PRO_LOAD_SPO_STATS)	SPO	N
Insert EOD Statistics Records (SP_PRO_LOAD_STATS)	EOD_SPO	Y
Insert Hours worked records (SP_SPO_HRS_WORKED)	SPO_HRS_WORKED	N
Insert EOD Hours worked records (SP_SPO_HRS_WORKED)	EOD_SPO_HRS_WORKED	Y

PRO_REQUESTOR Table

POS_STAGING_HDR_LOAD

Create a script to modify this view if required. This view is responsible for setting various flags and occasionally needs to be adjusted to accommodate a customer's business requirements. The following screenshot shows the bottleslip flag being set. If a customer had a second type of bottleslip, you would add the custom tender type for that custom bottleslip.



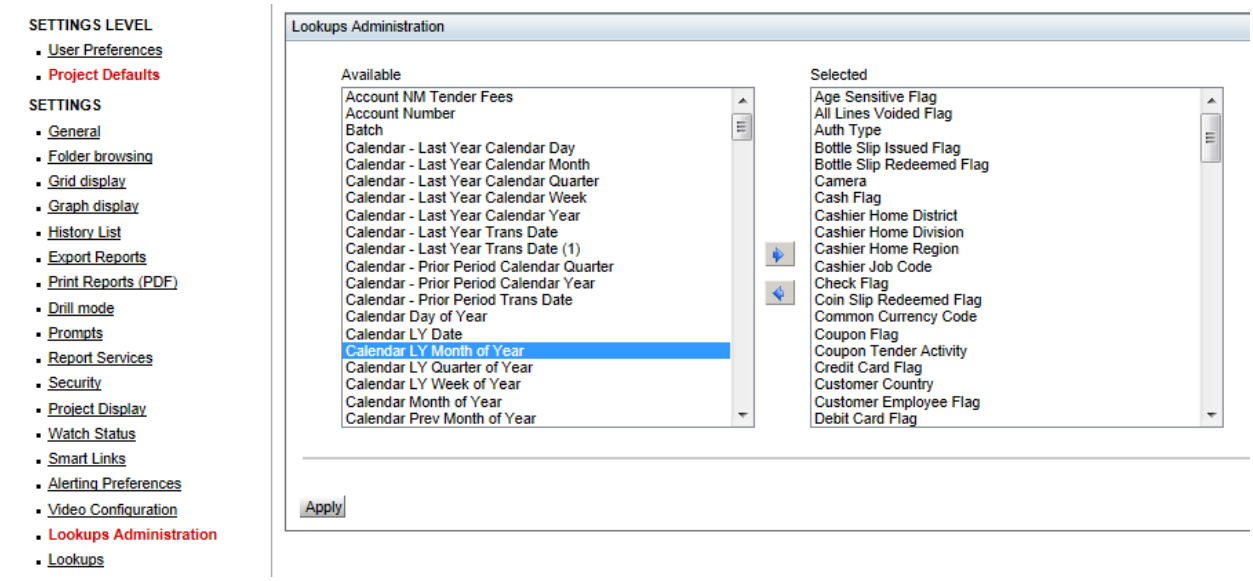
Script to set bottleslip

Enabling Attributes for Lookup Administration

Lookups are values (IDs and descriptions) for selected attributes. These values are stored in the lookup table for each attribute, and are used in reporting and for element browsing. The ID represents the stored value, that is, - Tender Type Code or District Number, and the description represents the textual description, i.e. - Tender Type Description or District Name. The ID, description, or both may be displayed on reporting. XBR*i* customer administrators have access to the Admin > Project Defaults > Lookups page, where they can create and modify lookup values for attributes. Core XBR*i* administrators also have access to a Lookups Administration page where they can add attributes to the list of those available on the Lookups page.

To enable attributes for lookups:

1. Log in to XBR*i* as an XBR*I* administrator.
2. From the Admin menu, choose **Project Defaults**.
3. Under Settings, click **Lookups Administration**. This displays the Lookups Administration page.



Project Defaults – Lookups Administration

4. In the Available list, select an attribute that you want to add, and click the **Add** icon. This moves it to the Selected list. Repeat this step for each attribute you want to add.
5. Click **Apply**.

> **Note:** You can remove an attribute by selecting it in the Selected list and clicking the **Remove** icon.

The attributes on the Selected list are available for XBR*I* customer administrators on the Lookups page.

Configuring Max Threads and Exceptions for Controls

Control Points are reports that track information on activity performed by a store, cashier, and so on, based on a defined threshold. An example of this would be a cashier who repeatedly exceeds the threshold amount for line discounts. When you run a control point report, it will create results if the values exceed the threshold value defined in the report. Those results are called Exceptions. Core XBR*i* administrators can set the number of control point reports that can run simultaneously, and the number of exceptions that can be generated by a control point report before a prompt is displayed asking if the user wants to see more exceptions.

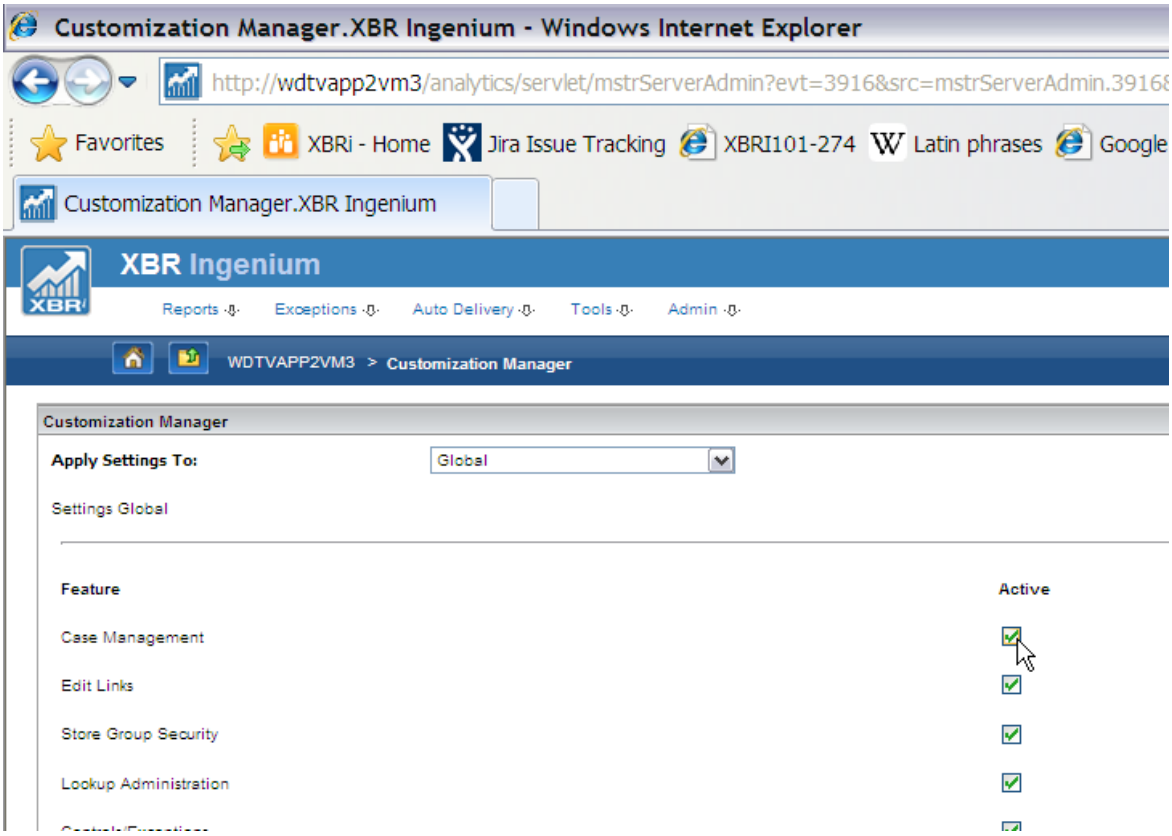
1. On the database server, use your frontend database viewer to navigate to the ADMIN_LP_VARIABLES table. You can set the parameters for the threads and exceptions allowed for controls, by organization.
2. Display the table rows in editable mode.
3. In the ORGID column locate the ID of the customer organization.
4. For the VAR_NAME, CONTROL_MAX_THREADS enter the VAR_VALUE. This is the number of control point reports that can be run simultaneously.
5. For the VAR_NAME, CONTROL_MAX_EXCEPTIONS, enter the VAR_VALUE. This is the number of exceptions that can be generated for a control point before an error message is displayed.

Enabling Case Management Integration for a Customer

After running the Upgrade program the final time for Case Management configuration, you can enable case management capability for an organization.

To enable case management capability in XBRi for an Organization:

1. Log in to XBRi as an XBRi Administrator user (that is, CoreXBRAdmin).
2. From the **Admin** menu, choose **Customization Manager**.



Customization Manager

3. Ensure that **Global** is selected in the **Apply Settings** dropdown.
4. Place a check in the **Case Management** check box and click **Apply**.
5. From the **Apply Settings to** drop-down list, ensure that the customer's project is selected.
6. Place a check in the **Case Management** check box and click **Apply**.
7. Click **OK**.

Report Linking – Attribute Form Mapping

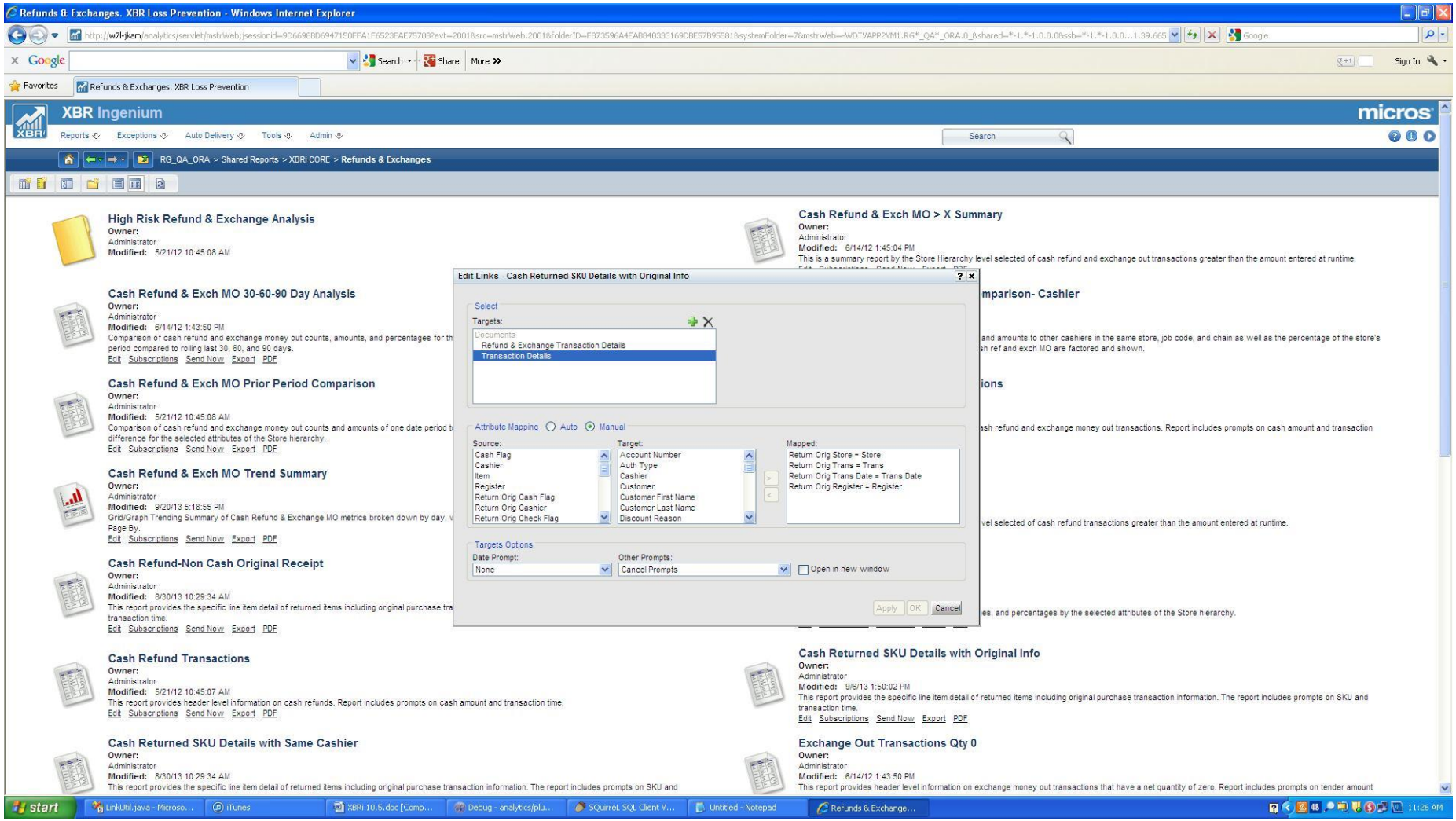
The report linking feature in XBR^i lets users create links to relevant target reports or documents from source reports or documents. This lets them execute the related report or document by dynamically passing the selected attribute elements from the source, instead of having to run another report or document and re-enter the information. Most of the attribute linking can be done within XBR^i by customers, but there some configuration steps that can only be done by Oracle administrators.

Linking Attributes with Different Attribute Form Structures

There are cases in which a source attribute has different key attribute form structures than the target attribute. For example, the Cash Returned SKU Details with Original Info report lists items returned for cash. For each returned item there is an associated Return Orig Trans attribute with a single key attribute form: #. The attribute form # provides the Transaction ID of the transaction in which the item was sold. For each returned item, users may want to link to the Transaction Details document to view detailed information concerning the original sale.

However, the Trans attribute in Transaction Details has six key attribute forms (Version, #, Register, Trans Time, Trans date, Store ID).

The ADM_LP_XLINKS_ATTRFORM_MAPPING table provides a mechanism with which to specify how the value of Return Orig Trans's key attribute form # is mapped to a key attribute form in the Trans attribute.



XBR*i* Link Editor

ADM_LP_XLINKS_ATTRFORM_MAPPING can also be used to support customers if their applications require linking from source attributes to target attributes that have different key attribute form structures.

Link Maintenance Interval

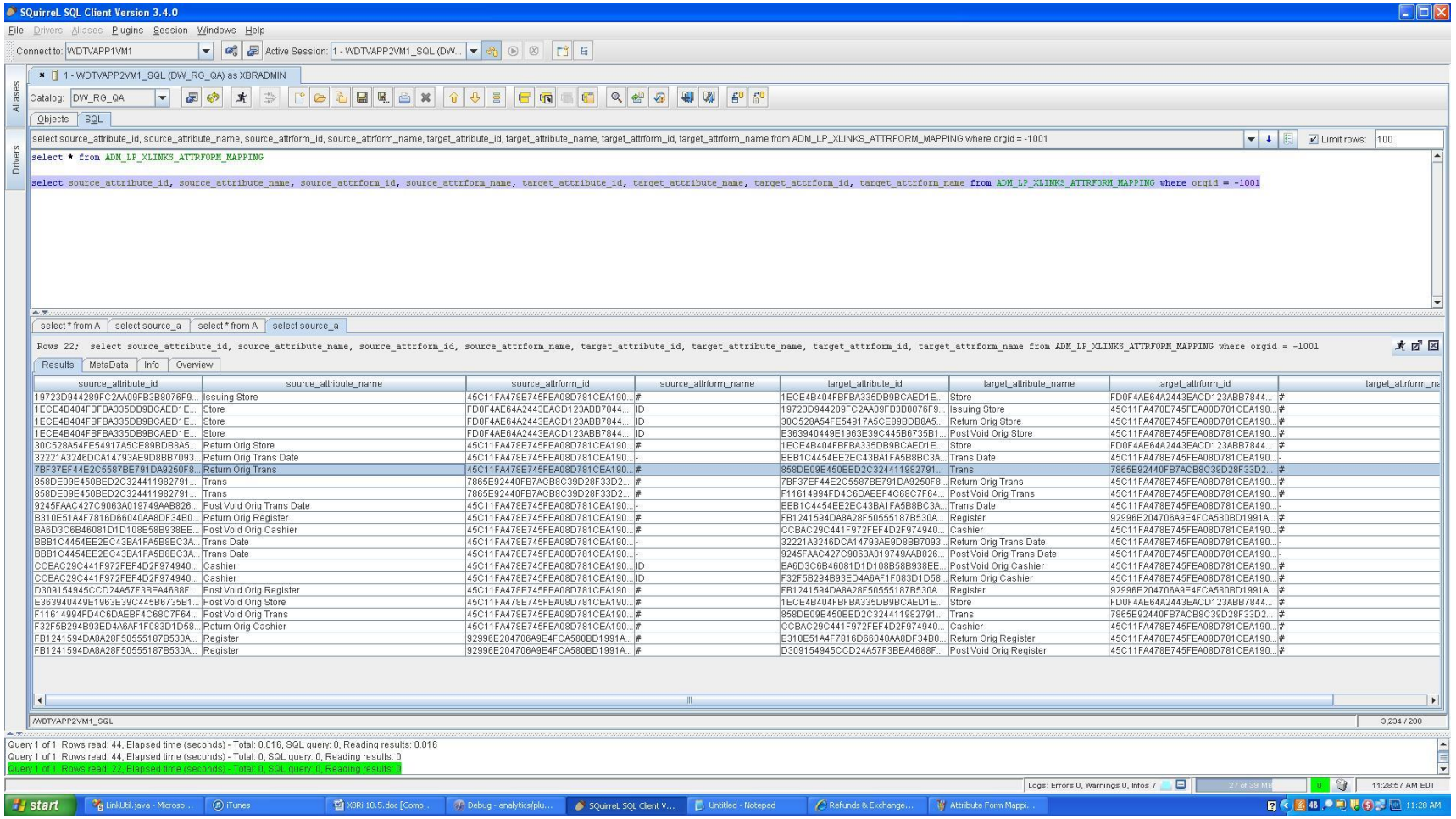
XBR currently supports preserving some, but not all report attribute link definitions. The scenario that follows is an example of link definitions that are supported.

Example: Supported Link Preservation

Suppose there are two folders, A1 and B1, under the XBR^i Core folder. Also suppose that under folder A1 there is a subfolder A2 that has two reports, Report1 and Report2, with a link definition defined from Report1 to Report2. If users make a copy of folder A2 (together with its content) into folder B1, XBR^i can preserve the link definition from the new copy of Report1 to the new copy of Report2 independently of whether the folder is copied in XBR^i or elsewhere.

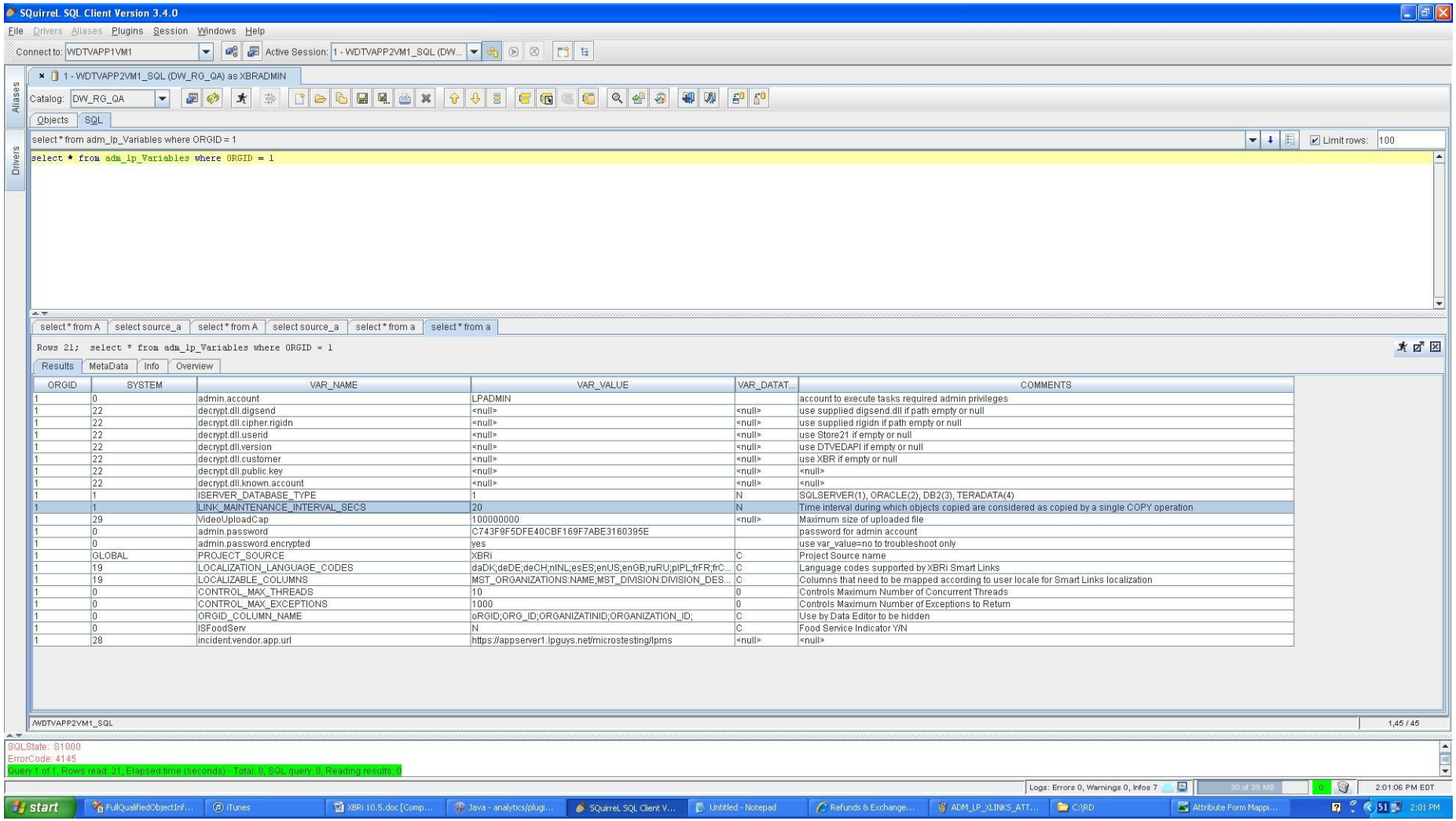
XBR^i currently supports link preservation as described in the scenario above. However, there are many ways users may copy folders in XBR^i. To achieve Link Preservation, XBR^i implements a Link Maintenance Interval, for example, every 20 seconds. Reports or documents copied within the Link Maintenance Interval are considered copies made by a single folder copy operation.

The default interval value of 20 seconds is chosen because it is long enough so that a report folder copying operation will likely complete in that time interval, but short enough so that it is very unlikely that two report folder copying operations would take place within that time interval.



ADM_LP_XLINKS_ATTRFORM_MAPPING table

You can adjust value of Link Maintenance Interval by modifying the LINK_MAINTENANCE_INTERVAL_SECS row in the adm_lp_Variables table:



adm_lp_Variables table

Video Linking Configuration

The video linking feature in XBR*i* allows users to retrieve the digital video that corresponds to one or more transactions. Video vendors provide the video services to customers that are used to record transactions at the point of sale and to display selected video at a later time.

Code for supported video vendors is included in the SP_PRO_VIDEO procedure that is invoked when user clicks on a video link in the video queue. The video queue is in the document that is displayed when a user clicks on the Video Link command in the Tools menu or toolbar in a report in which one or more rows with video-enabled attributes are selected. If the customer is using a non-supported vendor, the code for that vendor can be added to the SP_PRO_VIDEO procedure.

There are several other steps for configuring video linking, including associating cameras with registers, specifying attributes for video linking, and configuring the ID of the document that is displayed when the user clicks on the Video Link command.

Before you begin, get the following information from either the customer or the video vendor:

- Code for the SP_PRO_VIDEO procedure, if the vendor is not supported by XBR*i*
- Values for the MST_REGISTER_TAB table: DEVICE_STRING, VIDEO_FLAG, VIDEO_VENDOR and SITECODE (optional if web-based video)

Step 1: Adding a New Vendor to SP_PRO_VIDEO

> **Note:** Supported video vendors are already included in the SP_PRO_VIDEO procedure. You do not need to complete this step if the vendor appears on the supported vendors list, but this list is subject to change. Check with the account manager or database administrator to see if a vendor has been added.

XBR*i* supported video vendors

For a list of the XBR*i* supported video vendors, contact the XBR*i* product manager.

> **Note:** Video script is launched when the user clicks the video link in XBR*i*

To add a video vendor:

1. Get the SP_PRO_VIDEO procedure from your database Procedures folder and the code specific to the vendor. In this example the vendor is CLINTON.
2. Add an IF statement to existing code for the new vendor. **Note:** IF statements begin around line 200 to 250.

   For example, if the vendor is Clinton

   ```
   ELSIF UPPER (vs_video_vendor)='CLINTON'THEN
   goto CLINTON
   ```
3. Add the CLINTON code block just before the <<GETOUT>>: label. You can search for this label and create the code block just prior to this label. The code block must begin with the new label, which is defined by the GOTO target of the IF statement previously entered. The label is the GOTO name ended by a colon : The end of the block must be the statement "GOTO GETOUT". Those are the only standard requirements for this procedure, the code between the label CLINTON: and "GOTO GETOUT" is supplied by the video vendor

Example: SP_PRO_VIDEO script

```
<<CLINTON>>
-- d:\video\video.exe -s192.168.10.112 -c1 -d10042001 -b17:47:00 -e17:47:30 -uUSERID -pPASSWORD
-- Video parameters:
-- -s - store from mst_register_tab where storenum=pn_storenum
```

```
-- -c - camera number from MST_REGISTER_TAB where
-- storenum=pn_storenum and regnum=pn_regnum
-- -d - date (mmddccyy) passed pdt_transdate
-- -b - beginning time (hh:mm:ss) passed ps_start_time
-- -e - ending time (hh:mm:ss) passed ps_end_time
-- Uses a dash before the parameter letter
-- Note the space between parameters in the command line
-- store and camera already retrieved in beginning
-- add storenum
SELECT video_vendor, SITECODE, DEVICE_STRING
INTO vs_video_vendor, vs_sitecode, vs_camera
FROM MST_REGISTER_TAB
WHERE storenum = vn_storenum
AND regnum = pn_regnum;
vs_string := vs_string || '-s' || vs_sitecode || ' ';
-- add regnum
vs_string := vs_string || '-c' || vs_camera || ' ';
-- transdate needs to be mmddyyyy
IF pdt_transdate IS NOT NULL THEN
vs_string := vs_string || '-d' || TO_CHAR(pdt_transdate, 'mmddyyyy') || ' ';
END IF;
-- start time
IF ps_start_time IS NOT NULL THEN
vs_string := vs_string || '-b' || ps_start_time || ' ';
END IF;
-- end_time
IF ps_end_time IS NOT NULL THEN
vs_string := vs_string || '-e' || ps_end_time || ' ';
END IF;
-- User Name
temp_var := 'user';
BEGIN
SELECT var_value INTO temp_var
FROM PRO_SP_VARIABLES
WHERE UPPER(SYSTEM) = UPPER(vs_video_vendor)
AND UPPER(var_name) = 'USERNAME';

EXCEPTION
WHEN NO_DATA_FOUND THEN
temp_var := 'user';
WHEN OTHERS THEN
p_rtnstr1 := SUBSTR(SQLERRM,1,200);
RETURN;
END;
vs_string := vs_string || '-u ' || temp_var || ' ';

-- Password
BEGIN
SELECT var_value INTO temp_var
FROM PRO_SP_VARIABLES
WHERE UPPER(SYSTEM) = UPPER(vs_video_vendor)
AND UPPER(var_name) = 'PASSWORD' ;

EXCEPTION
WHEN NO_DATA_FOUND THEN
temp_var := 'user';
WHEN OTHERS THEN
p_rtnstr1 := SUBSTR(SQLERRM,1,200);
RETURN;
END;
vs_string := vs_string || '-p ' || temp_var || ' ';

GOTO GETOUT;
```

Step 2: Setting Variables in PRO_SP_VARIABLES

Several video vendors require specific information, such as usernames and passwords, in order to use their services. This general information is held within the PRO_SP_VARIABLES table in the XBR Database. These variables must be set before video vendor services can be used.

In addition to the vendor-specific variables, there are a few general variables that may need to be configured. They are:

PROACT	**REGRECEIPT**
PROACT VIDEO	VIDEO_VENDOR
PROACT VIDEO	SECONDS PRIOR
PROACT VIDEO	SECONDS AFTER
PROACT VIDEO	LENGTH

The variables that need to be set depend on your video vendor. The following tables detail the information each video vendor requires. Modify these rows with your information. If no table exists for your supported video vendor, no configuration of the PRO_SP_VARIABLES table is necessary.

Table 6.2 – Video Vendor Information Requirements

Vendor	Versions Installed/Comments	Video Application - Web or Desktop	Installed on - XBR or XBRi	Vendor in Core Stored Procedure for Video
3VR	www.3vr.com	Desktop	XBR, XBRi	Yes
Advanced Technology (AT) Video	www.atvideo.com		XBRi	Yes
Arrowsight	www.arrowsight.com	Web	XBR, XBRi	Yes
Click-It	www.clickitinc.com	Desktop	XBR, XBRi	Yes
Clinton	www.clintonelectronics.com	Desktop	XBR, XBRi	Yes
Costar	www.costarvideo.com		XBRi	
Dedicated Micros	www.dedicatedmicros.com		XBRi	Yes
DTT	www.dttusa.com	Desktop	XBR, XBRi	
Envysion	envysion.com	Web	XBR, XBRi	
ExacqVision	exacq.com	Desktop	XBR, XBRi	Yes
Genetec	www.genetec.com	Desktop	XBR, XBRi	
Geovision	www.geovision.com.tw	Desktop	XBR, XBRi	
I3 International (DVR)	www.i3international.com	Desktop	XBR, XBRi	Yes
Image Vault	www.imagevault.se		XBRi	Yes
Intellex (Tyco/American Dynamics)	www.americandynamics.net	Desktop	XBR, XBRi	
March Networks	www.marchnetworks.com	Desktop	XBR, XBRi	
Mirasys	www.mirasys.com		XBRi	Yes
Nice Video	www.nice.com		XBRi	Yes
RetailNext	retailnext.net	Web	XBR, XBRi	
Sensormatic	www.sensormatic.com	Desktop	XBR, XBRi	Yes

Short Circuit	shortcircuitinc.com	Desktop	XBR, XBRi

Step 3: Configuring Cameras and Registers

The MST_REGISTER_TAB table stores information that maps cameras to registers. You need to get the following information from the customer or video vendor to enter into this table: Values for the MST_REGISTER_TAB table:

DEVICE_STRING – the camera number

ACTIVE_FL – Indicates whether the video for the register in enabled, yes (Y) or no (N)

VIDEO_VENDOR – the video vendor directs the application to call the appropriate video viewer for this store/register combination.

SITECODE (optional if web-based video) – The IP address for the store's video system

To configure cameras and registers:

1. Open the XBRi_Video_Update.sql script.
2. Modify the UPDATE MST_REGISTER_TAB statement to populate the: columns DEVICE_STRING, VIDEO_FLAG, VIDEO_VENDOR and SITECODE, enter the values provided by the customer or video vendor.

   Example:

```
UPDATE XBRADMIN.MST_REGISTER_TAB
SET    DEVICE_STRING     = ',
       VIDEO_FLAG        = 'y',
       VIDEO_VENDOR      = 'CLINTON',
       SITECODE          = 'IP Address'
WHERE  DIVISION           = 1
AND    STORENUM           = 123
AND    REGNUM            = 20;
COMMIT;
```

3. Execute the script from Oracle client supporting PL/SQL.

Step 4: Configuring the Attributes for Video Linking

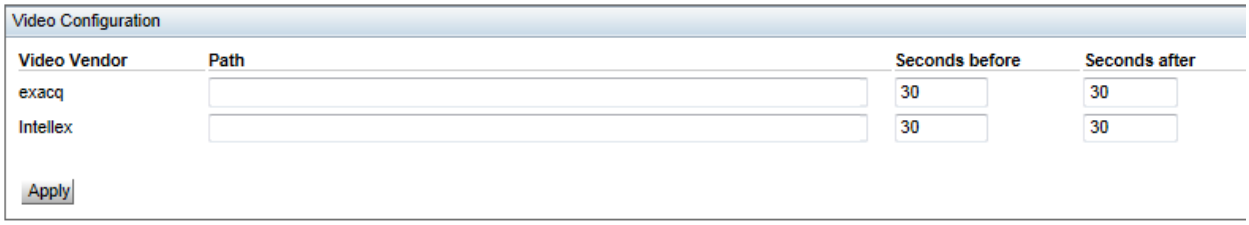
There are 6 core attributes that are defined in the ADM_LP_VIDEO_ MAPPING table such as Trans, Trans Time, Division, Trans Date, Store, and Register. These 6 attributes need to be defined in a report in order to enable video linking. When you select a report with video enabled, the Video link in the report Tools menu or toolbar is enabled

Step 5: Entering the Path to the Video Vendor in Project Defaults

The Video Configuration Defaults page lets you modify the local paths to the executable files of the video viewers used when executing video links. You can also specify the number of seconds before and after the transaction to be included in the video.

To set video linking project defaults:

1. From the Admin menu, choose Project Defaults.
2. Under Settings, click **Video Configuration**. This displays the Video Configuration page:



3. Edit the Path for the Video Vendor if necessary. This sets the local path to the video viewer used when executing a video link.
4. Change the number of Seconds Before or Seconds After defaults if necessary. This determines the default number of seconds before and after the transaction time when generating the start and end times in the Video Queue.
5. Click **Apply**. This applies the information and saves it to the ADM_LP_VIDEOCONFIG table.

9

Customer Configuration

After you install XBR[i] and complete the post-installation manual steps, you may need to configure a customer-specific deployment of the XBR[i] core project. These are the procedures that should be followed to properly modify or extend a customer environment, although not all procedures may be applied in each case.

> **WARNING:** Customer Supplemental tables should only be modified by BI architects since modifying core schema objects can have a significant impact on reporting as well as the upgrade path if not done properly.

Custom POS_STATISTICS Columns

There are 20 custom fact columns in the POS_STATISTICS table, 10 count and 10 amount. Customers can use these columns to build new fact expressions.

Using Custom Facts – POS_STATISTICS Columns

1. Open the database warehouse using TOAD or SSMS
2. Insert a record into the PRO_VIEW_SYNTAX table for the custom column you plan on using. Set the ACTIVE_FLAG = Y and the CUSTOM_FLAG = 1.
3. Test the SP_PRO_LOAD_STATS procedure to make sure the new custom column is populating with the correct data

The column is available in the application After you run the command script to unhide it. For instructions, see the following section: Using Metrics – POS_STATISTICS Columns.

In order to use the new custom fact, you must create a new metric for it. As described in the following procedure:

Using Metrics - POS_STATISTICS Columns

Unhide the custom facts columns using the following command manager script:

> **Note:** For all scripts referenced in the command line, the entire command must be in the script.

- To review how to run the scripts, refer to the section, Running Command Manager Scripts.
- Use the following Example scripts as templates.
- The text in red in the example scripts should be modified to meet customer requirements.

Template:

ALTER FACT "<fact_name>" IN FOLDER "<location_path>" HIDDEN (TRUE | FALSE) FOR PROJECT "<project_name>";

Example:

ALTER FACT "STAT_CUSTOM_AMOUNT1_FACT" IN FOLDER "\SCHEMA Objects\FACTS\Loss Prevention\STATISTICS\CUSTOM" HIDDEN FALSE FOR PROJECT "XBRI";

Customizing Core Statistics Buckets

You can add or update core statistics buckets in the PRO_VIEW_SYNTAX table. You use the CUSTOM_FLAG column in conjunction with the ACTIVE_FLAG column to customize statistics

To customize a core statistics bucket:

1. Open the database warehouse using TOAD or SSMS.
2. Add a new row for the statistics bucket that you are modifying and set the CUSTOM_FLAG = 1 and the ACTIVE_FLAG = Y.
3. Locate the row of the core statistics bucket that you are modifying and set the ACTIVE = N
4. Add the formula for the custom stat bucket.

Using Custom Core Header Attributes and Facts

To use a custom core attribute, the object should be translated for the Custom English language to the customer-specific context. It should not be renamed. Once you unhide the attribute and update the schema, it will be available for use.

This is the Command Manager script for un-hiding the attribute:

```
ALTER ATTRIBUTE "Custom SKU Char1" IN FOLDER "\Schema Objects\Attributes\Loss
Prevention\Detail\SKU\SKU Master" HIDDEN FALSE FOR PROJECT "XBRI";
```

There are two options for renaming the custom core attribute:

- Keep the custom core attribute name as is and change the column name when it is used in a report.
- Request that Oracle make the change to the custom core attribute. They will then provide the customer with a package.

Using Custom Detail Attributes and Facts

You can configure the custom core detail attributes and facts provided and if necessary, create a set of custom detail attributes and facts required by the customer. Use the custom core detail attribute or fact if it applies to ONLY one POS_* table, otherwise you would create a new attribute/fact in the custom detail folder. If you have an attribute that was put into a numeric custom field you would also need to create a new attribute.

The customer must make a request to Oracle to create a new attribute.

The following custom core detail attributes and facts are already provided:

(LDS, OTH, SKU,TAX,TDS, TND or PTC) CUSTOM CHAR *n(1-10)* – Use these for any attribute that does not require a lookup.

(LDS, OTH, SKU,TAX,TDS, TND or PTC) CUSTOM CODE *n(1-10)* – Use these for any attribute that has a lookup. (used in new installs only)

(LDS, OTH, SKU,TAX,TDS, TND or PTC) CUSTOM FLAG *n(1-10)* – Use these for any flag attributes with a N or Y value. (used in new installs only)

(LDS, OTH, SKU,TAX,TDS, TND or PTC) CUSTOM DATE *n(1-3)* – Use these for any custom date attribute.

(LDS, OTH, SKU,TAX,TDS, TND or PTC) CUSTOM NUM n(1-10) FACT - Use these only if the fact applies to only one POS_* table

Modifying Views and Stored Procedures Using Extensibility

Overview

Often during an installation, a stored procedure or database view has to be modified to meet specific customer requirements. This can lead to problems during an upgrade trying to preserve the custom procedures or views.

Certain views and stored procedures have been designated as core and, when extended, are saved as custom code and are marked as active. The corresponding core code is marked as inactive.

> **Note:** Before making any changes to core views or stored procedures, make sure you back up your database.

Stored Procedure Extensibility

The table, PRO_PROCEDURES, is populated with core procedure names as part of the installation process.

The Enablement team is responsible for comparing all custom procedures to the core procedures and based on their analysis, decide to activate either the core or custom code. If they decide to activate a core procedure, they will delete the custom procedure from the database and set the core procedure to active by setting CUSTOM_NAME field to Null in the PRO_PROCEDURES table.

Modifying a Core Stored Procedure

Use the following steps to modify an existing stored procedure:

1. Create your custom stored procedure with the naming convention of the core procedure plus "_E". For example: If you are customizing SP_MST_UPD_STORE then name your new procedure SP_MST_UPD_STORE_E
2. Update the custom_name value in the PRO_PROCEDURES table. For Example: Update PRO_PROCEDURES set custom_name = 'SP_MST_UPD_STORE_E' where core_name = 'SP_MST_UPD_STORE'.

   > **Note**: All the core procedures that are called by the ETL have a section of code in the beginning that looks for the existence of a custom procedure name in the pro_procedures table. If it exists, then the core procedure calls the custom procedure instead.

3. When a customer upgrades to a newer version of XBR[i], Oracle only updates the core procedure, never any custom procedures with the naming convention ending in "_E". It is the responsibility of the enablement team to compare the core procedure to the custom procedure and modify the custom procedure as necessary or revert to the core procedure.

Deleting a Custom Stored Procedure

1. To delete a custom procedure, simply remove the custom_name value from the pro_procedures table. Example: Update pro_procedures set custom_name = null where core_name = 'SP_MST_UPD_STORE'.
2. Drop the custom procedure from the database.

View Extensibility

The table, PRO_VIEWS, is populated with core views that are assigned "-1001" as the orgid. Run the stored procedure sp_create_ext_views to create the active views. The team is responsible for evaluating the difference between the activated custom views and their corresponding core views and determining whether to keep the custom or core view. They should remove any custom views that are inactive.

Adding New or Custom Views

Use the following steps to customize an existing Core view or create a new view:

> **Note:** When entering a new view name, make sure the name is in all upper case.

1. Create your new view with the same name as the core view.
2. Add a new row to the PRO_VIEWS table with the custom syntax. This is done in two steps, first add the new row in the PRO_VIEWS table and then update it using dbms_metadata.get_ddl to populate the view syntax.

   Example 1: Add the new row

   ```
   INSERT INTO PRO_VIEWS (ORGID, VIEW_NAME, CUSTOM_FLAG, ACTIVE_FLAG, PROCESSING_TYPE_FLAG)
   VALUES (<Company ORGID>, 'MST_STORE', 'Y', 'Y','BR');
   ```

   > **Note:** CUSTOM_FLAG should be Y, ACTIVE_FLAG should be Y and the PROCESSING_TYPE_FLAG can be B for batch, R for real time or BR if the view is to be used for both batch and real time processing. Do not update any row with orgid = -1001 in pro_views. This is the core view; when Oracle upgrades the core view may be modified.

   Example 2: Update the syntax using dbms_metadata.get_ddl

   ```
   UPDATE PRO_VIEWS v SET SYNTAX = dbms_metadata.get_ddl('VIEW',v.view_name) WHERE ORGID = '<
   Company ORGID >' and v.VIEW_NAME = 'MST_STORE' and CUSTOM_FLAG = 'Y' and PROCESSING_TYPE_FLAG
   = 'BR';
   ```

3. Turn off the core view by setting the ACTIVE_FLAG to N.

   Example

   ```
   UPDATE PRO_VIEWS SET ACTIVE_FLAG = 'N' WHERE ORGID = -1001 and VIEW_NAME ='MST_STORE' and
   CUSTOM_FLAG = 'N';
   ```

4. Run the create views procedure and verify the custom view is in place.

   Example

   ```
   EXEC SP_CREATE_EXT_VIEWS ('MST_STORE');
   ```

When an upgrade is done, you only update the core rows where orgid = -1001. It is the Enablement team's responsibility to compare the core view to the custom view and modify the custom view as necessary or revert to the core view.

Deleting Custom Views

1. To delete a custom view and revert to the core view, update the ACTIVE_FLAG to be Y on the core row then set the ACTIVE_FLAG to N on the custom row.

   Example

   ```
   UPDATE PRO_VIEWS SET ACTIVE_FLAG = 'Y' where ORGID = -1001 and VIEW_NAME = 'MST_STORE'
   UPDATE PRO_VIEWS SET ACTIVE_FLAG = 'N' where ORGID = <Company ORGID> and VIEW_NAME =
   'MST_STORE'
   ```

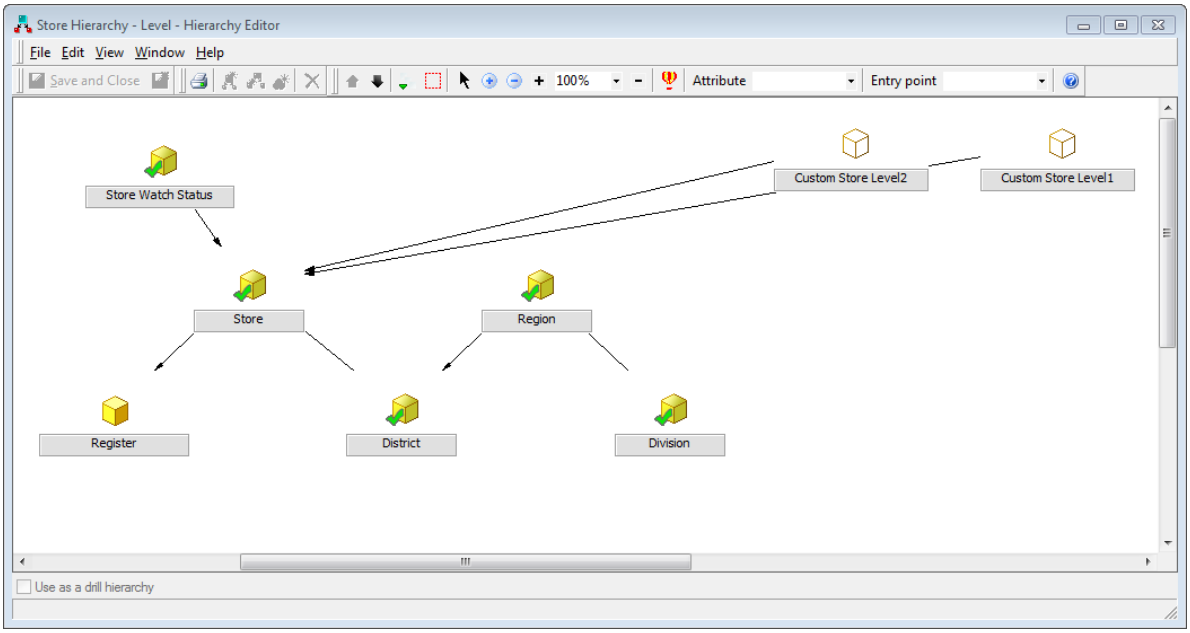
2. Run the create views procedure to verify the core view has reverted.

   Example

   ```
   EXEC SP_CREATE_EXT_VIEWS ('MST_STORE');
   ```

Updating Drill Hierarchies

This section describes how to update drill hierarchies.



Store Hierarchy diagram

Use the command prompt and command manager scripts to update drill hierarchies for the customer.

> **Note**: This procedure must be performed from a Windows-based command manager interface. If the implementation environment already has a Windows VM MicroStrategy installed, the script should be run from there, otherwise you must contact Oracle Support to run the script

To update drill hierarchies using command manager scripts:

> **Note:** For all scripts referenced in the command line, the entire command must be in the script.

- See: Running Command Manager Scripts in this guide to review how to run the scripts.
- Use the following Example scripts as templates.
- The text in red (lighter than surrounding text) in the example scripts should be modified to meet customer requirements.

Alter Hierarchy Template:

ALTER HIERARCHY "Hierarchy Name" NAME "Altered Hierarchy Name " IN FOLDER " Folder Path " DRILLHIERARCHY TRUE DESCRIPTION "XXXX" HIDDEN TRUE LOCATION " Folder Path " FOR PROJECT "Project Name";

Alter Hierarchy Example:

ALTER HIERARCHY "Copy of Time (Drilling)" NAME "Altered copy of Time (Drilling)" IN FOLDER "\Schema Objects" DRILLHIERARCHY FALSE DESCRIPTION "A modified copy of Time (Drilling)" HIDDEN TRUE LOCATION "\Public Objects" FOR PROJECT "MicroStrategy Tutorial";

Add Attribute to Hierarchy Template:

ADD ATTRIBUTE "Attribute Name" UNLOCKED IN FOLDER " Folder Path " ENTRYPOINT TRUE CHILDATTRIBUTES "Child Attribute Name " IN " Folder Path " TO HIERARCHY " Hierarchy Name " IN FOLDER " Folder Path " FOR PROJECT " Project Name ";

Add Attribute to Hierarchy Example:

ADD ATTRIBUTE "Month" UNLOCKED IN FOLDER "\Schema Objects\Attributes\Time" ENTRYPOINT TRUE CHILDATTRIBUTES "Day" IN "\Schema Objects\Attributes\Time" TO HIERARCHY "Copy of Time (Drilling)" IN FOLDER "\Schema Objects" FOR PROJECT "MicroStrategy Tutorial";

Remove Attribute to Hierarchy Template:

REMOVE ATTRIBUTE "Attribute Name" IN FOLDER " Folder Path " FROM HIERARCHY " Hierarchy Name " IN FOLDER " Folder Path " FOR PROJECT " Project Name ";

Remove Attribute to Hierarchy Example:

REMOVE ATTRIBUTE "Year" IN FOLDER "\Schema Objects\Attributes\Time" FROM HIERARCHY "Copy of Time (Drilling)" IN FOLDER "\Schema Objects" FOR PROJECT "MicroStrategy Tutorial"

10

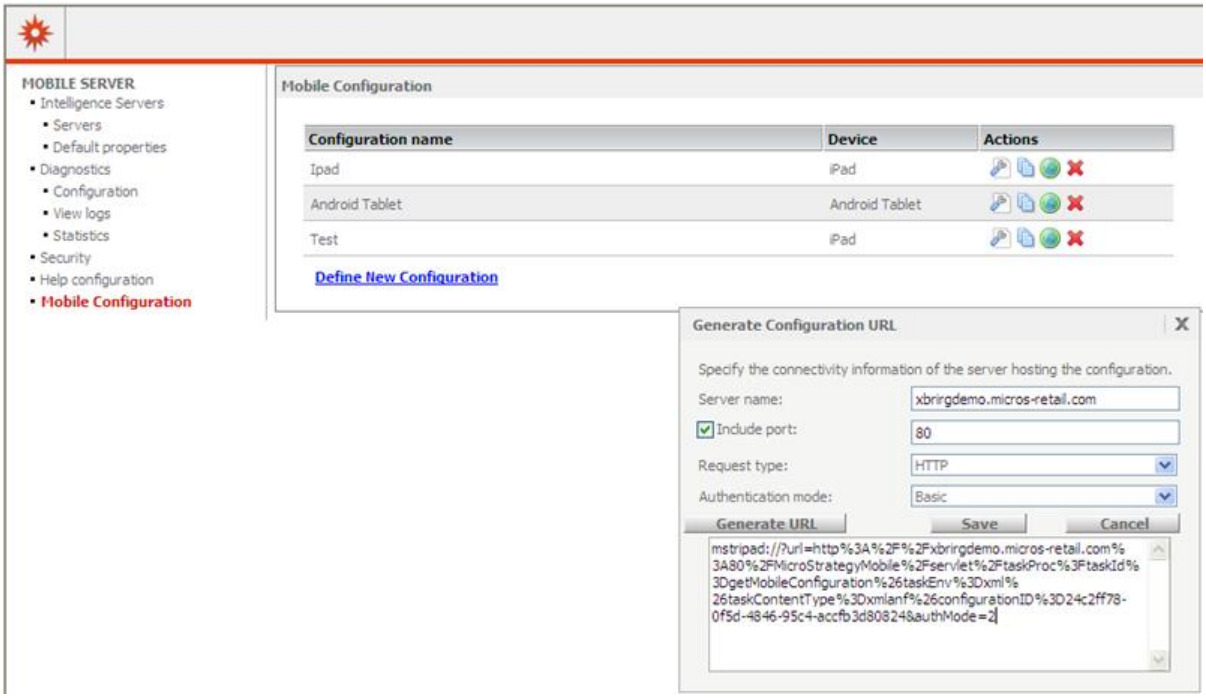
Oracle XBR^{i} Mobile Configuration

You can install the Oracle XBR^{i} app on an iPad. The app for iPad is available from the Apple App Store. Prior to installing the app, the Oracle Cloud installation team sets up the Apple device to connect to the XBR^{i} Mobile Server. Then generate the URL for the device, as described in the following section:

Generating the URL for the Mobile iPad

From the Mobile Configuration start page, you need to generate the URL for the iPad that has installed the Oracle XBR^{i} mobile app. Then send the URL to the user who has the mobile app installed. They use this link to log on to the mobile server.

> **Note**: The user profile must have access to both modules in order for the SP&LP link to work.



Mobile Configuration Start Page

Step 1: Generate and send the URL

1. Select the **Generate URL** icon under **Actions** for your configuration.
2. In the Generate Configuration URL dialog box, select **Basic** from the **Authentication Mode** drop-down list.
3. Click **Generate URL**.
4. In the text box, select the entire URL, copy it and change the first six characters from "mstr" to "xbripad", and paste it into an e-mail message. Send the message to the iPad user with the Oracle XBR^{i} Mobile app installed.

Step 2: Configure the URL on the iPad

1. Download the Oracle XBRi app from the Apple app store.

2. On the iPad, open the e-mail and copy the URL to the Notes app. This converts the long URL to a link.
3. Tap the link to connect to XBR^i Mobile Server.
4. Enter your XBR^i login credentials, with your User Name prefixed by your customer code and an underscore. For example, for user JSmith and customer XYZ, enter *xyz_jsmith.* Enter your XBR^i password.
5. Either tap the **GO** button on the virtual keyboard, or shrink the keyboard and click **OK** on the login prompt.

   This logs you in to the XBRi mobile app. Hereafter, when you click the Oracle XBR^i app icon, the saved credentials are used and you will not need to reenter them.

Logging out of the app

If you need to completely log out of the app, follow the steps below. To log back in, you must follow steps 3 and 4 above, making sure to prefix the User Name with the customer Org code.

To log out of the app:

1. Click on the **Settings** icon in the app.
2. Under Mobile Server, select the server.
3. Scroll to the bottom of the page, and click Delete. This logs you out of the app.

11

Troubleshooting

This sections describes issues that may arise after installation and how to resolve them.

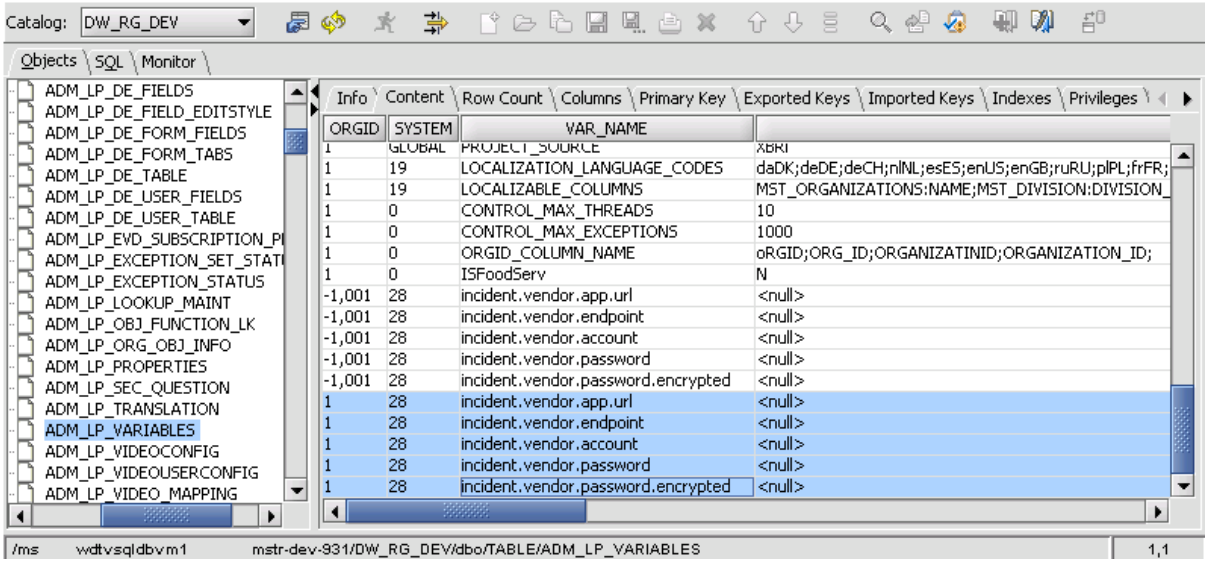
> **Note:** For the procedures in this Troubleshooting chapter, the ODI, UI, and database access is only available in the Implementation environment, not in Pre-Production or Production.

Validate Case Management Connection Settings in the Data Warehouse

You can validate Case Management connection settings in the XBRi Project Defaults – Case Management Settings page, or through the data warehouse. This can only be done in the Implementation environment.

To verify value configuration for case management in the data warehouse:

1. On the data warehouse server, go to your front-end database tool.
2. Locate the database instance of your project.
3. Under the tables folder, locate ADM_LP_Variables.



ADM_LP_Variables Table

4. For the rows with the ORGID of the customer, SYSTEM 28, and VAR_NAME beginning with incident., ensure that the values are correct:

   Incident.vendor.app.url = https://appserver1.lpguys.net/microstesting/lpms

   Incident.vendor.endpoint = https://appserver1.lpguys.net/microstesting/lpms/webservice/dataservice.asmx

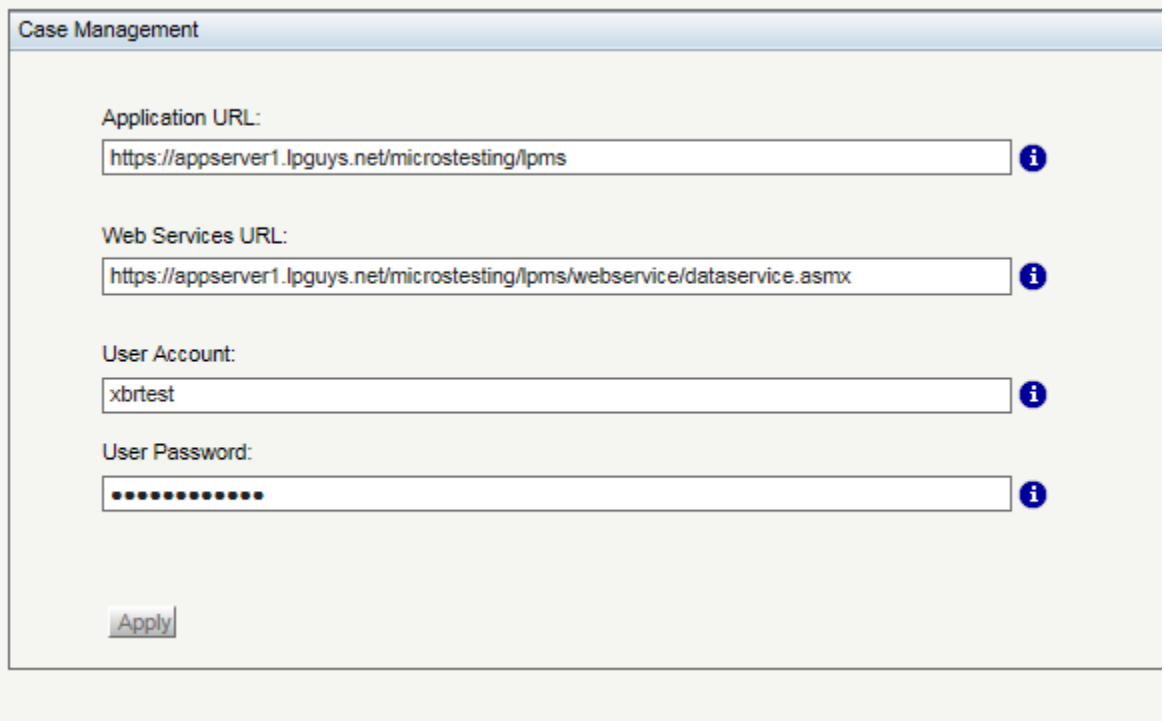
   Incident.vendor.account = userName

   Incident.vendor.password = userPassword (encryption through mr.key with {tomcat}/lib/MrCrypto.jar)

   Incident.vendor.password.encrypted = true/false

To verify value configuration for case management in Project Defaults:

1. Log in to XBRi as a customer administrator.
2. From the Admin menu, choose Project Defaults.
3. Under Settings, click **Case Management**. This displays the Case Management window:



Case Management Window

4. Verify that the following settings are correct:

   Application URL – Fully qualified URL to access case management online.

   Web Services URL – Fully qualified URL to access vendor web services.

   User Account – Eligible account to access web services.

   User Password – Password associated with the User Account.

5. Click **Apply** to save any changes.

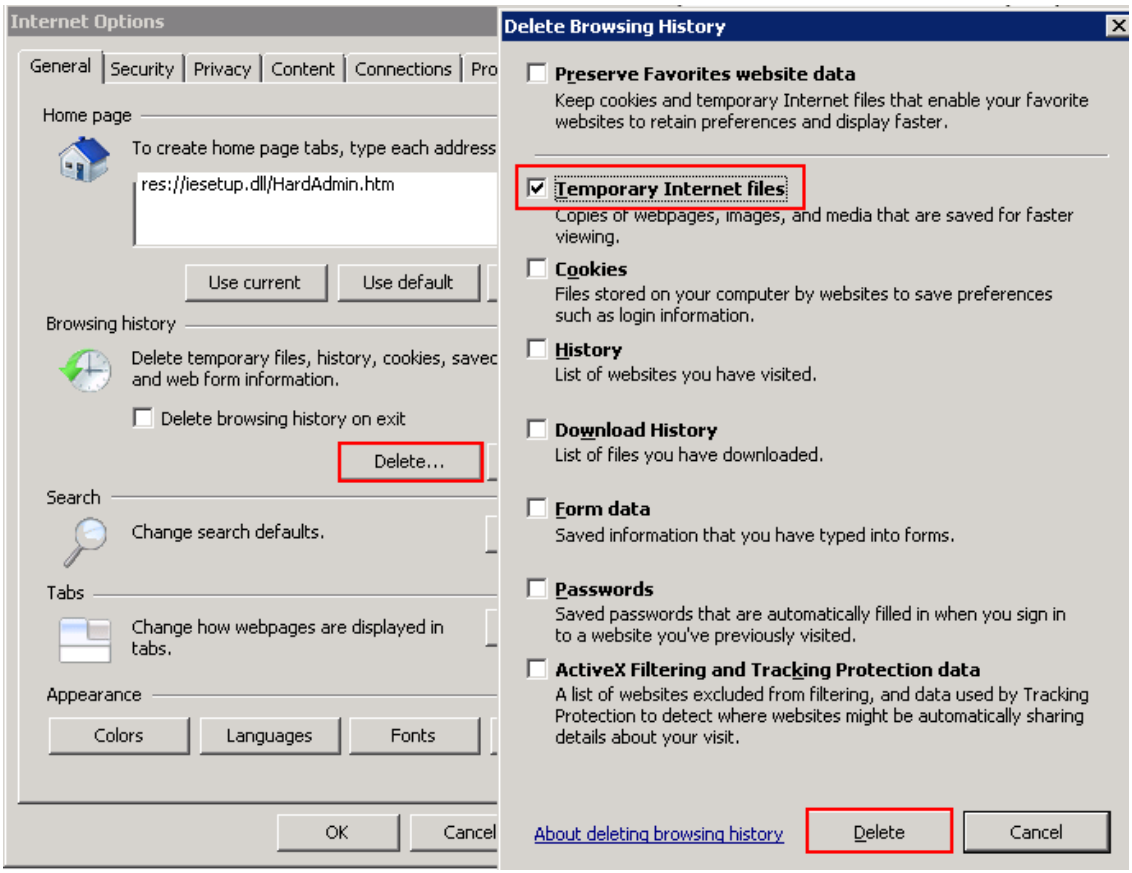
Dates Display Incorrectly in a Linked Report

If, after installation, the customer edits a link to a report, sets the date prompt to Prompt User and applys the change, when they run the linked report, and select a different date range, the dates do not display properly.

This problem has occurred with Internet Explorer.

To correct this problem:

1. Open Internet Explorer
2. Clear the browser cache of temporary internet files, as in the example below:



Delete Browsing History Dialog

3. From the Tools menu, select **Compatibility View Settings**.
4. Clear the Display intranet sites in Compatibility View check box.

New Users Do Not Receive E-mails

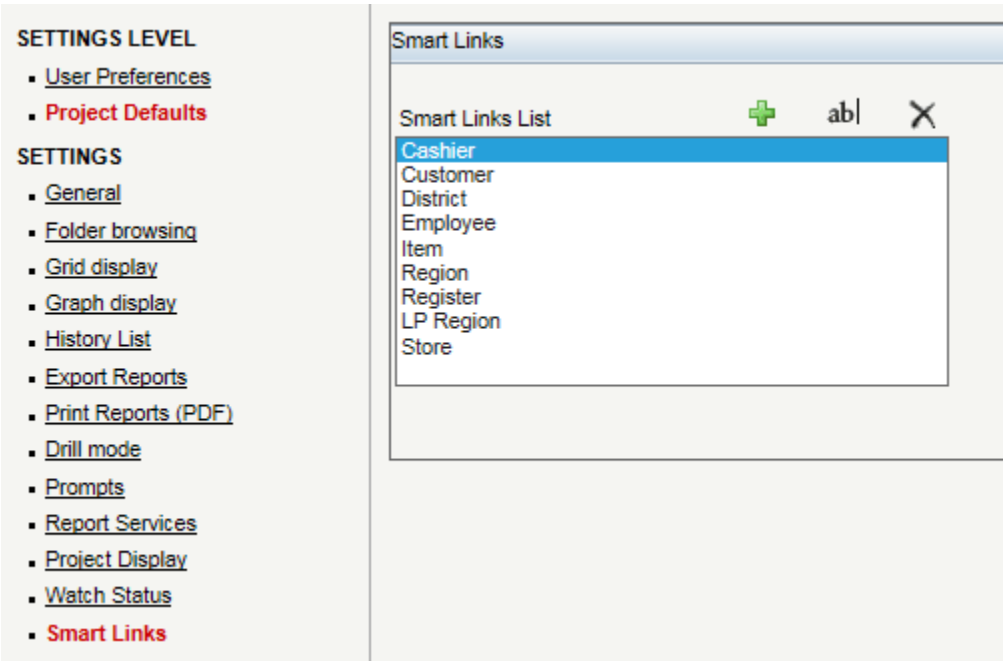
If your new users are not receiving e-mails with login and password information after their accounts are set up by the administrator, you may have the **Always Use Smart Host** option checked in the Device Editor and the Smart Host is not accessible. Perform these steps to correct the problem:

1. Log in to MicroStrategy Desktop.
2. Click on **XBR**^{i} (project Source) and click on **Administration.**
3. Navigate to **Delivery Managers > Devices**.
4. Right click on generic e-mail and select Edit from the context menu.
5. If the **Always Use Smart Host** option is selected, clear the check box.

Smart Links Do Not Work

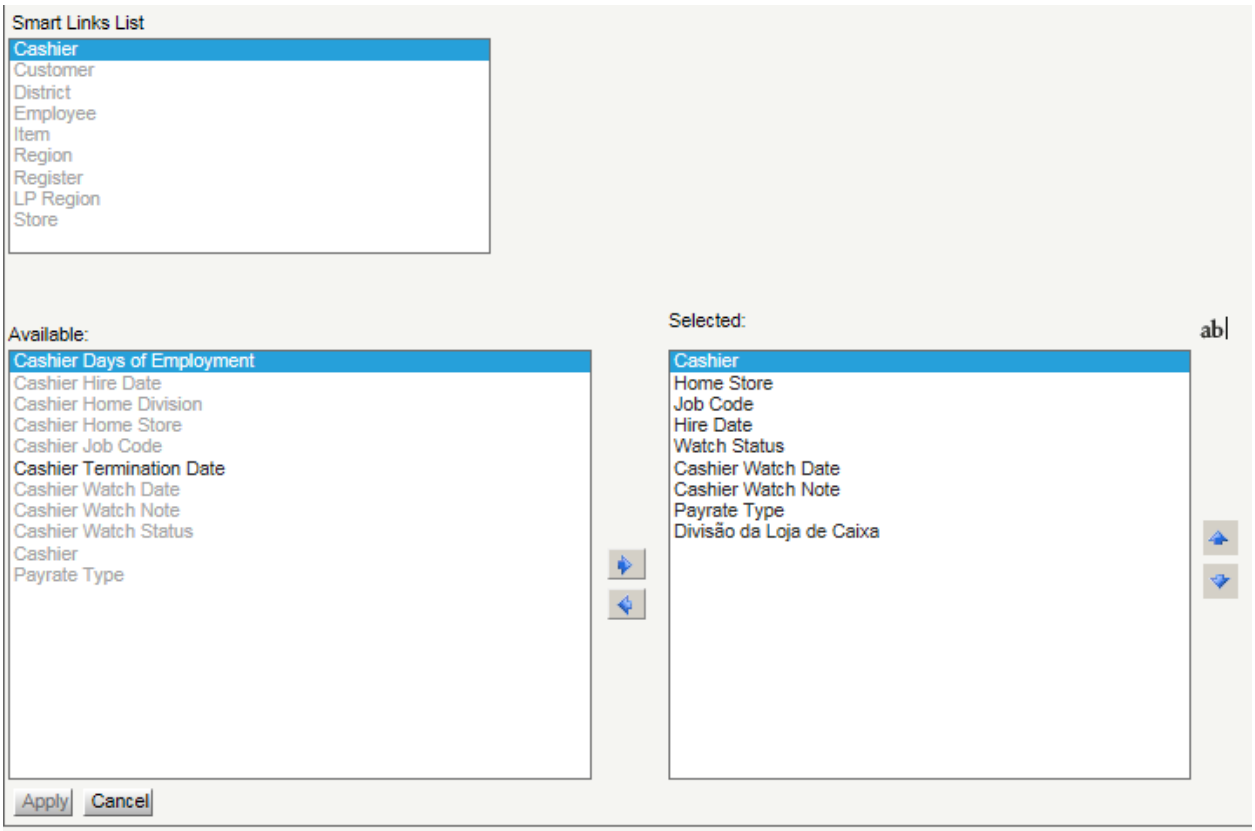
If some of the Smart Links that are set to be active after a new install or upgrade do not activate, you must re-apply the settings for those smart links in Projects.

1. Log in to XBR^i as the Customer Administrator
2. From the Admin menu, choose **Project Defaults**.
3. Under Settings, choose **Smart Links**. This displays the Smart Links window:



Smart Links Window

4. Select the attribute category under the Smart Links list and click the **Edit** icon. This displays the Smart Links editor:



Smart Links Editor

4. In the **Selected** box, highlight the smart link that is not working and click the **Edit** icon.
5. If necessary, edit any fields that need to be changed, and then click **OK.**
6. Click **Apply**.

Once you select **Apply**, the smart link should now work in the reports. Repeat the steps for any other smart links that you need to activate.

Calendar Visualization in Date Selection Prompts are Displaying Incorrect Dates or No Dates

If you run a report, document or control point and notice that the selected dates on the calendar visualization are out of date (for example, if you select Yesterday and the calendar displays the day before yesterday), it is likely that the date selection batch process did not run overnight as scheduled. To solve this problem, run the date selection batch process manually n the Linux Cloud server and check the authentication used to run the daily scheduled task.

To manually run the date selection batch process:

1. Navigate to the Linux sever using a Linux shell such as SSH or Putty and log into an account with sudo privileges. Check the following to identify the reason the calendar displays are out of date:
   - Check the date selection batch process to see if it is still scheduled to run daily based on the output of "sudo crontab -1" If there are errors in the cron job, correct them.
   - If the cron job is correct, review the logs in /var/log/xbri-batchProcess for any errors. Correct any errors that you find.
2. Check the date selection batch process to see if it is still scheduled to run dailybased on the output of "sudo crontab -1" I there are errors in the cron job, correct them.
3. If you determine that the date batch process needs to be run manually, Run it using these commands:

   ```
   cd /usr/local/xbri/batchProcess
   sudo ./dsBatch.sh
   ```

4. Check the log to make sure it ran successfully.

Linking from the Control Points Exception Dashboard Does Not Work

This problem may occur after an Oracle new installation or upgrade.

To fix broken linking from the Control Points Exception Dashboard:

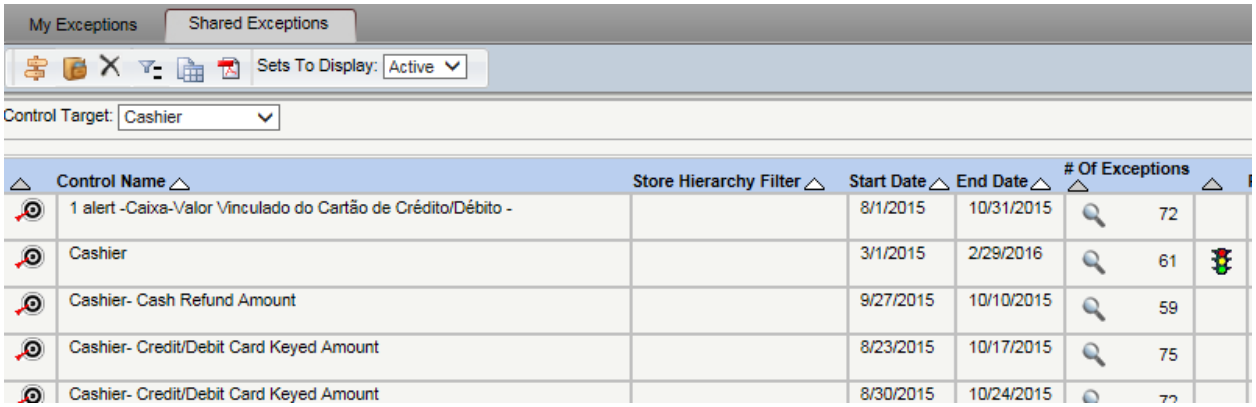
1. Navigate to the Oracle Database.
2. Insert the following into ADM_LP_VARIABLES:

```
(ORGID,SYSTEM,VAR_NAME,VAR_VALUE,VAR_DATATYPE,COMMENTS) VALUES (-
1001,'2','link.execution.oracle.edd.masked.format','yyyy-MM-dd',null,null)
```

3. Re-start Tomcat.

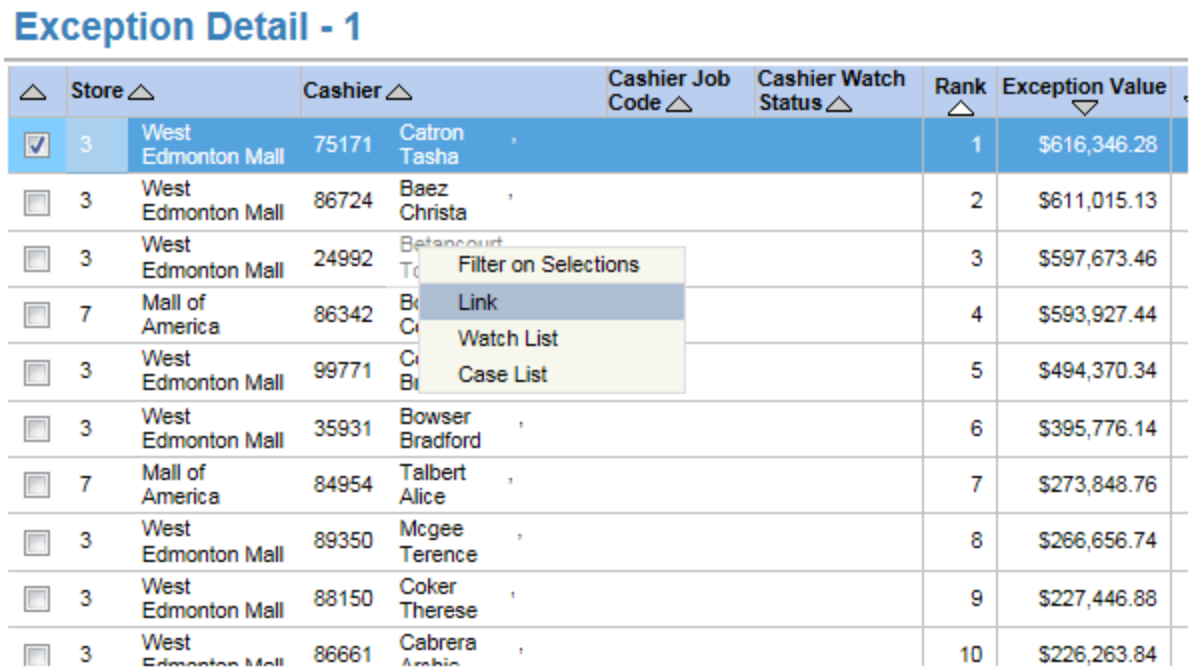
To verify that the links are working:

1. Log in to XBR*i* as the Customer Administrator.
2. From the Exceptions menu, select **Exception Results**.
3. Select the Shared Exceptions tab. This displays the Control Points Exceptions dashboard for all shared exception results:



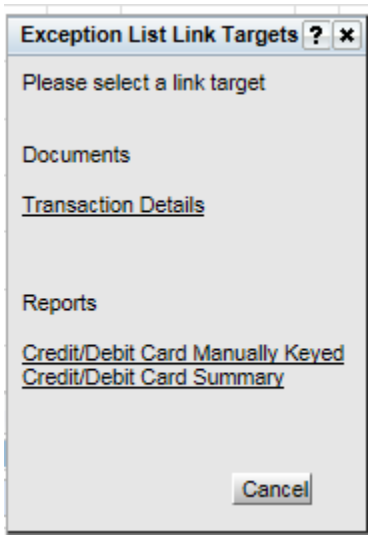
Shared Exceptions Dashboard

4. In a **# of Exceptions** cell with a value, click the magnifying glass icon. This displays an Exception Detail page. Right-click on an attribute and select **Link** from the menu.



Exception Detail

4. Verify that this opens a linked target report or document, or to a list of linked targets like the following image:



Link Targets

5. If presented with a list, click a link and verify that it opens the linked report or document.

Video Link Does Not Connect Successfully with Certain JRE Versions

If customers run into an issue with video linking and certain JRE versions, it is recommended that they upgrade to the latest JRE version (certified up to version 55).

Customers who are running an earlier version of XBR^{i} (prior to 10.7) and run into this issue should upgrade to the latest JRE version. They should also add the Analytics Web Server to the Exception Site list in the java control panel.

To Add URLs to the Exception Site list:

1. Go to the Java Control Panel (From Windows, click **Start** and then, **Configure Java**)
2. Click the Security tab
3. Click **Edit Site List**.
4. In the Exception Site List window, click **Add**.

> **Note:** JRE version 21 is not compatible with any version of XBR^{i}.

Gauges and Line Graphs Do Not Display on Activity Dashboards

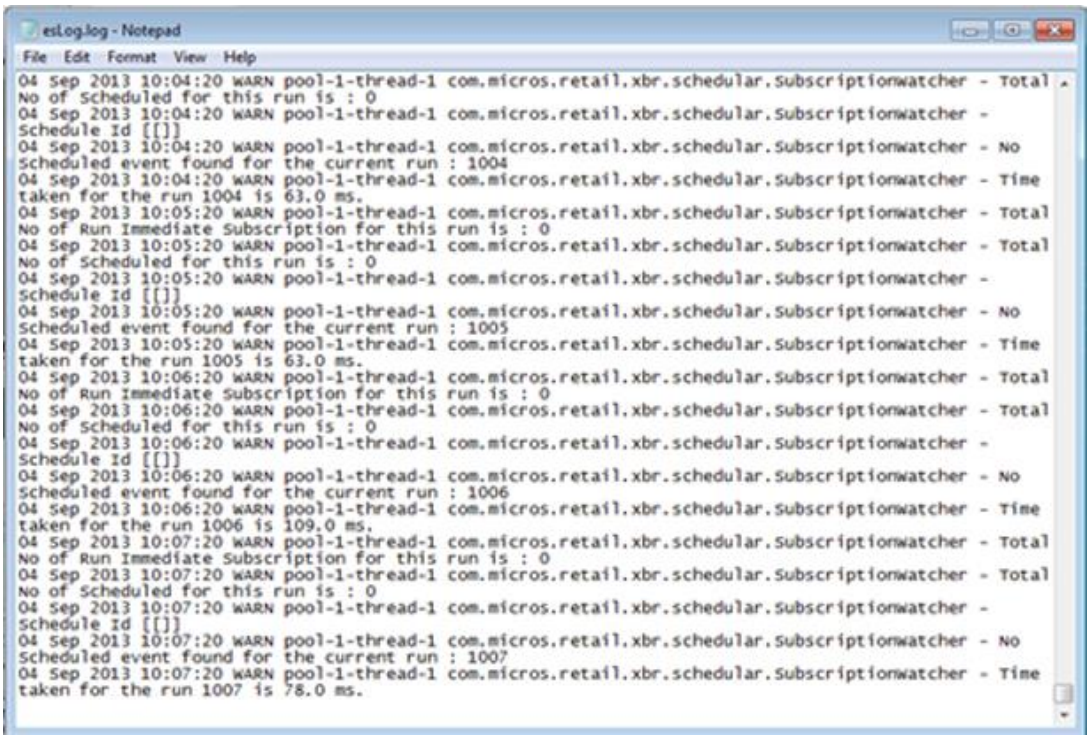
After installation, the gauges and line graphs do not populate on the Activity dashboards until you select a block in the heat map. This affects the Refund Activity, Void Activity, and Discount Activity dashboards for Retail installations. This problem occurs in the XBR^{i} application and in the Oracle XBR^{i} app for iPad. In order to have the dashboards show the gauges and graphs for each tab for future run times, you can select the heat map defaults for each tab and save them.

To correct this problem, for each dashboard:

1. Log in as a core XBR Administrator.
2. Run the dashboard.
3. Click in an area of the heat map to populate it. For example in the Discount Activity dashboard, click a District in the heat map.
4. Move to the next tab, for example, Store in a Discount Activity dashboard, and click in an area of the heat map to populate it.
5. Do the previous step for any remaining tabs..
6. From the toolbar, click **Save**.
7. Click **OK**.

Troubleshooting Schedules

You can troubleshoot external schedule distribution failures by viewing entries in the esLog.log file located in the XBR*i* application log folder: …XBRi\ExternalScheduler\log folder.



Sample esLog.log file

Viewing Tomcat Log Files to Monitor Application Issues

Because of enhanced security, it is necessary to take extra steps to view log files in the application Tomcat directories. You can either view the log file in place, or make a copy. The following steps use the example of the Tomcat catalina log, where usr is the current user and the example is dev:

To view the log file in place, enter the following in the command line:

```
sudo cat /dev/local/tomcat01/logs/xbri-tomcat01/catalina.2016-11-11.log
```

To make a copy of the log file, change owner to user "dev" and make it viewable in SSH by entering the following in the command line:

```
sudo cp /dev/local/tomcat01/logs/xbri-tomcat01/catalina.2016-11-11.log /home/dev
sudo chmod dev:dev /home/dev/catalina.2016-11-11.log
```

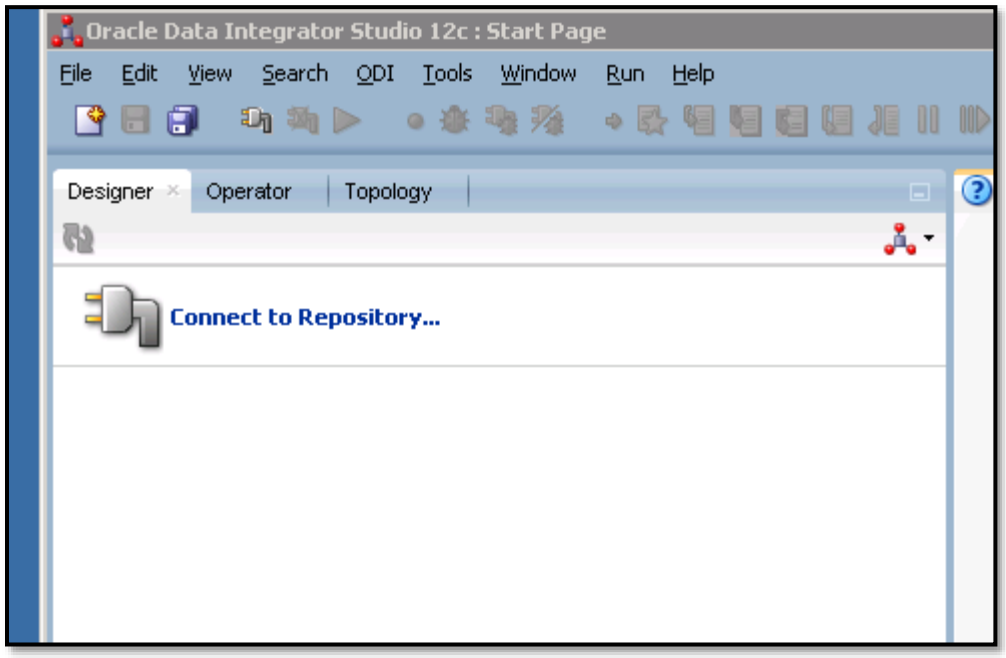
Debugging ETL failures

This section shows you how to troubleshoot ETL failures.

Review Errors Within ODI Studio

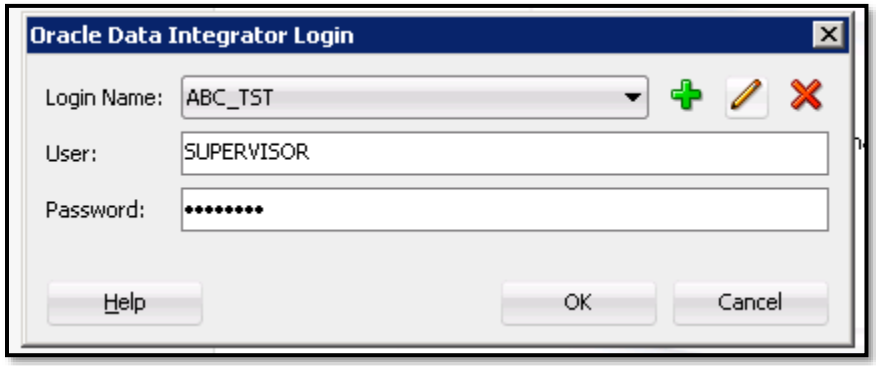
> **Note**: This procedure can only be done in the Implementation environment.

1. Launch ODI Studio (odi.exe).
2. Click **Connect to Repository**.



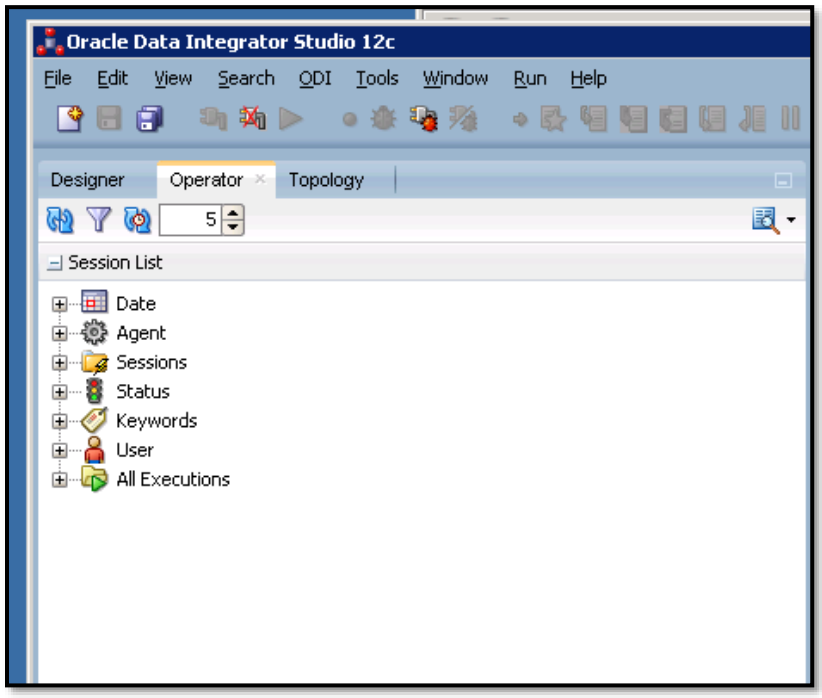
ODI Start Page

3. Select your login name from the Login Name drop-down list, and click **OK**.



ODI Login Prompt

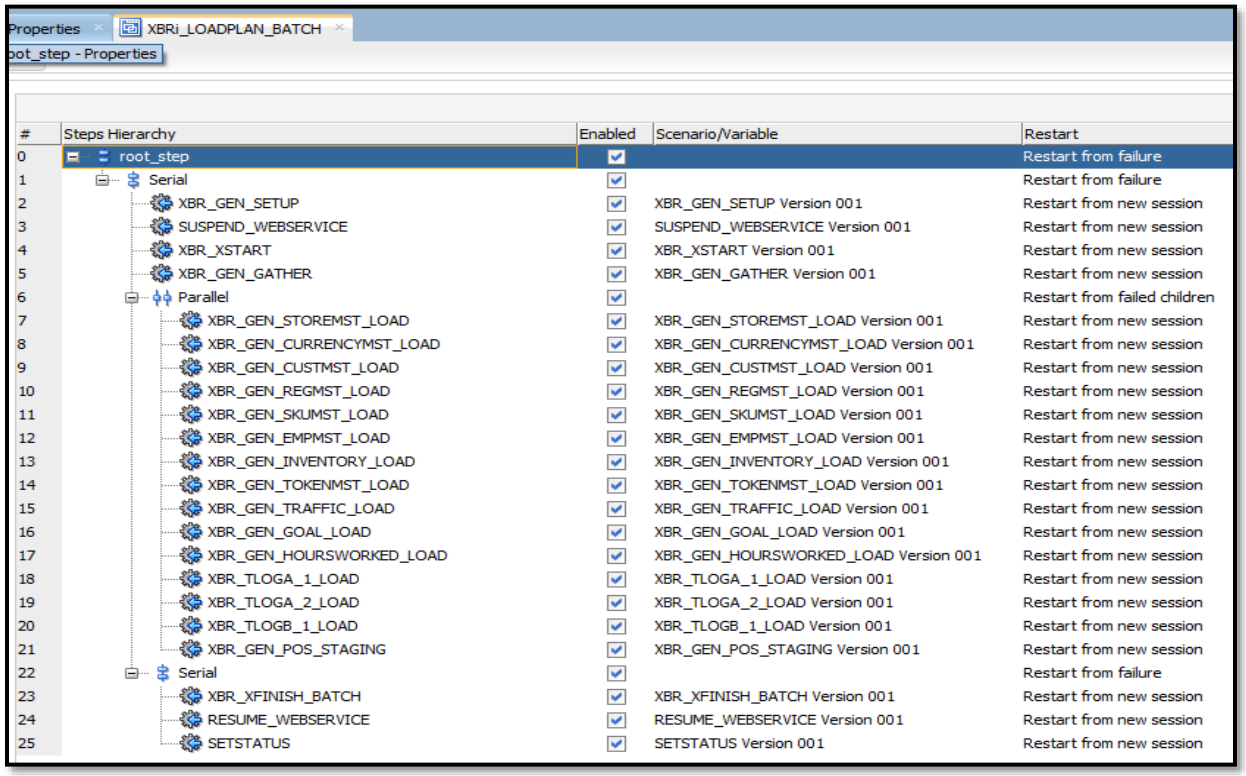
4. Select the Operator tab.



ODI Operator Tab

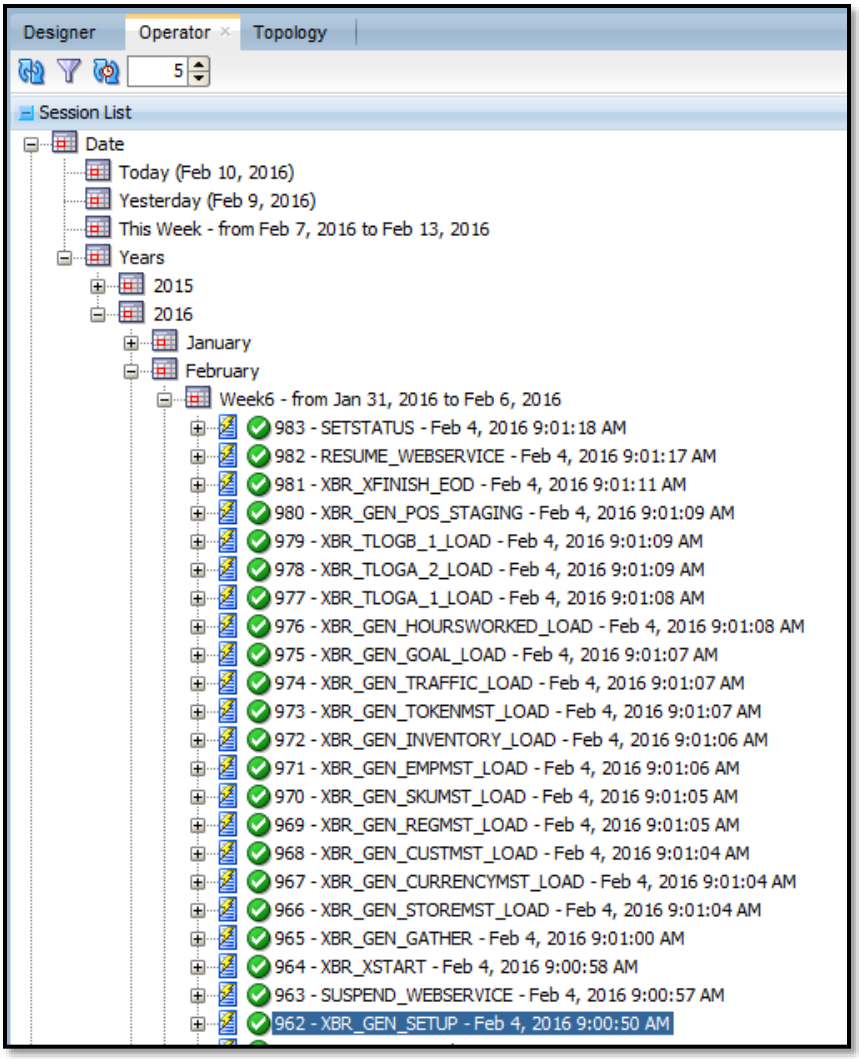
Viewing Load Status in the Operator Tab

The Operator Tab shows the status of ODI packages executed by the Load Plan:



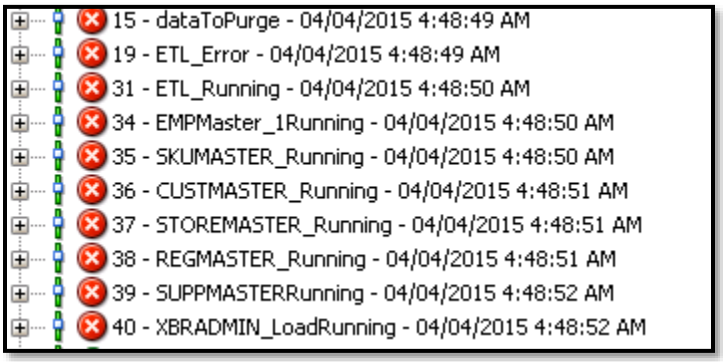
#	Steps Hierarchy	Enabled	Scenario/Variable	Restart
0	root_step	☑		Restart from failure
1	Serial	☑		Restart from failure
2	XBR_GEN_SETUP	☑	XBR_GEN_SETUP Version 001	Restart from new session
3	SUSPEND_WEBSERVICE	☑	SUSPEND_WEBSERVICE Version 001	Restart from new session
4	XBR_XSTART	☑	XBR_XSTART Version 001	Restart from new session
5	XBR_GEN_GATHER	☑	XBR_GEN_GATHER Version 001	Restart from new session
6	Parallel	☑		Restart from failed children
7	XBR_GEN_STOREMST_LOAD	☑	XBR_GEN_STOREMST_LOAD Version 001	Restart from new session
8	XBR_GEN_CURRENCYMST_LOAD	☑	XBR_GEN_CURRENCYMST_LOAD Version 001	Restart from new session
9	XBR_GEN_CUSTMST_LOAD	☑	XBR_GEN_CUSTMST_LOAD Version 001	Restart from new session
10	XBR_GEN_REGMST_LOAD	☑	XBR_GEN_REGMST_LOAD Version 001	Restart from new session
11	XBR_GEN_SKUMST_LOAD	☑	XBR_GEN_SKUMST_LOAD Version 001	Restart from new session
12	XBR_GEN_EMPMST_LOAD	☑	XBR_GEN_EMPMST_LOAD Version 001	Restart from new session
13	XBR_GEN_INVENTORY_LOAD	☑	XBR_GEN_INVENTORY_LOAD Version 001	Restart from new session
14	XBR_GEN_TOKENMST_LOAD	☑	XBR_GEN_TOKENMST_LOAD Version 001	Restart from new session
15	XBR_GEN_TRAFFIC_LOAD	☑	XBR_GEN_TRAFFIC_LOAD Version 001	Restart from new session
16	XBR_GEN_GOAL_LOAD	☑	XBR_GEN_GOAL_LOAD Version 001	Restart from new session
17	XBR_GEN_HOURSWORKED_LOAD	☑	XBR_GEN_HOURSWORKED_LOAD Version 001	Restart from new session
18	XBR_TLOGA_1_LOAD	☑	XBR_TLOGA_1_LOAD Version 001	Restart from new session
19	XBR_TLOGA_2_LOAD	☑	XBR_TLOGA_2_LOAD Version 001	Restart from new session
20	XBR_TLOGB_1_LOAD	☑	XBR_TLOGB_1_LOAD Version 001	Restart from new session
21	XBR_GEN_POS_STAGING	☑	XBR_GEN_POS_STAGING Version 001	Restart from new session
22	Serial	☑		Restart from failure
23	XBR_XFINISH_BATCH	☑	XBR_XFINISH_BATCH Version 001	Restart from new session
24	RESUME_WEBSERVICE	☑	RESUME_WEBSERVICE Version 001	Restart from new session
25	SETSTATUS	☑	SETSTATUS Version 001	Restart from new session

XBRi Load Plan Batch



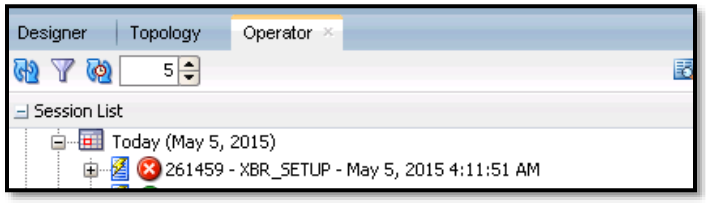
Corresponding Operator Tab

Red (X) is not always bad and green (checkmark) is not always good. Look at the step name and think of it as a question: "Is the ETL in the error state? Is the ETL running?" And so on.



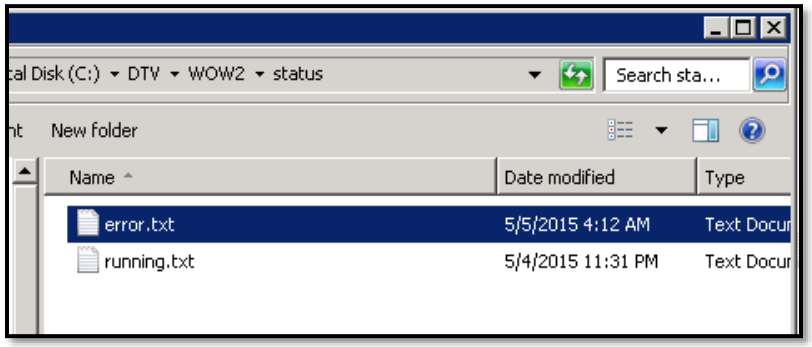
Red Status Examples

If ETL were terminated for any reason while processing (for example, a power outage), a running flag is left in status directory preventing ETL from starting again. The Setup Package will fail



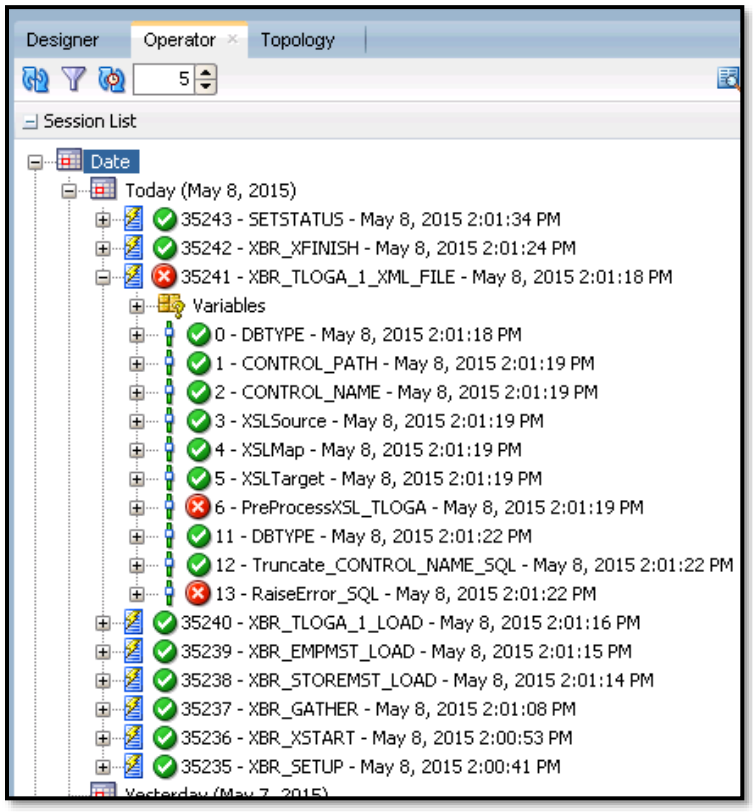
SETUP Failure Example

Once the cause of the failure is determined and it is verified that the problem has been fixed, You can delete the running or error flag, or both. Remove the running.txt and error.txt from the status directory.



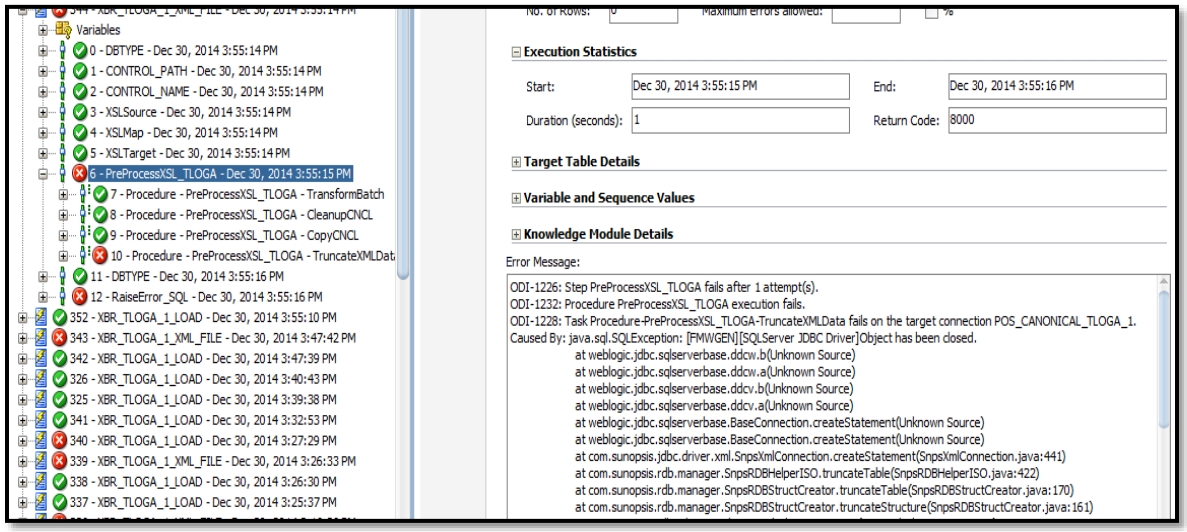
Status Directory

The following example shows a failure when processing an XML tlog through an XSL map:



XBR_TLOGA_1_LOAD Failure

If you expand Step 6 of session 35241, as shown in the previous image, items 7-10 are displayed:



Step 6 Expanded

This is caused by a Stand Alone Agent that is hung. Verify the POSCanonical.xml and POSCanonical.xsd files are valid and that they exist in the ../TRANSFORMS/TLOGA_1/xsl directory.

Resolution:

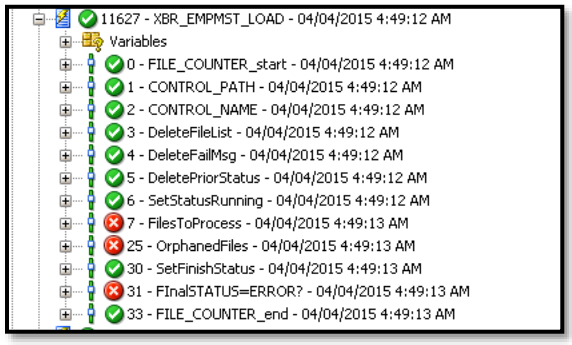
Restart the standalone agent.

Reviewing Master file packages in the Operator Tab

Files that are delivered in standard XBR^i format to the SFTP site will be moved to the INCOMING_FILES landing area in their respective folders. The ELT package XBR_GEN_GATHER moves the files to the control's /tmp directory for processing. Note that a POS_STAGING.dat file that is delivered in standard format falls under this category.

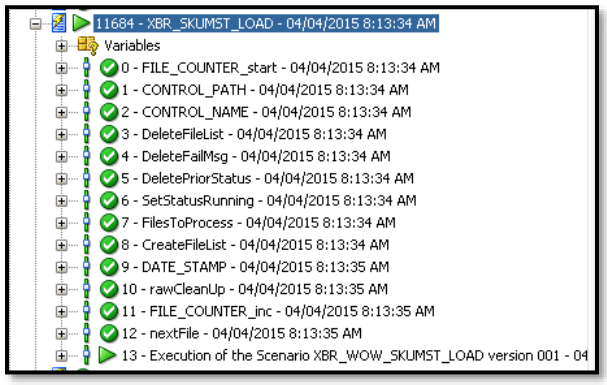
In the following example, no files were moved from LANDING_AREA to the control's /tmp directory for processing

> **Note:** The orphanedFiles step is a check to see if any rogue files are left in the tmp directory. Rogue files are zipped and then moved to the OUTPUT_ARCHIVES directory with a prefix of orphan.

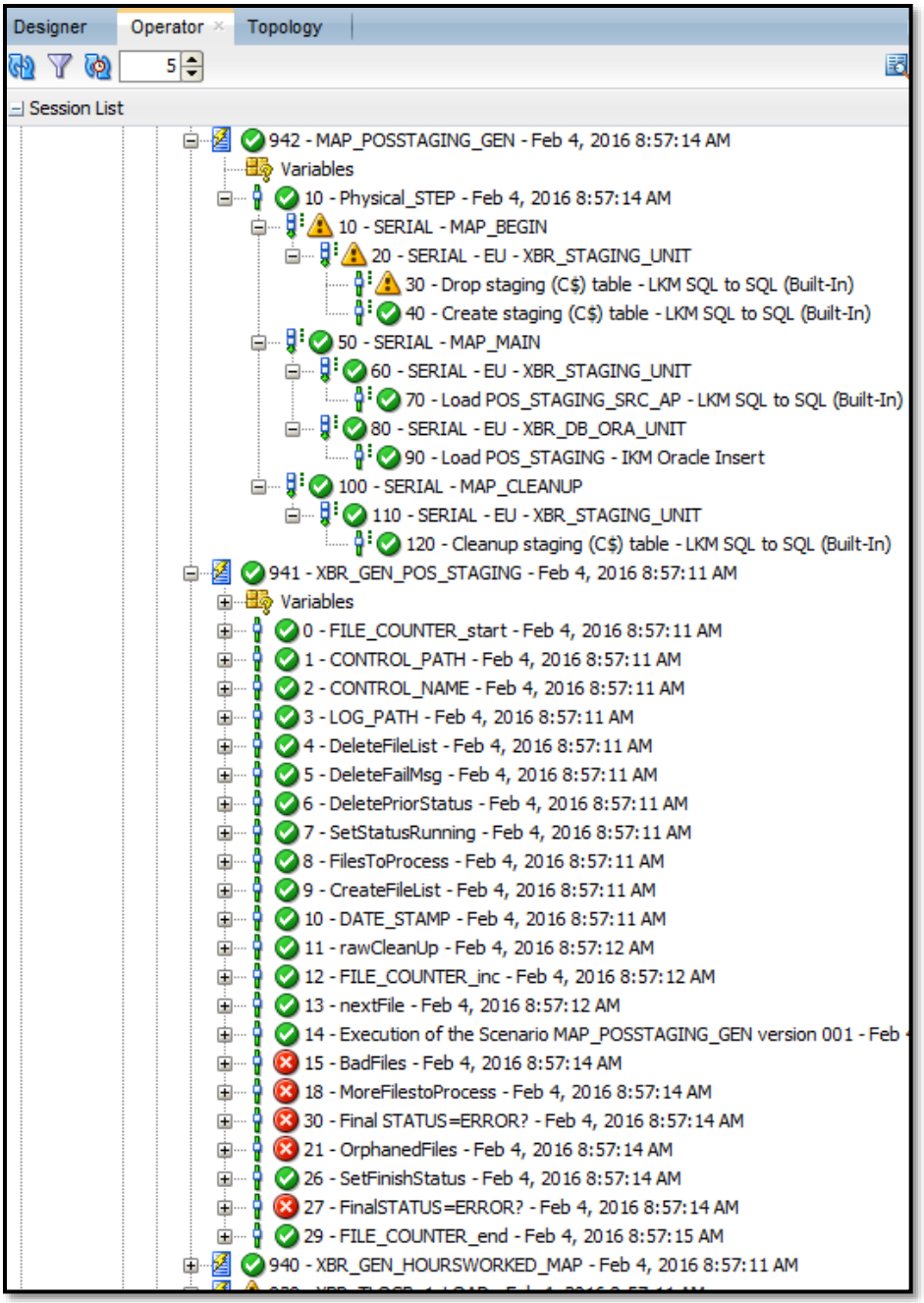


Example: No files Moved to TMP Directory

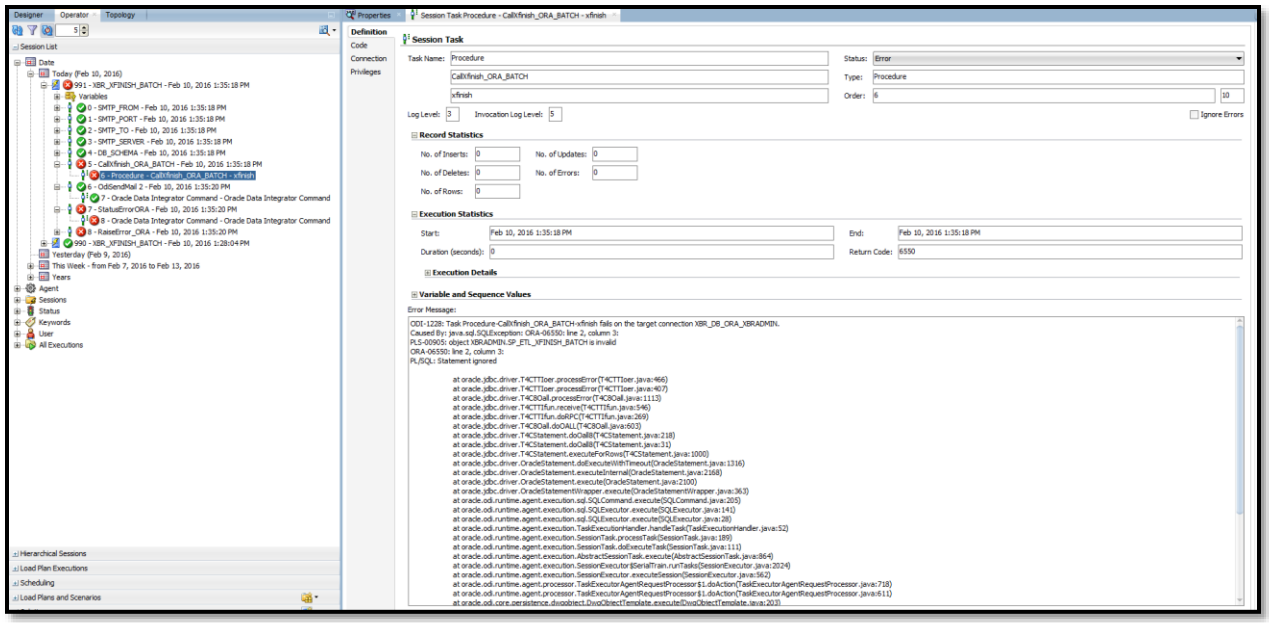
The following image shows that a file was found and is currently being processed (FilesToProcess is green). The map's scenario has been called.



Example: File Being Processed

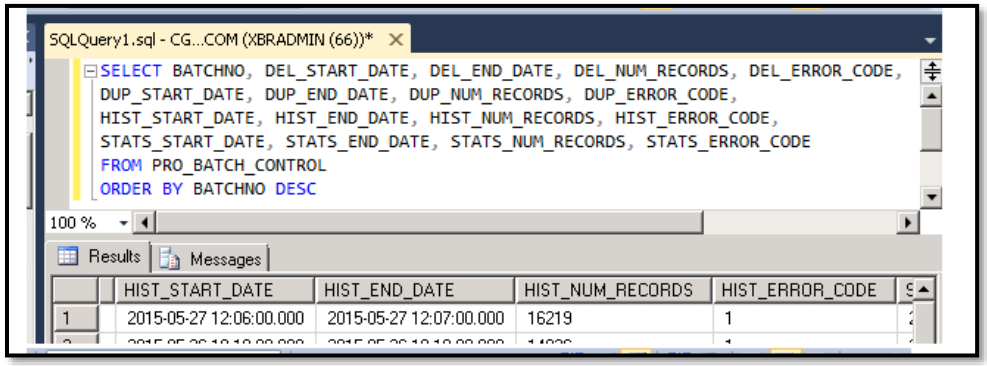


Reviewing Pos Staging File Load from the operator tab

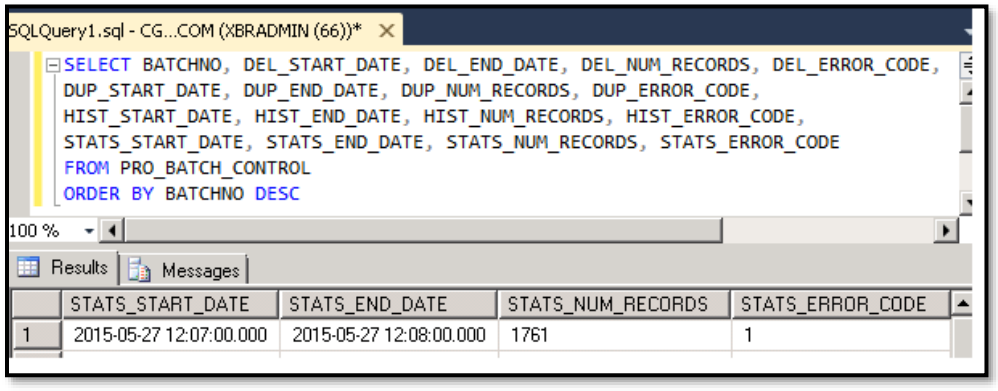


Database Stored Procedure failure

Connect to db, run select from PRO_BATCH_CONTROL table for most recent batch, verify that the HIST_NUM_RECORDS and STATS_NUM_RECORDS are populated.



HIST_NUM_RECORDS



STATS_NUM_RECORDS

Connect to db, run select from PRO_EVENTLOG table to identify the stored procedure responsible for the failure.

Columns | Indexes | Constraints | Triggers | Data | Script | Grants | Synonyms | Partitions | Subpartitions | Stats/Size | Referential | Used By | Auditing

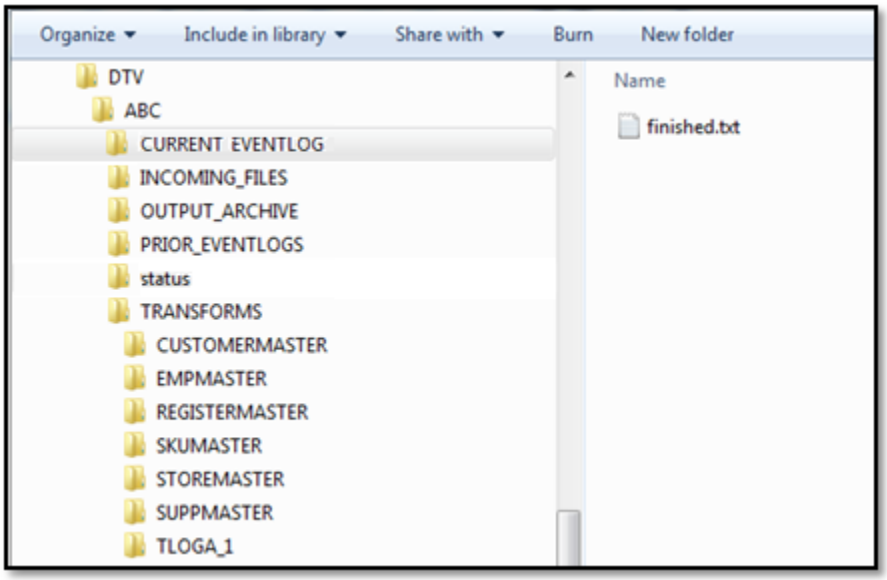
Sort by Primary Key Desc
Read Only Auto Refresh

LOGEVENT	LOGUSERID	LOGDATE	LOGTIME	LOGCODE	LOGMESSAGE	LOGINFO
start process	Sp_Pro_Load_Hist	2/10/2016	13:28:09			Sp_Pro_Load_Hist Move staging to Proact Procedure- process started.
end process	Sp_Pro_Load_Hist	2/10/2016	13:28:09			Sp_Pro_Load_Hist ended - no errors. new records loaded.
begin process	SP_PRO_LOAD_STATS	2/10/2016	13:28:09			sp_pro_load_Stats - process started.
end process	SP_PRO_LOAD_STATS	2/10/2016	13:28:10			sp_pro_load_Stats - process ended successfully.
start process	sp_pro_transdate_pu...	2/10/2016	13:28:10			sp_pro_transdate_purge stored procedure - starting history data purge...
end process	sp_pro_transdate_pu...	2/10/2016	13:28:11			sp_pro_transdate_purge stored procedure - Procedure ended successfully without error
UPDATE_BEGIN	SYSTEM	2/10/2016	13:28:08			SP_MST_UPD_LOOKUPS - MST_OTH_CUSTOM_CODE10_TAB completed

PRO_EVENTLOG

ODI ELT Framework Log Files

Log files generated by the ODI ELT framework within the packages are found in the CURRENT_EVENTLOG folder. Each ELT run zips up the files from the prior run and moves them to the PRIOR_EVENTLOG folder. Each run also creates status files in the status directory of the control (type of file being processed) and at the upper level of the ELT. The ODI standalone agent generates its own log files. You must request these from the Cloud team, as access to the Oracle middleware directory is limited.



CURRENT_EVENTLOG Folder

This folder contains the following log files

Eventlog.txt

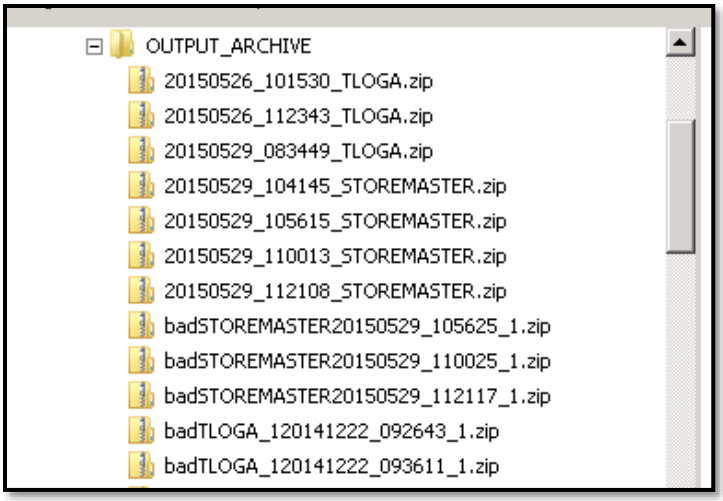
Failuremsg.txt

Status/finished.txt

Status/error.txt

Status/running.txt

Bad data is archived to the OUTPUT_ARCHIVE directory.



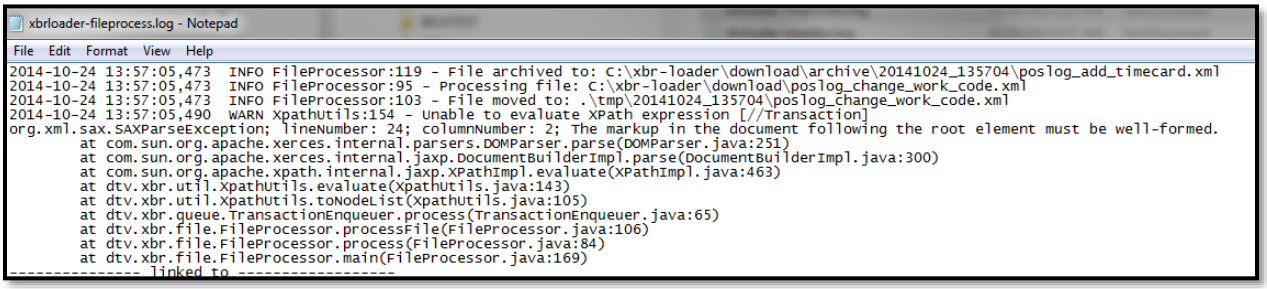
OUTPUT_ARCHIVE Directory

XBRLOADER Log Files

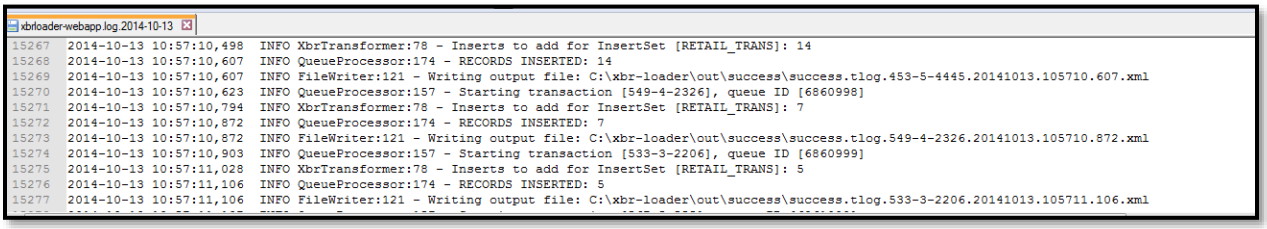
Failures in loading data to the POS_STAGING table in the database can be found in the following log files:

/usr/local/xbr-loader/log

xbrloader-fileprocess.log



xbrloader-webapp.log



Any transactions which are not processed and deemed bad by the web service are stored in:

/usr/local/xbr-loader/out/failures.