

Oracle® Retail XBRⁱ Cloud Services
Implementation Guide
Release 18.1

F21977-02

November 2019

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Barbara Clemmer-Dunn

Contributors: John Gaitens, Karen Bagdasarian, Bill Warrick

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

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

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

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

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

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

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

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

Contents

Send Us Your Comments	v
Preface	6
Audience	6
Documentation Accessibility	6
Access to Oracle Support	6
Related Documents	6
Customer Support	6
Improved Process for Oracle Retail Documentation Corrections	6
Oracle Retail Documentation on the Oracle Technology Network	7
Introduction	9
Scope of this Document	9
Cloud Components	9
Cloud Environments	9
User Acceptance Testing (UAT) Environment	9
Production Environment	10
Data Flow	11
Integration Components	12
The XBRⁱ Data Model	13
Transactional File Data	13
Transaction Functionality	13
Mapping Transactional Data to POS_STAGING Schema	14
Header Records	14
Detail Records	14
XBR ⁱ Database Stored Procedures	15
Internationalization	17
Translation	17
Multi-Language Setup	17
Scenario 1	18
Scenario 2	18
Scenario 3	18
Localizing Currency in XBR ⁱ	18
Implementing Currency Exchange Rates	18
Applying Updates for Currency Metrics	19
Data File Delivery	23
Introduction	23
Core Data Files	23
Core ELT and Associated Directory Structure	23
ODI XBRI_LOADPLAN	23
High Level ELT Description	24

Batch Mode	25
Real Time Mode	27
ELT Directory Structure	30
XBRLOADER Directory Structure	31
Data File Delivery Options	31
ODI ELT – Steps	35
Xstore/ XBRⁱ Integration	37
Introduction	37
XBR Loader Architecture	37
File Mode (Default)	37
Database Mode	38
Components of an Xstore- XBR ⁱ Web Service Integration	38
Xstore/ Xcenter 6.5 or Later	38
XBR ⁱ Broadcaster Enabled in Xcenter	38
XBR ⁱ Database	38
Tomcat Services Configured to Run XBR Loader Application	40
Submitting Transactions	40
Web Services Submission (File Processor)	40
POS_STAGING Data Load (Queue Processor)	41
Suspending and Resuming Poslog Dataload (CommandProcessor)	42
Purging (PurgeProcessor)	42
XSTORE poslogs through Web Service	42
Mapping Data through XPath	43
Basic Syntax	43
Root XML Elements	44
XPathAlias	44
InsertSet	44
Section	44
Translator	44
String Manipulation	47
XPath Resources	47
Web Services	49
Introduction	49
SOAP Based Web Services	49
XBR Loader SOAP-Based Web Services	49
SOAP-Based Web Services for POSLOG, Queue Processor, and Staging Processor	54
REST Based Web Services	55
Logging into REST Web Services	55
Available Rest Based WebServices	55

Send Us Your Comments

Oracle Retail XBRi Cloud Services, Implementation Guide, Release 18.1

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 XBRⁱ 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 18.1 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.)

Introduction

Oracle Retail XBRⁱ 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 XBRⁱ 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 XBRⁱ 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 XBRⁱ 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 housing Oracle Data Integrator (ODI) or XBRLoader web service or both. The XBRLoader Web Service is generally only used for XSTORE- XBRⁱ customers. (XSTORE 7.0 and higher). In a Cloud implementation, the necessary components are pre-installed.

Cloud Environments

When XBRⁱ is implemented in the Cloud, there are two environments: User Acceptance Testing (UAT) and Production. All environments are provisioned by the Cloud Team.

Note: The UAT environment will only be available as long as needed for customers and partners to perform the task requires after Cloud installation.

User Acceptance Testing (UAT) Environment

In the UAT 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 UAT environment before promoting configurations to the Staging Environment. In this environment, the customer is engaged to perform user acceptance testing. Customer access is limited to the XBRⁱ application front end. More information on the deployment process and how to address customer issues in this environment is covered in the *Oracle Retail XBRi Cloud Services Administration Guide*.

Production Environment

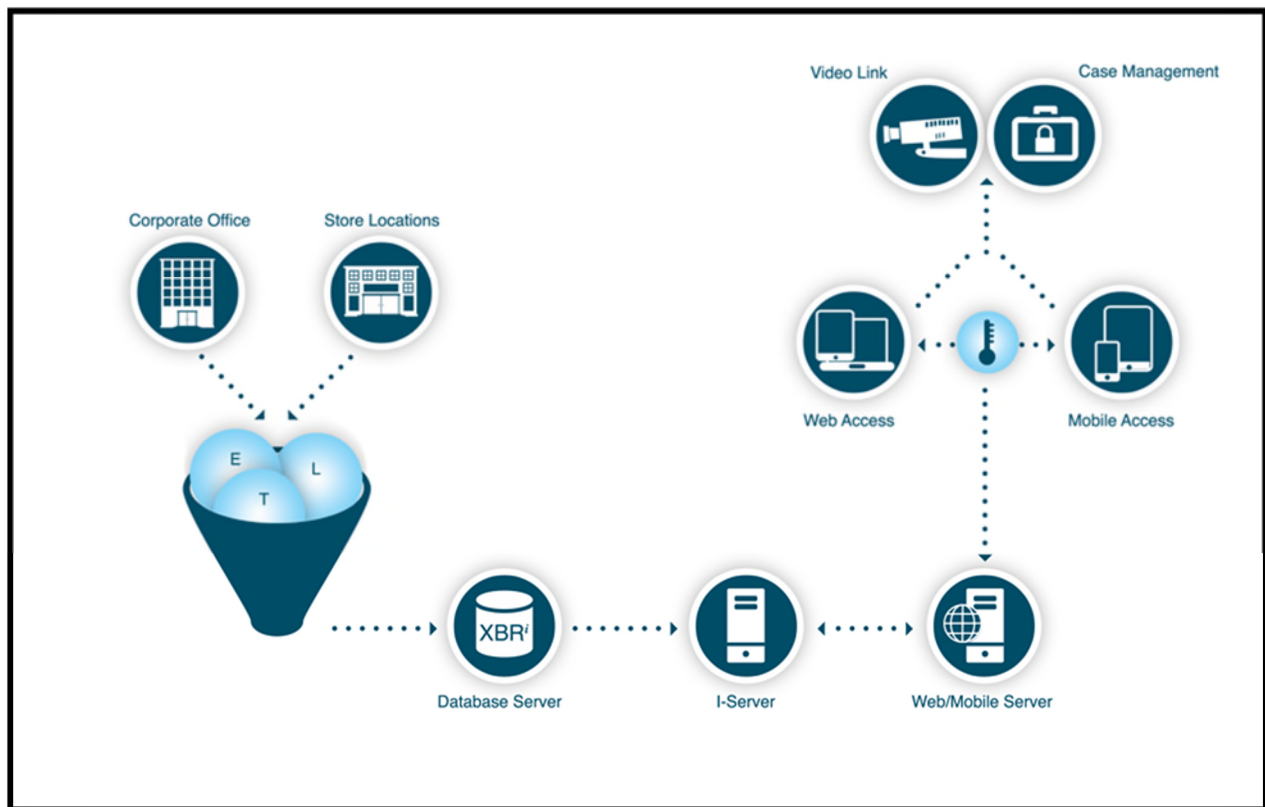
Production is the customer's go live environment.

Data Flow

At a high level, data flows into XBRⁱ 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 XBRⁱ database server. A set of stored procedures moves transactional data to historical tables. Supplemental data feeds are loaded to temporary tables in the XBRⁱ 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.

Note: The import data provided by customers is exposed to specified formats that are provided by Oracle. The ODI ELT framework is not directly accessible to customers.

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

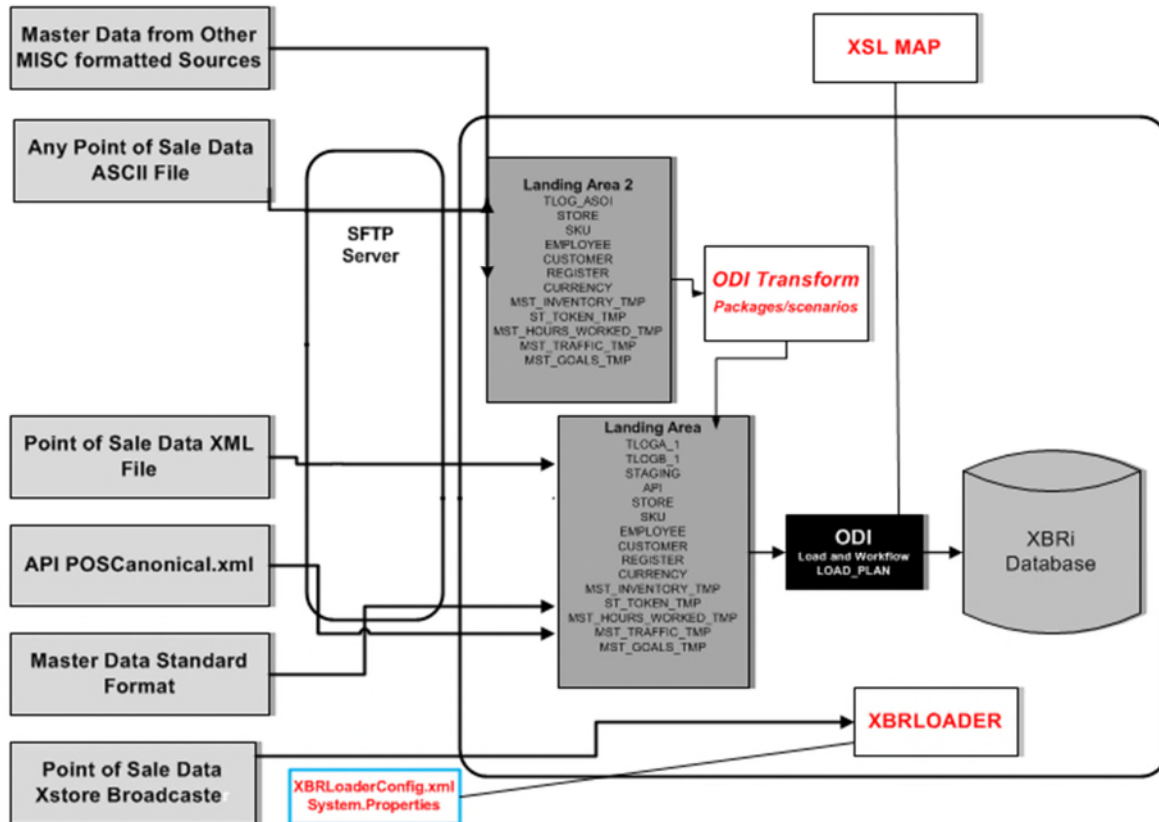


XBRⁱ Data Flow

Integration Components

Most of the XBRi integration is performed using XSL and XPATH configuration through web services.

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.



XBRi Integration Components

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

Setting Non-Standard Master Fiscal and Calendar Date Tables

When a new database is installed, the Fiscal Date Calendar format and date range to create for the master date tables follows the standard NRF Fiscal Calendar 4-5-4 monthly format, and the default fiscal calendar that was loaded with the core metadata generates back several years from fiscal year 2005 to 2030.

If the customer needs a different date range or uses the fiscal 4-4-5 format, you must enter an Oracle support request to have correct fiscal calendar loaded.

The XBRⁱ Data Model

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 or ASCII files.

The standard format of transactional data can be found in the XBRⁱ Cloud Services Core Field Mapping Guide, which is available on MOS. The POS Staging Schema Layout section in Appendix A of the XBRⁱ Cloud Services Core Field Mapping Guide. 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 XBRⁱ data model XML file. POSCanonical.xsd will be provided.

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 XBRⁱ, 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 XBRⁱ 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 XBRⁱ 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 Schema

All transactional data is mapped to the POS_STAGING schema in the XBRⁱ Database. A set of stored procedures (see: [XBRⁱ Database Stored Procedures](#)) move the data to historical schemas for viewing in the XBRⁱ application. Detailed mapping instructions are found in the XBRⁱ 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)

XBRⁱ Database Stored Procedures

All data is loaded to staging and temp schemas in the database. There are stored procedures that control the movement of this data to the appropriate today and historical schemas. An explanation of these stored procedures is shown in the following schema. The stored procedures are controlled through the ODI ETL process.

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 schema 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 schema 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 schemas to historical schemas. This procedure will run at the end of XBRI_LOADPLAN_EOD
SP_PRO_CLEAR_STAGE:	Clears the staging schema 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 schema and puts them in the pos_staging_dups schema
SP_PRO_LOAD_HIST:	Copies transactions from the staging schema to the Analytics history schemas. Calls sp_pro_nomatch_pvcancel and sp_pro_nomatch_returnexch to populate no match data.
SP_PRO_LOAD_STATS:	Loads the pos_statistics_tab schema for use in Analytics

SP_PRO_NOMATCH	Matches refunds and exchanges to their original transactions
SP_PRO_TRANSDATE_PURGE:	Purges POS historical schemas of old entries
SP_PRO_INVENTORY	Updates/Inserts inventory data from the temp schema
SP_PRO_LOAD_SPO_STATS	Aggregates data for Sales and Productivity summary schemas
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 schema and populates the hours worked schema
SP_SPO_UPD_GOALS	Updates/Inserts goal data from the temp schema
SP_SPO_UPD_TRAFFIC	Updates/Inserts traffic count data from the temp schema
SP_MST_UPD_STORE;	Updates/Inserts store data from the temp schema. Adds stores for new stores in the pos_staging schema
SP_MST_UPD_EMP	Updates/Inserts employee data from the temp schema. Adds employees for new employees in the pos_staging schema
SP_MST_UPD_SKU;	Updates/Inserts sku data from the temp schema. Adds skus for new skus in the pos_staging schema
SP_MST_UPD_REGNUM;	Updates/Inserts register data from the temp schema. Adds registers for new registers in the pos_staging schema
SP_MST_UPD_CUSTOMER;	Updates/Inserts customer data from the temp schema. Adds customers for new customers in the pos_staging schema
SP_MST_UPD_LOOKUPS;	Updates and inserts data in the lookup schemas
SP_MST_UPD_TOKEN	Updates pos_tnd_tab with token values which were delivered separately from the pos data

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. XBRⁱ 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

XBRⁱ data is supported in six languages. This section provides details of various scenarios that you may come across during implementation. Since multi-language data support in XBRⁱ 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:

- XBRⁱ 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, XBRⁱ 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 XBRⁱ primary language for data purposes to be supported within the source system.

Scenario 1

All the supported languages are implemented in XBRⁱ and the same set of languages are supported in the source system as well.

Multi-lingual data sets are enabled in both XBRⁱ 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 XBRⁱ, 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 XBRⁱ.

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, XBRⁱ will display the attribute in source system's primary language. For example, XBRⁱ requests data in German and English languages. In XBRⁱ 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 XBRⁱ 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 XBRⁱ primary language.

Scenario 3

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

Localizing Currency in XBRⁱ

If multi-currency reporting is required for an XBRⁱ implementation, the customer needs to provide a multi-currency exchange rate feed to the XBRⁱ 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 XBRⁱ Admin interface, as described in the *Oracle Retail XBRⁱ Cloud Services Administration Guide*.

Implementing Currency Exchange Rates

You can schedule XBRⁱ 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: XBRⁱ 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 XBRⁱ as the Customer Administrator.
2. From the Admin menu, choose **Project Defaults**.
3. Under Settings, choose Metric Bulk Update.

SETTINGS LEVEL	Metric Bulk Update
<ul style="list-style-type: none">▪ User Preferences▪ Project Defaults	
SETTINGS	
<ul style="list-style-type: none">▪ General▪ Folder browsing▪ Grid display▪ Graph display▪ History List▪ Export Reports▪ Print Reports (PDF)▪ Drill mode▪ Prompts▪ Report Services▪ Project Display▪ Watch Status▪ Smart Links▪ Alerting Preferences▪ Video Configuration▪ Lookups▪ Controls/Exceptions▪ Master File Distribution▪ Upload Goals/Sales▪ Salesperson Custom Stats▪ Store Status	<p>Scope: <input type="text" value="All Currency Metrics"/></p> <p>Currency Symbol: <input type="text" value="\$"/></p> <p>Currency Custom Mask: <input type="text" value="#.###.00"/></p> <p>Currency Position: <input type="text" value="Front"/></p> <p>Negative numbers: <input type="text" value="Red and parenthesis"/></p> <p>Decimal place (how many digits are after decimal separator). This option will be ignored if currency symbol is Custom: <input type="text" value="2"/></p> <p><input type="button" value="Apply"/></p>

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 XBRⁱ 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.

Data File Delivery

Introduction

This section describes how the core data files are delivered to the XBRⁱ database.

Core Data Files

File Names:

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

Note: The import data provided by customers is exposed to specified formats that are provided by Oracle. The ODI ELT framework is not directly accessible to customers.

Core ELT and Associated Directory Structure

The purpose of the ELT is to appropriately load the data files, as discussed in [Chapter 2 The XBRⁱ Data Model](#), to their corresponding tables in the XBRⁱ 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 XBRⁱ Cloud, the XBRⁱ ELT is a black box capable of loading Point of Sale (POS) data and supplemental data feeds into the XBRⁱ 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 XBRⁱ'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 XBRⁱ 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.

The screenshot shows the 'XBRi_LOADPLAN_BATCH' configuration window. It features a 'Steps' tab with a tree view on the left and a detailed table on the right. The tree view shows a hierarchy starting with 'root_step', followed by a 'Serial' block containing 'XBR_GEN_SETUP', 'SUSPEND_WEBSERVICE', 'XBR_XSTART', and 'XBR_GEN_GATHER'. This is followed by a 'Parallel' block containing multiple 'XBR_GEN_*_LOAD' steps (STOREMST_LOAD, CURRENCYMST_LOAD, CUSTMST_LOAD, REGMST_LOAD, SKUMST_LOAD, EMPMST_LOAD, INVENTORY_LOAD, TOKENMST_LOAD, TRAFFIC_LOAD, GOAL_LOAD, HOURSWORKED_LOAD, TLOGA_1_LOAD, TLOGA_2_LOAD, TLOGB_1_LOAD, POS_STAGING). The final 'Serial' block contains 'XBR_XFINISH_BATCH', 'RESUME_WEBSERVICE', and 'SETSTATUS'.

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

ODI XBRi_LOADPLAN_BATCH

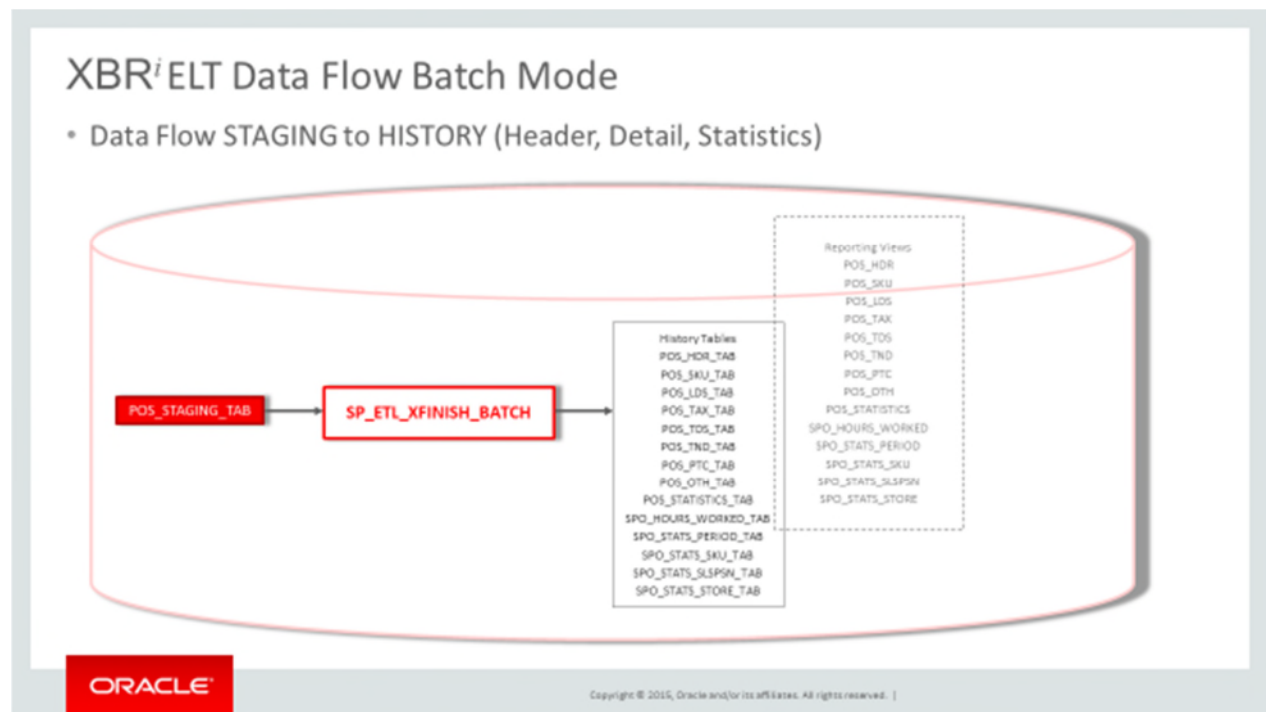
High Level ELT Description

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 XBRi's point or sale staging table format as well as transforming core master file data to prescribed XBRi 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 XBRi 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

Validate

Definition

Steps

Exceptions

Variables

Privileges

Version

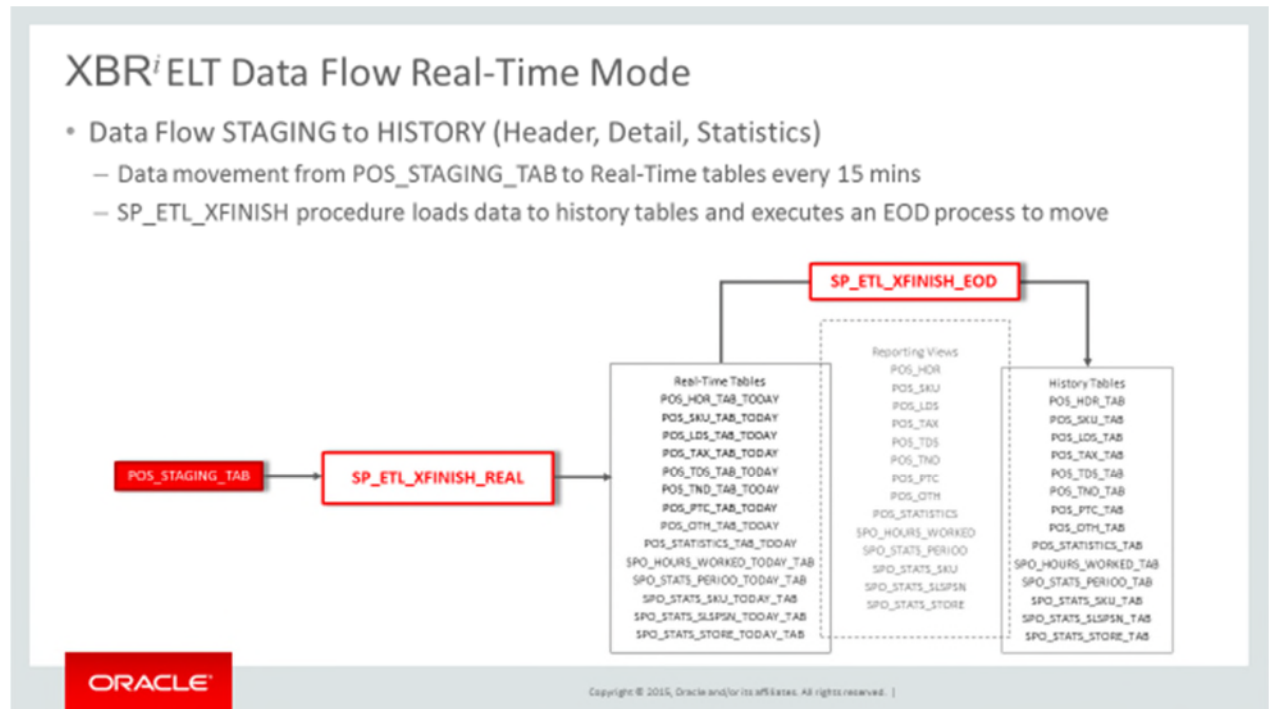
Flexfields

#	Steps Hierarchy	Enabled	Scenario/Variable	Restart
0	[-] root_step	<input checked="" type="checkbox"/>		Restart from failure
1	[-] Serial	<input checked="" type="checkbox"/>		Restart from failure
2	[-] XBR_GEN_SETUP	<input checked="" type="checkbox"/>	XBR_GEN_SETUP Version 001	Restart from new session
3	[-] SUSPEND_WEBSERVICE	<input checked="" type="checkbox"/>	SUSPEND_WEBSERVICE Version 001	Restart from new session
4	[-] XBR_XSTART	<input checked="" type="checkbox"/>	XBR_XSTART Version 001	Restart from new session
5	[-] XBR_GEN_GATHER	<input checked="" type="checkbox"/>	XBR_GEN_GATHER Version 001	Restart from new session
6	[-] Parallel	<input checked="" type="checkbox"/>		Restart from failed children
7	[-] XBR_GEN_STOREMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_STOREMST_LOAD Versi...	Restart from new session
8	[-] XBR_GEN_CURRENCYMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_CURRENCYMST_LOAD V...	Restart from new session
9	[-] XBR_GEN_CUSTMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_CUSTMST_LOAD Version...	Restart from new session
10	[-] XBR_GEN_REGMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_REGMST_LOAD Version ...	Restart from new session
11	[-] XBR_GEN_SKUMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_SKUMST_LOAD Version ...	Restart from new session
12	[-] XBR_GEN_EMPMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_EMPMST_LOAD Version ...	Restart from new session
13	[-] XBR_GEN_INVENTORY_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_INVENTORY_LOAD Versi...	Restart from new session
14	[-] XBR_GEN_TOKENMST_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_TOKENMST_LOAD Versi...	Restart from new session
15	[-] XBR_GEN_TRAFFIC_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_TRAFFIC_LOAD Version ...	Restart from new session
16	[-] XBR_GEN_GOAL_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_GOAL_LOAD Version 001	Restart from new session
17	[-] XBR_GEN_HOURSWORKED_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_HOURSWORKED_LOAD ...	Restart from new session
18	[-] XBR_TLOGA_1_LOAD	<input checked="" type="checkbox"/>	XBR_TLOGA_1_LOAD Version 001	Restart from new session
19	[-] XBR_TLOGA_2_LOAD	<input checked="" type="checkbox"/>	XBR_TLOGA_2_LOAD Version 001	Restart from new session
20	[-] XBR_TLOGB_1_LOAD	<input checked="" type="checkbox"/>	XBR_TLOGB_1_LOAD Version 001	Restart from new session
21	[-] XBR_GEN_POS_STAGING	<input checked="" type="checkbox"/>	XBR_GEN_POS_STAGING Version 0...	Restart from new session
22	[-] XBR_GEN_API_LOAD	<input checked="" type="checkbox"/>	XBR_GEN_API_LOAD Version 001	Restart from new session
23	[-] Serial	<input checked="" type="checkbox"/>		Restart from failure

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

XBRⁱ 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 XBRⁱ front end.

XBRI_LOADPLAN_REAL Validate

Definition

Steps

Exceptions

Variables

Privileges

Version

Flexfields

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

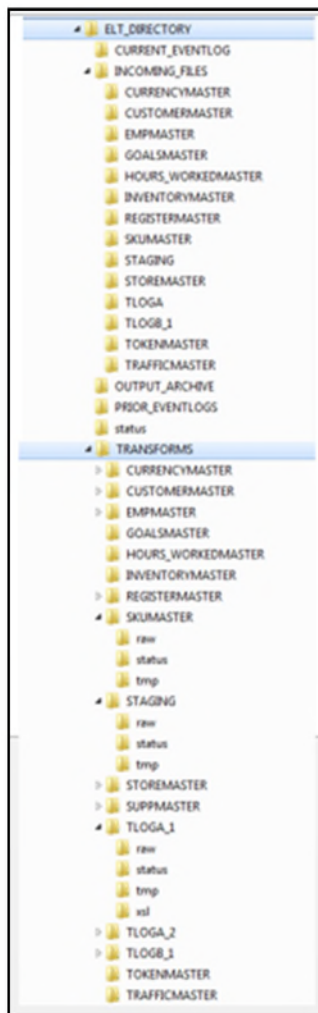
XBRI_LOADPLAN_REAL

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

XBRI_LOADPLAN_EOD

ELT Directory Structure

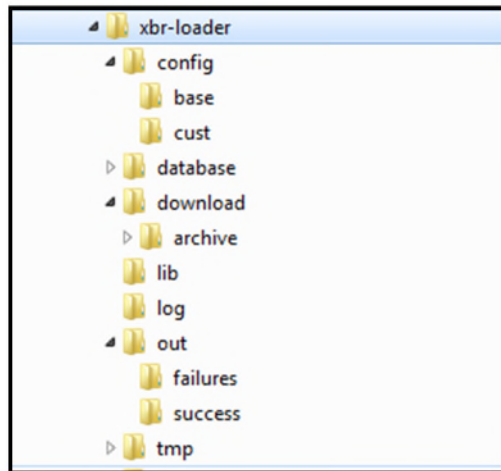
The workflow of the ODI ELT black box is dependent on the directory structure shown in the image that follows. INCOMING_FILES is the Landing area. There is a subdirectory for each core data feed to be loaded to the database. The CURRENT_EVENTLOG directory holds the log files generated by the ODI black 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

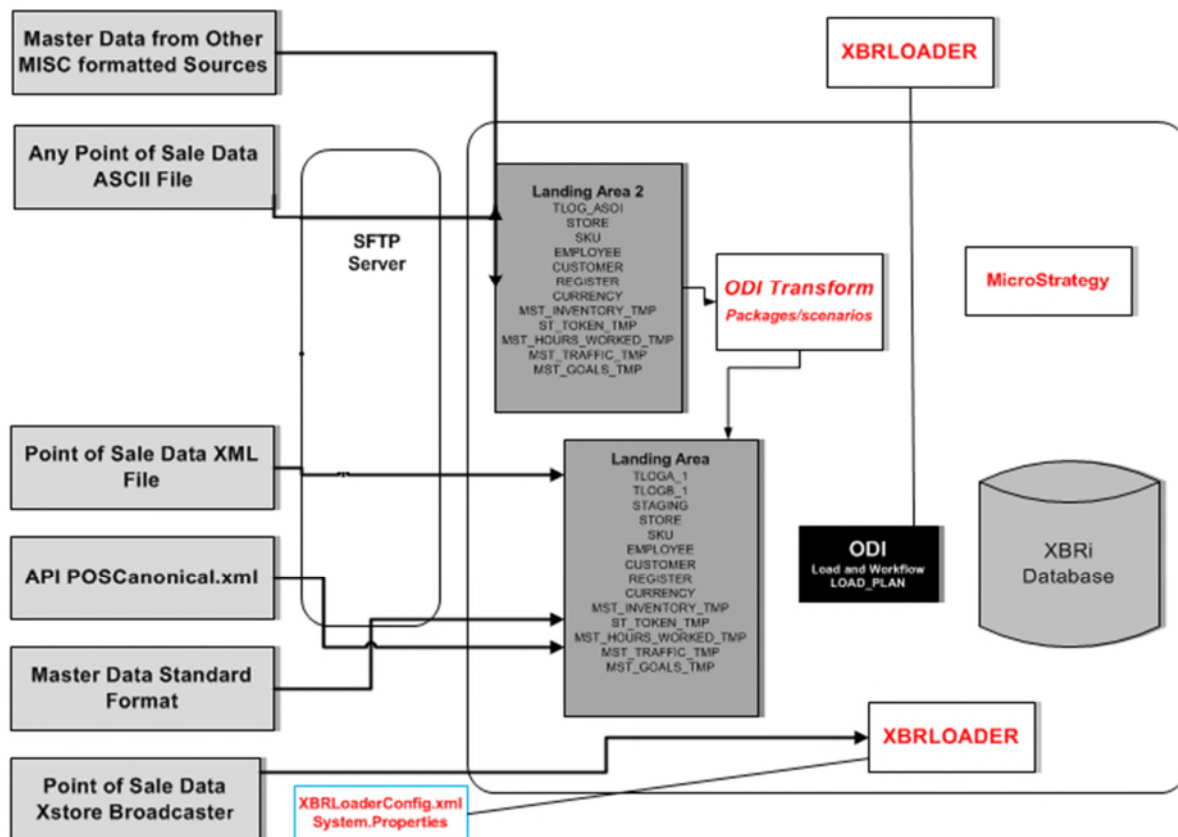
XBRLOADER Directory Structure

The following directory structure is only used for XSTORE-XBRⁱ web service implementations. This type of implementation is described in the [Data File Delivery Options](#) section and in [Chapter 6: Xstore/XBRⁱ Integration](#).

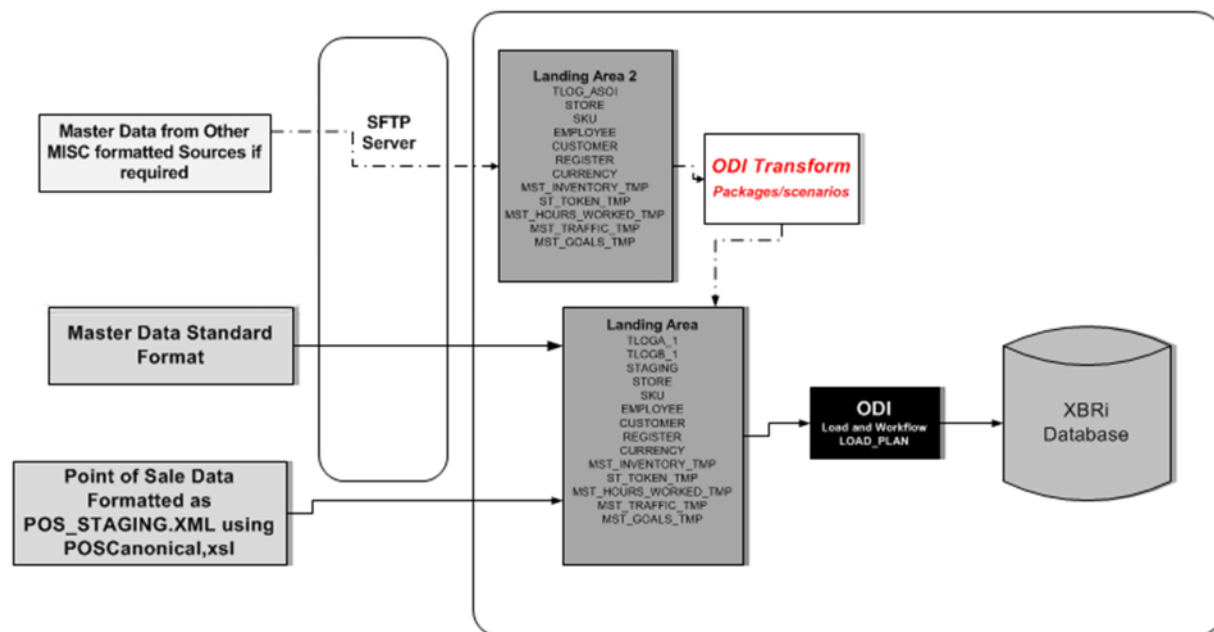


xbr-loader Directory Structure

Data File Delivery Options



Data File Delivery Options Diagram



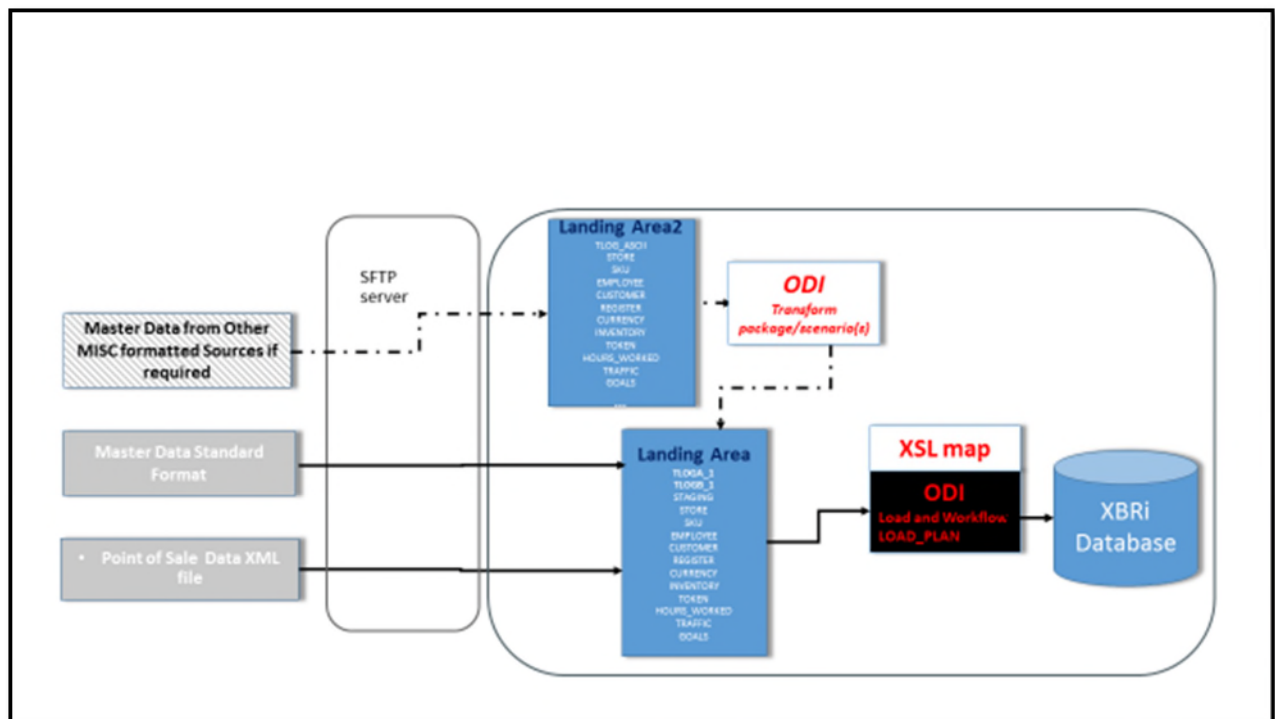
POSCanonical.xml API File Delivery Diagram

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.

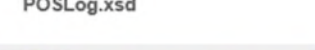


XML poslogs File Delivery Diagram

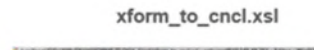
For point of sale transactional logs that are delivered as XML files (excluding XSTORE poslogs), you must create an XSL map that converts the customer's XML file to the XBRⁱ Canonical version defined by POSCanonical.xsd. The XSL map must be named: xform_to_cncl.xsl. [Chapter 4: Data File Delivery](#) 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.

• XSL map is required for Point of Sale transaction logs delivered as XML files


POSLog.xsd



xform_to_cncl.xsl



POSCanonical.xsd



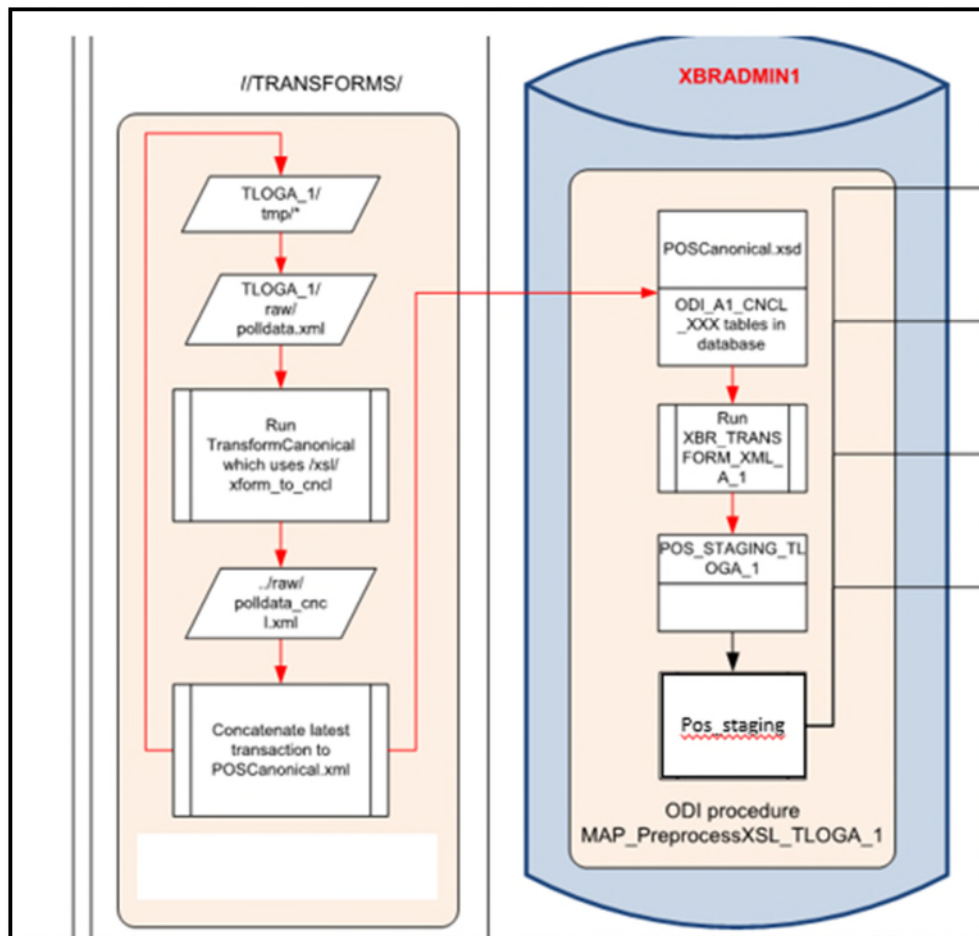
POS Logs for Incoming Files Landing Area

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 Schema.

ODI ELT – Steps

This section provides descriptions of all the steps performed by the ODI ELT process. For information on how to manage the settings for these steps, see the *Oracle Retail XBRi Cloud Services Administration Guide*.

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.

SUSPEND_WEBSERVICE

Suspends the XSTORE_XBRi web service from loading data to the pos_staging table.

XBR_XSTART

Executes the sp_xstart stored procedure in the database.

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_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_API_LOAD

Verifies that the import process is in the correct state, and prepares for the import process.

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_XFINISH_BATCH, XBR_GEN_XFINISH_REAL, XBR_GEN_XFINISH_EOD

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

RESUME_WEBSERVICE

Package that resumes the XSTORE-XBRi web service load to pos_staging.

SETSTATUS

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

Xstore/ XBRⁱ Integration

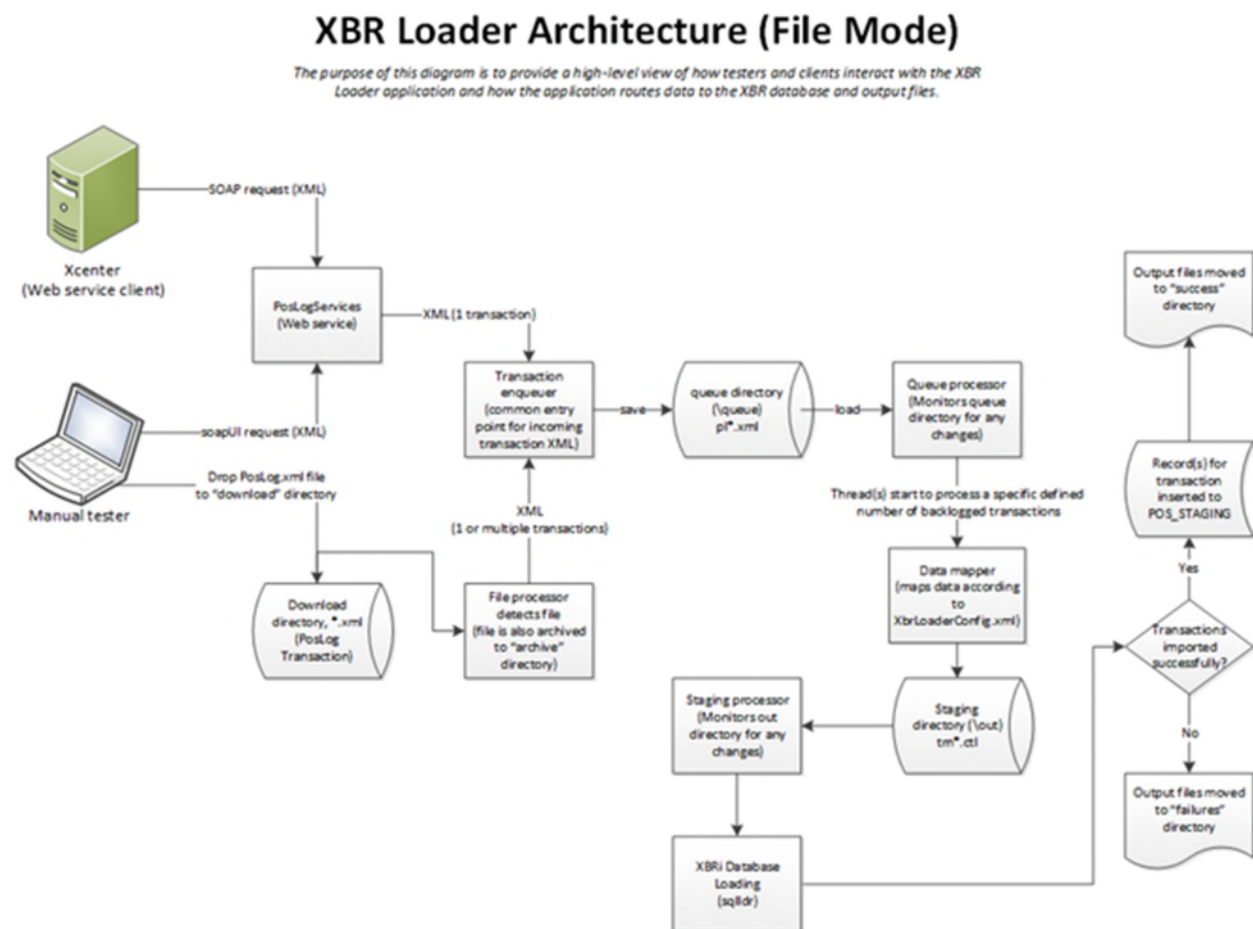
Introduction

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 XBRⁱ database where it is processed for use in an XBRⁱ 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 XBRⁱ 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.

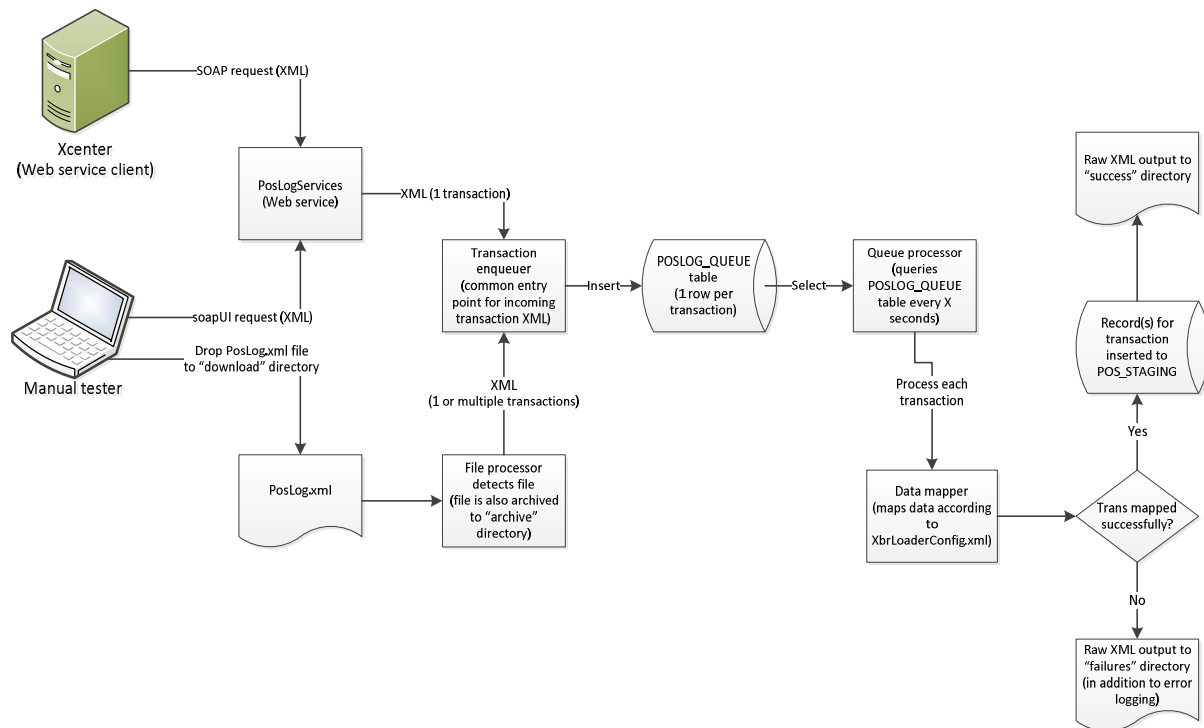
File Mode (Default)



Database Mode

XBR Loader Architecture

The purpose of this diagram is to provide a high-level view of how testers and clients interact with the XBR Loader application and how the application routes data to the XBR database and output files.



Components of an Xstore- XBRⁱ Web Service Integration

Xstore/Xcenter 6.5 or Later

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

XBRⁱ 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, XBRⁱ, and so on) through platform-independent technologies (such as Web services), as transactions flow through existing Xcenter servlet for persistence. The XBRⁱ 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.

XBRⁱ Database

There are two tables in the XBRⁱ 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

Schema to hold the individual transactions before they are transformed and loaded to the POS_STAGING table in XBRi.

POS_QUEUE_STATUS

Schema 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/xbr-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.

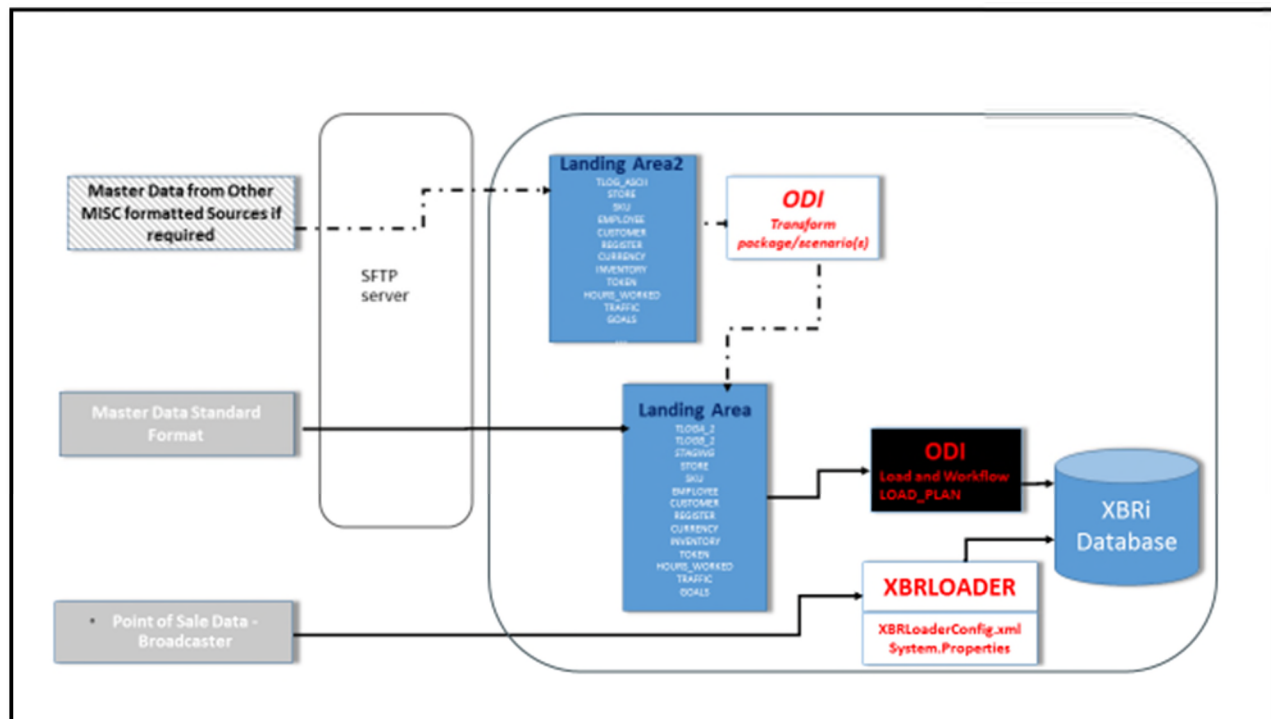
Note: Restarting the Tomcat service will automatically cause the queue processor to resume processing.

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.

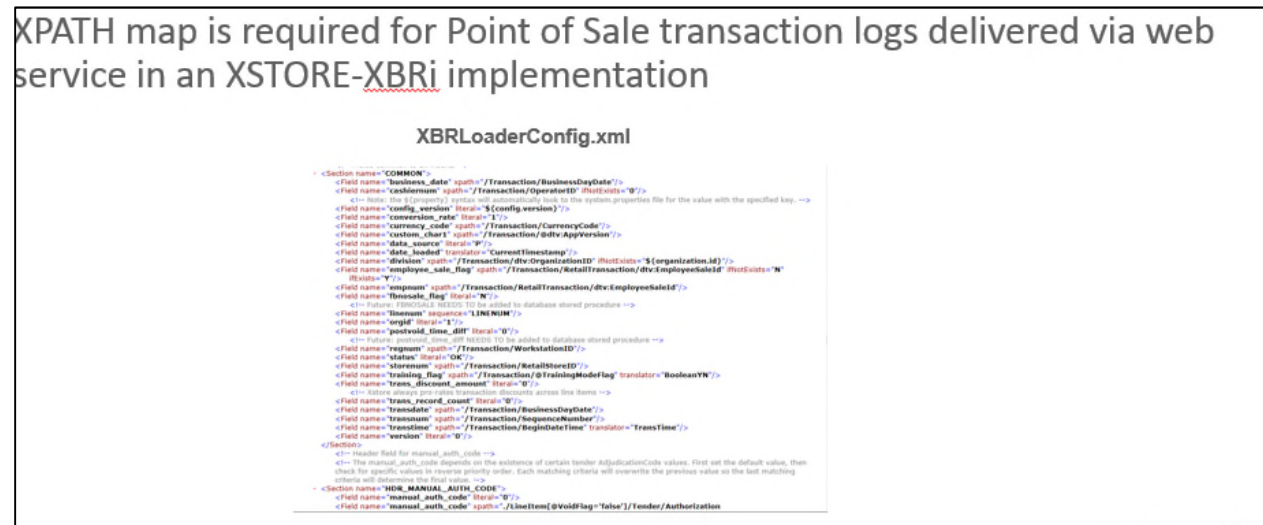
XSTORE poslogs through Web Service

XSTORE Poslogs delivered by the XCENTER Broadcaster to the XBRLOADER Web Service are transformed using a configuration. The coding is based on XPATH. The ODI black box framework handles the workflow. For Xstore - XBRⁱ 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-XBRi 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

Mapping Data through XPath

This section describes data mapping through XPath.

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/>

Web Services

Introduction

There are two styles of web service in v18.1

- Soap based web services
- REST based services

SOAP Based Web Services

You can analyse the SOAP based services by reviewing the WSDL, found on the ETL system with a relative path of: <https://<ETL system hostname>/xbr-loader/PosLogServices?wsdl>

XBR Loader SOAP-Based Web Services

You can use SOAP-Based web services to execute XBR Loader calls. The files in XBR-Loader's base/cust folders are not used if they have been executed using the web service APIs. You can confirm that API uploads were used by checking the data in ETL_Profile table in the XBRi Data Warehouse. The ETL_Profile table has an XPATH config base/cust column for any date being used. If APIs were used, the column will contain non-null fields.

The calls should be run from a Linux command or in a Windows sh environment.

Parameter Values

- <odi server> - Replace with the ODI
- <username> - Replace with XBRi user account
- <password> - Replacing with XBRi account

[data] – Replace with the data being uploaded. This must be converted with a escaping tool such as: <https://www.freeformatter.com/xml-escape.html#ad-output>

Example base configuration download:

```
curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml;charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:downloadBaseXbrLoaderConfig/>
  </soapenv:Body>
</soapenv:Envelope>' \
'https://<odi server>/xbr-loader/PosLogServices'
```

Example download customer configuration:

```
curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml;charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
```

```
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:downloadCustXbrLoaderConfig/>
  </soapenv:Body>
</soapenv:Envelope>' \
'https://<odi server>/xbr-loader/PosLogServices'
```

STOP XBR LOADER

```
curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml; charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:stopQueueProcessor/>
  </soapenv:Body>
</soapenv:Envelope>' \
'https://<odi server>/xbr-loader/PosLogServices'
```

START XBR LOADER

```
curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml; charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:startQueueProcessor/>
  </soapenv:Body>
</soapenv:Envelope>' \
'https://<odi server>/xbr-loader/PosLogServices'
```

Upload Customer Configuration:

```
curl -i -X PUT \
-H "SOAPAction:" \
-H "Content-Type:text/xml; charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:uploadCustXbrLoaderConfig>
      <custXbrLoaderConfig>[data]</custXbrLoaderConfig>
    </ws:uploadCustXbrLoaderConfig>
  </soapenv:Body>
</soapenv:Envelope>' \
'https://<odi server>/xbr-loader/PosLogServices'
```

Upload a New Configuration:

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
  '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ws="http://ws.xbr.dtv/">
    <soapenv:Header/>
    <soapenv:Body>
      <ws:uploadBaseXbrLoaderConfig>
        <baseXbrLoaderConfig>[data]</baseXbrLoaderConfig>
      </ws:uploadBaseXbrLoaderConfig>
    </soapenv:Body>
  </soapenv:Envelope>
  '\
  'https://<odi server>/xbr-loader/PosLogServices'
```

Download XPATH Config – Customer Override

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ws="http://ws.xbr.dtv/">
    <soapenv:Header/>
    <soapenv:Body>
      <ws:downloadCustXbrLoaderConfig/>
    </soapenv:Body>
  </soapenv:Envelope>
```

Upload XPATH Config – Base

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ws="http://ws.xbr.dtv/">
    <soapenv:Header/>
    <soapenv:Body>
      <ws:uploadBaseXbrLoaderConfig>
        <baseXbrLoaderConfig>?DATA?</baseXbrLoaderConfig>
      </ws:uploadBaseXbrLoaderConfig>
    </soapenv:Body>
  </soapenv:Envelope>
  --?DATA? would need swapping out for XPATH configuration. Because it is XML data, it must be
  escaped, for example: https://www.freeformatter.com/xml-escape.html#ad-output
```

Update XPATH Config – Customer Override

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
```

```

-d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:uploadCustXbrLoaderConfig>
      <custXbrLoaderConfig>?DATA?</custXbrLoaderConfig>
    </ws:uploadCustXbrLoaderConfig>
  </soapenv:Body>
</soapenv:Envelope>
--?DATA? would need swapping out for XPATH configuration. Because it is XML data, it must be
escaped, for example: https://www.freeformatter.com/xml-escape.html#ad-output

```

Stop XBR Loader

```

curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml; charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:stopQueueProcessor/>
  </soapenv:Body>
</soapenv:Envelope>

```

Start XBR Loader

```

curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml; charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:startQueueProcessor/>
  </soapenv:Body>
</soapenv:Envelope>

```

Suspend or Resume XBR Loader

```

curl -i -X POST \
-H "SOAPAction:" \
-H "Content-Type:text/xml; charset=UTF-8" \
-H "javax.xml.ws.security.auth.username:<username>" \
-H "javax.xml.ws.security.auth.password:<password>" \
-d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:suspendQueueProcessor>
      <argSuspend>false</argSuspend>
    </ws:suspendQueueProcessor>
  </soapenv:Body>
</soapenv:Envelope>

```

--**false** would need swapping out for **true** or **false**

Suspend or Resume XBR Loader Staging Only

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:suspendStagingProcessor>
      <argSuspend>false</argSuspend>
    </ws:suspendStagingProcessor>
  </soapenv:Body>
</soapenv:Envelope>
--false would need swapping out for true or false
```

Is XBR Loader Stopped?

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:isQueueProcessorStopped/>
  </soapenv:Body>
</soapenv:Envelope>
```

Is XBR Loader Suspended?

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:isQueueProcessorSuspended/>
  </soapenv:Body>
</soapenv:Envelope>
```

Is XBR Loader Staging Suspended?

```
curl -i -X POST \
  -H "SOAPAction:" \
  -H "Content-Type:text/xml;charset=UTF-8" \
  -H "javax.xml.ws.security.auth.username:<username>" \
  -H "javax.xml.ws.security.auth.password:<password>" \
  -d \
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.xbr.dtv/">
```

```
<soapenv:Header/>
<soapenv:Body>
  <ws:isStagingProcessorSuspended/>
</soapenv:Body>
</soapenv:Envelope>
```

Additional XBR Loader Services

In addition to those above, the following services, which perform similar functions are available:

downloadBaseXbrLoaderConfig - Download the base system XBR-LOADER XPATH mapping configuration. Returns the XPATH configuration in the downloadBaseXbrLoaderConfigResponse tag.

downloadXbrLoaderConfig - Download the XBR-LOADER XPATH mapping configuration, takes one parameter base, which can be true/false to say you want the base or customer configurations.

uploadXbrLoaderConfig - Upload the XBR-LOADER XPATH mapping configuration, within the body of the upload requires base - true/false, xbrLoaderConfig - XML XPATH configuration.

SOAP-Based Web Services for POSLOG, Queue Processor, and Staging Processor

downloadPOSLogSchemas - Download any configured XSDs for the XBR-LOADER processing. The response provides two tags, posLogSchema and posLogDTVSchema. The posLogSchema is the primary/base XSD. posLogDTVSchema is the optional child schema. The XStore import uses this secondary schema for the DTV add-ons.

uploadPOSLogSchemas - Provides the ability to associate an XML XSD

isQueueProcessorStopped - Checks to see if the POSLOG queue processing is stopped.

isQueueProcessorSuspended - Checks to see if the POSLOG queue is suspended

isStagingProcessorSuspended - Checks to see if the Staging processor is suspended

startQueueProcessor - Start the queue processor.

stopQueueProcessor - Stop the queue processor.

suspendQueueProcessor - Suspend the queue processor. Takes one parameter, argSuspend which can be true/false.

suspendStagingProcessor - Suspend the staging processor. Takes one parameters, argSuspend which can be true/false.

processPosLog - Submit a POSLOG into the XBRi system. Takes one parameter under the tag name, argPosLog which is the POSLOG XML schema. Keep in mind that submitting this data is best done via a CDATA block as the data is XML inside XML. The inner XML should really be encoded to allow it to fit within an XML tag. The response provides the number of POSLOG transactions it found in the request body. If zero, the data couldn't be processed.

Header Properties

In order to use these web service methods, two header properties are required in order to authenticate into the service. These settings are configurable so they may change, but out of the box they are:

javax.xml.ws.security.auth.username - header property that would be set with the services username.

javax.xml.ws.security.auth.password - header property that would be set with the services password.

XBR Loader Stages

The XBR Loader breaks out into two stages:

-
1. **Queue** - Every time a user submits a POSLOG into the system, the submission gets placed in this queue. From there the queue entries will be picked up, validated and then transformed based on any pre-defined XSD or XPATH configuration. To note, the XBR-LOADER application has built in validation so doesn't need an XSD to be defined. If one is supplied it will load that and use it. Further note, if an XSD is provided instead of using the internal validation it will slow down the processing via a small amount due to the additional time needed to perform the external validation. Once the validation and transformations have occurred, they output is placed on a staging queue. **Note:** the conversion of the raw data into the staging format is a very CPU intensive process, and may impact performance
 2. **Staging Queue** - Once the data gets into the staging queue the hard work is done, the data pending here is pushed into the data storage when and only when the storage is ready to accept it.

With the above information, the web service commands offer two types of suspend. Queue suspend and Staging suspend. The XBRi application uses the staging suspended service calls to stop data from coming into the data storage while it's busy doing some maintenance. Blocking only the staging allows the hard work done by the queue to continue. If you suspend the queue, the staging is also suspended. This effectively stops the entire chain of processing. Any further submissions will simply build up on the queue until such as time as the processing is resumed.

If any configuration changes are made to the XSD or XPATH configuration, the changes won't be picked up until a call to the stopQueueProcessor & startQueueProcessor is called.

REST Based Web Services

Before you are allowed to use any of the defined REST services, you must first log in.

Logging into REST Web Services

The Login web service is REST based and requires the following:

Protocol -HTTPS

Method - GET

URL - <XBRi Application system hostname>/analytics/rest/ext/v1/login?project=XBRI

Headers - **Authorization** – Basic <Base64 Encoded username:password>

The username and password are from an XBRi account. The username and password must be delimited by a colon before encoding with base64.

Response - Assuming a HTTP 200 status comes back, the response body will contain one of two things: If the user details are invalid, the message: "Error connecting to object" Or, a plain text version of the access token. Any further communications with XBRi web services will require this token to verify access.

Getting back an HTTP 500 generally means the format of the submission is incorrect, or there's a problem with the server.

Available Rest Based WebServices

Delete an ETL imported staging batch

Get an XSD from the ODI system hierarchy

Replace an XSD from the ODI system hierarchy

Delete a ETL imported staging batch

This rest service requires the following information:

Protocol - HTTPS

Method - DELETE

URL - <XBRI ETL system hostname>/xbr-odi/v1/batch/<batch#>

Where the <batch#> will be replaced with a numeric batch number previously submitted.

Headers - xbri-mst-token - <token>

The token that was generated during login.

Response

HTTP 200 - status comes back meaning the request was successful.

HTTP 401, 403 - status means that access was denied or the token given has expired.

HTTP 500 - problem occurred on the server, contact support is the problem persists.

Result Example: {"success": true, "serverOutput": "...audit of what occurred during delete process..."}

Get an XSD from the ODI system hierarchy

This rest service requires the following information:

Protocol - HTTPS

Method - GET

URL - <XBRI ETL system hostname>/xbr-odi/v1/xsd/<logType>

Where <logType> will be replaced by the ODI type, available options are:

TLOGA_1

TLOGA_2

TLOGB_1

Headers

xbri-mst-token - <token>

The token that was generated during login.

filename - Optional parameter, if not provided will assume POSCanonical.xsd

It can only be one of the following POSCanonical.xsd or POSCanonical.validate.xsd

Anything else will be rejected.

Accept - Optional parameter stating what format the output should be written in. If not provided, will assume "application/json". The other available option is "application/xml".

Note: The formatting of the output will depend on this option. There may be some conversion needed depending on the output chosen.

Response

HTTP 200 - status means the contents of the request will be placed in the response body.

HTTP 401, 403 - status means that access was denied or the token given has expired.

HTTP 500 - problem occurred on the server, contact support is the problem persists.

Body - The response will appear in a downloadXSResponse -> xsd tag if the format expected in XML, for JSON the root tag is skipped and the contents appear under the xsd tag.

Result Example (JSON):

{"xsd": "...XSD contents..."}

Replace an XSD from the ODI system hierarchy

This rest service requires the following information:

Protocol - HTTPS

Method - POST

URL - <XBRI ETL system hostname>/xbr-odi/v1/xsd/<logType>

Where <logType> will be replaced by the ODI type, available options are:

TLOGA_1

TLOGA_2

TLOGB_1

Headers

xbri-mst-token - <token> - The token that was generated during login.

filename - Optional parameter, if not provided will assume POSCanonical.xsd

It can only be one of the following POSCanonical.xsd or POSCanonical.validate.xsd

Anything else will be rejected.

Body - The body contents of the POST will be the XSD contents, there are no wrapper tags around it.

Response

HTTP 200 - status means the contents of the request will be placed in the response body.

HTTP 401, 403 - status means that access was denied or the token given has expired.

HTTP 500 - problem occurred on the server, contact support if the problem persists.