Prepared by: Buzz IT Company Limited

Author: Steven Chen

Date: 20/01/2017

Version: 0.9

Maxim’s POS Polling Enterprise Service Bus Implementation Service

**EL-FY16-902**

**Sales/Master/Pricing Data Exchange**

System Design Specification

# Document Control

## Document History

| Version | Date | Author | Revision Remark |
| --- | --- | --- | --- |
| 0.93 | 20/01/2017 | Steven Chen | 1st draft |

## Document/Design Owner

| Name | Title |
| --- | --- |
| Steven Chen | System Analyst |
| Edward Leung | System Analyst |
| Tommy Leung | System Analyst |

## Key Comments

| Name/Title |  | |
| --- | --- | --- |
| # | Comments |
| Comment | 1 |  |
| Response | 1 |  |

# Table of Content

[Document Control 2](#_Toc471750729)

[Document History 2](#_Toc471750730)

[Document/Design Owner 2](#_Toc471750731)

[Key Comments 2](#_Toc471750732)

[Table of Content 3](#_Toc471750733)

[1 Background 5](#_Toc471750734)

[1.1 Document Purpose 5](#_Toc471750735)

[1.2 Document Scope 5](#_Toc471750736)

[1.3 Document Audience 5](#_Toc471750737)

[1.4 Terms & Abbreviations 6](#_Toc471750738)

[1.5 Reference Materials 6](#_Toc471750739)

[2 Executive Summary 7](#_Toc471750740)

[3 Architecture Design 8](#_Toc471750741)

[3.1 Overall Architecture 8](#_Toc471750742)

[4 Data Model Design 9](#_Toc471750743)

[4.1 POS Client Data Source Data Model 9](#_Toc471750744)

[4.2 EDW Data Model 11](#_Toc471750745)

[4.3 Pricing/Master Data Model 12](#_Toc471750746)

[4.4 Staging Table Data Model 13](#_Toc471750747)

[4.5 Polling Application Work Tables 15](#_Toc471750748)

[5 High Level Interface Design 17](#_Toc471750749)

[6 Functional Design 18](#_Toc471750750)

[6.1 System Functional Overview 18](#_Toc471750751)

[6.2 Branch Info Maintenance 22](#_Toc471750752)

[6.3 Scheduler Jobs Maintenance 29](#_Toc471750753)

[6.4 Branch Scheme Maintenance 34](#_Toc471750754)

[6.5 Job Logs View 47](#_Toc471750755)

[6.6 Sales Data Real Time Polling Batch Job 48](#_Toc471750756)

[6.7 Sales Data EOD Polling Batch Job (POS-Staging) 53](#_Toc471750757)

[6.8 Pricing/Master Data Generation Batch Job (Pricing Server - Staging) 56](#_Toc471750758)

[6.9 Pricing/Master Data Distribution Batch Job (Staging-POS) 58](#_Toc471750759)

[7 Interface Design 60](#_Toc471750760)

[7.1 Sales Data Web Service 60](#_Toc471750761)

[7.2 Pricing Data Web Service 72](#_Toc471750762)

[7.3 EDW Checksum Web Service 74](#_Toc471750763)

# Background

## Document Purpose

The purpose of the System Design Specification (SDS) is to describe the detailed system design specification for a project and its main aim is to provide system design context for the project and its objectives. It will provide the input for high-level development activities.

The System Design Specification is part of the deliverables in the Business Case Development phase of Project Delivery Lifecycle.

## Document Scope

The scope of the System Design Specification (SDS) is to describe the architectural view of the system. It has section such as architecture design, data model design, high-level interface design, report design, etc. The technical designs and specifications of the impacted applications are not included in this document.

## Document Audience

The audience of this System Design Specification (SDS) is the technical staff of the IT department of the project owner.

## Terms & Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| ESB | Enterprise Service Bus |
| API | Application Programming Interface |
| EDW | Enterprise Data Warehouse |
| EOD | End of Day |
| POS Client | One Database Owner on the Sales side, provided by the POS machine vendor/manufacturer |
| DB | Database |
| POS | Point of Sales |
| Staging DB | The service bus database to stage the polling data |

## Reference Materials

| Document Names |
| --- |
| Maxim’s POS Polling ESB Implementation Service Proposal EL-FY16-902-v3.docx |
| POS Polling User Requirement Confirmation-20161121-Discussion Note.xlsx |

# Executive Summary

The aim of the ESB project is to migrate three data processing flows in Maxim’s current enterprise architecture using database provided technologies (linked server & stored procedure) to a new platform using Oracle Enterprise Services Bus technology. In the new ESB polling system (ESB system), it will have three data processing flows implemented.

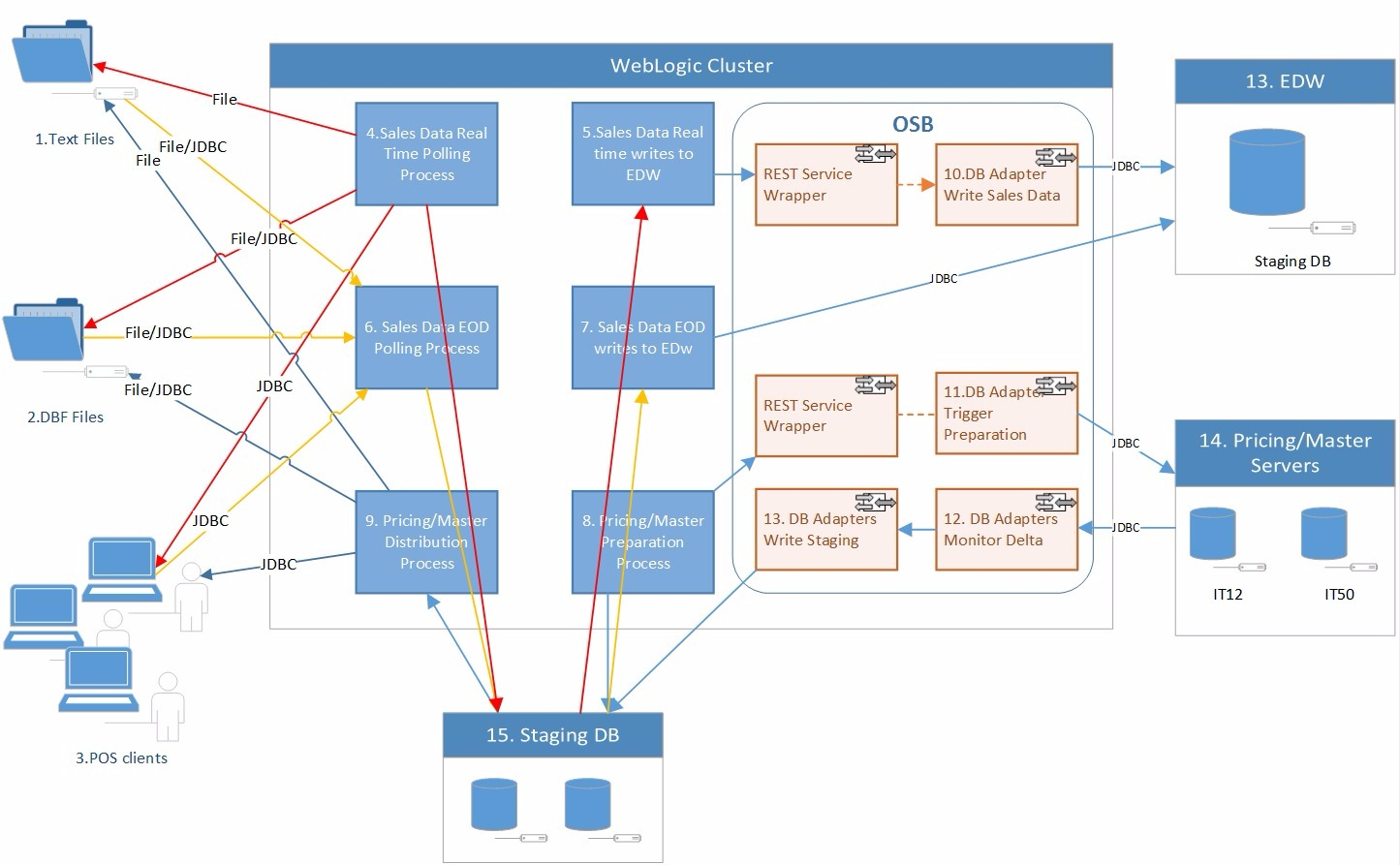
* Sales data real time processing to EDW
* Sales data EOD processing to EDW
* Synchronize master data to POS clients (e.g. Pricing/Master)

The ESB system will use JDBC to connect to the databases of existing POS clients in outlets by pre-configured connection information in order to collect sales data from the existing POS systems and update the pricing/master data back to the existing POS clients. Referring to the polling logic found in the production IT51 server, for any connection error, the ESB system will log down the error and retry data synchronization. After reaching maximum retry count, the ESB system will halt the synchronization for that particular POS client and generate alert to related parties for follow up.

All POS clients’ connection settings are configurable and maintainable by Maxim’s IT. It makes the adding of new POS client easy and no alteration of programming code is required. The ESB system can invoke several configurable concurrent threads to poll/push the data to/from POS clients concurrently for maximizing system performance.

# Architecture Design

## Overall Architecture



Dick:

1. any interface to synchronize the data between staging DB to existing POS Polling server?
2. For pricing/master data, what is DB adaptor trigger preparation and why it is triggered by REST service wrapper?
3. Is it too complicate to use DB Adaptor to write the staging DB, especially for **employee** table [w/o delta flag]?
4. What is the scheduler used to trigger the process? Weblogic library or crontab?

**Key Data Sources**

1. Sales Data text File
2. Sales Data DBF files
3. Sales Data POS Client SQL Server Database

13. EDW Database

14. Pricing/Master Database

15. Staging Database

Dick??? Existing POS Polling DB (for sales data synchronization)

**Key Process**

1. Sale Data Real Time Polling (POS -> Staging)
2. Sales Data Real Time to EDW (Staging -> EDW, depend on #4 completion)
3. Sales Data EOD (POS -> Staging)
4. Sales Data EOD to EDW (Staging –> EDW, depend on #6 completion)
5. Pricing/Master Generation/Preparation (Pricing/Master servers -> Staging)
6. Pricing/Master Distribution (Staging –> POS clients, depending on #8 completion)

Dick??? Staging DB to existing POS Polling servers (IT5x) for interim data synchronization

**Key OSB components**

1. Write EDW tables service (Virtual Branch Sales Inbound tables)
2. Pricing generation service (trigger SP udsp\_gen\_pricing\_group)
3. DB Adapter monitoring pricing/master tables delta
4. DB Adapter write Pricing/Master data to staging (triggered by #12)

# Data Model Design

## POS Client Data Source Data Model

Data Model refer to existing POS Client Database Tables as follows:

| **Table** | **Type** | **Update Freq.** | **Up/Down** | **Source** |
| --- | --- | --- | --- | --- |
| ACCOUNTS | Master | Every Polling | Download | Carl |
| COUPON\_CONTROL | Master | Every Polling | Download | Carl |
| COUPON\_RANGE | Master | Every Polling | Download | Carl |
| CURRENCY | Master | Every Polling | Download | Carl |
| EMPLOYEE | Master | Every Polling | Download | Carl |
| HIST\_CHECK\_LOGS | HIST | EOD | Upload | Carl |
| HIST\_COUPON\_SALES | HIST | EOD | Upload | Carl |
| HIST\_ITEM | HIST | EOD | Upload | Carl |
| HIST\_ITEMSTOCK | HIST | EOD | Upload | Carl |
| HIST\_ORDERS | HIST | EOD | Upload | Carl |
| HIST\_ORDERS\_EXTRA | HIST | EOD | Upload | Carl |
| HIST\_ORDERS\_PAY | HIST | EOD | Upload | Carl |
| HIST\_ORDERS\_PAY\_PROGRESS | HIST | EOD | Upload | Carl |
| HIST\_PAYFIG | HIST | EOD | Upload | Carl |
| HIST\_PAYSUM | HIST | EOD | Upload | Carl |
| HIST\_POSSYSTEM | HIST | EOD | Upload | Carl |
| HIST\_REDEEMED\_COUPON | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXCHECK | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXCHECKTENDER | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXINOUT | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXINOUTEXTENDINFO | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXPICKUP | HIST | EOD | Upload | Carl |
| HIST\_SESSIONINFO | HIST | EOD | Upload | Carl |
| HIST\_SESSIONTENDER | HIST | EOD | Upload | Carl |
| HIST\_STOCK\_MOVEMENT | HIST | EOD | Upload | Carl |
| HIST\_SUPP | HIST | EOD | Upload | Carl |
| HIST\_TRANS | HIST | EOD | Upload | Carl |
| HIST\_TRANS\_ECARD | HIST | EOD | Upload | Carl |
| HIST\_TRANS\_MODIFIER | HIST | EOD | Upload | Carl |
| INVITATION | Master | Every Polling | Download | Carl |
| ITEM | Master | Every Polling | Download | Carl |
| ITEM\_BARCODE | Master | Every Polling | Download | Carl |
| ITEM\_MODIFIER | Master | Every Polling | Download | Carl |
| ITEMANLY | Master | Every Polling | Download | Carl |
| ITEMDEPT | Master | Every Polling | Download | Carl |
| MENU | Master | Every Polling | Download | Carl |
| MENUITEM | Master | Every Polling | Download | Carl |
| MESSAGES | Master | Every Polling | Download | Carl |
| MODIFIER | Master | Every Polling | Download | Carl |
| MODIFIER\_GRP | Master | Every Polling | Download | Carl |
| MODIFIER\_LIST | Master | Every Polling | Download | Carl |
| ONHOUSE | Master | Every Polling | Download | Carl |
| OPTIONS | Master | Every Polling | Download | Carl |
| ORDERS | Sales Data | Every Polling | Upload | Carl |
| ORDERS\_EXTRA | Sales Data | Every Polling | Upload | Carl |
| ORDERS\_PAY | Sales Data | Every Polling | Upload | Carl |
| PAYCAT | Master | Every Polling | Download | Carl |
| PAYMENT | Master | Every Polling | Download | Carl |
| PMT\_ACTION | Master | Every Polling | Download | Carl |
| PMT\_CONDITION | Master | Every Polling | Download | Carl |
| PMT\_HDR | Master | Every Polling | Download | Carl |
| POSBUSDATE | Master | Every Polling | Download | Carl |
| ROLE | Master | Every Polling | Download | Carl |
| ROLE\_PERMISSION | Master | Every Polling | Download | Carl |
| SUPP | Sales Data | Every Polling | Upload | Carl |
| SYSSETTINGS | Master | Every Polling | Download | Carl |
| TRANS | Sales Data | Every Polling | Upload | Carl |
| TRANS\_ECARD | Sales Data | Every Polling | Upload | Carl |
| TRANS\_MODIFIER | Sales Data | Every Polling | Upload | Carl |
| TRANS\_TYPE | Master | Every Polling | Download | Carl |
| USER\_PERMISSION | Master | Every Polling | Download | Carl |
| USER\_ROLE | Master | Every Polling | Download | Carl |
| WIFI CODE | Master | Every Polling | Download | Carl |

## EDW Data Model

Data Model refer to existing EDW Database Tables as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tables | Source | Matched with Polling Tables | Remark |
| HIST\_COUPON\_SALES | EDW | HIST\_COUPON\_SALES | EOD |
| HIST\_ITEM | EDW | HIST\_ITEM | EOD |
| HIST\_ITEMSTOCK | EDW | HIST\_ITEMSTOCK | EOD |
| HIST\_ORDERS | EDW | HIST\_ORDERS | EOD |
| HIST\_ORDERS\_PAY | EDW | HIST\_ORDERS\_PAY | EOD |
| HIST\_PAYFIG | EDW | HIST\_PAYFIG | EOD |
| HIST\_PAYSUM | EDW | HIST\_PAYSUM | EOD |
| HIST\_POSSYSTEM | EDW | HIST\_POSSYSTEM | EOD |
| HIST\_REDEEMED\_COUPON | EDW | HIST\_REDEEMED\_COUPON | EOD |
| HIST\_SUPP | EDW | HIST\_SUPP | EOD |
| HIST\_TRANS | EDW | HIST\_TRANS | EOD |
| HIST\_TRANS\_ECARD | EDW | HIST\_TRANS\_ECARD | EOD |
| HIST\_TRANS\_MODIFIER | EDW | HIST\_TRANS\_MODIFIER | EOD |
| ITEM | EDW | ITEM | Master |
| TRANS\_TYPE | EDW | TRANS\_TYPE | Master |
| COUPON\_SALES | EDW | COUPON\_SALES | Real Time |
| ITEMSTOCK | EDW | ITEMSTOCK | Real Time |
| ORDERS | EDW | ORDERS | Real Time |
| ORDERS\_PAY | EDW | ORDERS\_PAY | Real Time |
| SUPP | EDW | SUPP | Real Time |
| TRANS | EDW | TRANS | Real Time |
| TRANS\_ECARD | EDW | TRANS\_ECARD | Real Time |
| TRANS\_MODIFIER | EDW | TRANS\_MODIFIER | Real Time |

Dick??? Real time data only handle the ORDERS, ORDER\_PAY and TRANS only. Do we need to synchronize SUPP or TRANS\_ECARD in **real time**? If yes, how to do it, using ESB message or JDBC copy or other?

Dick??? EDW receive data at either EOD or Real Time? Does ‘Master’ means real time? If yes, who will synchronize this data?

## Pricing/Master Data Model

Among the POS client data model, the following tables belongs to master table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Type** | **Update Freq.** | **Up/Down** | **Source** |
| ACCOUNTS | Master | Every Polling | Download | Carl |
| COUPON\_CONTROL | Master | Every Polling | Download | Carl |
| COUPON\_RANGE | Master | Every Polling | Download | Carl |
| CURRENCY | Master | Every Polling | Download | Carl |
| EMPLOYEE | Master | Every Polling | Download | Carl |
| INVITATION | Master | Every Polling | Download | Carl |
| ITEM | Master | Every Polling | Download | Carl |
| ITEM\_BARCODE | Master | Every Polling | Download | Carl |
| ITEM\_MODIFIER | Master | Every Polling | Download | Carl |
| ITEMANLY | Master | Every Polling | Download | Carl |
| ITEMDEPT | Master | Every Polling | Download | Carl |
| MENU | Master | Every Polling | Download | Carl |
| MENUITEM | Master | Every Polling | Download | Carl |
| MESSAGES | Master | Every Polling | Download | Carl |
| MODIFIER | Master | Every Polling | Download | Carl |
| MODIFIER\_GRP | Master | Every Polling | Download | Carl |
| MODIFIER\_LIST | Master | Every Polling | Download | Carl |
| ONHOUSE | Master | Every Polling | Download | Carl |
| OPTIONS | Master | Every Polling | Download | Carl |
| PAYCAT | Master | Every Polling | Download | Carl |
| PAYMENT | Master | Every Polling | Download | Carl |
| PMT\_ACTION | Master | Every Polling | Download | Carl |
| PMT\_CONDITION | Master | Every Polling | Download | Carl |
| PMT\_HDR | Master | Every Polling | Download | Carl |
| POSBUSDATE | Master | Every Polling | Download | Carl |
| ROLE | Master | Every Polling | Download | Carl |
| ROLE\_PERMISSION | Master | Every Polling | Download | Carl |
| SYSSETTINGS | Master | Every Polling | Download | Carl |
| TRANS\_TYPE | Master | Every Polling | Download | Carl |
| USER\_PERMISSION | Master | Every Polling | Download | Carl |
| USER\_ROLE | Master | Every Polling | Download | Carl |
| WIFI CODE | Master | Every Polling | Download | Carl |

## Staging Table Data Model

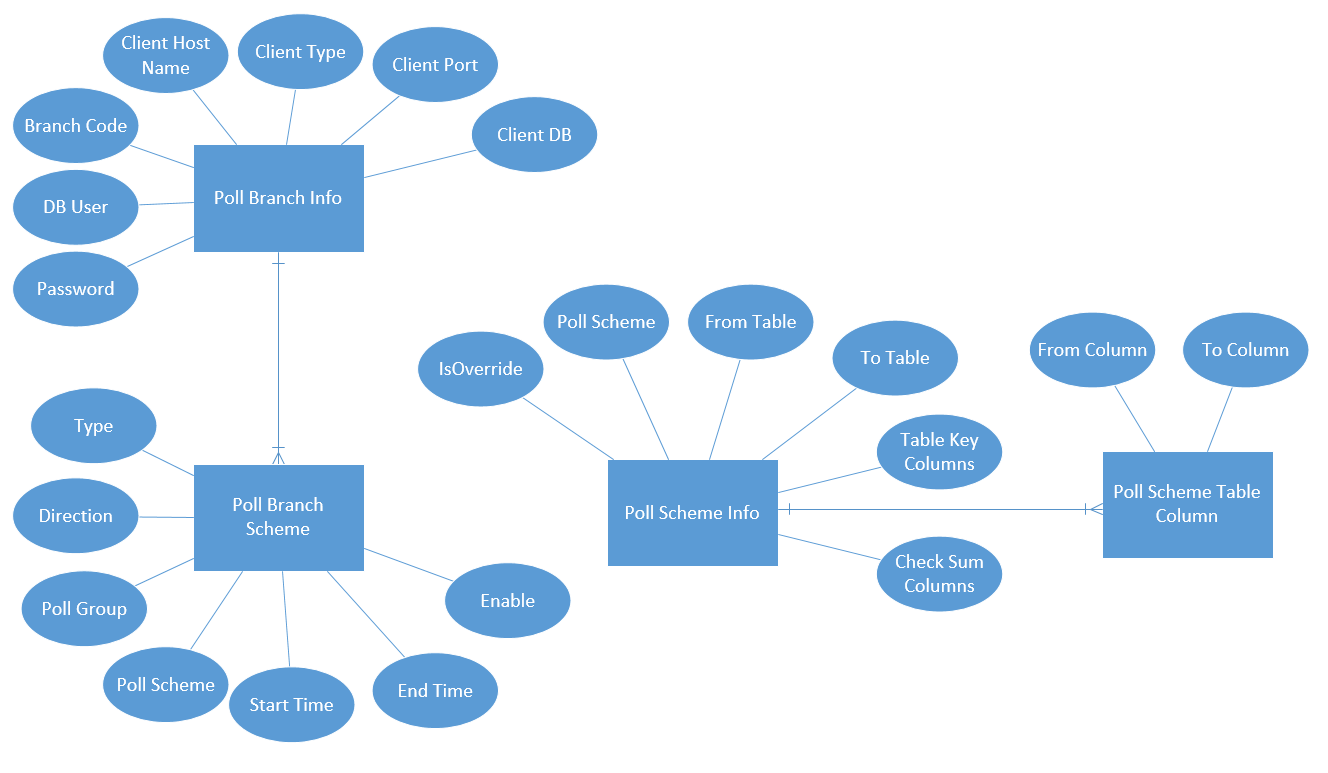
Data Model refer to existing POS Client Database Tables as follows:

| **Table** | **Type** | **Update Freq.** | **Up/Down** | **Source** |
| --- | --- | --- | --- | --- |
| ACCOUNTS | Master | Every Polling | Download | Carl |
| COUPON\_CONTROL | Master | Every Polling | Download | Carl |
| COUPON\_RANGE | Master | Every Polling | Download | Carl |
| CURRENCY | Master | Every Polling | Download | Carl |
| EMPLOYEE | Master | Every Polling | Download | Carl |
| HIST\_CHECK\_LOGS | HIST | EOD | Upload | Carl |
| HIST\_COUPON\_SALES | HIST | EOD | Upload | Carl |
| HIST\_ITEM | HIST | EOD | Upload | Carl |
| HIST\_ITEMSTOCK | HIST | EOD | Upload | Carl |
| HIST\_ORDERS | HIST | EOD | Upload | Carl |
| HIST\_ORDERS\_EXTRA | HIST | EOD | Upload | Carl |
| HIST\_ORDERS\_PAY | HIST | EOD | Upload | Carl |
| HIST\_ORDERS\_PAY\_PROGRESS | HIST | EOD | Upload | Carl |
| HIST\_PAYFIG | HIST | EOD | Upload | Carl |
| HIST\_PAYSUM | HIST | EOD | Upload | Carl |
| HIST\_POSSYSTEM | HIST | EOD | Upload | Carl |
| HIST\_REDEEMED\_COUPON | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXCHECK | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXCHECKTENDER | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXINOUT | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXINOUTEXTENDINFO | HIST | EOD | Upload | Carl |
| HIST\_SAFEBOXPICKUP | HIST | EOD | Upload | Carl |
| HIST\_SESSIONINFO | HIST | EOD | Upload | Carl |
| HIST\_SESSIONTENDER | HIST | EOD | Upload | Carl |
| HIST\_STOCK\_MOVEMENT | HIST | EOD | Upload | Carl |
| HIST\_SUPP | HIST | EOD | Upload | Carl |
| HIST\_TRANS | HIST | EOD | Upload | Carl |
| HIST\_TRANS\_ECARD | HIST | EOD | Upload | Carl |
| HIST\_TRANS\_MODIFIER | HIST | EOD | Upload | Carl |
| INVITATION | Master | Every Polling | Download | Carl |
| ITEM | Master | Every Polling | Download | Carl |
| ITEM\_BARCODE | Master | Every Polling | Download | Carl |
| ITEM\_MODIFIER | Master | Every Polling | Download | Carl |
| ITEMANLY | Master | Every Polling | Download | Carl |
| ITEMDEPT | Master | Every Polling | Download | Carl |
| MENU | Master | Every Polling | Download | Carl |
| MENUITEM | Master | Every Polling | Download | Carl |
| MESSAGES | Master | Every Polling | Download | Carl |
| MODIFIER | Master | Every Polling | Download | Carl |
| MODIFIER\_GRP | Master | Every Polling | Download | Carl |
| MODIFIER\_LIST | Master | Every Polling | Download | Carl |
| ONHOUSE | Master | Every Polling | Download | Carl |
| OPTIONS | Master | Every Polling | Download | Carl |
| ORDERS | Sales Data | Every Polling | Upload | Carl |
| ORDERS\_EXTRA | Sales Data | Every Polling | Upload | Carl |
| ORDERS\_PAY | Sales Data | Every Polling | Upload | Carl |
| PAYCAT | Master | Every Polling | Download | Carl |
| PAYMENT | Master | Every Polling | Download | Carl |
| PMT\_ACTION | Master | Every Polling | Download | Carl |
| PMT\_CONDITION | Master | Every Polling | Download | Carl |
| PMT\_HDR | Master | Every Polling | Download | Carl |
| POSBUSDATE | Master | Every Polling | Download | Carl |
| ROLE | Master | Every Polling | Download | Carl |
| ROLE\_PERMISSION | Master | Every Polling | Download | Carl |
| SUPP | Sales Data | Every Polling | Upload | Carl |
| SYSSETTINGS | Master | Every Polling | Download | Carl |
| TRANS | Sales Data | Every Polling | Upload | Carl |
| TRANS\_ECARD | Sales Data | Every Polling | Upload | Carl |
| TRANS\_MODIFIER | Sales Data | Every Polling | Upload | Carl |
| TRANS\_TYPE | Master | Every Polling | Download | Carl |
| USER\_PERMISSION | Master | Every Polling | Download | Carl |
| USER\_ROLE | Master | Every Polling | Download | Carl |
| WIFI CODE | Master | Every Polling | Download | Carl |

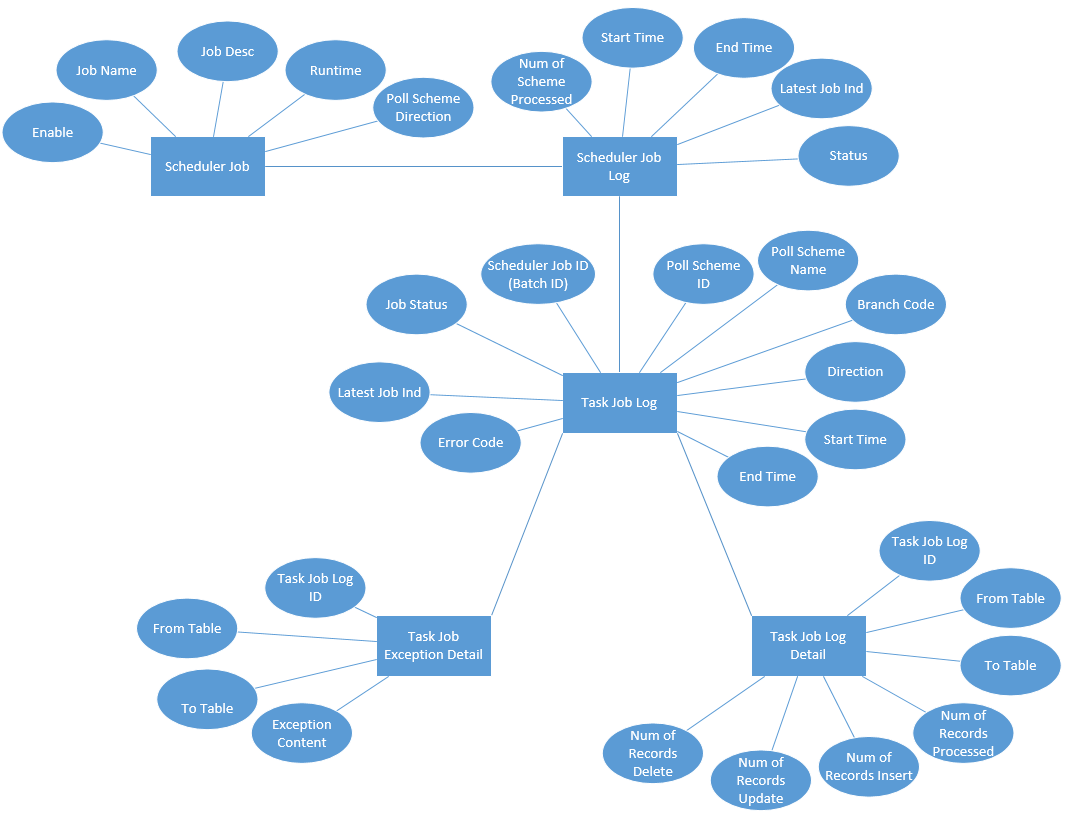
Dick??? Do we check the mapping **every** between POS client DB tables, Staging DB tables, EDW database and Master/Pricing tables?

## Polling Application Work Tables

**Polling Schemes**



**Job Control (Job Logs, Exceptional Data)**



Dick??? What is the usage of “Latest Job Ind”?

Dick??? What is the logic behind the flow for the scheduler and task? Could you explain the flow?

Dick??? As mentioned by Wing, we need to interface to existing Maxim’s Branch Master to retrieve the latest branch information, e.g. Branch Code, IP Address, etc. Any overlap with ‘Poll Scheme Info’?

Dick??? As mentioned by Wing, Carl had a new enhancement in POS (we need to check with Carl).

# High Level Interface Design

POS Client

POS DB

POS Client

POS File

Sales Data Real Time Service

Web Logic Application

Sales Data EOD Service

Pricing / Master Service

Administrative Service

Job Monitor Service

Report Service

FTP

JDBC

HTTP

File System

Oracle Service Bus

wWsdssadsadsad

Sales Data Web Service

Pricing / Master Data Web Service

EDW Checksum Web Service

JDBC

POS Server Synchronization

JDBC

EDW Database

Staging Database

JDBC

Pricing Database

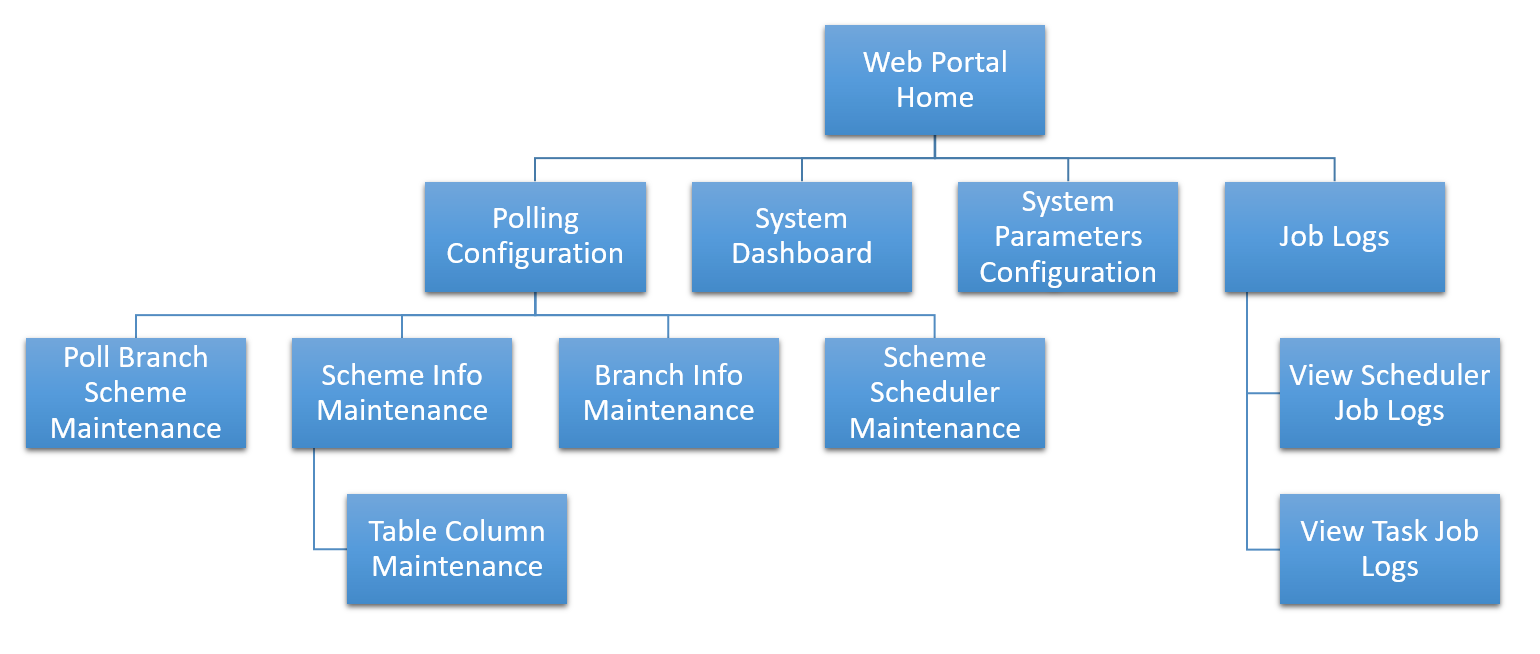
IT 5x

# Functional Design

## System Functional Overview

### Application Context

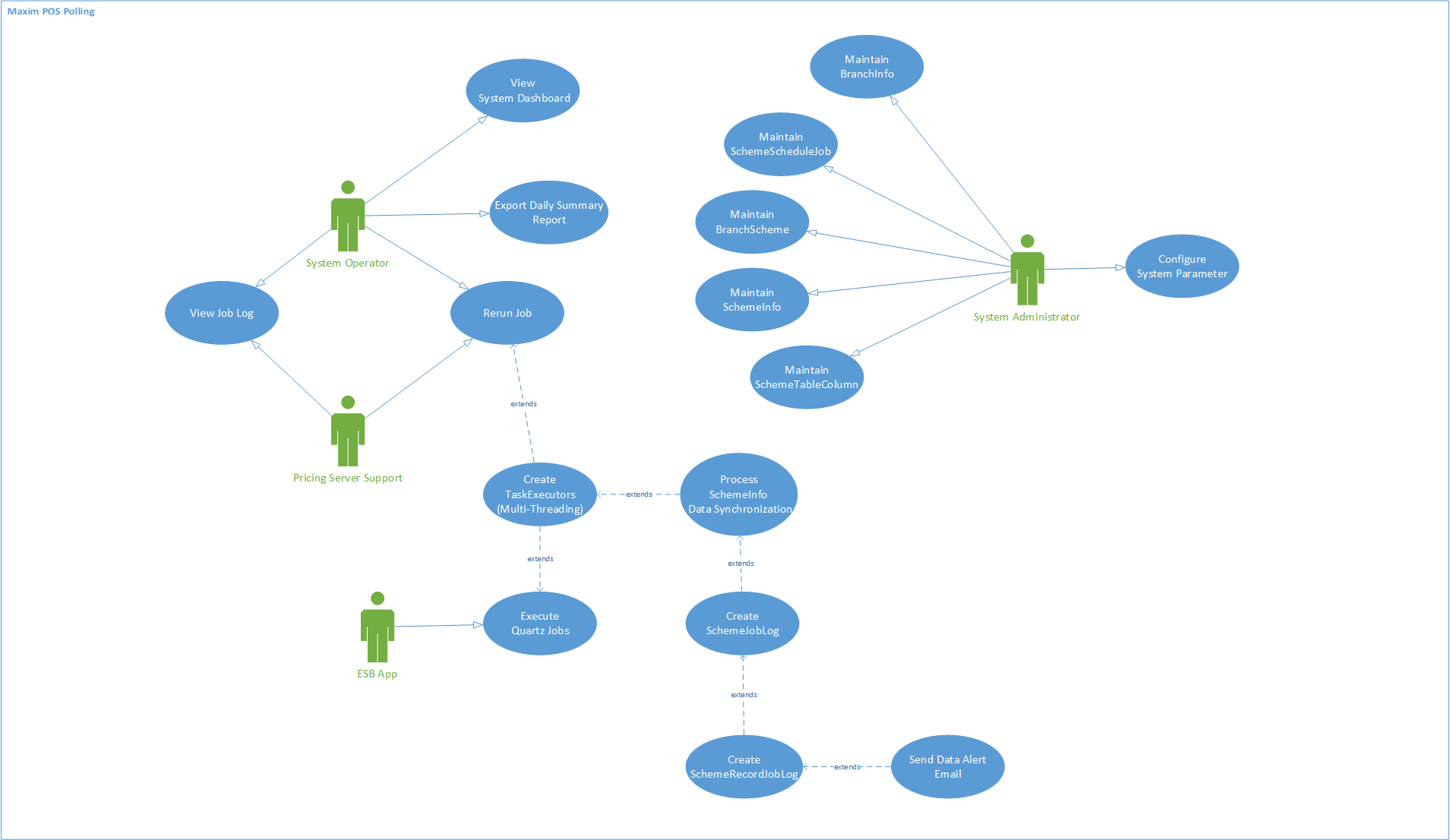
### Site Map



Dick??? Branch Info Maintenance might overlap with Maxim’s branch master.

Dick??? Where to define the Email Alert? In system parameters configuration?

### Use Case (Add back user requirement matrix id) / add back matrix



Dick: How to handle delta sync and full set sync?

Dick: Table Design and Flow Design

### Interfaces

## Technology Used

Application Server side:

*Oracle WebLogic 12c, Oracle Service Bus 12c, JAVA (With JRE V1.8)*

Database Server side:

*MS SQL Server 2012, Oracle 11.2g (PL/SQL), MS SQL Server 2000*

Client Side Browser:

*Chrome 55 or above*

*Internet Explorer 11 or above*

## Branch Info Maintenance

### Description

The “Branch Info Maintenance” function allows user’s manipulation on POS client data source definition. The data should be initially input based on current polling server. The view of these data are available for the access from both Service Bus User and POS User, but Service Bus user are allowed to use operations including Create, Edit and Delete. On the Add or Modification of the branch info data, the data source connectivity will be validated upon saving.

Use Case functions

# Maintain Branch Info (List)

# Maintain Branch Info (Create/Edit/Delete) 🡸 Dick??? Should read from Maxim’s branch master.

### Input

N/A

### Output

N/A

### Maintain Branch Info (List)

#### Process/Work Flow

This function is to display branch info records according the user input criteria. The branch info will be displayed into the table within the page, and specific columns will be displayed on screen, as some of them enabled as sorting function upon user’s requirements.

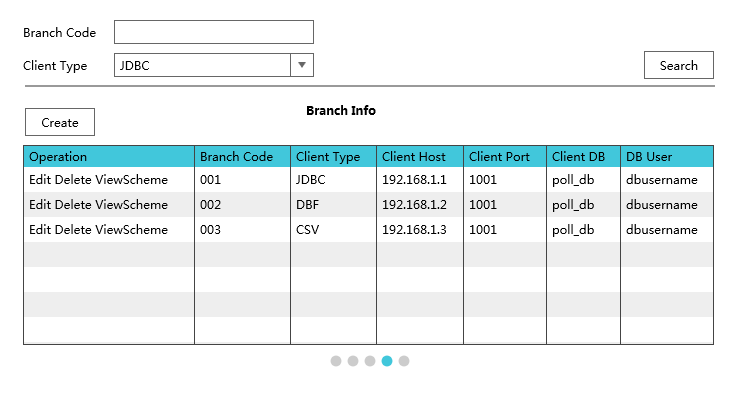
**Business Logic**

The branch info is to describe the data source configuration of the POS clients sales data, which affect sales data polling process described in the latter session. Each record represents a single data source, which could be in type of SQL connection, database file or fixed format text file.

1. By default, when the page is accessed, there’s no search criteria provided and all branch info will be displayed on the page with pagination
2. User could conduct manipulations on searched-out records such as page switching, sorting by specific columns and refining.
3. User could conduct operational functions upon searched-out records such as Edit/Delete and view its details in the “Operation” column. (details included in session 6.2.5)

#### Screen

**Maintain Branch Info (List)**



Dick??? Do we need branch name as no one will remember all branch codes? Besides, can support partial searching, e.g. 1% to search all branch starting with ‘1’.

Dick??? Suggestion: use link instead of “ViewScheme”. Check box for multiple selections, e.g. delete multiple branch. Put “Operation” to the top, e.g. Create, Delete to save some space.

#### Data Fields & Presentation Logic

1. Default Sorting of Branch Info List, Priority from top to bottom

* Branch Code (ascending)

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Branch Code** | Read-only text  Dick??? Any edit screen? | Prefilled | - | A17 | - | Branch Code  Dick??? Any validation for this field if editable? Size, type or LOV? |
| **Client Type** | Read-only text | Prefilled | - | A | - | Data source type, could be DB connection, database file or text file |
| **Client Host** | Read-only text | Prefilled | - | A | - | The location of the clie3nt, it could be a URL with host name, IP address or file directory. |
| **Client Port** | Read-only text | Prefilled | - | A | - | Client port number, will be blank if the client access does not require a port, e.g. shared folder |
| **Client DB** | Read-only text | Prefilled | - | A | - | Database name if the client is a DB Connection |
| **Client User** | Read-only text | Prefilled | - | A | - | A username to access the client |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Create | Button | Create a new entry of branch info | N/A |
| Operation -**Edit** | Button | Edit the corresponding branch info record | N/A |
| Operation -**Delete** | Button | Delete the corresponding branch info | N/A |
| Operation -**Detail** | Button | View the corresponding branch info detail | N/A |

#### User/Security Group

This function can be accessed by the following user(s):

* System Administrator

#### Assumptions/Constraints

N/A

#### Error & Exception List

### Maintain Branch Info (Create/Edit/Delete)

#### Process/Work Flow

**Maintain Branch Info (Create)**

A pop-up dialog will appear to receive input from the user by clicking the “Create” button. On clicking the “save” button, the system will first validate the source’s connectivity, and save to the branch info table upon successful validation. User could also click on the “test” button to validate its connectivity separately after input all mandatory fields before saving them

**Maintain Branch Info (Edit)**

A pop-up dialog contain in original branch info detail values will appear to receive input from the user by clicking the “Edit” button. Same as record creation, On clicking the “save” button, the system also validate the source’s connectivity, and save the record’s update. “Test” button works for the same validation.

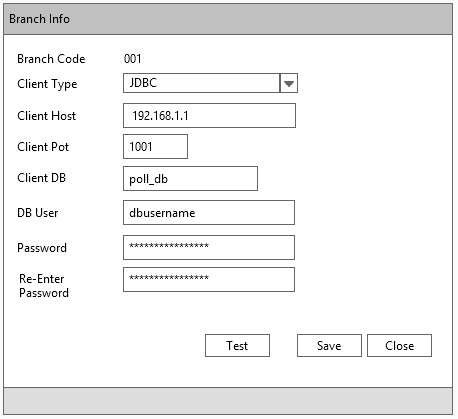
**Maintain Branch Info (Delete)**

Home> Polling Configuration> Branch Scheme> Search Record> Delete

A pop-up dialog will appear to asking the user for the confirmation of the deletion of the corresponding record. Once user clicks yes, the system will check whether the record has relations to other data, if no, the target record will be deleted.

#### Screen

**Maintain Branch Info (Create/Edit)**



#### Data Fields & Presentation Logic

**Create Branch info Layout**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Branch Code** | Textbox | Blank | - | A17 | - | Branch Code |
| **Client Type** | Dropdown box | Blank | - | A | - | Options   * SQL Server * DBF * Text file |
| **Client Host** | Textbox | Blank | - | A | - | Must be a valid location of the client, it could be a URL with host name, IP address or file directory. |
| **Client Port** | Textbox | Blank | - | N(5) | - | Client port number, will be blank if the client access does not require a port, e.g. shared folder |
| **Client DB** | Textbox | Blank | - | A | - | Database name if the client is a DB Connection |
| **Client User** | Textbox | Blank | - | A | - | A username to access the client |
| **Password** | Textbox | Blank | - | A | - | The value of this textbox will be masked |
| **Re-enter password** | Textbox | Blank | - | A | - | The value of this textbox will be masked |
| **Dick??? Any audit information for the records, e.g. create/amend datetime and user?** |  |  |  |  |  |  |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Test | Button | Test the connectivity of the source |  |
| Save | Button | Save the created/edited content of the corresponding record | N/A |
| Close | Button | Cancel and discard all changes of corresponding created/edited record and back to the list page | - |

#### User/Security Group

This function can be accessed by the following user(s):

* System Administrator

#### Assumptions/Constraints

N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
| S0001 | N/A | Connection Test Failed! |
| Successful message  Dick??? It is NOT an exception. | N/A | Branch info saved successfully. |
| S0002 |  | This branch info has maintained relation with other data.  Dick??? What mean? |
| Successful message  Dick??? Not exception. | N/A | Branch info deleted successfully |
|  |  | Dick??? How about other exception like duplicate branch code, branch not exit in Maxim’s branch master, missing hostname or port number, etc. |

## Scheduler Jobs Maintenance

### Description

The “Scheduler Job Maintenance” function allows user’s manipulation upon service bus schedule jobs running for the data processing between POS client and staging, staging and EDW, staging and pricing server. Because the job running is a customized and standalone service, the jobs cannot be created or deleted by user. But user is able to change the scheduler triggering time, job names or enable/disable the jobs.

Use Case functions

# Maintain Scheduler Jobs (List)

# Maintain Scheduler Jobs (Edit)

### Input

N/A

### Output

N/A

### Maintain Scheduler Jobs (List)

#### Process/Work Flow

This function is to display all scheduler jobs details. The scheduler job records will be displayed into the table within the page, and only specific columns allow editing.

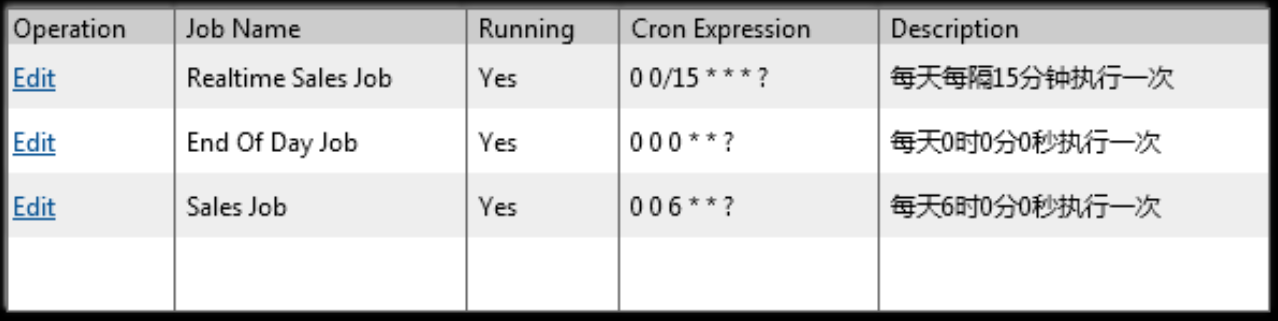
**Business Logic**

The branch scheme is to describe the POS polling process scheduling details. Each scheduler will run upon target poll branch scheme to take on data processing. Each record represents a single schedule tasks and triggers by its configured time to submit separate tasks upon multiple poll schemes.

1. By default, all scheduler jobs will be displayed in the table within the page.
2. User could conduct “Edit” functions upon the target scheduler job records such as Edit in the “Operation” column. (details included in session 6.4.5) so as to open a job detail panel to edit.

#### Screen

**Maintain Scheduler Jobs (List)**

****

#### Data Fields & Presentation Logic

1. Default Sorting of Branch Info List, Priority from top to bottom

* Job Group (ascending)
* Job Name (ascending)

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Job Name** | Read-only text | Prefilled | - | A50 | - | Job name |
| **Job Description** | Read-only text | Prefilled | - | A200 | - | Job description |
| **Cron Expression** | Read-only text | Prefilled | - | A50 | - | The cron expression represents the job’s running time |
| **Enable** | Read-only text | Prefilled | - | A1 | - | * Y (enabled) * N (disabled) |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Operation -**Edit** | Button | Edit the corresponding scheduler job record | N/A |

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Data Fields & Presentation Logic

N/A

#### Assumptions/Constraints

N/A

#### Error & Exception List

N/A

### Maintain Scheduler Jobs (List)

#### Process/Work Flow

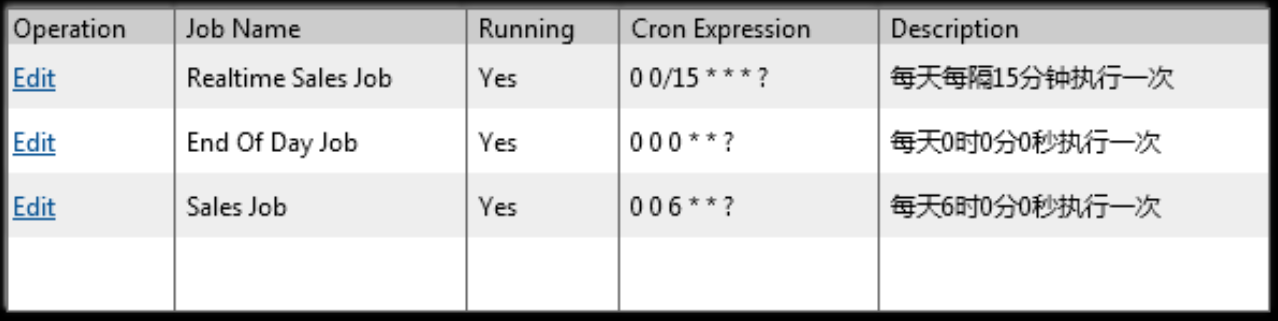
**Maintain Branch Info (Edit)**

A pop-up dialog contain in original scheduler info detail values will appear to receive input from the user by clicking the “Edit” button. Same as record creation, On clicking the “save” button, the system also validate the source’s connectivity, and save the record’s update. “Test” button works for the same validation.

Because each single job has differences in the actual processing, any scheduler job records cannot be created or deleted from the table. The creation and deletion can only be maintained by IT department. Dick??? What is the meaning “has differences in the actual processing”? Besides, the function should be maintained by “IT department”.

#### Screen

**Maintain Scheduler Jobs (Edit)**

****

#### Data Fields & Presentation Logic

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Job Name** | Textbox | Original Value | - | A50 | - | Job name |
| **Job Description** | Textbox | Original Value | - | A200 | - | Job description |
| **Cron Expression** | Textbox | Original Value | - | A50 | - | The cron expression represents the job’s running time |
| **Enable** | Dropdown box | Original Value | - | A1 | - | * Y (enabled) * N (disabled) |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Save | Button | Save the created/edited content of the corresponding record | N/A |
| Close | Button | Cancel and discard all changes of corresponding created/edited record and back to the list page | N/A |

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Assumptions/Constraints

N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
| S | N/A | The cron expression is not valid! |
| Successful Message | N/A | The job information is updated successfully |

## Branch Scheme Maintenance

### Description

The “Branch Scheme Maintenance” function allows user’s manipulation on POS client data processing logic data. The data is input/configured by service bus administrator according to the poll branch scheme data in current polling server. The view of these data are available for the access from both Service Bus, but Service, bus Admin user are allowed to use operations including Create, Edit and Delete.

Use Case functions

# Maintain Branch Scheme (List)

# Maintain Branch Scheme (Create/Edit/Delete)

# Maintain Scheme Info (List)

# Maintain Scheme Info (Create/Edit/Delete)

# Maintain Table Column (List)

# Maintain Table Column (Create/Edit/Delete)

### Input

N/A

### Output

N/A

### Maintain Branch Scheme (List)

#### Process/Work Flow

This function is to display branch scheme records according the user input criteria. The branch scheme records will be displayed into the table within the page, and specific columns will be displayed on screen, as some of them enabled as sorting function upon user’s requirements.

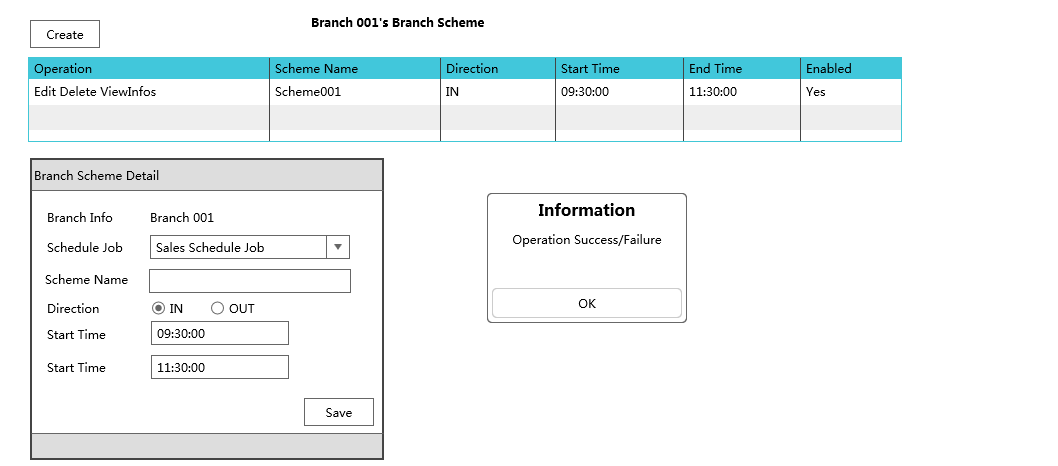
**Business Logic**

The branch scheme is to describe the POS polling data process configuration upon a certain POS client (branch), which affect data process job of both inbound and outbound to the POS client. Each record represents a single data processing logic in field level definition.

1. By default, when the page is accessed, there’s no search criteria provided and all branch info will be displayed on the page with pagination 🡸 Dick??? Beware where is the point #1 and #2? Why there is no searching screen? Basically, it is not maintainable as over thousands of records, 900+ branch times 3 scheme = 2,700+ records.
2. User could conduct manipulations on searched-out records such as page switching, sorting by specific columns and refining.
3. User could conduct operational functions upon searched-out records such as Edit/Delete and view its details in the “Operation” column. (details included in session 6.4.5)

#### Screen

**Maintain Branch Scheme (List)**



Dick??? What is Operation Success/Failure??? Confusing…

#### Data Fields & Presentation Logic

1. Default Sorting of Branch Info List, Priority from top to bottom

* Scheme Name (ascending)

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Poll Scheme Name** | Read-only text | Prefilled | - | A17 | - | Poll scheme name |
| **Poll Scheme Type** | Read-only text | Prefilled | - | A20  Dick??? Change to LOV | - | Poll Scheme Type is used to defined the set of tables this scheme will be covered   * SALES\_REALTIME * SALES\_EOD * MASTER |
| **Direction** | Read-only text | Prefilled | - | A40 | - | Describe the direction of the data |
| **Start Time** | Read-only text | Prefilled | - | Time – hh24:mi:ss  Dick??? Really need seconds? mi:**ss** | - | Start time of the poll scheme |
| **End Time** | Read-only text | Prefilled | - | Time – hh24:mi:ss | - |  |
| **Enable** | Read-only text | Prefilled | - | A1 | - | * Y (enabled) * N (disabled) |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Create | Button | Create a new entry of poll scheme | N/A |
| Operation -**Edit** | Button | Edit the corresponding poll scheme record | N/A |
| Operation -**Delete** | Button | Delete the corresponding poll scheme | N/A |
| Operation -**Detail** | Button | View the corresponding poll scheme detail | N/A |

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Data Fields & Presentation Logic

N/A

#### Assumptions/Constraints

N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Dick??? No validation or incomplete??? |

### Maintain Branch Scheme (Create/Edit/Delete)

#### Process/Work Flow

This function is to manipulate the detail of branch scheme records. The branch scheme records will be displayed into the table within the page, and specific columns will be displayed on screen, as some of them enabled as sorting function upon user’s requirements.

**Business Logic**

By below operation from the “Maintain Branch Scheme” page, user is able to utilize different functions upon the poll branch scheme data.

**Maintain Branch Scheme (Create)**

Home> Polling Configuration> Branch Scheme> Create

A pop-up dialog will appear to receive input from the user by clicking the “Create” button. On clicking the “save” button, the system will first validate fields’ mandatory and validity, and save to the branch scheme table upon successful validation.

**Maintain Branch Scheme (Edit)**

Home> Polling Configuration> Branch Scheme> Search Record> Edit

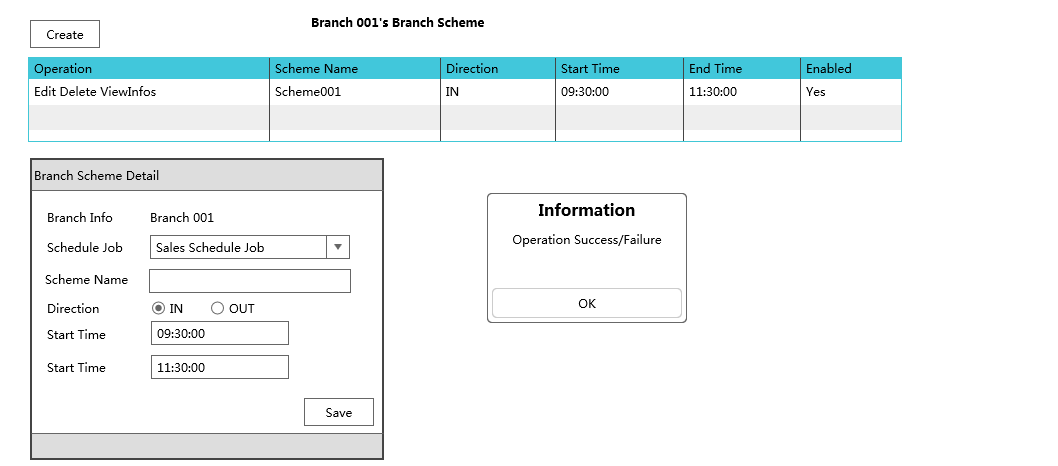
A pop-up dialog contain in original branch info detail values will appear to receive input from the user by clicking the “Edit” button. Same as record creation, On clicking the “save” button, the system also validate upon the values, and save the record’s update.

**Maintain Branch Scheme (Delete)**

Home> Polling Configuration> Branch Scheme> Search Record> Delete

A pop-up dialog will appear to asking the user for the confirmation of the deletion of the corresponding record. Once user clicks yes, the system will check whether the record has relations to other data, if no, the target record will be deleted.

#### Screen



#### Data Fields & Presentation Logic

1. Default Sorting of Branch Info List, Priority from top to bottom

* Scheme Name (ascending)

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Poll Scheme Name** | textbox | Prefilled | - | A17 | - | Poll scheme name |
| **Poll Scheme Type** | Dropdown box | Prefilled | - | A20 | - | Poll Scheme Type is used to defined the set of tables this scheme will be covered   * SALES\_REALTIME * SALES\_EOD * MASTER |
| **Direction** | Dropdown box | Prefilled | - | A40 | - | Describe the direction of the data |
| **Start Time** | Timepicker | Prefilled | - | Time – hh24:mi:ss | - | Start time of the poll scheme |
| **End Time** | Timepicker | Prefilled | - | Time – hh24:mi:ss | - |  |
| **Enable** | Radiobox | Prefilled | - | A1 | - | * Y (enabled) * N (disabled) |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Save | Button | Save the created/edited content of the corresponding record | N/A |
| Close | Button | Cancel and discard all changes of corresponding created/edited record and back to the list page | - |

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Data Fields & Presentation Logic

N/A

#### Assumptions/Constraints

N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  |  |

### Maintain Scheme Info (List)

#### Process/Work Flow

This function is to display scheme info detail records according the user input criteria. The scheme info records will be displayed into the table within the page, and specific columns will be displayed on screen, as some of them enabled as sorting function upon user’s requirements.

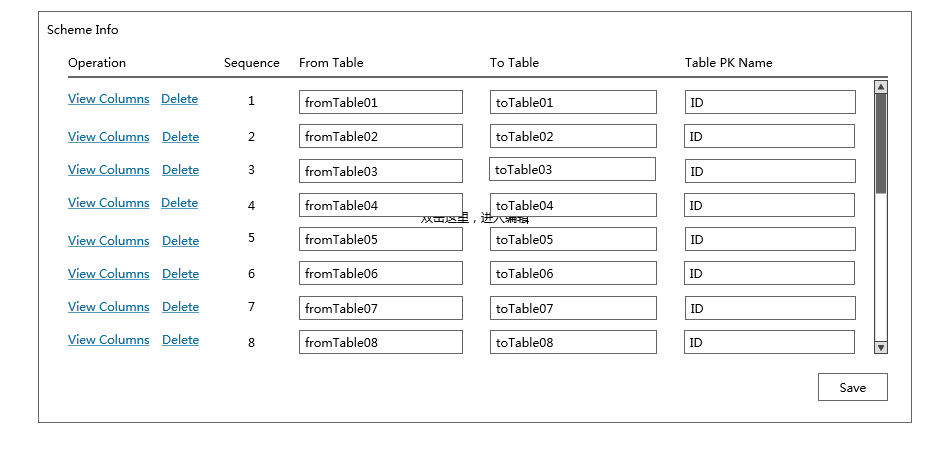
**Business Logic**

The scheme info is to describe the set of table to be processed in a branch scheme under the category of field “poll scheme type”, So the poll branch scheme could use its poll scheme types to find tables to process according to the poll scheme type. Each scheme info record represents a single table processing logic.

1. By default, when the page is accessed, there’s no search criteria provided and all branch info will be displayed on the page with pagination
2. User could conduct manipulations on searched-out records such as page switching, sorting by specific columns and refining.
3. User could conduct operational functions upon searched-out records such as Edit/Delete and view its details in the “Operation” column. (details included in session 6.4.7)

Dick???? can we allow to import the scheme info using CSV file? Or, copy the scheme info from another branch.

#### Screen



#### Data Fields & Presentation Logic

1. Default Sorting of Branch Info List, Priority from top to bottom

* Poll scheme type (ascending)
* From table (ascending)
* To table (ascending)

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Poll scheme type** | Read-only text | Prefilled | - | A(20) | - | Poll scheme type |
| **From table** | Read-only text | Prefilled | - | A(100) | - | From table name |
| **To table** | Read-only text | Prefilled | - | A(100) | - | To table name |
| **PK columns** | Read-only text | Prefilled | - | A(10) | - | Primary key column names |
| **Check sum columns** | Read-only text | Prefilled | - | A(100) | - | Check sum column names |
| **Override** | Read-only text | Prefilled | - | A(1) | - | * Y (override to table) * N (not override) |

#### Screen Objects & Action

N/A

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Assumptions/Constraints

N/A

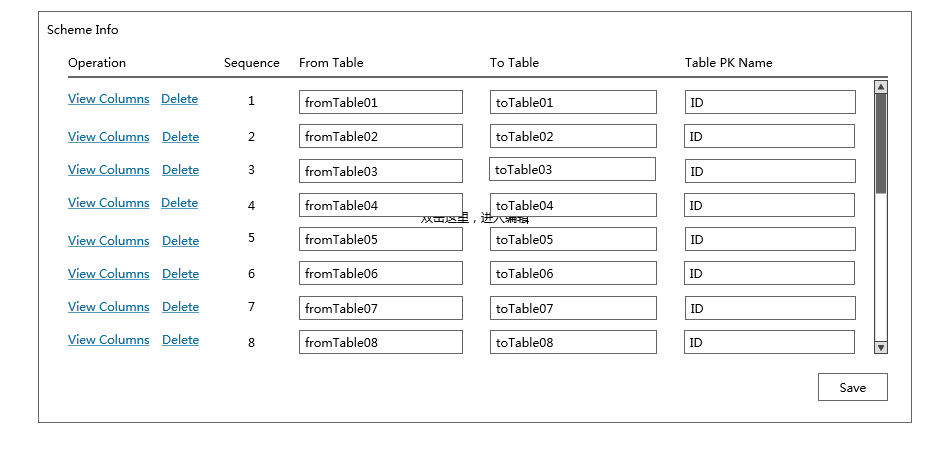
#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  | N/A |  |

### Maintain Scheme Info (Create/Edit/Delete)

#### Process/Work Flow

#### Screen



#### Data Fields & Presentation Logic

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| Poll scheme type | Dropdown box | Prefilled | - | A(20) | - | Pre-configured options   * SALE\_REALTIME * SALES\_EOD * MASTER |
| From table | textbox | Prefilled | - | A(100) | Validate the table name correctness | Free text table name |
| To table | textbox | Prefilled | - | A(100) | Validate the table name correctness | Free text table name |
| PK columns | textbox | Prefilled | - | A(10) | Validate the column name correctness | Free text primary key column names on table table, separated by ‘,’ |
| Check sum columns | textbox | Prefilled | - | A(100) | Validate the column name correctness | Free text check sum column names on target able, separated by ‘,’ |
| Override | Dropdown box | Prefilled | - | A(1) | - | * Y (override to table) * N (not override) |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Save | Button | Save the created/edited content of the corresponding record | N/A |
| Close | Button | Cancel and discard all changes of corresponding created/edited record and back to the list page | - |

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Assumptions/Constraints

N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  |  |

### Maintain Table Column Info (List/Create/Edit/Delete)

#### Process/Work Flow

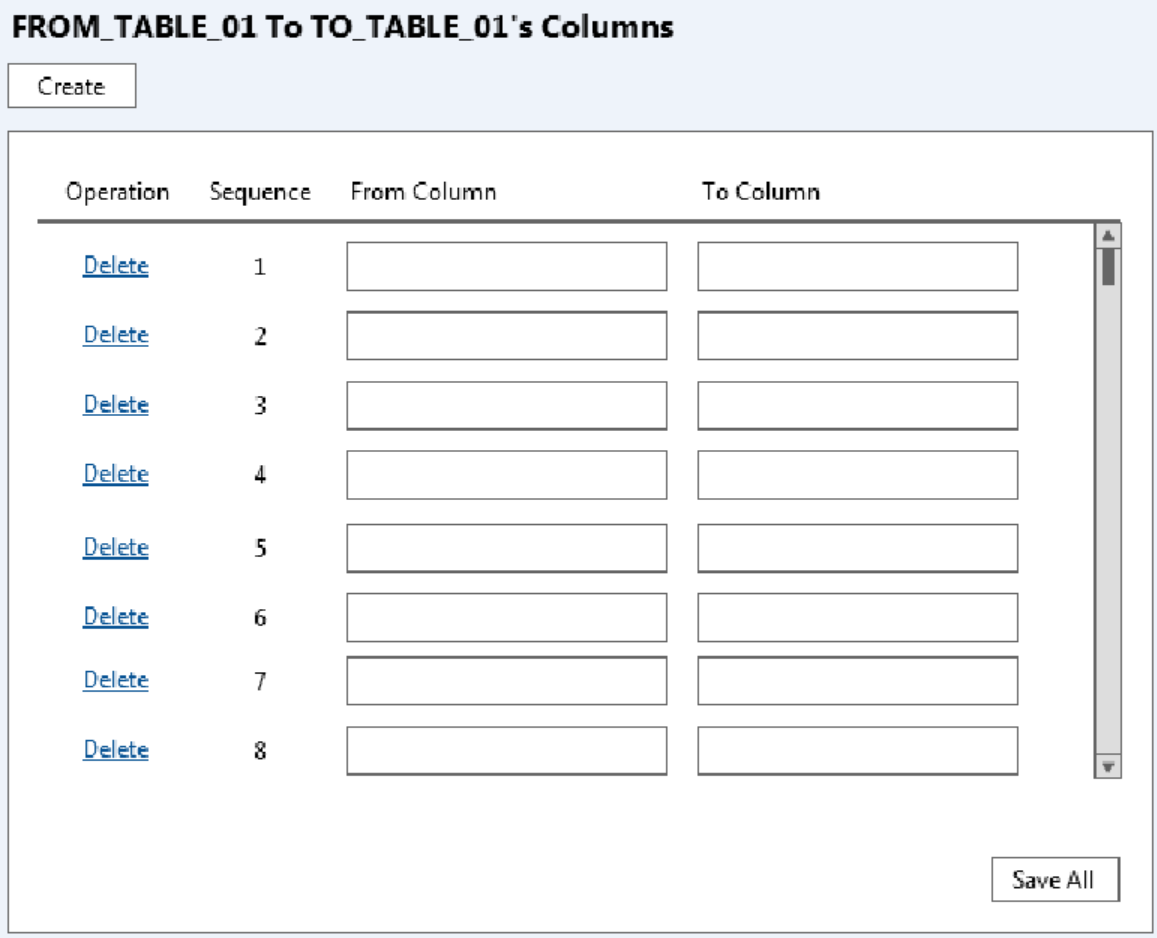
This function is to display table column detail records of the corresponding poll scheme info. The scheme table columns details will be displayed when user opens the detail panel of a single scheme info. Within the scheme info detail panel, the table of table columns allows user to view the full list of table column, as well as create/edit and delete each row.

**Business Logic**

The table columns of a scheme info is to describe the column mapping from table to table of a scheme info. So the poll scheme info could use these columns to format a SQL string for the certain polling process on a table. Each table column represents a from table to table column mapping in the polling process.

1. By default, when the table columns table is displayed when the detail panel of a poll scheme info appears.
2. User could edit the from column to column to define the details of the table column, which represents the column mapping for the poll data process. Column mapping here is limited to one-to-one mapping.
3. User could conduct operational functions including “Add” “Edit” or “Remove” upon each row of a table column, from the “Operation” column.

#### Screen



#### Data Fields & Presentation Logic

1. Default Sorting of Branch Info List, Priority from top to bottom

* Sequence (ascending)

1. Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| **Sequence** | Read-only text | Prefilled | - | N(\*) | - | A Sequence for display and reference only |
| **From column** | Textbox | Original Value  /Blank | M | A(100) |  | The from-column name |
| **To Column** | Textbox | Original Value  /Blank | M | A(100) |  | The to-column name |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Create | Button | Append a new and empty table column setting record at the bottom of the table | N/A |
| Delete | Button | Remove the corresponding table column setting record from the table | N/A |
| Save | Button | Save the created/edited tables columns list for the editing poll scheme info | N/A |
| Close | Button | Cancel and discard all changes of corresponding created/edited/delete tables columns and back to the list page | N/A |

#### User/Security Group

This function can be accessed by the following user(s):

* System Admin

#### Assumptions/Constraints

N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  |  |

## Job Logs View

### Description

The “Job Logs View” function will display the status of the batch job.

Use Case functions

# Job Logs View(List)

# Job Logs View(Detail)

### Input

N/A

### Output

N/A

### Job Logs View (List)

#### Process/Work Flow

This function is to display the job logs records according the user input criteria. The log info will be displayed into the table within the page, and specific columns will be displayed on screen, as some of them enabled as sorting function upon user’s requirements.

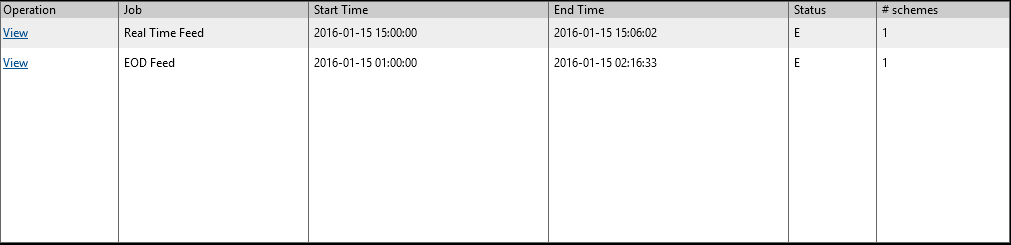
**Business Logic**

The job logs view is to describe the status of the schedule jobs. Each record represents a scheduled job.

1. By default, when the page is accessed, there’s no search criteria provided and all job logs will be displayed on the page with pagination
2. User could conduct manipulations on searched-out records such as page switching, sorting by specific columns and refining.
3. User could view its details in the “Operation” column. (details included in session 6.5.5)

#### Screen

**Job Logs View (List)**



#### Data Fields & Presentation Logic

1. Default Sorting of Job Logs View List, Priority from top to bottom

* Start Time (descending)

1. Layout

| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| --- | --- | --- | --- | --- | --- | --- |
| **Job** | Read-only text | Prefilled | - | A | - | Job Name |
| **Start Time** | Read-only text | Prefilled | - | A | - | The Start Time of the Job |
| **End Time** | Read-only text | Prefilled | - | A | - | The End Time of the Job |
| **Status** | Read-only text | Prefilled | - | A | - | The Status of the Job |
| **# Scheme** | Read-only text | Prefilled | - | A | - | The number of the scheme of the job |

#### Screen Objects & Action

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen Object** | **Object Type** | **Action / Event / Response** | **Remarks** |
| Operation -View | Button | View the corresponding job log detail | N/A |

#### User/Security Group

This function can be accessed by the following user(s):

#### Assumptions/Constraints

N/A

#### Error & Exception List

N/A

### Job Logs View (Detail)

#### Process/Work Flow

**Job Logs View (Detail)**

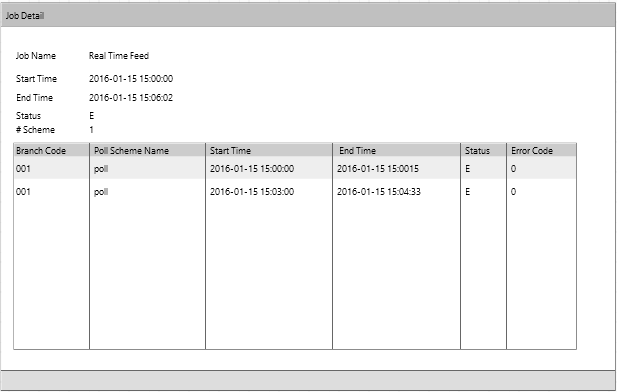
This function is to display the detail log of a job according to the user selected. The detail will be shown and the related task log info will be displayed into the table and the specific columns will be displayed on screen, as some of them enabled as sorting function upon user’s requirements.

**Business Logic**

The job logs view detail is to describe the status of the detail information of the schedule jobs. Each record represents a scheduled job.

#### Screen

**Job Logs View (Detail)**



#### Data Fields & Presentation Logic

1. Default Sorting of the task list, Priority from top to bottom

* Start Time (descending)

1. Layout

| **Field** | **Object Type** | **Default Value** | **Mandatory (M/O/C)** | **Format** | **Action / Event / Response** | **Description** |
| --- | --- | --- | --- | --- | --- | --- |
| **Job Name** | Read-only text | Prefilled | - | A | - | Job Name |
| **Start Time** | Read-only text | Prefilled | - | A | - | The Start Time of the Job |
| **End Time** | Read-only text | Prefilled | - | A | - | The End Time of the Job |
| **Status** | Read-only text | Prefilled | - | A | - | The Status of the Job |
| **# Scheme** | Read-only text | Prefilled | - | A | - | The number of the scheme of the job |
| **Branch Code (List)** | Read-only text | Prefilled | - | A | - | The Branch Code of the Task |
| **Poll Scheme Name (List)** | Read-only text | Prefilled | - | A | - | The Poll Scheme Name of the Task |
| **Start Time (List)** | Read-only text | Prefilled | - | A | - | The Start Time of the Task |
| **End Time (List)** | Read-only text | Prefilled | - | A | - | The End Time of the Task |
| **Status (List(** | Read-only text | Prefilled | - | A | - | The Status of the Task |

#### Screen Objects & Action

N/A

#### User/Security Group

This function can be accessed by the following user(s):

#### Assumptions/Constraints

N/A

#### Error & Exception List

N/A

## Sales Data Real Time Polling Batch Job

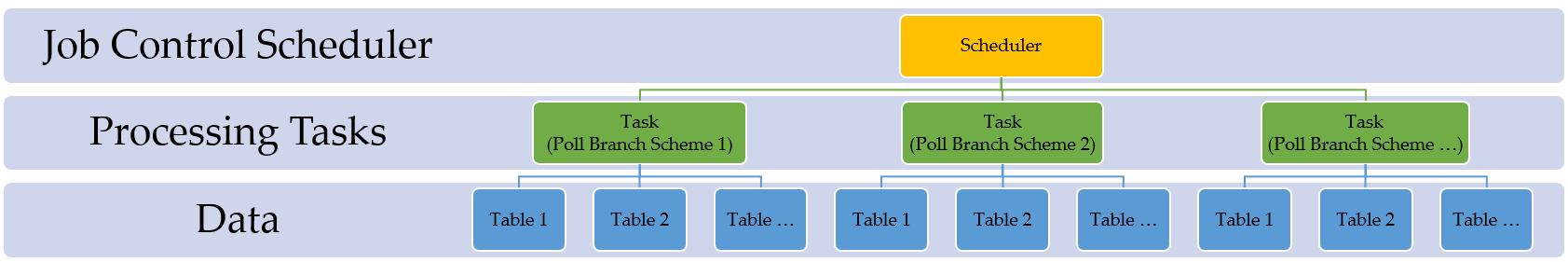
### Description

The scheduled batch job runs in the configurable time interval (e.g. 15 mins) to download POS client sales data from all active POS clients into the staging database. This is the pre-requisite batch job of the other batch which push real time sales data from staging database into EDW.

The job is triggered by a scheduler and it works as a task controller, submitting separated standalone tasks. Each task handles one available branch scheme. The job will handle the data in the pre-defined tables under the scheme according to the scheme configuration (e.g. direction, override logic, and validation).

In below session there will be the details descriptions about

1. Sales Data Real Time (POS - Staging) Scheduler
2. Sales Data Real Time (POS - Staging) Task



### Input

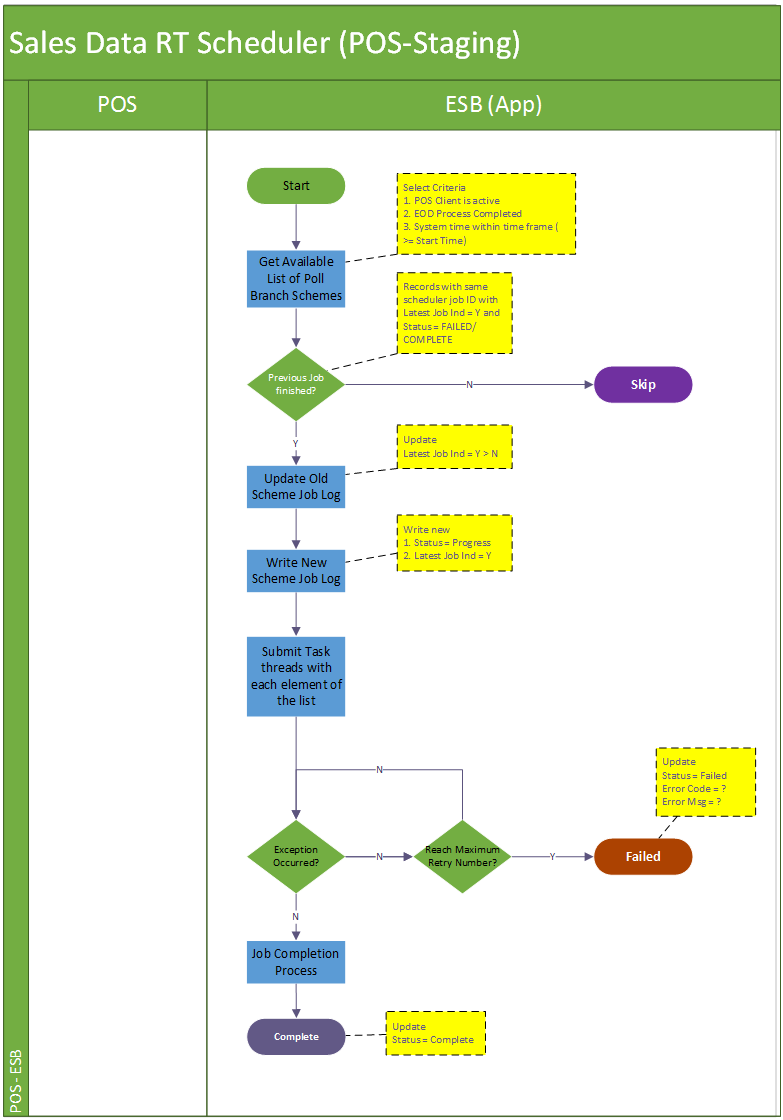
POS Client Sales Data (Real time)

### Output

Staging Sales Data (Real time)

### Sales Data Real Time (POS - Staging) Scheduler

#### Process/Work Flow



**Job Details**

The scheduler job is considered as the commander to handle the life cycle of poll scheme tasks which it’s responsible for. With a proper criteria selection, the scheduler is supposed to submit a series of tasks into the pull and queue up for their own process. And it logs down the running process in job logs and log files.

Summarily the schedule executes in below steps

Step1. This batch job will be triggered every 15 mins (configured). It would iterate all available data accumulated in poll branch scheme table (with condition POS client is enabled, the POS client has not yet conducted EOD process and the current time is greater or equal to the start time)

Step 2. Update scheduler job log. Firstly update the previous job log’s latest job indicator from ‘Y’ to ‘N’. Onwards newly create scheduler job log is having the status “PROGRESS”. The latest batch indicator for the new record will be marked as “Y”, and that of the current batch will be marked as “P”.

Step 3. Submit processing tasks for all eligible poll branch schemes, which are selected in step 1.

Step 4. Upon any errors in the task submission and cannot be resolved (e.g. by retry). The job will abort and keep the job status to reflect the problematic stage “FAILED”.

Step 5. If all tasks of the poll branch schemes are successfully submitted, the scheduler is regarded as complete and update the job status to “COMPLETE” with a record number of schemes processed

|  |  |  |
| --- | --- | --- |
| **Job Name** | **Status** | **Description** |
| Sales Real Time Polling Scheduler | “PROGRESS”  “FAILED”  “COMPLETE” | “PROGRESS” when scheduler is invoked and started  “FAILED” when encounter errors reading poll scheme data from database or errors submitting tasks.  “COMPLETE” when scheduler job iterates all available poll schemes and submit related tasks of them and all these tasks are either completed or failed. |

**Logging**

Log directory: /repos/esb/polling/log/sales\_rt\_yyyymmdd.log

Upon each poll branch scheme task submission: the scheme info will be written into follow format:

[INFO] [Timestamp] [Scheme name]-[branch code]-[scheme type]-[direction]

Dick??? Any description to update the exception log in DB?

**Entry & Exit Criteria**

The processing job starts once the schedule job invokes.

In the process of the scheduler job, it will be regarded as job failure and send alert email to IT support according to the error’s severity

Error occurs when error reading poll branch scheme data from staging database.

Error occurs when submitting poll branch scheme tasks into subsystem.

#### Screen

N/A

#### Data Fields & Presentation Logic

N/A

#### Screen Objects & Action

N/A

#### User/Security Group

This is a batch process belonging to the SYSTEM. So all auditable fields

#### Assumptions/Constraints

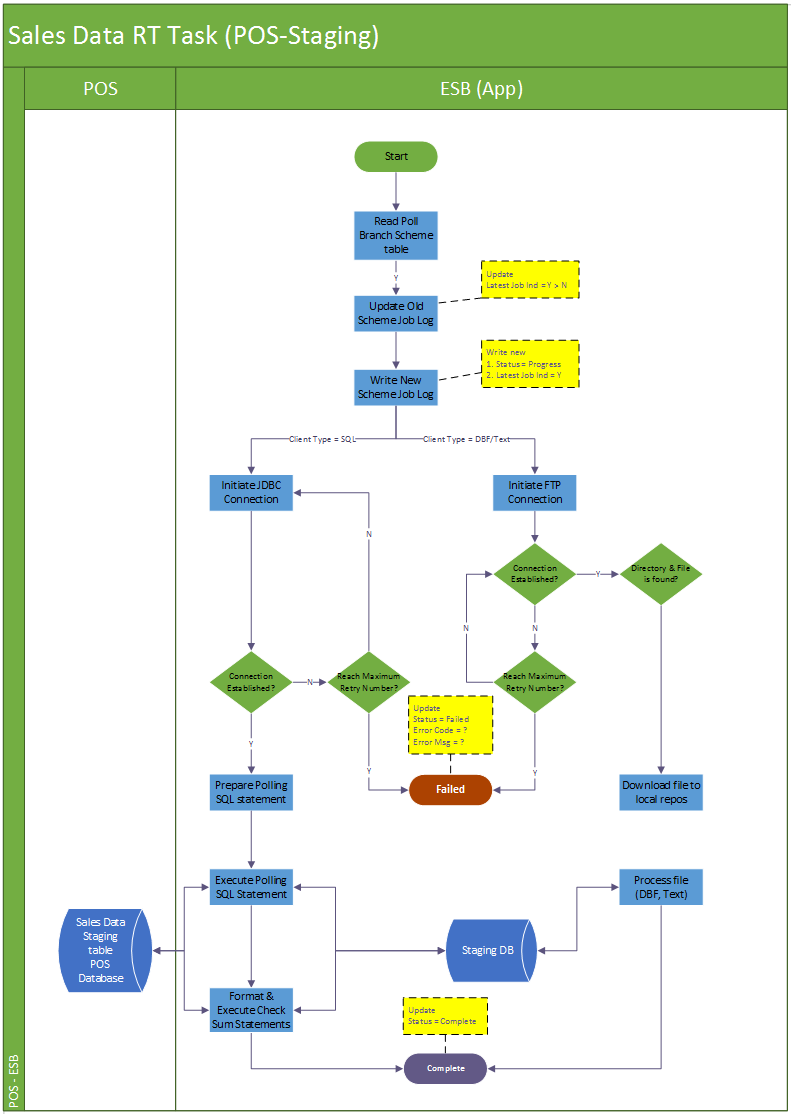
N/A

#### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Failed to read poll scheme data. |
|  |  | Failed submit poll scheme processing task. |
|  |  | Failed to download target file |
|  |  | Failed to process records into staging database |

### Sales Data Real Time (POS - Staging) Task

#### Process/Work Flow



Add back arrow

**Job Details**

The scheduler job is considered as the commander to handle the life cycle of poll scheme tasks which it’s responsible for. With a proper criteria selection, the scheduler is supposed to submit a series of tasks into the pull and queue up for their own process. And it logs down the running process in job logs and log files.

Summarily the task executes in below steps:

Step1. Firstly, it will update the previous job log’s latest job indicator from ‘Y’ to ‘N’. Onwards newly create scheduler job log is having the status “PROGRESS”. The latest batch indicator for the new record will be marked as “Y”.

Step 2. Initiate the JDBC connection or file input from target source according to the client type.

Step 3. When the client type is SQL, use the poll scheme info to format the polling SQL statement to processing the polling. If the client type is DBF, copy the file to local source folder, and initiate a JDBC connection upon the file to process. If the client type is text file, also copy the file to local source folder and use program process.

Step 4. After the polling SQL is successfully executed, a check sum will check upon the processed record upon the total record counts

Step 5. Upon any errors in the task submission and cannot be resolved (e.g. by retry). The job will abort and keep the job status to reflect the problematic stage “FAILED”. If the processing source is file-based, move the file to error folder.

Step 5. If all steps of the task are successfully submitted, the task is regarded as complete and update the job status to “COMPLETE” with a record number of schemes processed. If the processing source is file-based, move the file to backup folder.

|  |  |  |
| --- | --- | --- |
| **Job Name** | **Status** | **Description** |
| Sales Real Time Polling Task | “PROGRESS”  “FAILED”  “COMPLETE” | “PROGRESS” when task is submitted and successfully insert log into DB.  “FAILED” when encounter errors processing the source and the poll branch scheme.  “COMPLETE” when the completes the polling data against the target branch schemes without any errors. |

Dick??? Can we use an example to explain the above operation?

**Logging**

Log directory: /repos/esb/polling/log/sales\_rt\_yyyymmdd.log

Upon each poll branch scheme task submission: the scheme info will be written into follow format:

**File System**

/repos/esb/polling/input/dbf/source/filename\_yyyymmddhh24miss.dfb

/repos/esb/polling/input/dbf/backup/filename\_yyyymmddhh24miss.dfb 🡸 dick??? dbf file? Suggest to use more realistic file name instead of filename\_yyyy….

/repos/esb/polling/input/dbf/error/filename\_yyyymmddhh24miss.dbf

/repos/esb/polling/input/text/source/filename\_yyyymmddhh24miss.txt

/repos/esb/polling/input/text/backup/filename\_yyyymmddhh24miss.txt

/repos/esb/polling/input/text/error/filename\_yyyymmddhh24miss.txt

[INFO] [Timestamp] [Scheme name]-[branch code]-[scheme type]-[direction]

**Entry & Exit Criteria**

The sales data real time processing task is submitted by the sales data real time scheduler job (session 6.6.4).

In the process of the task, it will be regarded as job failure and send alert email to IT support according to the error’s severity

Error occurs when error initiating DB connection to target data source.

Error occurs when error retrieving target source file from target directory.

Error occurs when submitting poll branch scheme tasks into subsystem.

#### Screen

N/A

##### Data Fields & Presentation Logic

N/A

##### Screen Objects & Action

N/A

##### User/Security Group

This is a batch process belonging to the SYSTEM. So all auditable fields

##### Assumptions/Constraints

N/A

##### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Failed to establish FTP connection. |
|  |  | Target directory or file not found. |
|  |  | Failed to download target file. |
|  |  | Failed to establish JDBC connection. |
|  |  | Failed to process records into staging database. |
|  |  | Check sum failed on data in destination table [Table Name]. |

## Sales Data EOD Polling Batch Job (POS-Staging)

### Description

The scheduled batch job runs in the configurable time interval (e.g. 5 mins) to monitor EOD indication data table (e.g. HIST\_POSSYSTEM). This is the pre-requisite batch job of the other batch which push real time sales data from staging database into EDW.

The job is triggered by a scheduler and it works as a task controller, submitting separated standalone tasks. Each task handles one available branch scheme. The job will handle the data in the pre-defined tables under the scheme according to the scheme configuration (e.g. direction, override logic, and validation).

In below session there will be the details descriptions about

1. Sales Data Real Time (POS - Staging) Scheduler
2. Sales Data Real Time (POS - Staging) Task

### Input

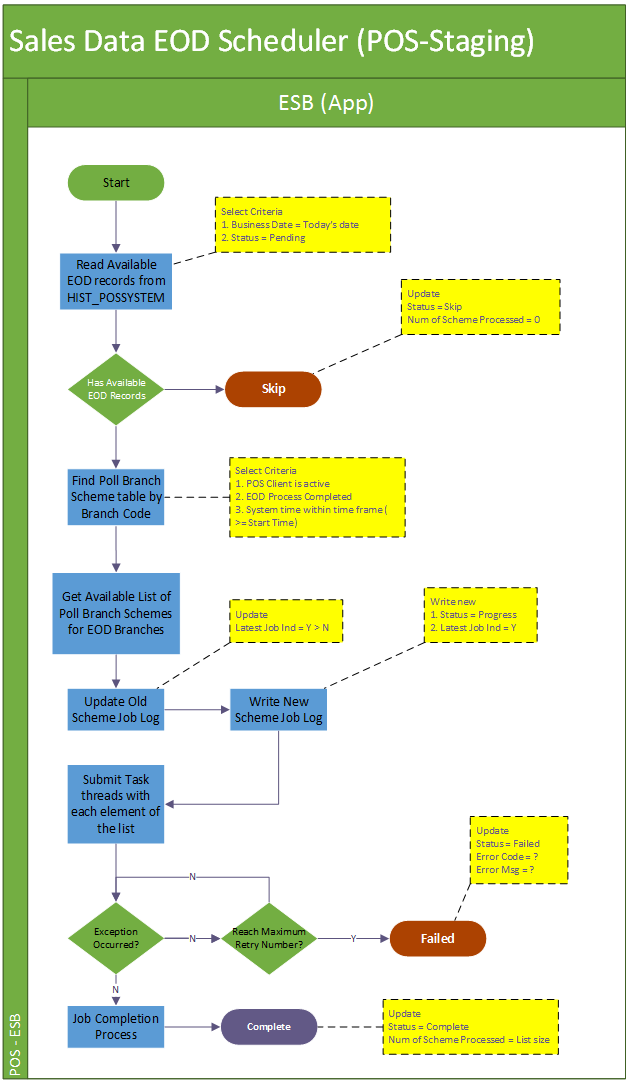
EOD Sales Data in POS client database

### Output

EOD Sales Data in staging database

### Sales Data EOD (POS - Staging) Scheduler

#### Process/Work Flow



**Dick??? Now to do ‘business date = today’s date’ checking. Today’s date of ESB application server? How to cater the today after mid-night, using today’s date - 1? How about 7x24 handling? Some branch will not operate on Sunday. What happen when polling this branch on Sunday?**

**Job Details**

The scheduler job is considered as the commander to handle the life cycle of EOD tasks which it is responsible for. It monitors the EOD tables’ records from POS client, when the EOD records is updated, the scheduler is supposed to submit the EOD task for the satisfied POS client. And it logs down the running process in job logs and log files.

Summarily the schedule executes in below steps

Step1. This batch job will be triggered every 5 mins (configured). It would iterate all available data source in poll branch scheme table for the EOD indicator. For SQL, there should be the business date record in HIST\_POSSYSTEM table, for file-based data source, it should be the new EOD file is written in the target directory.

Step 2. Once the EOD indicator is available, update scheduler job log. Firstly update the previous job log’s latest job indicator from ‘Y’ to ‘N’. Onwards newly create scheduler job log is having the status “PROGRESS”. The latest batch indicator for the new record will be marked as “Y”.

Step 3. Submit EOD processing tasks for all eligible poll branch schemes which has the EOD indicator available (verified in step 1).

Step 4. Upon any errors in the task submission and cannot be resolved (e.g. by retry). The job will abort and keep the job status to reflect the problematic stage “FAILED”.

Step 5. After the task submission, the scheduler is regarded as complete and update the job status to “COMPLETE” with a record number of schemes processed.

|  |  |  |
| --- | --- | --- |
| **Job Name** | **Status** | **Description** |
| Sales EOD Polling Scheduler | “PROGRESS”  “FAILED”  “COMPLETE” | “PROGRESS” when scheduler is invoked and started  “FAILED” when encounter errors reading poll scheme data from database or errors submitting tasks.  “COMPLETE” when scheduler job iterates all available poll schemes and submit related tasks of them and all these tasks are either completed or failed. |

**Logging**

Log directory: /repos/esb/polling/log/sales\_eod\_yyyymmdd.log

Upon each poll branch scheme EOD task submission: the scheme info will be written into follow format:

[INFO] [Timestamp] EOD-[Scheme name]-[branch code]-[scheme type]-[direction]

**Entry & Exit Criteria**

The processing job starts once the schedule job invokes.

In the process of the scheduler job, it will be regarded as job failure and send alert email to IT support according to the error’s severity

Error occurs when reading EOD indicator data from the target data sources

Error occurs when looking up/downloading EOD file from target file directory

Error occurs when error reading poll branch scheme data from staging database.

Error occurs when submitting EOD processing tasks into subsystem.

#### Screen

N/A

##### Data Fields & Presentation Logic

N/A

##### Screen Objects & Action

N/A

##### User/Security Group

This is a batch process belonging to the SYSTEM. So all auditable fields

##### Assumptions/Constraints

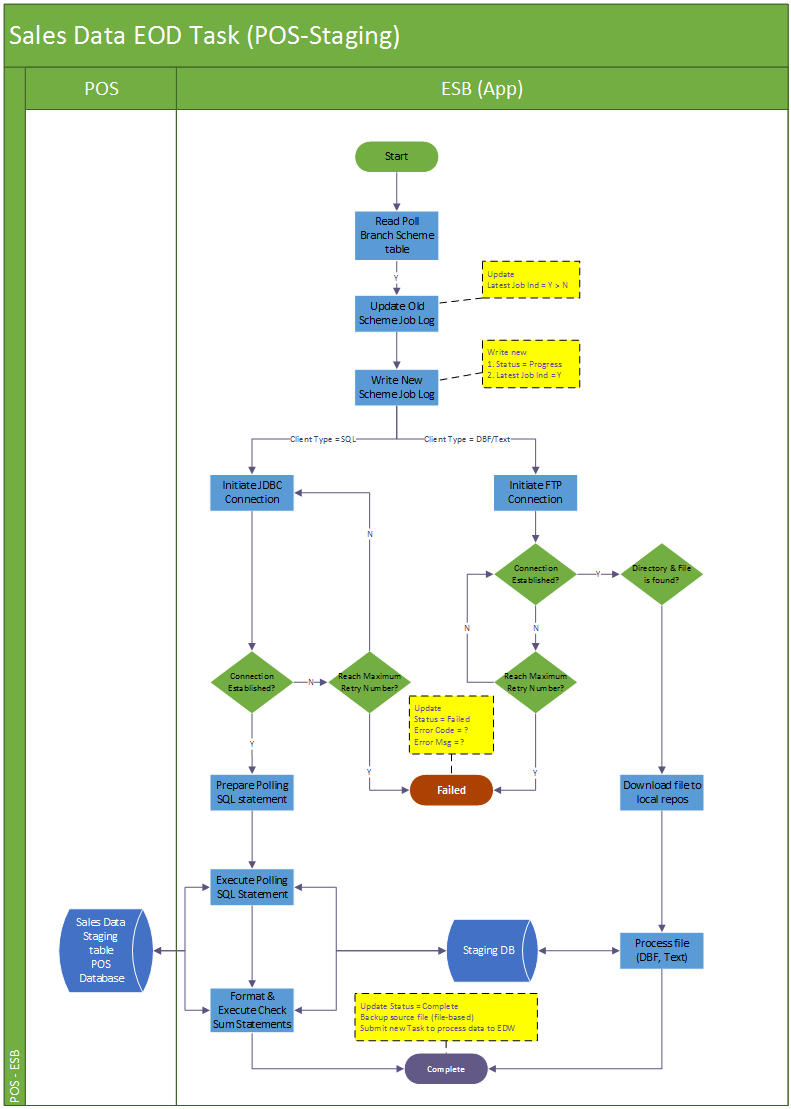
N/A

##### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Failed to establish FTP connection. |
|  |  | Failed to download target file. |
|  |  | Failed to read target text/DBF file. File is broken. |
|  |  | Failed to establish JDBC connection. |

### Sales Data EOD (POS - Staging) Task

#### Process/Work Flow



**Job Details**

The single EOD task job is considered as a single EOD process towards the target poll branch scheme, which is submitted by the EOD scheduler.

Summarily the task executes in below steps:

Step1. Firstly, it will update the previous job log’s latest job indicator from ‘Y’ to ‘N’. Onwards newly create scheduler job log is having the status “PROGRESS”. The latest batch indicator for the new record will be marked as “Y”.

Step 2. Though the parent scheduler has already very the data source (SQL or file), because this is a separate process, the task will also try to initiate the JDBC connection or try to download the file from target source according to the client type.

Step 3. When the client type is SQL, use the poll scheme info to format the polling SQL statement to processing the polling. If the client type is DBF, copy the file to local source folder, and initiate a JDBC connection upon the file to process. If the client type is text file, also copy the file to local source folder and use program process. 🡸 Dick??? how “use program process”?

Step 4. After the polling SQL is successfully executed, a check sum will check upon the processed record upon the total record counts

Step 5. Upon any errors in the task submission and cannot be resolved (e.g. by retry). The job will abort and keep the job status to reflect the problematic stage “FAILED”. If the processing source is file-based, move the file to error folder.

Step 5. If all steps of the task are successfully submitted, the task is regarded as complete and update the job status to “COMPLETE” with a record number of schemes processed. If the processing source is file-based, move the file to backup folder. At the end of the successful completion process, the task will change the poll scheme’s direction to “STG\_TO\_EDW” and submit a new task upon it.

|  |  |  |
| --- | --- | --- |
| **Job Name** | **Status** | **Description** |
| Sales EOD Polling Task | “PROGRESS”  “FAILED”  “COMPLETE” | “PROGRESS” when task is submitted and successfully insert log into DB.  “FAILED” when encounter errors processing the source and the poll branch scheme.  “COMPLETE” when the completes the polling data against the target branch schemes without any errors. |

**Logging**

Log directory: /repos/esb/polling/log/sales\_eod\_yyyymmdd.log

Upon each EOD task submission: the scheme info will be written into follow format:

**File System**

/repos/esb/polling/input/dbf/source/eod\_filename\_yyyymmddhh24miss.dfb

/repos/esb/polling /input/dbf/backup/eod\_filename\_yyyymmddhh24miss.dfb

/repos/esb/polling/input/dbf/error/eod\_filename\_yyyymmddhh24miss.dbf

/repos/esb/polling/input/text/source/eod\_filename\_yyyymmddhh24miss.txt

/repos/esb/polling/input/text/backup/eod\_filename\_yyyymmddhh24miss.txt

/repos/esb/polling/input/text/error/eod\_filename\_yyyymmddhh24miss.txt

(original file name)

Logging details:

[INFO] [Timestamp] EOD-[schemeName]-[branchCode]-[schemeType]-[direction] Start

[INFO] [Timestamp] EOD-[schemeType]-[fromTable]-[toTable]-[PK=?]-[checksum=?]-[override = ?] start execution

[INFO] [Timestamp] EOD-[schemeType]-[fromTable]-[toTable]-[PK=?]-[checksum=?]-[override = ?] processed = ?

[ERROR] [Timestamp] EOD-[schemeType]-[fromTable]-[toTable]-[PK=?]-[checksum=?]-[override = ?] SQL Error return Code = ?

….

[INFO] [Timestamp] EOD-[schemeName]-[branchCode]-[schemeType]-[direction] End

**Entry & Exit Criteria**

The sales data EOD processing task is submitted by the sales data EOD scheduler job (session 6.7.4) and triggered by corresponding EOD indicator the POS client from different kinds of data sources

In the process of the task, it will be regarded as job failure and send alert email to IT support according to the error’s severity.

Error occurs when error initiating DB connection to target data source.

Error occurs when error retrieving target source file from target directory.

Error occurs when submitting poll branch scheme tasks into subsystem.

#### Screen

N/A

##### Data Fields & Presentation Logic

N/A

##### Screen Objects & Action

N/A

##### User/Security Group

This is a batch process belonging to the SYSTEM. So all auditable fields

##### Assumptions/Constraints

N/A

##### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Failed to establish FTP connection |
|  |  | Target directory or file not found |
|  |  | Failed to download target file |
|  |  | Failed to process records into staging database |

## Pricing/Master Data Generation (Pricing Server - Staging)

### Description

The generation of the pricing/master data will be considered to migrate the corresponding SQL server agent job from IT51/52/53 into a transitional database server (details described in impact analysis report).

The pricing/master generation process can be regarded as two main process

1. The original SQL server agent job which prepare the data in the pricing server tables
2. The oracle service bus DB adapter to monitor the delta changes of these tables
   1. DB adapters to monitor the master data table updates
   2. DB adapters to monitor the indicating data which represents the completion of the generation (impact provided and detail is explained in the impact analysis report)

### Input

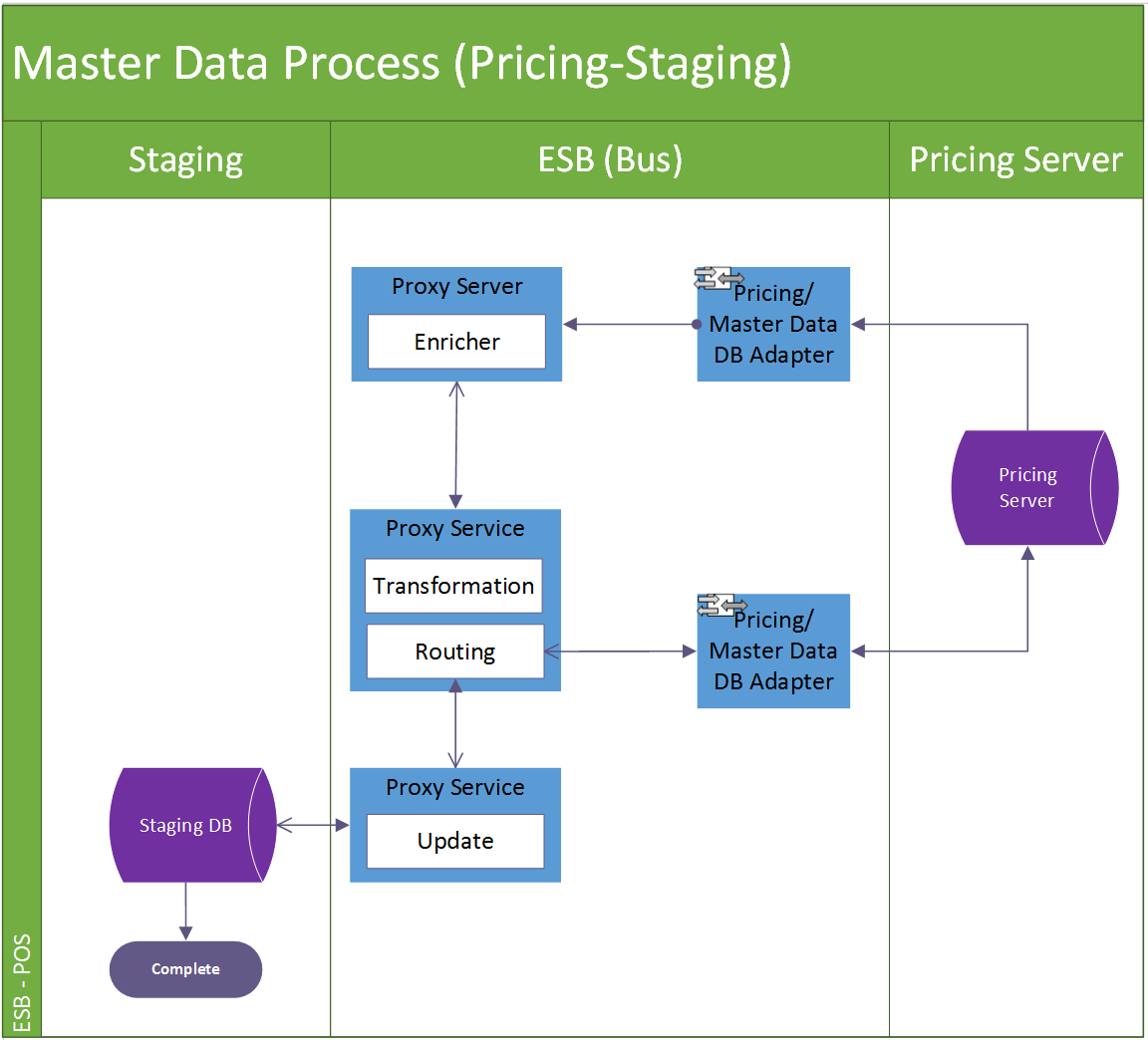
Pricing/master data in pricing server

### Output

Pricing/master data in staging database

### Pricing/Master Data Process (EDW - Staging)

#### Process/Work Flow



**Data process detail**

Step 1. DB adapters monitor in add/update in the target master tables. When there is new data coming in, capture the delta’s data sequencing ID as response. 🡸 Dick??? Explain in more detail more to monitor the data changes or what is the criteria for monitoring? Do we need to discuss with Maxim’s?

Step 2. The pre-defined proxy service will use the captured ID to invoke an message enricher service so as to retrieved all details of the delta record.

Step 3. The enricher service will conduct a certain transformation and routing if necessary, eventually call the web service to update the delta data into staging DB.

#### Screen

N/A

##### Data Fields & Presentation Logic

N/A

##### Screen Objects & Action

N/A

##### User/Security Group

N/A

##### Assumptions/Constraints

N/A

##### Error & Exception List

N/A

## Pricing/Master Data Distribution Batch Job (Staging-POS)

### Description

While the pricing/master data is finished being updated into staging database, the main batch scheduler will run in every 15mins, to distribute the up-to-date pricing/master data into corresponding POS client according to their branch code as long as the pricing/master data generation is indicated with completion upon certain branch.

The pricing/master data distribution tasks is triggered by a proxy server and it works as a task controller, submitting separated standalone tasks. Each task handles one available branch scheme. The job will handle the data in the pre-defined tables under the scheme according to the scheme configuration (e.g. direction, override logic, and validation) and from table to table. Dick??? We need to explain what is the override logic or validation, etc?

In below session there will be the details descriptions about

1. Pricing/Master data distribution (POS - Staging) Scheduler
2. Pricing/Master data distribution (POS - Staging) Task

### Input

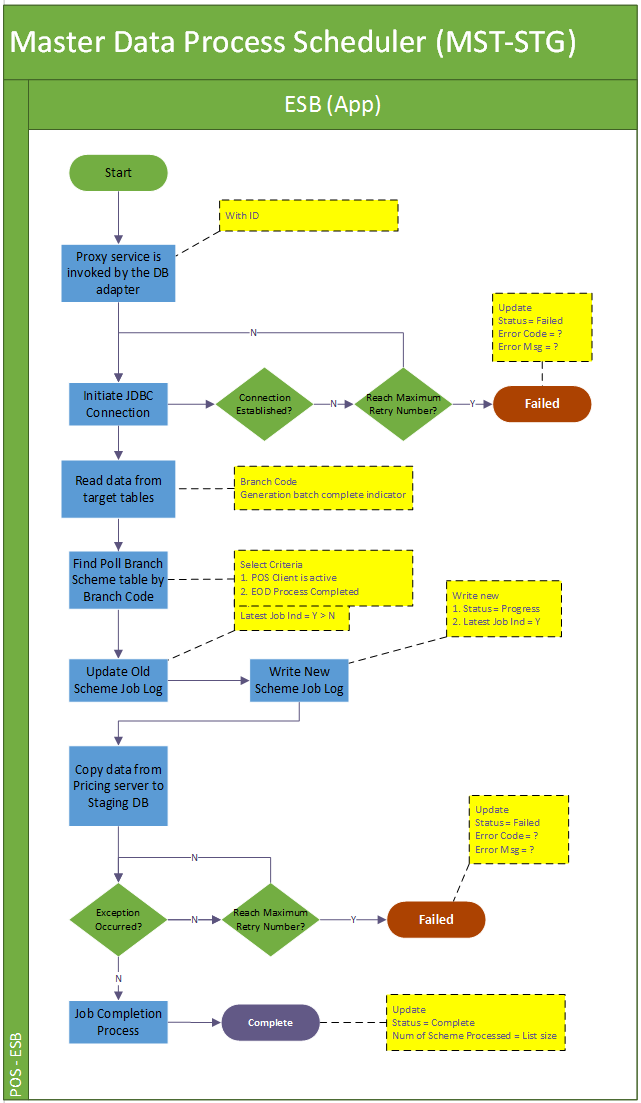
Pricing/Master data in staging DB

### Output

Pricing/Master data in POS clients

### Pricing/Master Data Distribution Scheduler (Staging-POS)

#### Process/Work Flow



**Job Details**

The scheduler job is the main controller of all the pricing/master data distribution. It is invoked by the proxy service which is trigger by the OSB DB Adapter moniting the indication data of the pricing/master data generation in pricing server. The scheduler not only submits pricing/master data distribution task upon specific poll branch scheme but also logs down the running process in job logs and log files.

Summarily the schedule executes in below steps

Step1. This batch job will be triggered every 15 mins (configured). It would iterate all available data source in poll branch scheme table which are under the poll scheme type = “MASTER” and direction = “STG\_TO\_POS”.

Step 2. Upon the scheduler job’s triggering, firstly it updates the previous job log’s latest job indicator from ‘Y’ to ‘N’. Onwards newly create scheduler job log is having the status “PROGRESS”. The latest batch indicator for the new record will be marked as “Y”.

Step 3. Submit pricing/master data distribution tasks for all eligible poll branch schemes selected in step 1.

Step 4. Upon any errors in the task submission and cannot be resolved (e.g. by retry). The job will abort and keep the job status to reflect the problematic stage “FAILED”. Dick??? What means “cannot be resolved”?

Step 5. After the task submission, the scheduler is regarded as complete and update the job status to “COMPLETE” with a record number of schemes processed.

|  |  |  |
| --- | --- | --- |
| **Job Name** | **Status** | **Description** |
| Sales EOD Polling Scheduler | “PROGRESS”  “FAILED”  “COMPLETE” | “PROGRESS” when scheduler is invoked and started  “FAILED” when encounter errors reading poll scheme data from database or errors submitting tasks.  “COMPLETE” when scheduler job iterates all available poll schemes and submit related tasks of them and all these tasks are either completed or failed. |

**Logging**

Log directory: /repos/esb/polling/log/master\_yyyymmdd.log

Upon each poll branch scheme pricing/master distribution task submission: the scheme info will be written into follow format:

[INFO] [Timestamp] MASTER-[Scheme name]-[branch code]-[scheme type]-[direction]

**Entry & Exit Criteria**

The processing job starts once the schedule job is invoked by the proxy service.

In the process of the scheduler job, it will be regarded as job failure and send alert email to IT support according to the error’s severity

Error occurs when reading pricing/master data from the target data sources

Error occurs when error initiating JDBC connection toward poll branch scheme data source.

Error occurs when submitting distribution task into subsystem.

#### Screen

N/A

##### Data Fields & Presentation Logic

N/A

##### Screen Objects & Action

N/A

##### User/Security Group

This is a batch process belonging to the SYSTEM. So all auditable fields

##### Assumptions/Constraints

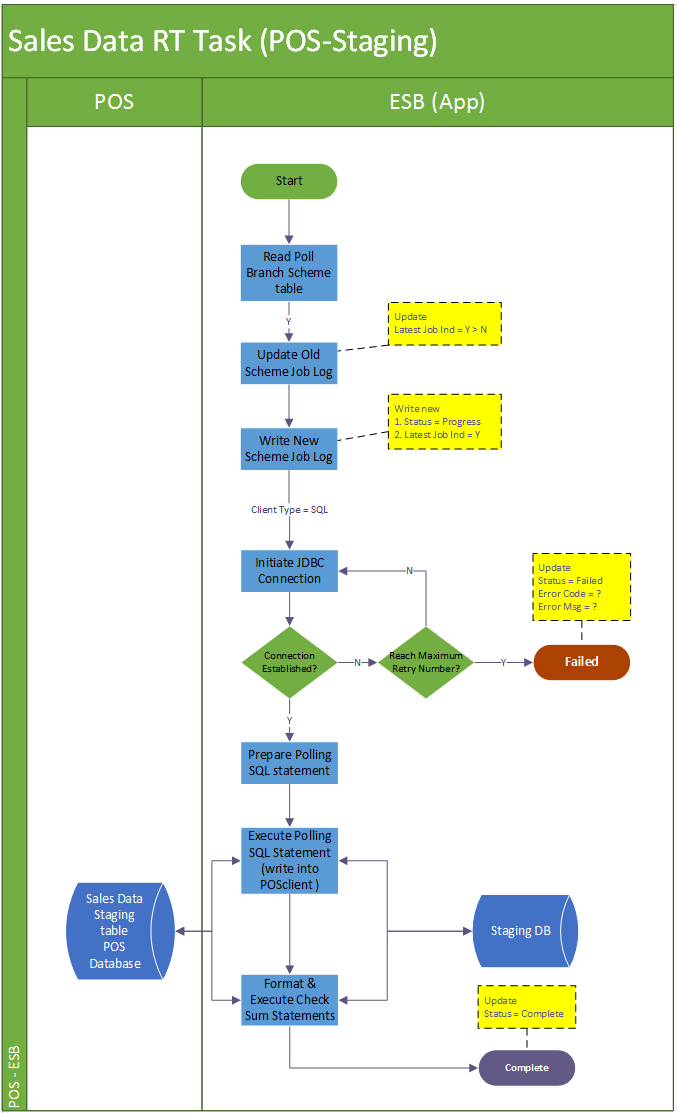
N/A

##### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Failed to submit distribution task. |
|  |  | Failed to establish JDBC connection. |

### Pricing/Master Data Distribution Task (Staging-POS)

#### Process/Work Flow



**Job Details**

The single single distribution task job is designed to handle master data towards the target poll branch scheme.

Summarily the task executes in below steps:

Step1. Firstly, it will update the previous job log’s latest job indicator from ‘Y’ to ‘N’. Onwards newly create scheduler job log is having the status “PROGRESS”. The latest batch indicator for the new record will be marked as “Y”.

Step 2. The task will also try to initiate the JDBC connection toward the target POS client.

Step 3. Use the poll scheme info to format the polling SQL statement to processing the polling.

Step 4. After the distribution SQL is successfully executed and copy all wanted master data into POS client, update the corresponding indicator in the master data table in staging database representing the staging and POS client are synchronized.

Step 5. Upon any errors in the task submission and cannot be resolved (e.g. by retry). The job will roll back to discard the change, abort and keep the job status to reflect the problematic stage “FAILED”. If the processing source is file-based, move the file to error folder.

Step 5. If all steps of the task are run successfully, the task is regarded as complete and update the job status to “COMPLETE” with a record number of schemes processed.

|  |  |  |
| --- | --- | --- |
| **Job Name** | **Status** | **Description** |
| Sales EOD Polling Task | “PROGRESS”  “FAILED”  “COMPLETE” | “PROGRESS” when task is submitted and successfully insert log into DB.  “FAILED” when encounter errors processing the source and the poll branch scheme.  “COMPLETE” when the completes the polling data against the target branch schemes without any errors. |

**Logging**

Log directory: /repos/esb/polling/log/master\_yyyymmdd.log

Upon each master distribution task processing: the scheme info will be written into follow format:

Logging details:

[INFO] [Timestamp] MASTER -[schemeName]-[branchCode]-[schemeType]-[direction] Start

[INFO] [Timestamp] MASTER -[schemeType]-[fromTable]-[toTable]-[PK=?]-[checksum=?]-[override = ?] Start Execution

[INFO] [Timestamp] MASTER -[schemeType]-[fromTable]-[toTable]-[PK=?]-[checksum=?]-[override = ?] Processed = ?

[ERROR] [Timestamp] MASTER -[schemeType]-[fromTable]-[toTable]-[PK=?]-[checksum=?]-[override = ?] SQL Error return Code = ?

….

[INFO] [Timestamp] MASTER-[schemeName]-[branchCode]-[schemeType]-[direction] End

**Entry & Exit Criteria**

The pricing/master data distribution task is submitted by the pricing/master data distribution scheduler job (session 6.10.4).

In the process of the task, it will be regarded as job failure and send alert email to IT support according to the error’s severity.

Error occurs when error initiating DB connection to target data source with retry.

Error occurs when executing distribution SQL statement to write pricing/master data into POS client.

#### Screen

N/A

##### Data Fields & Presentation Logic

Refer to Appendix C “Master Data Table Field Mapping”

##### Screen Objects & Action

N/A

##### User/Security Group

This is a batch process belonging to the SYSTEM. So all auditable fields

##### Assumptions/Constraints

N/A

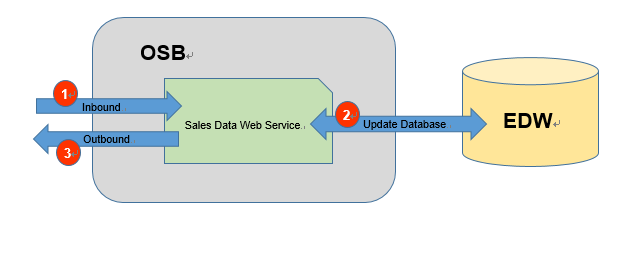
##### Error & Exception List

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Severity** | **Error Message** |
|  |  | Failed to establish JDBC connection |
|  |  | Failed to process records into target POS client [branch code = ?] [SQLCOD = ?]. |

# Interface Design

## Sales Data Web Service

### Process/Work Flow



1. Send the request the OSB Restful Sales Data Web Service through HTTP.
2. The service will update the EDW by the request content.
3. The Service will return the status to indicate whether the update is success.

Interface Design

Inbound Interface Format (Remarks and JSON Example to be provided)

<salesOrder>

| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | branchId | Maxim Branch ID | Y | String |  | <branchId>MXM  </branchId> | {“branchId”:”MXM”} |
| 2 | businessDate | Transaction Business Date | Y | Date |  | <businessDate>2016-01-01</businessDate> | {“businessDate”:”2016-01-01”} |
| 3 | orderNo | Transaction order no. | Y | String |  | <orderNo>OR12345  </orderNo> | {“orderNo”:”OR12345”} |
| 4 | orderSequence | Transaction order sequence | Y | Numeric (5,0) |  | < orderSequence>1  </orderSequence> | {“orderSequence”:”1”} |
| 5 | eventNo | Event Order no. | N | String |  | <eventNo>  </eventNo> | {"eventNo":""} |
| 6 | tableNo | Table no. | Y | String |  | <tableNo>  </tableNo> | {"tableNo":""} |
| 7 | guest | Number of guests | Y | Numeric (5,0) |  | <guest>  </guest> | {"guest":""} |
| 8 | printTimes | Number of times the pay check is printed | Y | Numeric (5,0) |  | <printTimes>  </printTimes> | {"printTimes":""} |
| 9 | printUserId1 | Print User ID 1 | N | String |  | <printUserId1>  </printUserId1> | {"printUserId1":""} |
| 10 | printUserId2 | Print User ID 2 | N | String |  | <printUserId2>  </printUserId2> | {"printUserId2":""} |
| 11 | printUserId3 | Print User ID 3 | N | String |  | <printUserId3>  </printUserId3> | {"printUserId3":""} |
| 12 | currencyCode | Currency Code | Y | String |  | <currencyCode>  </currencyCode> | {"currencyCode":""} |
| 13 | bevamt | Beverage Sales Amount | Y | Numeric (19,4) |  | <bevamt>  </bevamt> | {"bevamt":""} |
| 14 | foodamt | Food Sales Amount | Y | Numeric (19,4) |  | <foodamt>  </foodamt> | {"foodamt":""} |
| 15 | servamt | Service Charge Amount | Y | Numeric (19,4) |  | <servamt>  </servamt> | {"servamt":""} |
| 16 | discCode | Discount Code | N | String |  | <discCode>  </discCode> | {"discCode":""} |
| 17 | discRatio | Discount Ratio | Y | Numeric (19,4) |  | <discRatio>  </discRatio> | {"discRatio":""} |
| 18 | discAmt | Discount Amount | Y | Numeric (19,4) |  | <discAmt>  </discAmt> | {"discAmt":""} |
| 19 | taxCode | Tax Code | N | String |  | <taxCode>  </taxCode> | {"taxCode":""} |
| 20 | taxAmt | Tax Amount | Y | Numeric (19,4) |  | <taxAmt>  </taxAmt> | {"taxAmt":""} |
| 21 | orderType | Order type | Y | String |  | <orderType>  </orderType> | {"orderType":""} |
| 22 | spTransType |  | N | String |  | <spTransType>  </spTransType> | {"spTransType":""} |
| 23 | transType | Transaction type | Y | String |  | <transType>  </transType> | {"transType":""} |
| 24 | chargeDept | Charging Department | N | String |  | <chargeDept>  </chargeDept> | {"chargeDept":""} |
| 25 | chargeReason | Charging Reason | N | String |  | <chargeReason>  </chargeReason> | {"chargeReason":""} |
| 26 | chargeUserId | Charge User ID | N | String |  | <chargeUserId>  </chargeUserId> | {"chargeUserId":""} |
| 27 | openTillNo | Till machine on which the sales order is opened | Y | String |  | <openTillNo>  </openTillNo> | {"openTillNo":""} |
| 28 | closeTillNo | Till machine on which the sales order is closed | Y | String |  | <closeTillNo>  </closeTillNo> | {"closeTillNo":""} |
| 29 | openUserId | User ID of staff who opens the sales order | Y | String |  | <openUserId>  </openUserId> | {"openUserId":""} |
| 30 | closeUserId | User ID of staff who closes the sales order | Y | String |  | <closeUserId>  </closeUserId> | {"closeUserId":""} |
| 31 | checkInDateTime | Sales order check-in date time | Y | datetime |  | <checkInDateTime>  </checkInDateTime> | {"checkInDateTime":""} |
| 32 | checkOutDateTime | Sales order check-out date time | lY | datetime |  | <checkOutDateTime>  </checkOutDateTime> | {"checkOutDateTime":""} |
| 33 | voidFlag | Void sales order flag | Y | String |  | <voidFlag>  </voidFlag> | {"voidFlag":""} |
| 34 | voidDateTime | Void sale date time | Y | datetime |  | <voidDateTime>  </voidDateTime> | {"voidDateTime":""} |
| 35 | voidId | Void ID | N | String |  | <voidId>  </voidId> | {"voidId":""} |
| 36 | reasonCode | Void reason | N | String |  | <reasonCode>  </reasonCode> | {"reasonCode":""} |
| 37 | remarks01 | Remark 1 | N | String |  | <remarks01>  </remarks01> | {"remarks01":""} |
| 38 | remarks02 | Remark 2 | N | String |  | <remarks02>  </remarks02> | {"remarks02":""} |
| 39 | remarks03 | Remark 3 | N | String |  | <remarks03>  </remarks03> | {"remarks03":""} |
| 40 | dataSource | Data Source of the sales (e.g. “OM” for Order Management; “Lunch” for Lunch Box system) | Y | String |  | <dataSource>  </dataSource> | {"dataSource":""} |
| 41 | lastUpdateUser | Last update by (user id by which the convert process or adjustment is done) | Y | String |  | <lastUpdateUser>  </lastUpdateUser> | {"lastUpdateUser":""} |
| 42 | lastUpdateTime | Last update date time | Y | datetime |  | <lastUpdateTime>  </lastUpdateTime> | {"lastUpdateTime":""} |
| 43 | workstationName | Server at which the convert process take place | Y | String |  | <workstationName>  </workstationName> | {"workstationName":""} |
| 44 | appNmae | Program name of convert process | Y | String |  | <appNmae>  </appNmae> | {"appNmae":""} |
| 45 | runno | Run ID of the convert process | Y | Numeric (5,0) |  | <runno>  </runno> | {"runno":""} |
| 46 | status | Whether or not the sales data is counted as store sales | N | String |  | <status>  </status> | {"status":""} |

<salesItem>

| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | branchId | Maxim Branch ID | Y | String |  | < branchId>MXM  </branchId> | {“branchId”:”MXM”} |
| 2 | businessDate | Transaction Business Date | Y | Date |  | < businessDate>2016-01-01  </businessDate> | {“businessDate”:”2016-01-01”} |
| 3 | orderNo | Transaction order no. | Y | String |  | < orderNo>OR12345  </orderNo> | {“orderNo”:”OR12345”} |
| 4 | orderSequence | Transaction order sequence | Y | Numeric (5,0) |  | < orderSequence>1  </orderSequence> | {“orderSequence”:”1”} |
| 5 | eventNo | Event Order no. | N | String |  | <eventNo>  </eventNo> | {"eventNo":""} |
| 6 | itemSequence | Transaction item sequence | Y | Numeric (5,0) |  | <itemSequence>  </itemSequence> | {"itemSequence":""} |
| 7 | subitemSequence | Transaction item sub-sequence | Y | Numeric (5,0) |  | <subitemSequence>  </subitemSequence> | {"subitemSequence":""} |
| 8 | itemno | Item no. | Y | Numeric (5,0) |  | <itemno>  </itemno> | {"itemno":""} |
| 9 | category | Item category | Y | String |  | <category>  </category> | {"category":""} |
| 10 | itemDescription | Item code description | Y | String |  | <itemDescription>  </itemDescription> | {"itemDescription":""} |
| 11 | listPrice | Listed Unit price | Y | Numeric (19,4) |  | <listPrice>  </listPrice> | {"listPrice":""} |
| 12 | price | Unit price | Y | Numeric (19,4) |  | <price>  </price> | {"price":""} |
| 13 | qty | Transaction Quantity | Y | Numeric (19,4) |  | <qty>  </qty> | {"qty":""} |
| 14 | itemTot | Item sales total | Y | Numeric (19,4) |  | <itemTot>  </itemTot> | {"itemTot":""} |
| 15 | servFlag | Service type | Y | String |  | <servFlag>  </servFlag> | {"servFlag":""} |
| 16 | setmenu | Set menu item code | N | String |  | <setmenu>  </setmenu> | {"setmenu":""} |
| 17 | setFlag | Set menu item indicator | Y | String |  | <setFlag>  </setFlag> | {"setFlag":""} |
| 18 | discCode | Item Discount Code | N | String |  | <discCode>  </discCode> | {"discCode":""} |
| 19 | discRatio | Item Discount Ratio | Y | Numeric (19,4) |  | <discRatio>  </discRatio> | {"discRatio":""} |
| 20 | discAmt | Item Discount Amount | Y | Numeric (19,4) |  | <discAmt>  </discAmt> | {"discAmt":""} |
| 21 | modifiedAmt | Modified Amount | Y | Numeric (19,4) |  | <modifiedAmt>  </modifiedAmt> | {"modifiedAmt":""} |
| 22 | cost | Item Cost | Y | Numeric (19,4) |  | <cost>  </cost> | {"cost":""} |
| 23 | taxCode | Tax Code | N | String |  | <taxCode>  </taxCode> | {"taxCode":""} |
| 24 | taxAmt | Tax Amount | Y | Numeric (19,4) |  | <taxAmt>  </taxAmt> | {"taxAmt":""} |
| 25 | orderType | Order type | Y | String |  | <orderType>  </orderType> | {"orderType":""} |
| 26 | transType | Transaction type | Y | String |  | <transType>  </transType> | {"transType":""} |
| 27 | tillNo | Till machine on which the ordered item is made | Y | String |  | <tillNo>  </tillNo> | {"tillNo":""} |
| 28 | inputDateTime | Sales item input date time | Y | datetime |  | <inputDateTime>  </inputDateTime> | {"inputDateTime":""} |
| 29 | inputId | User ID of staff who input the item | Y | String |  | <inputId>  </inputId> | {"inputId":""} |
| 30 | voidFlag | Void sales item flag | Y | String |  | <voidFlag>  </voidFlag> | {"voidFlag":""} |
| 31 | voidDateTime | Void sale date time | Y | datetime |  | <voidDateTime>  </voidDateTime> | {"voidDateTime":""} |
| 32 | voidId | Void ID | Y | String |  | <voidId>  </voidId> | {"voidId":""} |
| 33 | reasonCode | Void reason | N | String |  | <reasonCode>  </reasonCode> | {"reasonCode":""} |
| 34 | remarks01 | Remark 1 | N | String |  | <remarks01>  </remarks01> | {"remarks01":""} |
| 35 | remarks02 | Remark 2 | N | String |  | <remarks02>  </remarks02> | {"remarks02":""} |
| 36 | remarks03 | Remark 3 | N | String |  | <remarks03>  </remarks03> | {"remarks03":""} |
| 37 | dataSource | Data Source of the sales (e.g. “OM” for Order Management; “Lunch” for Lunch Box system) | Y | String |  | <dataSource>  </dataSource> | {"dataSource":""} |
| 38 | lastUpdateUser | Last update by (user id by which the convert process or adjustment is done) | Y | datetime |  | <lastUpdateUser>  </lastUpdateUser> | {"lastUpdateUser":""} |
| 39 | lastUpdateTime | Last update date time | Y | String |  | <lastUpdateTime>  </lastUpdateTime> | {"lastUpdateTime":""} |
| 40 | workstationName | Server at which the convert process take place | Y | String |  | <workstationName>  </workstationName> | {"workstationName":""} |
| 41 | appNmae | Program name of convert process | Y | String |  | <appNmae>  </appNmae> | {"appNmae":""} |
| 42 | runno | Run ID of the convert process | Y | Numeric (10,0) |  | <runno>  </runno> | {"runno":""} |
| 43 | status | Whether or not the sales data is counted as store sales | N | String |  | <status>  </status> | {"status":""} |

<salesPay>

| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | branchId | Maxim Branch ID | Y | String |  | < branchId>MXM  </branchId> | {“branchId”:”MXM”} |
| 2 | businessDate | Transaction Business Date | Y | Date |  | < businessDate>2016-01-01  </businessDate> | {“businessDate”:”2016-01-01”} |
| 3 | orderNo | Transaction order no. | Y | String |  | < orderNo>OR12345  </orderNo> | {“orderNo”:”OR12345”} |
| 4 | orderSequence | Transaction order sequence | Y | Numeric (5,0) |  | < orderSequence>1  </orderSequence> | {“orderSequence”:”1”} |
| 5 | eventNo | Event Order no. | N | String |  | <eventNo>  </eventNo> | {"eventNo":""} |
| 6 | paymentSequence | Payment sequence | Y | Numeric (5,0) |  | <paymentSequence> </paymentSequence> | {"paymentSequence":""} |
| 7 | paymenType | Payment Type | Y | String |  | <paymenType> </paymenType> | {"paymenType":""} |
| 8 | posPayType | POS Payment Type | Y | String |  | <posPayType> </posPayType> | {"posPayType":""} |
| 9 | paidQuantity | Paid quantity | Y | Numeric (19,4) |  | <paidQuantity> </paidQuantity> | {"paidQuantity ":""} |
| 10 | paidCurrency | Paid currency | Y | String |  | <paidCurrency> </paidCurrency> | {"paidCurrency":""} |
| 11 | paidAmount | Paid amount | Y | Numeric (19,4) |  | <paidAmount> </paidAmount> | {"paidAmount":""} |
| 12 | change |  | Y | Numeric (19,4) |  | <change> </change> | {"change":""} |
| 13 | tips | Tips paid | Y | Numeric (19,4) |  | <tips> </tips> | {"tips":""} |
| 14 | localCurrency | POS local currency | Y | String |  | <localCurrency> </localCurrency> | {"localCurrency":""} |
| 15 | localAmount | Paid amount in local currency | Y | Numeric (19,4) |  | <localAmount> </localAmount> | {"localAmount":""} |
| 16 | exchangeRate | Exchange rate to change Paid amount to Local amount | Y | Numeric (19,4) |  | <exchangeRate> </exchangeRate> | {"exchangeRate":""} |
| 17 | discType | Payment discount type | Y | String |  | <discType> </discType> | {"discType":""} |
| 18 | discAmt | Payment discount amount | Y | Numeric (19,4) |  | <discAmt> </discAmt> | {"discAmt":""} |
| 19 | cardType | Card type (e.g. Visa, membership card, etc) | N | String |  | <cardType> </cardType> | {"cardType":""} |
| 20 | cardNo | Card Number | N | String |  | <cardNo> </cardNo> | {"cardNo":""} |
| 21 | cardUser | Card holder name | N | String |  | <cardUser> </cardUser> | {"cardUser":""} |
| 22 | refNo | Reference No. | N | String |  | <refNo> </refNo> | {"refNo":""} |
| 23 | traceNo | Trace no. | N | String |  | <traceNo> </traceNo> | {"traceNo":""} |
| 24 | approvalCode | Approval code | N | String |  | <approvalCode> </approvalCode> | {"approvalCode":""} |
| 25 | orderType | Order type | N | String |  | <orderType> </orderType> | {"orderType":""} |
| 26 | transType | Transaction type | N | String |  | <transType> </transType> | {"transType":""} |
| 27 | tillNo | Till machine on which the ordered item is made | Y | String |  | <tillNo> </tillNo> | {"tillNo":""} |
| 28 | inputDateTime | Sales item input date time | Y | datetime |  | <inputDateTime> </inputDateTime> | {"inputDateTime":""} |
| 29 | inputId | User ID of staff who input the item | Y | String |  | <inputId> </inputId> | {"inputId":""} |
| 30 | voidFlag | Void sales item flag | Y | String |  | <voidFlag> </voidFlag> | {"voidFlag":""} |
| 31 | voidDateTime | Void sale date time | Y | datetime |  | <voidDateTime> </voidDateTime> | {"voidDateTime":""} |
| 32 | voidId | Void ID | N | String |  | <voidId> </voidId> | {"voidId":""} |
| 33 | reasonCode | Void reason | N | String |  | <reasonCode> </reasonCode> | {"reasonCode":""} |
| 34 | remarks01 | Remark 1 | N | String |  | <remarks01>  </remarks01> | {"remarks01":""} |
| 35 | remarks02 | Remark 2 | N | String |  | <remarks02>  </remarks02> | {"remarks02":""} |
| 36 | remarks03 | Remark 3 | N | String |  | <remarks03>  </remarks03> | {"remarks03":""} |
| 37 | dataSource | Data Source of the sales (e.g. “OM” for Order Management; “Lunch” for Lunch Box system) | Y | String |  | <dataSource>  </dataSource> | {"dataSource":""} |
| 38 | lastUpdateUser | Last update by (user id by which the convert process or adjustment is done) | Y | datetime |  | <lastUpdateUser>  </lastUpdateUser> | {"lastUpdateUser":""} |
| 39 | lastUpdateTime | Last update date time | Y | String |  | <lastUpdateTime>  </lastUpdateTime> | {"lastUpdateTime":""} |
| 40 | workstationName | Server at which the convert process take place | Y | String |  | <workstationName>  </workstationName> | {"workstationName":""} |
| 41 | appNmae | Program name of convert process | Y | String |  | <appNmae>  </appNmae> | {"appNmae":""} |
| 42 | runno | Run ID of the convert process | Y | Numeric (10,0) |  | <runno>  </runno> | {"runno":""} |
| 43 | status | Whether or not the sales data is counted as store sales | Y | String |  | <status>  </status> | {"status":""} |

<salesPayfig>

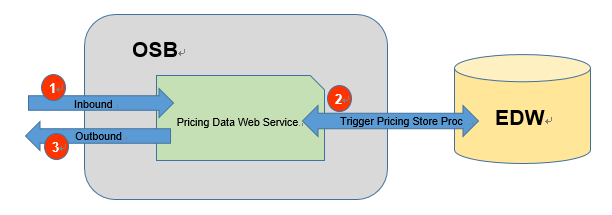
| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | branchId | Maxim’s branch ID | Y | String |  | < branchId>MXM  </branchId> | {“branchId”:”MXM”} |
| 2 | businessDate | Transaction Business Date | Y | Date |  | < businessDate>2016-01-01  </businessDate> | {“businessDate”:”2016-01-01”} |
| 3 | rectype | Record type | Y | String |  | <rectype> </rectype> | {"rectype":""} |
| 4 | recdesc | Record type description | Y | Numeric (5,0) |  | <recdesc> </recdesc> | {"recdesc":""} |
| 5 | currencyCode | Currency Code for the “total amount” | Y | String |  | <currencyCode> </currencyCode> | {"currencyCode":""} |
| 6 | totalAmount | Total amount for this Record type | Y | Numeric (5,0) |  | <totalAmount> </totalAmount> | {"totalAmount":""} |
| 7 | qty | Total quantity for this Record type | Y | String |  | <qty> </qty> | {"qty":""} |
| 8 | author | Author | Y | String |  | <author> </author> | {"author":""} |
| 9 | localCurrency | Currency Code for the “local amount” | Y | String |  | <localCurrency> </localCurrency> | {"localCurrency":""} |
| 10 | localAmount | Local amount for this Record type | Y | Numeric (19,4) |  | <localAmount> </localAmount> | {"localAmount":""} |
| 11 | exchangeRate | Exchange rate ( = Local currency amount / total amount) | Y | Numeric (19,4) |  | <exchangeRate> </exchangeRate> | {"exchangeRate":""} |
| 12 | inputDateTime | Input Date Time | Y | Numeric (19,4) |  | <inputDateTime> </inputDateTime> | {"inputDateTime":""} |
| 13 | inputId | Input ID | N | String |  | <reasonCode> </reasonCode> | {"reasonCode":""} |
| 14 | reasonCode | Reason code | N | String |  | <remarks01>  </remarks01> | {"remarks01":""} |
| 15 | remarks01 | Remark 1 | N | String |  | <remarks02>  </remarks02> | {"remarks02":""} |
| 16 | remarks02 | Remark 2 | N | String |  | <remarks03>  </remarks03> | {"remarks03":""} |
| 17 | remarks03 | Remark 3 | N | String |  | <dataSource>  </dataSource> | {"dataSource":""} |
| 18 | dataSource | Data Source | Y | datetime |  | <lastUpdateUser>  </lastUpdateUser> | {"lastUpdateUser":""} |
| 19 | lastUpdateUser | Last update user | Y | String |  | <lastUpdateTime>  </lastUpdateTime> | {"lastUpdateTime":""} |
| 20 | lastUpdateTime | Last update time | Y | String |  | <workstationName>  </workstationName> | {"workstationName":""} |
| 21 | workstationName | Server at which the convert process take place | Y | String |  | <appNmae>  </appNmae> | {"appNmae":""} |
| 22 | appNmae | Program name of convert process | Y | Numeric (10,0) |  | <runno>  </runno> | {"runno":""} |
| 23 | runno | Run ID of the convert process | Y | String |  | <status>  </status> | {"status":""} |
| 24 | status | Whether or not the sales data is counted as store sales | Y | String |  | <status>  </status> | {"status":""} |

Outbound Interface Format

| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | status | Update Status | Y | String |  | <status>MXM  </branchId> | {“status”:”A”} |
| 2 | remark | remark | Y | String |  | < remark>  </remark> | {“remark”:””} |

## Pricing Data Web Service

### Process/Work Flow



1. Send the request the OSB Restful Pricing Data Web Service through HTTP.
2. The service will trigger the pricing store procedure to update the EDW pricing tables in EDW.
3. The service will return the status to indicate whether the update is success.

Inbound Interface Format

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |

Inbound Interface Format

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| 1 | status | Update Status | Y | String |  | <status>MXM  </branchId> | {“status”:”A”} |
| 2 | remark | remark | Y | String |  | <remark>  </remark> | {“remark”:””} |

## EDW Checksum Web Service

### Process/Work Flow

1. Send the request the OSB Restful EDW Checksum Web Service through HTTP.
2. The service will get the information from the EDW.
3. The service will return the result and status to indicate whether the request is success.

Inbound Interface Format

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| 1 | branchId | Maxim’s branch ID | Y | String |  | < branchId>MXM  </branchId> | {“branchId”:”MXM”} |
| 2 | businessDate | Transaction Business Date | Y | Date |  | < businessDate>2016-01-01  </businessDate> | {“businessDate”:”2016-01-01”} |

Outbound Interface Format

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Description** | **Is Required** | **Format** | **Remarks** | **XML Example** | **JSON Example** |
| 1 | noOfRecord | noOfRecord | Y | Numeric (19,0) |  | <noOfRecord>10  </noOfRecord> | {“noOfRecord”:”1-”} |
| 2 | checksum | checksum | Y | Numeric (19,4) |  | <checksum>1234  </checksum> | {“checksum”:”1234”} |
| 3 | status | Update Status | Y | String |  | <status>MXM  </branchId> | {“status”:”A”} |
| 4 | remark | remark | Y | String |  | <remark>  </remark> | {“remark”:””} |

- End -