

Promax PX Technical Documentation

Promotions – Pricing Conditions

359PROM – PMXPXI03 – PXIATL02

**Updated by:**

**Chris Horn**

**08th November, 2013**

**Table of Contents**

[1. Introduction 3](#_Toc366065728)

[2. Interface Details 4](#_Toc366065729)

[3. Promax PX 4](#_Toc366065730)

[4. PX Interfacing Hub 5](#_Toc366065731)

[4.1. Validate the information from Promax PX 5](#_Toc366065732)

[4.2. Lookup additional information for interface to ATLAS 5](#_Toc366065733)

[4.3. Interface Logic. 6](#_Toc366065734)

[4.3.1. Inbound 6](#_Toc366065735)

[4.3.2. Outbound 7](#_Toc366065736)

[5. ATLAS (APP) 8](#_Toc366065737)

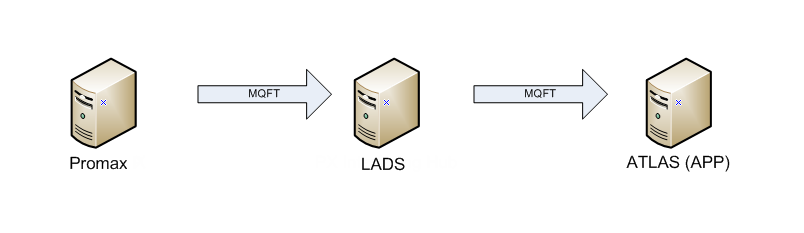
# Introduction

This document provides the technical processes involved for Interfacing Promotions out of Promax PX to SAP through the PMXPXI03\_LOADER interface; This interface is also known as ‘359 Promotions’ in the Promax PX Interface specification and ‘Promotion Pricing Conditions’ when discussing with the business.

This document does not cover the specific business processes that are involved at each step. They are covered in the any functional documentation.

See the diagrams below for an overview of what is covered:

**Technical Overview:**



CISATL14

PXIATL02

PMXPXI03

359PROM

**Note: I**nterface is sent through LADS and transformed processed and on sent to SAP.

# Interface Details

Production Interface Details

**Source Server:** WODNTS5 **Source Queue:** XPMXP01  
**Source Directory:** D:\apps\pmx\prod\outbound

**Destination Server:** WODLX008 **Destination Queue:** QM0217P  
**Destination Directory:** /ics/lad/prod/inbound  
**Destination Filename:** 359PROM.txt

# Promax PX

Scripting details to be entered here.

# PX Interfacing Hub

The PX Interfacing Hub is the destination from the transfer from Promax PX.

When the file arrives to the PX Interfacing Hub it is configured to validate the data that has been sent, lookup the additional information that is required for loading Pricing Conditions into ATLAS (APP), and then perform the actual send the file.

As there are many steps within the PX Interfacing Hub, they have been split into subsections below:

## Validate the information from Promax PX

The file that is received from the PX interfacing Hub is loaded through a new interface that has been created for Promax PX data.

## Lookup additional information for interface to ATLAS

## Interface Logic.

The interface performs the translation required to apply Promax PX Promotions to SAP (Atlas AAP) Pricing Conditions – this interface is a combination of Inbound and Outbound within the same package.

Inbound processing entry points are [on\_start], [on\_data] and [on\_end], with outbound processing entry point on [execute].

### Inbound

The inbound interface “[LADS:PMXPXI03] Promax NZ > LADS : Promotion (Pricing Conditions) : 359 : 359PROM” processes on receipt the [359PROM.txt] file from Promax PX, file specification below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Field Desc | Field Name | Field Name Len | Position | Length | Required | Type | Format |
| IC Record Type | ic\_record\_type | 14 | 0 | 6 | Yes | Alpha |  |
| PX Company Code | px\_company\_code | 15 | 6 | 3 | Yes | Alpha |  |
| PX Division Code | px\_division\_code | 16 | 9 | 3 | Yes | Alpha |  |
| Customer Hierarchy | customer\_hierarchy | 18 | 12 | 10 | No | Alpha |  |
| Sales Deal | sales\_deal | 10 | 22 | 10 | No | Alpha |  |
| Material | Material | 8 | 32 | 18 | No | Alpha |  |
| Buy Start Date | buy\_start\_date | 14 | 50 | 8 | Yes | Date | yyyymmdd |
| Buy Stop Date | buy\_stop\_date | 13 | 58 | 8 | Yes | Date | yyyymmdd |
| Transaction Code | transaction\_code | 16 | 66 | 4 | No | Alpha |  |
| Description | Description | 11 | 70 | 40 | No | Alpha |  |
| Sales Org | sales\_org | 9 | 110 | 4 | No | Alpha |  |
| Rate | Rate | 4 | 114 | 12 | No | Numeric | 999999999.99 |
| User 1 | user\_1 | 6 | 126 | 10 | No | Alpha |  |
| User 2 | user\_2 | 6 | 136 | 10 | No | Alpha |  |
| Action Code | action\_code | 11 | 146 | 1 | No | Alpha |  |
| Bonus Stock Description | bonus\_stock\_description | 23 | 147 | 100 | No | Alpha |  |
| Bonus Stock Hurdle | bonus\_stock\_hurdle | 18 | 247 | 9 | No | Numeric | 999999.99 |
| Bonus Stock Receive | bonus\_stock\_receive | 19 | 256 | 9 | No | Numeric | 999999.99 |
| Bonus Stock SKU Code | bonus\_stock\_sku\_code | 20 | 265 | 18 | No | Alpha |  |
| Rate Unit | rate\_unit | 9 | 283 | 5 | No | Alpha |  |
| Condition Pricing Unit | condition\_pricing\_unit | 22 | 288 | 5 | No | Alpha |  |
| Condition UOM | condition\_uom | 13 | 293 | 3 | No | Alpha |  |
| SAP Promo Number | sap\_promo\_number | 16 | 296 | 10 | No | Alpha |  |
| Currency | Currency | 8 | 306 | 3 | No | Alpha |  |
| UOM Str Unit | uom\_str\_unit | 12 | 309 | 3 | No | Alpha |  |
| UOM Str Saleable | uom\_str\_saleable | 16 | 312 | 3 | No | Alpha |  |
| Promo Price Saleable | promo\_price\_saleable | 20 | 315 | 10 | No | Alpha |  |
| Promo Price Unit | promo\_price\_unit | 16 | 325 | 10 | No | Alpha |  |
| Transaction Amount | transaction\_amount | 18 | 335 | 10 | No | Alpha |  |
| Payer Code | payer\_code | 10 | 345 | 20 | No | Alpha |  |

The interface references three other database objects:

* **PMX\_PROM\_CONFIG**Table of pricing condition transformation information.
* **PMX\_359\_PROMOTIONS\_SEQ**Sequence in support of a unique batch sequence for PMS\_359\_PROMOTIONS table.
* **PMX\_359\_PROMOTIONS**  
  Table of ALL processed promotions records.
* **PMX\_PRICE\_CONDTIONS**Table containing the generated timeline of all the pricing conditions.

The promotions file [**359PROM.txt**] is transactional, meaning that if a promotion is created, modified and or deleted before an extract is created, multiple records representing each change in state will exist for the same promotion.

Fields are parsed by the FFLU utilities, conforming to the file specification for type, format and required.

Records without [customer\_hierarchy] values are “Header Records” which are ignored.

The functional specification indicates that there are 4 possible [action\_code]’s:

* A = Add
* M = Modify
* C = Close : This has currently being implemented as a Delete. However it should really be implemented as per an M so as not to loose the history, but needs confirmation with the business.
* D = Delete

To reduce the possibility of processing a transaction out of order, a check is performed to ensure that transaction [xactn\_seq], as identified by a transaction id encoded into the records [description] field after a colon [:] character, is not less than the previous processed transaction – initialised with the value of the largest previously processed [xactn\_seq] from the [pmx\_359\_promotions] table.

Now the interface logic moves onto “Enhancing” the information received in the interface file.

* Firstly ensuring correctly formatted customer hierarchy [customer\_hierarchy > new\_customer\_hierarchy] and material [material > new\_material] information via the [pxi\_common.full\_cust\_code(customer\_hierarchy)] and [pxi\_common.full\_matl\_code(material)] utilities.
  + A null value for either raises an exception.
* Next adding [business\_segment] via a call to the [pxi\_utils.determine\_bus\_segment(px\_company\_code,new\_material].
* Next calculate the [condition\_flag] ..

if fflu\_data.get\_char\_field(pc\_condition\_pricing\_unit) = pc\_condition\_unit\_dollar then -- ‘1’

prv\_inbound.condition\_flag := pc\_condition\_flag\_dollar; -- ‘F’

else

prv\_inbound.condition\_flag := pc\_condition\_flag\_percentage; -- ‘T’

end if;

* Next [rate\_multiplier], [condition\_type\_code], [pricing\_condition\_code], [condition\_table\_ref], [cust\_div\_code] and [order\_type\_code] are referenced from the [pxm\_prom\_config] table, keyed on [px\_company\_code], [business\_segment], [user\_1] and [condition\_flag].
  + Not finding a matching value, or null values for [condition\_type\_code], [pricing\_condition\_code], [condition\_table\_ref] or [cust\_div\_code] raised an exception.
* Next construct the [vakey], varying on [order\_type\_code] ..

if nvl(prv\_inbound.order\_type\_code, ' ') = 'ZORB' then

prv\_inbound.vakey := rpad(

rpad(prv\_inbound.px\_company\_code, 3)

|| ' '

|| prv\_inbound.cust\_div\_code

|| prv\_inbound.order\_type\_code

|| prv\_inbound.new\_customer\_hierarchy

|| prv\_inbound.new\_material,

50

);

else

prv\_inbound.vakey := rpad(

rpad(prv\_inbound.px\_company\_code, 3)

|| ' '

|| prv\_inbound.cust\_div\_code

|| prv\_inbound.new\_customer\_hierarchy

|| prv\_inbound.new\_material,

50

);

end if;

* Next calculate [new\_rate], [new\_rate\_unit] and [new\_rate\_multiplier] ..

-- Calculate New Rate

if nvl(prv\_inbound.condition\_flag, ' ') = pc\_condition\_flag\_dollar and prv\_inbound.rate\_multiplier is not null then -- Dollar = F

prv\_inbound.new\_rate := (-1 \* nvl(prv\_inbound.rate, 0) \* prv\_inbound.rate\_multiplier);

else

prv\_inbound.new\_rate := (-1 \* nvl(prv\_inbound.rate, 0));

end if;

-- Calculate New Rate Unit

if nvl(prv\_inbound.condition\_flag, ' ') = pc\_condition\_flag\_percentage then -- Percentage = T

prv\_inbound.new\_rate\_unit := null;

else

prv\_inbound.new\_rate\_unit := prv\_inbound.currency;

end if;

-- Format New Rate Multiplier (Number to Text)

if nvl(prv\_inbound.rate\_multiplier,0) > 0 then

prv\_inbound.new\_rate\_multiplier := trim(to\_char(prv\_inbound.rate\_multiplier, '00000'));

else

prv\_inbound.new\_rate\_multiplier := rpad(' ', 5);

end if;

At this point the inbound transactions have been individually validated, and “Enhanced” with calculated or referenced information.

We now add a unique [xactn\_seq], and a unique [batch\_rec\_seq] (per [batch\_seq]) to the record, and insert into the [pmx\_359\_promotions] table, though not committed pending successful outbound processing.

### Outbound

The outbound processing [execute] is called from the [on\_end] of the inbound segment of the same package, and is passed the [pv\_previous\_xactn\_seq] which is the last transaction sequence successfully processed and thus committed to the [pmx\_359\_promotions] table.

**A promotion within Promax PX is effectively keyed on [vakey], [pricing\_condition\_code] and [sales\_deal], while the equivalent pricing condition with SAP is simply keyed on [vakey] and [pricing\_condition\_code]. This mismatch on keying, and the further desire to reduce the number of updates affected in SAP, leads too much of the complexity of this interface.**

**The remainder of this section walks through the process of applying a 3 element key Promax PX promotions to a 2 element key SAP pricing condition.**

All the remaining processing is applied against the [pmx\_359\_promotions] table, which contains ALL previously successful transactions (committed), and the current transactions visible to this process only.

Processing each of the latest Promax PX promotions [further referred to as “current transaction”], we first check for previous state of a transaction, within a previously committed batches, and failing a match, check for a previous state of a transaction within the current batch, but prior to the current transaction [further referred to as “previous transaction”].

The system generates a timeline starting out the earliest date ever seen for this vakey and pricing condition. It then applies every instruction ever seen up to the batch being processed to that timeline. So that in memory we now have a timeline indicating what has happened for all time. That timeline is the aggregated into a set of start and stop dates, based on where there is a change in the price or sales deal information. Those agreegated records are then written to the pmx\_price\_conditions table and then written out in the interface to SAP.

How each instruction is applied to the timeline is below.

* A = Add, The record is applied from the start date to the stop date.
* M = Modify, The whole timeline is searched for any previous entries with the same sales deal and zeroes the rate. Then the current record is applied as if it was an add.
* C = Cancel, D=Delete, The specified date range is zeroed.

Given success to this point, the current transactions are committed to the [pmx\_359\_promotions] table, and processing is complete, otherwise on any error, the current transactions are rolled back and processing halted – with ICS reporting status.

Below is the output format for on sending the data to SAP.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Source Field** |  | **Target Field** | **Position** | **Length** | **Required** | **Type** | **Format** | **Constant** | **Constant Value** |  | **Format Type** |
|  |  | UsageConditionCode | 0 | 1 | Yes | Alpha |  | Yes | A |  |  |
| cndtn\_table\_ref |  | CondTable | 1 | 3 | Yes | Alpha |  | No |  |  |  |
|  |  | Application | 4 | 1 | Yes | Alpha |  | Yes | V |  |  |
| VAKEY |  | VAKEY | 5 | 50 | Yes | Alpha |  | No |  |  |  |
| px\_company\_code |  | CompanyCode | 55 | 3 | Yes | Alpha |  | No |  |  |  |
| cust\_div\_code |  | Division | 58 | 2 | Yes | Alpha |  | No |  |  |  |
| cust\_hier |  | Customer | 60 | 10 | Yes | Alpha |  | No |  |  |  |
| Material |  | Material | 70 | 18 | Yes | Alpha |  | No |  |  |  |
| buy\_start\_date |  | ValidFrom | 88 | 8 | Yes | Date | yyyymmdd | No |  |  |  |
| buy\_stop\_date |  | ValidTo | 96 | 8 | Yes | Date | yyyymmdd | No |  |  |  |
| pricing\_cndtn\_code |  | Condition | 104 | 4 | Yes | Alpha |  | No |  |  |  |
| condition\_type\_code |  | ConditionType | 108 | 1 | Yes | Alpha |  | No |  |  |  |
| Rate |  | Rate | 109 | 11 | Yes | Numeric | s9999990.00 | No |  |  |  |
| rate\_unit |  | RateUnit | 120 | 5 | No | Alpha |  | No |  |  |  |
|  |  | UOM | 125 | 3 | Yes | Alpha |  | Yes | EA |  |  |
| sales\_deal |  | PromoNum | 128 | 10 | Yes | Alpha |  | No |  |  |  |
| rate\_multiplier |  | PriceUnit | 138 | 5 | No | Alpha |  | No |  |  |  |
| order\_type\_code |  | OrderType | 143 | 4 | No | Alpha |  | No |  |  |  |

# Reconciliation Report

After the pricing conditions have been sent to SAP, it is expected that this pricing information will be at some point in the future be interfaced back to LADS. This presents an opportunity to perform a reconciliation to ensure everything is in sync.

# As such a job runs each day at 2 am to perform this check that everything is in sync with what has actually been sent. This is done by a procedure called PMXPXI03\_LOADER.RECONCILE\_PRICING\_CONDITIONS.

It basically takes each current pricing condition in the PMX\_PRICE\_CONDITIONS table and looks for a corresponding entry in the LADS\_PRC\_LST\_HDR and LADS\_PRC\_LST\_DET tables. Any missing or incorrect entries are report and sent as an email. The destination email group is configured as a ICS Setting <TODO Insert Here>

# ATLAS (APP)

ATLAS (APP) is the destination from the transfer from PX Interfacing Hub.

The interface is loaded into ATLAS against message type ZACC\_DOCUMENT, and then processed into ATLAS.