

Promax PX

336 COGS – Cost of Goods Sold Interface

**Updated by:**

**Chris Horn**

**17st September, 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)

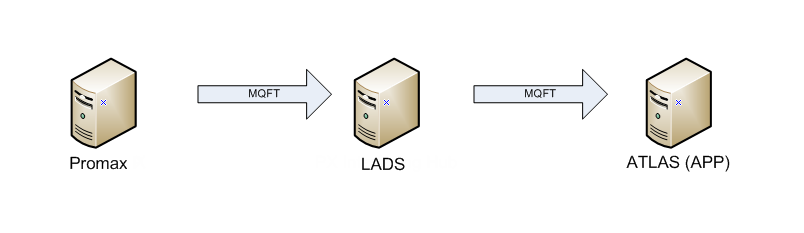
# Introduction

This document provides the technical processes involved for Interfacing Cost of Goods Sold information into Promax PX AP Claims Data out of Promax PX to SAP through the PXIPMX02 interface; This interface is also known as ‘331 Payments’ in the Promax PX Interface specification and ‘AP Claims’ 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 AP Claims Business Process Document. Nor does it cover the

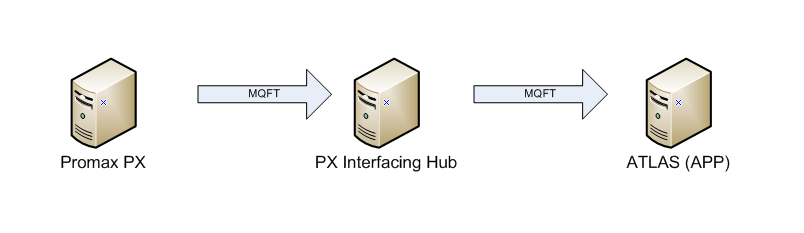
See the diagrams below for an overview of what is covered:

**Functional Overview (As-Is):**



**Note:** Current interface is sent through LADS. No logic is applied, it is a standard passthru interface.

**Functional Overview (To-Be):**



The following sections details what occurs at each of the systems in the above To-Be process.

# 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:** 331CLAIMS.txt

**Trigger Command:**

<file name="/ics/lad/prod/inbound/331CLAIMS.txt" label="PMXPXI02">  
 <job type="mqft\_schedexit">  
 <command><![CDATA[/ics/lad/test/bin/ics\_inbound\_mqft.sh %f PMXPXI02]]></command>  
 <system>Simple</system>  
 </job>  
</file>

# 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 AP Claims into ATLAS (APP), and then send the file onto ATLAS (APP).

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 for NZ.

The following fields require validation:

* Spend Amount (Confirm it equals base +tax)
* Material
* Validate the amount field equals the tax amount field plus the base amount field.
* Validate the ‘Posting Key’ field from the file, if it is not equal to one of the following, then raise an error:
  + ‘Posting Key Payment Credit’
  + ‘Posting Key Reversal Credit’
  + ‘Posting Key Payment Debit’
  + ‘Posting Key Reversal Debit’

## Lookup additional information for interface to ATLAS

The next step is to lookup the additional information that is not provided by Promax PX.

The following reference information is looked up :

* Business Segment
* Material Determination for Traded Unit
* Profit Centre
* Plant Code

## 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.

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 \*\*\* NOT IMPLIMENTED \*\*\*
* D = Delete

However, no business logic was agreed for Close, therefore if an [action code] of “C” is encountered, an exception is raised.

* It is conceivable that Close transactions would be handled per Delete, however this needs to be confirmed.

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\_259\_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”].

If we find the current transaction code is Delete, and previous state is found in current batch, then ignore the superfluous delete, ie. do NOT pass it onto SAP, as the transaction was both created and deleted with the current batch, so never existed within SAP.

Next we raise exceptions should any of the following conditions exist.

* The current transaction code is Add, however a previous matching transaction exists.
* The current transaction code is Delete, however the previously committed transaction was also a Delete.
* The current transaction code is either Modify or Delete, and NO previous transaction was found.
* The current transaction code is either Add or Modify, and it would clash with an earlier active (NOT Deleted) transaction – when applying the 3 element key Promax PX promotion to a 2 element key SAP pricing condition.
  + Promax PX Promotion Key - [vakey], [pricing\_condition\_code] and [sales\_deal]
  + SAP Pricing Condition Key - [vakey] and [pricing\_condition\_code]

After passing the above tests, we now forward the transactions necessary to have SAP pricing condition match the latest Promax PX promotions state.

* If a previous transaction was found, outside of the current batch, then append this previous transaction with a zeroed [new\_rate], to the outbound interface.
  + This has a nice side-effect of also taking care of Deletes.
* If the current transaction is NOT a Delete, append the current transaction to the outbound interface.

The “append” procedure formats the outgoing rows per the file specification below, and keeps a count of the outgoing row count, producing a NEW outbound interface for each [pxi\_common.gc\_max\_idoc\_rows] records .. this max rows is currently 909.

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.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |  |  |  |

# COGS interface.

In order to be able to use historic (actual) costs for good sold in Promax PX, we need to load these historic costs into the system.Need to upload a CSV file once per period with the Standard COGS cost per case. This value is calculated by finance from the costs incurred in the previous period.

## Upload file definition

The spreadsheet is of the following format:

|  |  |  |
| --- | --- | --- |
| **PERIOD** | **ZREP\_** | **COGS** |
| 201309 | 253713 | 11.50 |
| 201309 | 262460 | 62.74 |
| 201309 | 267464 | 30.27 |
| 201309 | 214501 | 14.14 |
| 201309 | 237255 | 10.39 |
| 201309 | 253293 | 27.48 |
| 201309 | 247347 | 20.04 |
| 201309 | 226122 | 15.05 |
| 201309 | 272086 | 0.89 |
| 201309 | 272087 | 4.66 |

This should be stored in VENUS (db1270p) in the PXI schema. The data in the file should be referenced by period, company code and division code.

## Table Definition

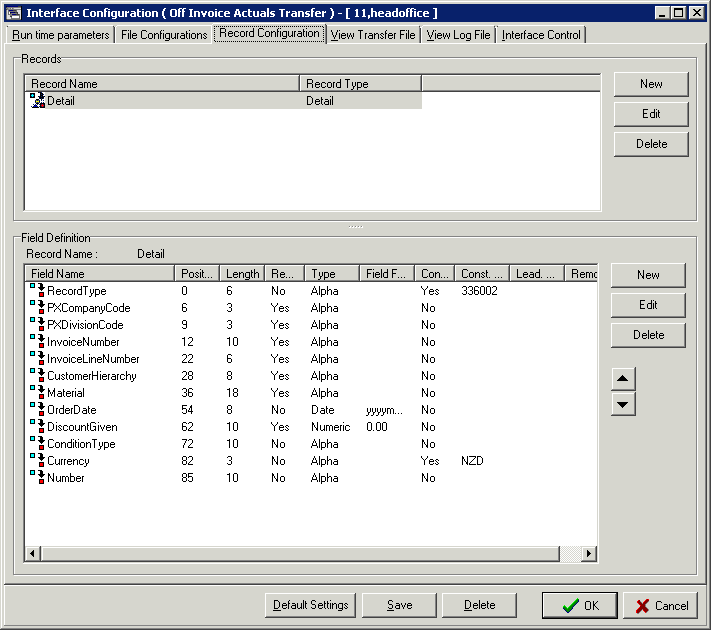
|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN** | **DATA TYPE** | **SIZE** | **EXAMPLE** |
| CMPNY\_CODE | CHAR | 3 | 149 |
| DIV\_CODE | CHAR | 3 | 149 |
| MARS\_PERIOD | NUMBER | 6,0 | 092013 |
| ZREP\_MATL\_CODE | CHAR | 18 | 253293 |
| COST | NUMBER | 7,2 | 27.48 |
| LAST\_UPDATE\_DATE | DATE |  | SYSDATE |
| LAST\_UPDATE\_USER | CHAR | 30 | DREWCRA |

The upload should default the PXCompanyCode and the PXDivisionCode determined by the interface type:

|  |  |  |
| --- | --- | --- |
| **Company** | **PXCompanyCode** | **PXDivisionCode** |
| New Zealand | “149” | “149” |
| Petcare | “147” | “05” |
| Food | “147” | “02” |
| Chocolate | “147” | “01” |

New interface to Promax PX. This uses the off invoice actuals format.

The interface is to be sent weekly , if a material cannot be found in the cost standard table then the whole interface should fail, alerting to ANZ Apps support team.

Promax File Configuration

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Value** |
| RecordType | Record Identifier | 336003 |
| PXCompanyCode | Hard coded to New Zealand | “149” |
| PXDivisionCode | Hard coded to New Zealand market | “149” |
| InvoiceNumber | Invoice number from sales |  |
| InvoiceLineNumber | Invoice Line Number from sales |  |
| CustomerHierarchy  Material | Sold to code of the customer | Sold\_to\_cust\_code |
| Material | ZREP Material Code | Rep\_Item |
| OrderDate | Order date from the sale | nvl(decode(to\_char(t2.order\_eff\_date,'DY'),'SUN',t2.order\_eff\_date +1,t2.order\_eff\_date),decode(to\_char(t1.billing\_eff\_date,'DY'),'SUN',t1.billing\_eff\_date +1,t1.billing\_eff\_date)) as order\_eff\_date |
| DiscountGiven | Sale qty \* standard from table | billed\_qty\_base\_uom \* COST joined by rep\_item from matl\_dim |
| ConditionType | Hard coded to cost type | “500” |
| Currency | Hard coded to New Zealand Dollars | “NZD” |
| Number | Left blank |  |

Interface will be scheduled to run weekly on a Sunday. The file should contain data for the previous full mars Week (Sunday to Saturday).

eg.

If run on Sunday 22nd September, the file should contain data for Invoices from Sunday 15th September to Saturday 21st.

If run on Saturday 21st September, the file should contain data for Invoices from Sunday 8th September to Saturday 14th September.

If a material cannot be found in the cost standard table then the whole interface should fail. It should raise an alert (email) to the ANZ applications support team.

# 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.

TODO: Document the tasks to process and looking up these values in SAP.