Data Warehouse Billing Tool (DWBT)

Developer Manual

Contents

[Description 4](#_Toc2033018251)

[Acronyms 4](#_Toc2019732581)

[Glossary 4](#_Toc72283201)

[Business Structure/Scope 5](#_Toc1015241951)

[Business Units 5](#_Toc2062431703)

[Physical Sites 6](#_Toc50429723)

[Data Management 6](#_Toc121937929)

[DWBT History 6](#_Toc1788445464)

[DWBT Uses 6](#_Toc1867010194)

[Technologies 7](#_Toc1778539557)

[Report Server 7](#_Toc6358944)

[Robotic Process Automation (RPA) 7](#_Toc2144440085)

[Windows Server 7](#_Toc1883338637)

[File Transfer Protocol (FTP) 7](#_Toc684385098)

[Python 7](#_Toc1938394851)

[Anaconda 7](#_Toc82746487)

[JSON 8](#_Toc1382948154)

[PostgreSQL 8](#_Toc357535160)

[DBeaver 8](#_Toc1481706455)

[GIT 8](#_Toc24504537)

[Excel 8](#_Toc1378719943)

[VBA 8](#_Toc310275804)

[DOS Batch Scripts 9](#_Toc214276181)

[Windows Task Scheduler 9](#_Toc479521716)

[DWBT Process 9](#_Toc1622800775)

[High Level 9](#_Toc757849564)

[Data Flow Diagram 9](#_Toc630975115)

[Data Sources (Input) 9](#_Toc63529047)

[Matrix WMS 9](#_Toc2141474450)

[WIMS 12](#_Toc1472858491)

[Pentana 13](#_Toc72334238)

[SAP 13](#_Toc831378890)

[FMS (Freight Management System) 14](#_Toc1326803577)

[EmpLive 15](#_Toc1390844120)

[Customers 15](#_Toc1644282846)

[Internal ad-hoc 16](#_Toc997353794)

[Other 16](#_Toc617460976)

[Manually Input Data 16](#_Toc396797348)

[Data Ingestion 17](#_Toc903105017)

[Billing Server 17](#_Toc1404290947)

[FTP Site 18](#_Toc920410983)

[PROD - startFolderWatch.bat 18](#_Toc296575480)

[\Supporting Code\Python\WatchFolder.py 18](#_Toc1326549725)

[\Configs\generalConfig.json 18](#_Toc621774095)

[Logs 19](#_Toc1489446066)

[\Supporting Code\Python\ Import.py 19](#_Toc1930970916)

[\Configs\importConfig.json 19](#_Toc419433642)

[\Supporting Code\Python\globalFunctions.py 20](#_Toc1750658054)

[Source Data Landing Zone 21](#_Toc1440890304)

[Converted Source Data 21](#_Toc681138714)

[Formatted Source Data 21](#_Toc7829832)

[Database Landing Zone 21](#_Toc695662713)

[Adding a New Report 21](#_Toc329404579)

[Reporting 22](#_Toc1109802277)

[\Supporting Code\Python\ generateReport.py 22](#_Toc1789946957)

[\Supporting Code\Config\reportConfig.json 22](#_Toc1715340687)

[Scheduling 23](#_Toc44033105)

[Database Architecture 24](#_Toc1858881436)

[Environments 24](#_Toc1688606934)

[Creating Environments 24](#_Toc1415903553)

[SQL Formatting and Naming Conventions 25](#_Toc912437029)

[Database Objects 27](#_Toc210818757)

[Schemas 27](#_Toc752463536)

[PostgreSQL Import Functions 27](#_Toc340070601)

[PostgreSQL Load Functions 27](#_Toc803918891)

[PostgreSQL Charges Functions 27](#_Toc1259662062)

[PostgreSQL Tables 30](#_Toc589238633)

[PostgreSQL Views 36](#_Toc2065628015)

[PostgreSQL Functions 45](#_Toc1234434621)

[PostgreSQL Users 48](#_Toc610722127)

[Related Columns 48](#_Toc594139471)

[Metadata columns 49](#_Toc280467563)

[Output/User Interaction 49](#_Toc756809538)

[Billing Tool 50](#_Toc1566746836)

[Excel Tools 51](#_Toc1460293828)

[Freight Management Functionality 53](#_Toc2124859840)

[Tools Not Related to DWBT 53](#_Toc239786603)

[ETAFormatConverter 53](#_Toc1535705829)

[ETAFormatConverter 54](#_Toc278248641)

[Database Risk Management 54](#_Toc511397470)

[Checking for Source Reports 54](#_Toc391939091)

[Database Backups 55](#_Toc473912571)

[Database Syncs 55](#_Toc630153083)

[DevOps 55](#_Toc31919796)

[Managed Materials 55](#_Toc1826003961)

[Back End 55](#_Toc208231598)

[Git Ecosystem 56](#_Toc1706717877)

[Front End 56](#_Toc1182332943)

[Lifecycle 56](#_Toc2025661501)

[Australia 57](#_Toc469613123)

[Server Details 57](#_Toc1722583072)

[FTP Details 57](#_Toc2144749624)

[Front End Details 57](#_Toc637523744)

[Training Exercises 57](#_Toc458743469)

[Charges Maintenance 57](#_Toc631764313)

# Description

## Acronyms

* CL – Contract Logistics
* FM – Freight Management
* DWBT - Data Warehouse Billing Tool
* DIFOT – Despatch In Full On Time
* QS – Qlik Sense
* QV – Qlik View
* RPA – Robotic Process Automation. Used to pull Matrix Reports by mimicking a user
* BPE - Business Process Enhancement. Business Process Engineer
* SCA - Site Classification Audit
* PRIMO - Pricing Model. Used to determine viability of new customers, contract renewals
* DG - Dangerous Goods
* WMS - Warehouse Management System. MATRIX WMS, WIMS are WMS’s. Matrix has additional software solutions, but in NZ ‘Matrix’ usually refers to the WMS.
* FMS/TMS - Freight Management System/Transport Management System
* SG - Storage
* FR - Freight
* CONSOL - Consolidated
* HAWB - Air shipment number identifier
* SLA - Service Level Agreements
* SOH - Stock On Hand
* TXN - Transaction
* IB - Inbound
* OB - Outbound
* EMP Live - Employee Live system for tracking employee data
* GL - Global Ledger. Accounting tool.
* WEB3 - Internal ordering system for things non-recurring
* DFM - Internal ordering system for things recurring/expected
* ASN - Advanced service notice.

## Glossary

* Tag ID - Pallet identifier for inbound. Can contain multiple SKUs, and the same SKU
* Pallet ID - Outbound pallet identifier. Can contain multiple orders. Contain multiple serial numbers which are each assigned a container.
* Pre-advice-id - The unique identifier for receipted items for inbound from Matrix. Pre-advice-id can be given to us by the customers or created internally to identify the items. Pre-advice-id can mean different things for different customers as it can represent the container it came in. Usually represents the logical unit that inventory arrived in.
* Reference - The unique identifier for orders made by customers in the outbound process. It is made up of Lines, SKU's and quantities of each SKU
* Order\_id - The unique identifier for items in DIFOT.
* Consignee – Person receiving inventory on behalf of our customers. For instance, a car dealership could be a consignee of our customer, Holden.
* KAIZEN - Internal Improvements which are tracked and reported each year at each site.
* Line – The combination of unique order and SKU. EX: Order 1 contains SKUs A and B. Order 1 SKU A is a line, Order 1 SKU B is a line.
* Pickrun – A term used for the customer Accent. A collection of orders. Why we work with pickruns is an Accent trade secret.

## Business Structure/Scope

CEVA operations are split into 2 entities: Contract Logistics and Freight Management. Contract Logistics handles the unloading, inbounding, storage, outbounding of inventory. Transport is a piece of CL that handles the transportation of inventory to destination. Freight Management (or freight forwarding) handles the international transport of inventory. There is no storage element. Inventory is received and immediately prepared for sending. **The scope of DWBT is CL/Transport.** A stock lifecycle can be found at [transaction\_types](file:///C:/Users/hentzd/AppData/Roaming/Microsoft/Word/transaction_types).



## Business Units

A business unit (BU) is an accounting bucket to which costs and revenue are recorded. A customer has its own business unit if it surpasses a revenue threshold. If it does not surpass this threshold, it is bundled into a business unit that contains multiple customers: Multi User Warehouse (MUWH). CEVA uses a Global Ledger (GL) for accounting purposes. The GL assigns numeric values to different cost/revenue buckets for different customers.

* BU: Holden warehouse AKL (GL revenue bucket: 40400003) – Holden only
* BU: Goodyear warehouse AKL (GL revenue bucket: 40400015) – GDT AKL only
* BU: Goodyear warehouse CHC (GL revenue bucket: 40400027) - GDT CHC only
* BU: Multi users warehouse AKL (GL revenue bucket: 40400042)
  + All Good Organics
  + Goddards
  + HK Wentworth
  + Karma Cola
  + Allwin
  + Integrated Packaging
* BU: Acer warehouse AKL (GL revenue bucket: 40400092) – Acer only
* BU: Light Source Solutions WHS AKL (GL revenue bucket: 40400104) – Light Source Solutions only
* BU: Accent warehouse AKL (GL revenue bucket: 40400140) – Accent only
* BU: Kogan warehouse AKL (GL revenue bucket: 40400166) – Kogan only
* BU: Lululemon warehouse AKL (GL revenue bucket: 40400170) – Lululemon only
* BU: MG Motors NZ (GL revenue bucket: 40400173) - MG Motors only
* BU: Coco Republic warehouse AKL (GL revenue bucket: 40400174) – Coco Republic only

## Physical Sites

At the time of this writing, CEVA NZ has 4 warehouses:

* Auckland Airport – Multiple customers
* Auckland Holden – Warehouse exclusively for Holden customer
* GDT Auckland - Warehouse exclusively for GDT customer
* GDT Christchurch - Warehouse exclusively for GDT customer

## Data Management

The 3 core elements of Contract Logistics are Inbounding (receiving, sorting, putting away inventory), Storage, and Outbounding (receiving customer orders, picking products, preparing for shipment, and shipment). These processes are managed by software called Warehouse Management Systems. At the time of this writing, CEVA NZ uses 4 WMS’s outlined in the ‘Source Systems’ section. There are multiple ways to interact with a WMS:

* Web user interface – for looking at data, pulling reports, updating data.
* Scanners used on the floor – whenever an action takes place, inventory is scanned. Scanned upon receipt, putaway, shipment, etc.
* EDI (Electronic Data Interchange) – Customers can place orders, let us know inventory is coming.

## DWBT History

CEVA does not have a CL billing system to automatically calculate charges for its customers. To calculate charges, complicated Excel workbooks were constructed. These workbooks accepted reports pulled from the various WMS’s as input and calculate the charges. This method was very error prone: one wrong formula or copy/paste could result in missed charges and error notification was non-existent. To combat revenue leakage from the Excel workbooks, the DWBT project was commissioned in early 2020.

## DWBT Uses

DWBT was originally created to automate billing calculations, but since the data was centralised, it has expanded to several other functions. These are currently, but not limited to:

* Automate the billing calculations.
* Automate KPI (key performance indicators) calculation.
* Allow easy access to WMS data.
* Automatically send reports.
* Source data for BI dashboards.
* Answer ad-hoc questions that would be complicated using Excel.

# Technologies

## Report Server

Report Server is a server containing Matrix WMS data. CEVA Australia is working to write queries to mimic the reports that the Matrix WMS UI can produce. This would be more reliable than RPA for sourcing Matrix WMS data. Work in progress as of this writing. Steve Wong [Steve.Wong@cevalogistics.com](mailto:Steve.Wong@cevalogistics.com) is heading the project.

## Robotic Process Automation (RPA)

* Description: A tool used to mimic human keystrokes.
* Application(s): Used to pull reports that are normally pulled by billing admins.

## Windows Server

* Description: Microsoft Server OS.
* Application(s): Host for all DWBT technologies except for user interface.
* Technical details: IP:
  + NZ: 10.76.156.24
  + AUS: 10.68.1.1

## File Transfer Protocol (FTP)

* Description: Protocol used to send files.
* Application(s): Used by RPA to send reports to the server.
* Technical details:
  + Server: 10.76.156.24
  + User: ftpUser
  + Pwd: Tr4n5f3r!
  + Port: 20

## Python

* Description: A general-purpose programming language
* Application(s):
  + Data integration – taking various reports/data and importing to the database.
  + Kicking off billing calculations
  + Email notifications
  + Logging
  + Verifying expected reports were received
  + Sending reports out based on views in the database

## Anaconda

* Description: A distribution of the Python programming language
* Application(s): Used to manage and deploy Python packages

## JSON

* Description: A standardised file format
* Application(s):
  + Configuration files
    - Which reports should be accepted, which columns are accepted and in which order, which SQL processes should be run in association with those reports, and general application configurations, etc.

## PostgreSQL

* Description: An open-source relational database management system
* Application(s):
  + Importing data into the database
  + Storing dimensional data – calendar, rate card, etc
  + Calculations such as billing charges, KPI performance, and ad-hoc queries
  + Serving data to clients

## DBeaver

* Description: Open-source database administration client software tool
* Application(s): Used for all SQL development.

## GIT

* Description: Open-source version control software
* Application(s): Keep track of code changes. Ex: While developing for customer A, need to implement hotfix for customer B. Allows promotion of hotfix while keeping track of all changes made to same files for customer A.
* Currently, our GIT repo lives on the server as any testing involving Python must be done on the server.

## Excel

* Description: Microsoft spreadsheet software
* Application(s): user-interface for billing admins to interact with the database.
* Technical details:
  + Requires PostgreSQL ODBC driver to be installed on user’s machine.

## VBA

* Description: Microsoft programming language
* Application(s):
  + Link the Excel UI to the database
  + Forms that user interacts with within Excel
  + Export supporting data and revenue forms

## DOS Batch Scripts

* Description: Microsoft programming language to execute operating system commands and local programs
* Application(s):
  + Take backup of the PROD environment
  + Synchronise DEV and TEST environments to PROD using the most recent backup
  + Run reports

## Windows Task Scheduler

* Description: Microsoft scheduler that allows the launch of programs at specified intervals
* Application(s):
  + Kick off the backup of PROD environment
  + Kick off the Python script that checks to make sure all reports were received
  + Schedule automated reports
* Notes:
  + Tasks are created by a user and whenever the user changes their password, they must update the user group that created the task.

# DWBT Process

## High Level

At a high level, DWBT is a database that ingests data from multiple, varied source systems into a database. It performs transformations/business logic on the source data. It then makes the data C.R.U.D. accessible to users using Excel workbooks on the Shared Drive.

## Data Flow Diagram

Data flow diagram can be found [here](https://cevalogisticsoffice365.sharepoint.com/:u:/r/sites/SEAPAC-NZ-AKL-DataEngineering/_layouts/15/Doc.aspx?sourcedoc=%7BC1E120FF-C198-4467-B5DD-DBD113536623%7D&file=DWBT%20Process%20Flow.vsdx&action=default&mobileredirect=true). Many of the sections below correspond to entities in this diagram.

## Data Sources (Input)

Matrix WMS

**Description:** Used by Auckland airport site. Preferred WMS, attempting to migrate all customers to this system.

**Data Instantiation:** Matrix WMS data is instantiated from the Matrix UI. This used to be done by the billing admin, but now a technology called RPA mimics a user and pulls the reports. In the future, ReportServer will be used to source the data.

**Customers:**

* ACER
* ALL GOOD ORGANICS
* COCO REPUBLIC
* GODDARDS
* HK WENTWORTH
* KARMA COLA
* KOGAN
* LULULEMON
* LIGHT SOURCE SOLUTIONS
* MG MOTORS NZ

**Reports:**

* Consolidated Transactions by Date Range
  + Transaction report. Shows all transactions that happened. Can be used to derive the other reports. Other reports may contain columns not in the transaction report which is why we need the others.
  + Database table: TXN\_MATRIX\_CONSOL
  + Receive frequency: Daily
  + Data range: Previous day
* Receipt Activity by Date Range
  + Inbound report showing ‘Receipt’, ‘Receipt Reverse’, ‘Return’ transactions. Shows when inventory was receipted into the system. Inventory may arrive to the warehouse prior to the dates shown in this report.
  + Database table: IB\_MATRIX\_RECEIPT\_ACTIVITY\_BY\_DATE\_RANGE
  + Receive frequency: Daily
  + Data range: Previous day
* Picking Activity by Date Range
  + Outbound report showing ‘Pick’ and ‘UnPick’ transactions. Shows when inventory was picked from its storage location to be sent to consignee.
  + Database table: OB\_MATRIX\_PICKING\_ACTIVITY
  + Receive frequency: Daily
  + Data range: Previous day
* Shipping Activity by Date Range
  + Outbound report showing ‘Shipment’ transactions. Shows when inventory was marked as shipped which indicates its ready to be picked up by courier to be delivered. Contains a column with serial numbers which is unique among other outbound reports.
  + Database table: OB\_MATRIX\_SHIPPING\_ACTIVITY\_BY\_DATE\_RANGE
  + Receive frequency: Daily
  + Data range: Previous day
* Detailed Order Receipt to Shipped Performance
  + Outbound report showing ‘Shipment’ transactions. Shows when inventory was marked as shipped which indicates its ready to be picked up by courier to be delivered. Contains some columns that are different from other OB reports
  + Database table: OB\_MATRIX\_DETAILED\_ORDER\_RECEIPT
  + Receive frequency: Daily
  + Data range: Previous 3 days
* Warehouse Occupancy
  + Point-in-time storage report. When pulled, shows the inventory occupying the warehouse at that moment.
  + Database table: SG\_MATRIX\_WAREHOUSE\_OCCUPANCY
  + Receive frequency: Daily
  + Data range: N/A. Point-in-time report.
* INVENTORY\_DETAILS\_by\_date\_range
  + Not automatically generated. Not used. Matrix report Was added to keep track of which Acer serial numbers are in the warehouse at the current moment.
  + Database table: SG\_MATRIX\_INVENTORY\_DETAILS
  + Receive frequency: Daily
  + Data range: N/A. Point-in-time report.

**Pulling Reports (How to pull reports):**

1. Talk to IT Admin to get Matrix WMS access.
2. Ask the Matrix WMS admin (Carlos) to create a user.
3. Visit <https://citrixapps.logistics.corp/Citrix/WebStore01Web/> and login with your credentials.
4. Click on ‘Matrix WMS\_02\_PROD’
5. Click ‘CORE+ (02P)’
6. Enter your credentials from step 2.
7. Click ‘Reports’ menu at the top
8. Click ‘Report Selection’
9. Keep the default ‘Print to printer’. Click ‘Next’
10. You can click on the ‘Search’ tab if you know the name of the report you want. Click ‘Next’
11. Select the report you want. For DWBT we are only interested in report prefixed with ‘CSV REPORT’. Click ‘Next’
12. Fill out the date range (if necessary) and email. Click ‘Next’.
13. Click ‘Done’.

**Transaction Types:** Please see visual below. This diagram can be found [here](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/_Documentation/Stock%20Lifecycle%20Diagram.vsdx):

**Diagram

Description automatically generated**

* Return – When we receive inventory that we had previously outbounded.
* Receipt – When inventory first comes into the system.
* Putaway – When stock is put into a storage location.
* Relocate – When stock is moved from one location to another. Sometimes used in place of putaway transactions.
* Receipt Reverse – To undo a receipt transaction.
* Inv Split – Split stock from one location to multiple. Allows us to produce different tag id.
* Batch Update – Way to group products together that share characteristics. EX: Karma Cola uses batch field with a number reflecting when it was produced in case there is a batch recall.
* Config Update – When a pack config is updated.
* Expiry Update – Some stock may have an expiry date. Updates this date.
* Cond Update – The condition field tracks the condition of stock (ex: damaged). This transaction updates this field.
* Stock Check – When verifying stock is in a location/has the right quantity.
* Replenish – Signals the restock of a pick face.
* Adjustment – A stock check verifies stock, then an adjustment is used to move stock off the system. Ex: Expecting 5 units, stock check yields 4. Adjustment used to update the system with actual stock level of 4.
* Release – When an order is placed by a customer, this transaction serves as acknowledgement of the order. This transaction is not visible in DB\_LOAD.TXN\_MATRIX\_CONSOL.
* Allocate – When an order is placed by a customer, this transaction allocates available stock to the order.
* Deallocate – Undo an allocate transaction
* Pick – When stock is picked from a storage location to be outbounded.
* UnPick – Undo a pick transaction
* Repack – Telling system you are packaging goods. Also used to trigger transport label to print.
* Vehicle Load – When inventory is loaded on the outbound vessel
* Vehicle UnLoad – Undo a vehicle load transaction
* Shipment – When the inventory is ready for the carrier. Fully closes off the order. Similar to vehicle load, but like a rubber stamp.

WIMS

**Description:** WMS developed in-house by CEVA. To be decommissioned and customers migrated to Matrix WMS.

**Data Instantiation:** Since it is an in-house developed tool, IT was able to automate the pulling of some reports. They are sent to the DWBT email address: [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com)

**Customers:** Accent

**Reports:**

* transaction\_extract\_rpt
  + Transaction report. Automated report comes as inbound transactions only but can contain all transactions. Working towards the transition to all transactions.
  + Database table: TXN\_WIMS\_TRANSACTION\_EXTRACT
  + Receive frequency: Daily
  + Data range: 3 previous days.
* so\_pick\_rpt
  + Outbound report showing when inventory was picked.
  + Database table: OB\_WIMS\_SO\_PICK
  + Receive frequency: Every 15 minutes during work hours.
  + Data range: current day.
* so\_alloc\_pick\_kpi
  + Not used at this time.
  + Database table: OPS\_WIMS\_SO\_ALLOC\_PICK\_KPI
  + Receive frequency: Every 15 minutes during work hours.
  + Data range: current day.
* so\_rpt\_
  + Not used at this time.
  + Database table: OB\_WIMS\_SO\_RPT
  + Receive frequency: Monthly.
  + Data range: 2 months.
* so\_items\_rpt
  + Not used at this time.
  + Database table: OB\_WIMS\_SO\_ITEMS
  + Receive frequency: N/A – not active
  + Data range: N/A – not active
* Stock Summary
  + Point-in-time storage report. When pulled, shows the inventory occupying the warehouse at that moment.
  + Database table: SG\_WIMS\_STOCK\_SUMMARY
  + Receive frequency: Daily
  + Data range: N/A. Point-in-time report.

**Pulling Reports (How to pull reports):**

1. Talk to IT admin about getting WIMS access.
2. Visit <https://cloudapps-aus.cevalogistics.com/vpn/index.html> and login.
3. Click ‘ANZ CEVA Logistics’
4. If an ‘ics’ file is downloaded, click it.
5. Click ‘WIMS’ in the popup.
6. Click ‘Reports’ in the popup.
7. From here you can select the report you need and the checkbox to send to your email.

Pentana

**Description:** *THIS DATA IS NOT IN DWBT*. WMS used by Holden warehouse in Auckland. Ownership of the software belongs to Holden and Pentana resides on their network.

Is currently being used to fill out the Daily KPI report for Holden found [here.](file://aklwsfps001/Groups/HNZ%20-%20KPI%27s/KPI%20Report%20Consolidated)

SAP

**Description:** WMS used by GDT Auckland and Christchurch. Ownership of the software belongs to GDT and SAP resides on their network.

**Data Instantiation:** Since the system is on GDT’s network, someone at the GDT site needs to use a GDT computer/VPN to access the system. For now, Bob Nandan < Bob.Nandan@Cevalogistics.com> pulls the required reports and sends them to the DWBT email address: SHD-NZ-CLBILLINGTOOL@Cevalogistics.com

**Reports:**

* IB Chch
  + Inbound report.
  + Database table: IB\_SAP\_CHC
  + Receive frequency: Weekly Monday
  + Data range: Previous Monday thru Sunday
* OB Chch
  + Outbound report.
  + Database table: OB\_SAP\_CHC
  + Receive frequency: Weekly Monday
  + Data range: Previous Monday thru Sunday

### FMS (Freight Management System)

**Description:** Used for transport/trucking data. Tracks when inventory was dispatched to a consignee, which destination, the rate we paid the courier, the rate we charged the customer, and more. Developed by ISAC, a company in Auckland.

**Data Instantiation:** Bruce Tonkin ([bruce@isac.nz](mailto:bruce@isac.nz)) is the founder of ISAC. He automated the reports we need by sending them to the DWBT email address: [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com)

**Reports:**

* CNM\_INFO
  + Contains SKU-level despatch data. Consolakl does not have SKU-level data which is why this report is important. Used to inform Kogan of orders that have been despatched.
  + Database table: FR\_FMS\_CNM\_INFO
  + Receive frequency: Daily
  + Data range: Current Day
* Phant
  + Not used right now. Shows which orders could not be delivered
  + Database table: FR\_FMS\_PHANT
  + Receive frequency: Daily
  + Data range: Previous week
* Consolakl
  + Shows all despatched lines, destinations, consignees, costs, and more. Primary report used for billing
  + Database table: FR\_FMS\_CONSOLAKL
  + Receive frequency: Multiple times daily
  + Data range: Current day
* ERR\_RPT\_FMS.KOGAN
  + Not used right now. Previously used to charge Kogan for address errors since fixing their errors took us time to do.
  + Database table: OB\_FMS\_ERR\_RPT\_KOGAN
  + Receive frequency: Daily
  + Data range: Current day

### EmpLive

**Description:** Used for blue collar labour hours data. Used for some billing calculations and KPIs.

**Data Instantiation:** Reports can be automatically generated using the [EmpLive website](https://secure14.emplivecloud.com/SignIn.aspx?ReturnURL=/Configuration/Task.aspx) and sent to an email. Currently, they are set up to be sent to the DWBT email address: [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com)

**Reports:**

* Payroll – Hrs breakdown (Weekly)
  + A detailed breakdown of hours spent, costs, rosters, and a whole range of other information. This is the base Emplive raw data.
  + Database table: LBR\_EMPLIVE\_HRSBREAKDOWN
  + Receive frequency: Weekly
    - This report is collected weekly rather than daily because supervisors can go into Emplive and make changes to entries which affect the accuracy of the data and causes redundancy/duplication. This report comes in every Tuesday which allows for supervisors to make the necessary changes on Monday before the report comes in.
  + Data range: Last 7 days

### Customers

**Description:** Sometimes we need to use data that is provided by customers. A common one is a report that forecasts the containers we expect to receive. This helps us understand future workload since inbounding inventory is about 60% of CL work.

**Data Instantiation:** Reports are sent to the DWBT email address: [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com)

**Reports:**

* Kogan CEVA Backorders Report
  + Shows us Kogan orders that are on backorder. Used to send them an update for the orders in the report
  + Database table: OB\_KOGAN\_CEVA\_BACKORDERS
  + Receive frequency: Daily
  + Data range: N/A – has missing orders
* ACCENT INBOUND SCHEDULE
  + Contains data letting CEVA know when containers are expected to arrive to be inbounded for Accent customer
  + Database table: IB\_ACCENT\_FORECAST
  + Receive frequency: Daily
  + Data range: Previous 3 months
* CEVA - Inbound Forecast
  + Contains data letting CEVA know when containers are expected to arrive to be inbounded for Kogan customer
  + Database table: IB\_KOGAN\_FORECAST
  + Receive frequency: Daily
  + Data range: Year to date
* NZ Pickrun Tracker
  + From customer Accent - a list of outbound pickruns for which they want the status of. We send them a daily report in response.
  + Database table: OB\_ACCENT\_PICKRUN
  + Receive frequency: Daily
  + Data range: Previous year

### Internal ad-hoc

**Description:** We currently have an internal report that is sent out on a scheduled basis through the report generation functionality described in the Reporting section. These reports are SLA commentary reports which show items where we have not met our KPI for inbound, outbound, and DIFOT. These reports require floor staff/managers to look at and provide commentary as to whether they are to be excluded or not and why. This report gets sent back to [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com) and the relevant tables are updated.

**Data Instantiation:** Reports are sent to the DWBT email address: [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com)

**Reports:**

* Inbound SLA commentary (TBC)
* Outbound SLA commentary
* DIFOT SLA commentary (TBC)

### Other

**Description:** These are reports that have unknown/no source system

**Data Instantiation:** Reports are manually pulled and either sent to the DWBT email address [SHD-NZ-CLBILLINGTOOL@Cevalogistics.com](mailto:SHD-NZ-CLBILLINGTOOL@Cevalogistics.com) or FTP’d to the billing server using WinSCP.

**Reports:**

* HouseBill\_Air\_Destination
  + FM report used to display KPI’s for shipments. Data is ingested then sent in an email body to relevant users.
  + Database table: TPT\_KPI\_POD
  + Receive frequency: Daily
  + Data range: N/A – I believe this is for open transport orders
* NZ COMMERCIAL INVOICES
  + Not used. Was going to be used for Coco Republic billing
  + Database table: IB\_COCO\_REP\_COMM\_INV
  + Receive frequency: N/A
  + Data range: N/A

### Manually Input Data

**Description:** Some data must be entered into DWBT manually via a user interface instead of a report.

**Data Instantiation:** User Interface such as an Excel workbook.

**Examples:**

* Manual charges
  + Some charges cannot be derived from reports and must be manually entered using Excel forms in the file ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
  + Destination table: DB\_ACCESS.FT\_CHARGES\_SUMMARY
* Consumables
  + Consumables (shrinkwrap, cardboard, etc) are manually entered into an Excel form using the file ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Consumables Master.xlsm’. These require a separate tool because we buy consumables in bulk and have to attribute them to individual customers and bill them.
  + Destination table: DB\_LOAD.CS\_CONSUMABLES\_MASTER
* Yard Management Tool
  + Yard Management encompasses the handling of inbound containers in the ‘yard’ which is the area under the awning just outside the warehouse. Containers are entered into an Excel form using the file ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Yard Management Tool V5 - Shortcut.xlsm’ (actual file is stored elsewhere). This tool provides data for inbound charges that are charged per container.
  + Destination table: DB\_LOAD.YARD\_MANAGEMENT\_TOOL.

## Data Ingestion

Except for user-entered form data, all data is ingested the same way regardless of how it is instantiated. The basic process is: Report lands in FTP directory -> Python script detects it -> Python script checks filename, columns, file format -> Python script inserts into database. Please reference the [data\_flow\_diagram](#data_flow_diagram).

### Billing Server

The NZ Billing server is hosted in NZ in the computing room. It is a virtual server running Windows Server 2019. It is only accessible from within the CEVA network. To request access, call the help desk and create a ticket. Your server user must be a different user than your normal CEVA user. It should have -adm appended. For instance, my server login is ‘hentzd-adm’. You can connect using the Remote Desktop application. Details below:

Computer Name: NZMAE01VWAPP002

IP: 10.76.156.24

IT contact (for hosting issues, backups, etc): Michael Keirs <michael.keirs@cevalogistics.com>

### FTP Site

An FTP site was created on the server using the program ‘Internet Information Services (IIS) Manager’. In essence it is a folder within the server that can be accessed by other people using FTP. To use FTP to transfer files to the server, you need the following credentials:

**Path**: D:/FTP

**IP**: 10.76.156.24

**Port**: 20

**File Protocol**: FTP

**Username**: ftpUser

**Password**: Tr4n5f3r!

Files can be transferred to this FTP site programmatically using most languages or using programs like WinSCP.

### PROD - startFolderWatch.bat

This batch file kicks of the data ingestion process by running \Supporting Code\Python\WatchFolder.py. Each developer should have their own version of this to kick off to test the data ingestion process.

### \Supporting Code\Python\WatchFolder.py

This file is responsible for detecting new files that land on the FTP site and initiating the next step of data ingestion. This file should always be running.

### \Configs\generalConfig.json

This JSON file contains configuration used by any Python file in the project that needs it.

This file should not be tracked by git as the configuration determines the environment you work in such as the database, where the email errors go to, etc. For example, my workspace has the following configuration:

{

"database": {

"server": "localhost"

, "port": 5430

, "user": "access\_user"

, "pwd": "coffee"

, "dbase": "akl"

}

, "emailErrors": {

"sender": "[DWBT-NZ-DEV@cevalogistics.com](mailto:DWBT-NZ-DEV@cevalogistics.com)"

, "receivers": ["[phillip.liu@cevalogistics.com](mailto:phillip.liu@cevalogistics.com)"]

}

, "reportCheckerEmailErrors": {

"sender": "[DWBT-NZ-DEV@cevalogistics.com](mailto:DWBT-NZ-DEV@cevalogistics.com)"

, "receivers": ["[phillip.liu@cevalogistics.com](mailto:phillip.liu@cevalogistics.com)"]

}

, "RPAreportCheckerEmailErrors": {

"sender": "[DWBT-NZ-DEV@cevalogistics.com](mailto:DWBT-NZ-DEV@cevalogistics.com)"

, "receivers": ["[phillip.liu@cevalogistics.com](mailto:phillip.liu@cevalogistics.com)"]

}

, "log": "Logs\\"

, "manualWatchFolderPath": "<\\\\akl2wsfps001\\clshare$\\MUWH> - Finance\\Drew DB Folder\\Watch Folder"

, "FTPWatchFolderPath": "C:\\Users\\liuph-adm\\Documents\\Local FTP"

, "condaBat": "C:\\ProgramData\\Anaconda3\\condabin\\conda.bat"

, "sourceDataZone": "D:\\Source Data\\"

, "reportDataZone": "D:\\Generated Reports\\"

}

Whereas the PROD workspace has the following configuration:

{

"database": {

"server": "localhost"

, "port": 5432

, "user": "postgres"

, "pwd": "popplenose"

, "dbase": "akl"

}

, "emailErrors": {

"sender": "[DWBT-NZ-PROD@cevalogistics.com](mailto:DWBT-NZ-PROD@cevalogistics.com)"

, "receivers": ["[Phillip.Liu@Cevalogistics.com](mailto:Phillip.Liu@Cevalogistics.com)"]

}

, "reportCheckerEmailErrors": {

"sender": "[DWBT-NZ-PROD@cevalogistics.com](mailto:DWBT-NZ-PROD@cevalogistics.com)"

, "receivers": ["[Phillip.Liu@Cevalogistics.com](mailto:Phillip.Liu@Cevalogistics.com)"]

}

, "RPAreportCheckerEmailErrors": {

"sender": "[DWBT-NZ-PROD@cevalogistics.com](mailto:DWBT-NZ-PROD@cevalogistics.com)"

, "receivers": ["[Phillip.Liu@Cevalogistics.com](mailto:Phillip.Liu@Cevalogistics.com)"]

}

, "log": "Logs\\"

, "manualWatchFolderPath": "<\\\\syddvwdfs001\\Groups$\\Brooklyn\\Common\\CL> AU WHS Billing Reports"

, "FTPWatchFolderPath": "D:\\FTP"

, "condaBat": "C:\\ProgramData\\Anaconda3\\condabin\\conda.bat"

, "sourceDataZone": "D:\\Source Data\\"

, "reportDataZone": "D:\\Generated Reports\\"

}

Below are descriptions of the properties:

**database:** an object with properties describing how to connect to the database. Changing the port can change the environment you’re working in.

**emailErrors:** an object with sender email and list of receiver emails for errors. EX: an unrecognised report is detected and emailed to receivers list.

**reportCheckerEmailErrors:** an object with sender email and list of receiver emails for ReportChecker errors. For more information on ReportChecker see [check\_for\_source\_report](#check_for_source_report).

**RPAreportCheckerEmailErrors:** an object with sender email and list of receiver emails for RPA ReportChecker errors. For more information on ReportChecker see [check\_for\_source\_report](#check_for_source_report).

**log:** The relative path to the parent directory containing DWBT logs

**manualWatchFolderPath:** Depcrecated. Was used to specify the absolute path to an alternate folder that could be watched for report ingestion.

**FTPWatchFolderPath:** The absolute path to the folder being watched by WatchFolder.py

**condaBat:** The absolute path to the Anaconda batch file. This file is used to run Python scripts in an Anaconda environment.

**sourceDataZone**: The absolute path to folder where reports are downloaded from the FTP site and converted to a format that is digestible by the database.

### Logs

Logs are stored in the directory indicated by generalConfig.json “log” property. The file name takes the form: DWBT\_Log\_<user>\_<date>\_T<time>.txt. The parent directory contains a ‘Test Log’ folder and <year> folder(s) which organise logs by year. The code responsible for managing logs is in \Supporting Code\Python\globalFunctions.py.

### \Supporting Code\Python\ Import.py

The guts of the data ingestions process. Has the following responsibilities:

* Download the file from the FTP site specified in generalConfig.json “FTPWatchFolderPath” property.
* Determine if the file is valid, excluded, unrecognised. This is done using configurations in importConfig.json and excludeConfig.json.
* Convert the file to csv. Can do this for zip, Z, xls, xlsx formats.
* Filter for expected columns and ensure they are in the correct order. Column order determined from importConfig.json “expectedColumnOrder” properties.
* Call PostgreSQL function to upload data to the database landing schema (DB\_STAGE).
* Call PostgreSQL function to load data into the core schema (DB\_LOAD).
* Call PostgreSQL function to process data if there are tables that derive their data from the report.

### \Configs\importConfig.json

A configuration file used for ingesting reports. Contains an array of objects, each object contains the properties described below:

**reportNameRegex:** The name of an acceptable report for which the rest of the properties apply. Can contain regular expressions.

**automationStrategy:** Describes how the report is generated/arrives at the FTP site.

**system:** The name of the system that the report comes from.

**matrixInstance:** When system is “MATRIX”, the instance of Matrix WMS the report comes from. Less applicable now, but previously an Express instance of Matrix was used for smaller customers. There were slight differences between instances and this property was necessary.

**reportCategory:** The category the report falls under. Existing categories are: INBOUND, TRANSACTION, OUTBOUND, STORAGE, FREIGHT, OPS, LABOUR, TRANSPORT. Most applicable for storage reports because the date the report was pulled is in the file name, and needs to be extracted. The exact time storage reports are pulled is extremely important. Whereas other reports should pull the same data if you apply the same date range (unless the range includes the current day and new activity occurs).

**siteId:** When system is “MATRIX”, the site ID for the report. This property addresses a future use case where multiple sites are included in DWBT.

**localTZ:** The time zone of the admin user who completes the billing. Must be one of the time zones in the Python package pytz. It is important for Storage reports. In NZ this time zone will always be the same, but this property will be important if other time zones are ever included. EX: the exact time a storage report was run must be converted to the time zone of the admin. They need to know the time of the snapshot from their perspective.

**nativeTZ:** The time zone used by the report’s source system. Must be one of the time zones in the Python package pytz. It is important for Storage reports. EX: the exact time and time zone a storage report was run must be converted to the time zone of the admin. They need to know the time of the snapshot from their perspective.

**filenameDateRegex:** Sometimes a timestamp must be extracted from the filename (EX: storage reports). This property contains the regular expression required to pull the timestamp from the file name.

**filenameDateFormat:** Sometimes a timestamp must be extracted from the filename (EX: storage reports). This property contains the Python datetime.strptime formats to interpret the timestamp in the filename.

**dataStartLine:** Sometimes reports have data rows that don’t start on the first row. This indicates the row number the data starts.

**sheetName:** If the report is an Excel file, the sheet from which to extract data.

**targetSchema:** The name of the schema the data will land in. Should always be DB\_STAGE.

**targetTable:** The name of the table the data will land in.

**importFunctionName:** The name of the function that will upload the report data to the database.

**loadFunctionCall:** The SQL statement that calls the PostgreSQL function that assigns data types and moves the data from the landing zone (DB\_STAGE) to the core zone (DB\_LOAD).

**chargesFunctionCall:** The SQL statement that calls the PostgreSQL function that populates tables containing derivative data of the reports. EX: calculating billing charges.

**expectedColumnOrder:** An array of column names expected in the report being processed. A column name should exactly match what is in the source report. The column order should exactly match the landing zone table in DB\_STAGE.

### \Supporting Code\Python\globalFunctions.py

Contains classes/function that may be used by other Python files.

**logObj:** Class that is responsible for log management and log entries relating to Python processes.

**ftpObj:** Currently not used. Skeleton class that may eventually be used to FTP files.

**logSubprocessOutcome:** Function that is used to log the outcome of external programs kicked off using Windows via a Python script.

**getJsonData:** Function that gets json data from a file

**sendMail:** Function that sends an email

**getEnvFromPort:** Function that returns the environment based on the port number. Port number is the differentiator of database environments.

### Source Data Landing Zone

Source data is moved from the FTP site to the source data landing zone. The path is specified in the generalConfig.json property “sourceDataZone”. From this property’s path, there is a sub folder for each environment. Then there is a sub folder for each day. Depending on whether the report is recognised, unrecognised, excluded, it will first land in the Downloaded, Unrecognised, Excluded sub folder respectively.

### Converted Source Data

Once a report is in the source data landing zone, it will be converted to a different format if:

* The source report is recognised and should be imported
* The source report came in format that is not csv

The converted source report will appear in the same folder it landed in (the ‘Downloaded’ folder)

### Formatted Source Data

Once a source report is in the source data landing zone and has been converted to csv. If necessary, the file will be transformed and moved to the Formatted data zone. This is in the ‘Formatted’ folder. The main difference between the original source report/converted report and the formatted source report is:

* Only the expected columns are present, and they are put in the expected order
* Storage reports add a column with the date the report was run

### Database Landing Zone

The source report is uploaded to the database in the DB\_STAGE schema. See [import\_functions](#import_functions) for more details.

### Adding a New Report

It’s common to add a new report to DWBT i.e., ingesting a new data source into the data warehouse. Referencing the [data\_flow\_diagram](#data_flow_diagram) is helpful for this task. Generally, the report will land in the FTP folder one way or another. How this is done depends on the source system. This section will focus on the steps to get the data in the database after it lands in the FTP folder. Below are the steps but be sure to look below the steps for a Python tool that does most of the work for you.

1. Create a table in DB\_STAGE where the data will land. This table will have only ‘TEXT’ data types.
2. Use another SP\_IMPORT function as a template to create an import function that will upload data to the DB\_STAGE table.
3. Create a config section in \Configs\importConfig.json. Fill out the key-value pairs. See [import\_config](#import_config) for details. The order of the columns in this section should exactly match the order of the columns in DB\_STAGE.
4. Try running your developer version of ‘PROD - startFolderWatch.bat’ (see [start\_watch\_folder](#start_watch_folder)) and dropping the report in the FTP folder
5. When the report data successfully lands in the DB\_STAGE table, create a DB\_LOAD table with data types assigned to columns.
6. Create a DB\_ACCESS pass-through view.
7. Use another SP\_LOAD function as a template to create a load function that will move data from DB\_STAGE to DB\_LOAD an assign data types.
8. Update \Configs\importConfig.json to reference the LOAD function that was created.

As mentioned above, there is a Python script that can generate code skeletons of all the above objects. Located: C:\Users\Public\Documents\Tools\Python\New Data Source\newDataSourceSkeleton.py. It looks in the folder ‘C:/Users/hentzd-adm/Documents/Tools/Python/New Data Source/Input/’ for files and reads the first line of the file to determine column names for the skeletons. Column names should be tab separated. Output skeleton will be put in ‘C:/Users/hentzd-adm/Documents/Tools/Python/New Data Source/Output/’

## Reporting

This section describes reporting done with DWBT NZ. Reporting includes but is not limited to:

* Reports sent to customers such as KPI status, inbound/outbound units
* Reports sent internally to aid in business decisions
* Reports sent internally to allow for exception handling of items not meeting the KPI

DWBT NZ reporting is done using a python script called generateReport.py and a JSON config file called reportConfig.json, both of which are tracked by the DWBT NZ git repository.

### \Supporting Code\Python\ generateReport.py

This is the main script for generating reports. Please note that this is far from perfect and improvements would be very helpful.

Like the import.py script, it uses the same Anaconda billing environment to run Python. And like the import.py script, it writes to the same log file.

It is primarily made up of a Report Class that gets instantiated into separate Report Objects. Each Object has functions which their functionalities are described in comments in the script itself. At the end of the file there is a main() function call which takes the argument passed when running the script as its main parameter to determine which configurations to use for sending the report.

I understand that this is likely not best practice, therefore, improvements and suggestions are welcomed.

### \Supporting Code\Config\reportConfig.json

Reports are sent based on what is in the JSON configuration file. The current parameters are:

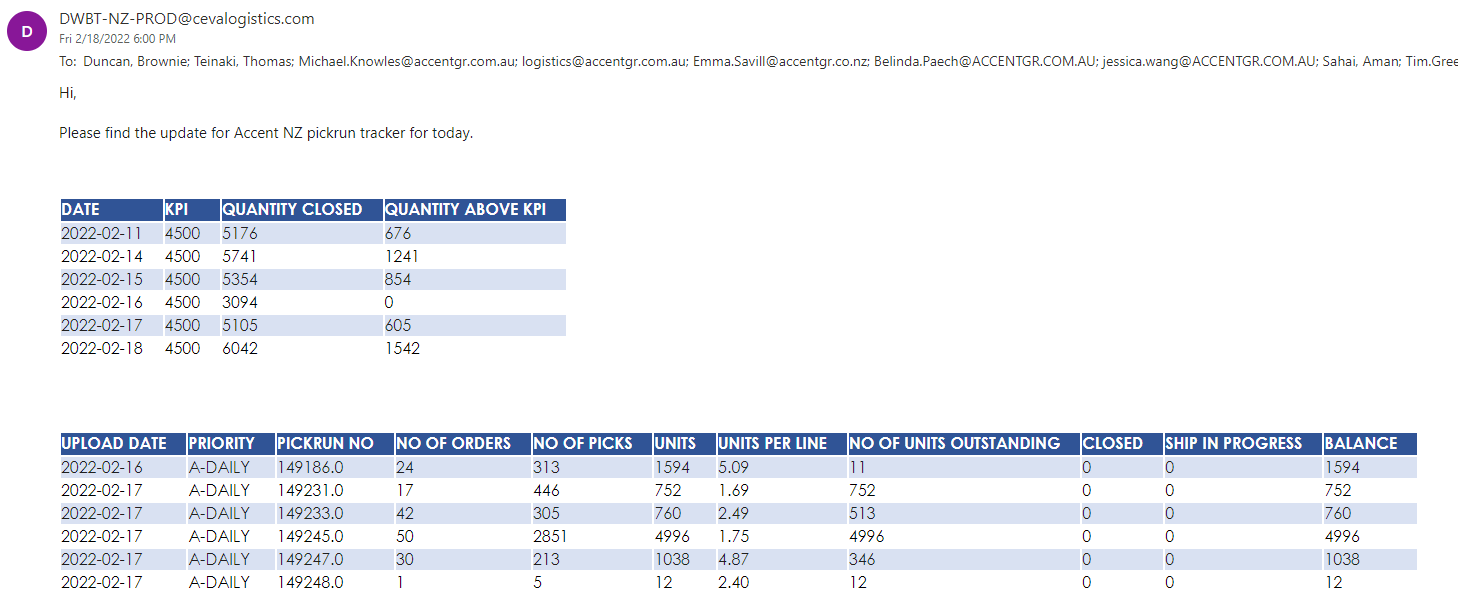
* **reportName**:
  + Type: String
  + Description: Name of the report which is used as a command line argument when calling the script to determine which report configuration to use
* **viewNames**:
  + Type: Array
  + Description: The views in the database from the DB\_ACCESS schema that we want to use to create this report
* **mailSender**:
  + Type: String
  + Description: The mail sender. It will either be [DWBT-NZ-PROD@cevalogistics.com](mailto:DWBT-NZ-PROD@cevalogistics.com) or [DWBT-NZ-DEV@cevalogistics.com](mailto:DWBT-NZ-PROD@cevalogistics.com)
* **mailRecipients**:
  + Type: Array
  + Description: Who to send the report to
* **outputFile**:
  + Type: Boolean
  + Description: Determines whether to send the report as an XLSX file or as a data frame in the email body
* **outputSheetNames**:
  + Type: Array
  + Description: The name of the sheets in the XLSX file if sending the report as an attachment. Correlates 1 to 1 with the number of views we are querying
* **graphs**:
  + Type: JSON object
  + Description: Determines whether we want to send a graph or not and what parameters to send it with. Refer to the createGraphImage function in generateReport.py to see what the script does with these parameters
* **timePeriod**:
  + Type: String
  + Description: Determines the naming of the report the mail recipients receive. Can either be “daily”, “weekly” or “monthly”

### Scheduling

Reports are currently scheduled using WTS (Windows Task Scheduler) on the server. WTS runs a cmd script at custom set periods. The cmd script runs the generateReport.py file with desired the name of the report as the only script argument.

Note that when running from your own workspace it will use the DEV environment and when running it from the Public PROD workspace it will use the PROD environment. The script uses the generalConfig.json file, which is not, and shouldn’t be tracked by git to determine which environment you’re working in. When calling the script from the PROD workspace it uses the generalConfig within the PROD workspace and therefore will query the PROD database and use the PROD config in the reportConfig.json file and vice versa with the DEV environment.

For example, the Accent pickrun report looks like this:

This report is simply a view in the database in the DB\_ACCESS schema, the script connects to the database, queries the view, and transforms the query results into a dataframe, applies formatting and then sends the report based on config parameters.

Windows Task Scheduler handles the scheduling of these reports.

## Database Architecture

### Environments

DWBT has 3 parallel instances: DEV, TEST, and PROD. This is industry-standard structure and allows for development and testing to happen while being isolated from the live version of the tool. The only difference between the environments is the port on which the database is active. See table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **IP Address** | **Port** | **Data Storage Path** |
| DEV | 10.76.156.24 | 5430 | D:\PostgreSQL DEV |
| TEST | 10.76.156.24 | 5431 | D:\PostgreSQL TEST |
| PROD | 10.76.156.24 | 5432 | D:\PostgreSQL PROD |

The PROD environment is the only environment actively receiving reports. You can make configuration changes to use other environments while developing/testing. The DEV and TEST environments are not automatically refreshed with PROD data, you can do this refresh yourself when needed.

To refresh an environment, run the following command in cmd:

"D:\Backups\PostgreSQL\Batch Files\pgEnvironSync.bat" <lowercase environment>

EX: "D:\Backups\PostgreSQL\Batch Files\pgEnvironSync.bat" test

### Creating Environments

Usually, different environments live on different servers, but I didn’t have time to work with IT to create the infrastructure for lower environments. That’s why all three environments are on the same server with different ports. I created the PROD environment with the normal installation file. However, you can’t use the installation file to install a second instance on the same machine. I followed the YouTube video below:

<https://www.youtube.com/watch?v=RYtd8j62rA0&authuser=0>

**The steps:**

1. Make a copy of the program files
   1. Find the existing program files and copy them within the same directory. Rename.
   2. EX: Make a copy of “C:\Program Files\PostgreSQLDEV” and rename to “C:\Program Files\PostgreSQLTEST”
2. Make a copy of the data directory
   1. Find the existing data directory and copy them within the same directory. Rename.
   2. EX: Make a copy of “D:\PostgreSQL DEV” and rename to “D:\PostgreSQL TEST”
3. Use the new program files to register a service that is linked to the data directory:
   1. "C:\Program Files\PostgreSQLTEST\9.6\bin\pg\_ctl" register -N postgresql-x64-9.6-TEST -U "NT AUTHORITY\NetworkService" -D "D:\PostgreSQL TEST"
4. Update the new postgresql.conf to use the port you want
   1. EX: “D:\PostgreSQL TEST\postgresql.conf”
5. Change the new postmaster.opts to new install location and data location
   1. EX: Change “D:\PostgreSQL TEST\postmaster.opts” to contain C:/Program Files/PostgreSQLTEST/9.6/bin/postgres.exe "-D" "D:\PostgreSQL TEST"
6. Delete the .pid file
   1. EX: Delete “D:\PostgreSQL TEST\postmaster.pid”
7. Start the service
   1. Use the services application to start the service you registered in step 3.

### SQL Formatting and Naming Conventions

**Formatting Rules:**

* Object name words separated by underscores.
* Everything in uppercase. This helps prevent camel-casing, and enforces underscore word separators.
* SQL keywords on their own line: SELECT, FROM, INNER JOIN, etc. Makes it easier to move lines around in text editors.
* Opening parentheses at the end of the line.
* Keyword continuations are tabbed (EX: columns being selected)
* All tables are aliased, and respective columns are fully qualified (helps trace which columns come from which table).
* Joined tables always have their columns on the left of the equal sign.
* Sub-queries are tabbed

**Formatting Example:**

**SELECT**

BASE.BILLING\_YEAR

, BASE.BILLING\_PERIOD

, SPLITS.CLIENT\_GROUP

, SPLITS.EMPLOYMENT\_TYPE

, BASE.TOTAL\_HOURS \* SPLITS.SPLIT **AS** TOTAL\_HOURS

, BASE.TOTAL\_COST \* SPLITS.SPLIT **AS** TOTAL\_COST

**FROM** (

**SELECT**

XBF.BILLING\_YEAR

, XBF.BILLING\_PERIOD

, LEH.EMPLOYMENT\_TYPE

, **SUM**(LEH.TOTAL\_HOURS) **AS** TOTAL\_HOURS

, **SUM**(LEH.TOTAL\_COST) **AS** TOTAL\_COST

**FROM**

DB\_ACCESS.LBR\_EMPLIVE\_HRSBREAKDOWN **AS** LEH

**INNER** **JOIN**

DB\_LOAD.DIM\_DATE **AS** DD

**ON** DD.DD\_DATE = LEH.**DATE**

**INNER** **JOIN**

DB\_LOAD.XWALK\_BILLING\_FREQUENCY **AS** XBF

**ON** XBF.DIM\_DATE\_CK = DD.DIM\_DATE\_CK

**AND** XBF.BILLING\_FREQUENCY = 'WEEKLY\_MONDAY'

**GROUP** **BY** 1, 2, 3

) **AS** BASE

**INNER** **JOIN**

DB\_ACCESS.VW\_DIM\_BU\_LBRHR\_SPLITS **AS** SPLITS

**ON** SPLITS.EMPLOYMENT\_TYPE = BASE.EMPLOYMENT\_TYPE

**Naming Conventions:**

* Incoming Reports – Tables containing data that originated from a report.
  + <Report Category>\_<Source System>\_<Report Name>
    - Report Categories
      * FR – Freight
      * IB – Inbound
      * OB – Outbound
      * OPS – Operations
      * SG – Storage
      * TPT – Transport
      * TXN – Transaction
      * CS - Consumables
    - EX: SG\_MATRIX\_WAREHOUSE\_OCCUPANCY
* Dimension Tables – Tables containing auxiliary data needed for calculations
  + DIM\_<Description>
  + EX: DIM\_ACCT
* Outgoing Reports – Reports sent from the database pertaining to a view or views from the DB\_ACCESS schema.
  + VW\_RPT\_ <Client\_name>\_ <Report Category>\_<Report\_Name>\_<Frequency>
  + EX: VW\_RPT\_ACCENT\_IBOB\_UNITS\_DAILY

## Database Objects

### Schemas

The database follows industry best-practice for schemas:

**DB\_STAGE:** The landing zone for source reports. Each source report has a corresponding table in this schema. The import PostgreSQL functions are used to upload data from the formatted csv file to the corresponding table in DB\_STAGE. Tables in DB\_STAGE do not have data types assigned, they are all Text/Varchar data types. This minimises the chance of failure when uploading csv data.

**DB\_LOAD:** Source report data is moved from DB\_STAGE to DB\_LOAD using the load PostgreSQL functions. Tables in DB\_STAGE have a one-to-one relationship with tables in DB\_LOAD. The only difference is the DB\_LOAD tables have primary keys, indices, and data types. This schema also contains auxiliary dimension tables used in calculations.

**DB\_ACCESS:** Contains objects that users interact with. For instance, one-to-one source report pass-through views and the billing charges table.

### PostgreSQL Import Functions

These functions are called by Import.py using importConfig.json “importFunctionName” property. They accept the path of the formatted source report as an argument and handle the upload to the DB\_STAGE table in importConfig.json. The import function structure is the same across all import functions, with only the target table name changed in each one. Dynamic SQL using a single import procedure was attempted but was too slow. The function always executes the following steps:

* Creates a copy of the target DB\_STAGE table as a temporary table and remove te metadata columns (insert user, timestamp, etc).
* Upload data to the temporary table.
* Insert/Select into the target table, adding in the metadata columns.

The reason there are 3 steps is because the metadata columns are important for troubleshooting, and there was no way to include the metadata columns as part of the PostgreSQL COPY command that does the upload.

### PostgreSQL Load Functions

These functions are called by Import.py using importConfig.json “loadFunctionCall” property. They move source report data from DB\_STAGE to DB\_LOAD. They assign data types and handle primary key conflicts.

### PostgreSQL Charges Functions

These functions are called by Import.py using importConfig.json “chargesFunctionCall” property. Each source report triggers its own importConfig.json “chargesFunctionCall” property value. Meaning each source report calls a function that calculates something derived from the data in the report. They perform data transformations that must be stored in a table. EX: RECEIPT\_ACTIVITY\_by\_date\_range report triggers DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_IB\_MATRIX\_RECEIPT() which calculates billing charges related to the report and stores them in DB\_ACCESS.FT\_CHARGES\_SUMMARY.

**Dependencies:** Sometimes the PostgreSQL Charges Functions are dependent on other functions. Best explained with an example:

CL charges customers for storage occupied weekly. However, the storage report is a point-in-time report. Therefore, inventory could be inbounded after the moment storage report was run. To capture and charge all storage, you must take the storage report at the beginning of the week and add all inventory inbounded during the week. The storage charge calculation therefore requires storage data and inbound data which come from 2 different reports. If either of these reports is received into DWBT, the calculation must be redone. Therefore, when storage data is received, the storage charges function runs. When inbound data is received, the inbound charges function runs AND the storage charges function runs. These dependencies are noted in the descriptions below:

* SP\_CHARGES\_SUMMARY\_NZ\_GDT\_IB\_SAP\_AKL
  + Runs
    - DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - GDT Inbound Passenger
    - GDT Inbound Light Truck / SUV
    - GDT Inbound Truck / Commercial
    - GDT Inbound Aviation
* SP\_CHARGES\_SUMMARY\_NZ\_GDT\_OB\_SAP\_AKL
  + Runs
    - DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - GDT Outbound Passenger
    - GDT Outbound Light Truck / SUV
    - GDT Outbound Truck / Commercial
    - GDT Outbound Aviation
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_CONSUM
  + Runs
    - DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Consumables for all customers that have consumables charges.
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_FMS\_CONSOLAKL
  + Runs
    - DB\_LOAD. SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_CONSUM
    - Which runs
      * DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - All transport charges for all customers that use transport.
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_IB\_MATRIX\_RECEIPT
  + Runs
    - DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Karma Inwards - Pallet
    - Karma Inwards - Carton
    - LSS Inbound Lines
    - LSS - Returns - Advice per line/ per order
    - HK Put Away Fee
    - HK - Customer Returns
    - All Good Inwards - Carton
    - All Good Inwards - Pallet
    - All Good Inwards - PO
    - Lululemon Inbound - Carton Receipts
    - Acer serial number tracking: New serial numbers
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_IB\_YMT
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Container charges for all customers that have them.
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Fills charge dummy rows (charges with zero units). This allows the billing admin to see all the charges even if they do not contain anything.
    - Fixed charges for all customers (charges that always happen once a week)
    - GDT AKL Labour
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_OB\_MATRIX\_PICK
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - HKW Outbound Orders
    - HKW Outbound Lines
    - Acer Outbound - Picking
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_OB\_MATRIX\_SHIP
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - None, but updates Acer serial number table. Serial numbers are a column unique to this report among other Matrix OB reports.
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_OB\_WIMS\_PICK
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Accent outbound units - apparel and shoes
    - Accent outbound surcharge units
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_SG\_MATRIX
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Runs DB\_LOAD.SP\_PACK\_CONFIG
  + Charges calculated
    - Kogan Storage charges
    - Karma Storage - Pallet 1M x 1.2M x 1.4M
    - HKW Storage charges - Pallet Bulk, Shelved Pick Face Locations
    - Goddard Storage charges - Oversize Pallets Per Week or Part Thereof, Storage Per Racking Location Per Week or Part Thereof
    - MG Double Pallet Storage
    - MG Pallet Storage (1200x1000mmx1350mm)
    - LSS Storage: High and Low Pallets
    - All Good Organics Storage - Pallet
    - MG Pallet Storage (1200x1000mmx1350mm)
    - LSS Storage: High and Low Pallets
    - All Good Organics Storage - Pallet
    - Lululemon Storage - Oversize Pallets 1.0m x 1.20m x 1.41m to 1.70m, Storage - Pallets 1.0m x 1.20m x 1.40m
    - Acer Storage - Pallets
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_SG\_MATRIX\_INV\_DET
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - None, but helps with Acer Serial Number tracking
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_SG\_WIMS\_STOCK\_SUMM
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Accent Overflow units >200K
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_TXN\_MATRIX
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Kogan Single SKU pallet put away to storage location and Shrink Wrapping
    - Weekend and holiday management fee Kogan
    - LSS Order Assembly - Line Items
    - LSS Order Assembly - Sales Orders
    - Kogan outbound charges
    - Acer scanning serial numbers
    - Karma outbound
    - Goddards OB picking per pallet
    - MG IB Line Receipt and Put away
    - MG OB Pick, pack, ship
    - All Good Outwards - Pallet pick
    - All Good Outwards - Carton pick
    - All Good Outwards - Sales Order, Outwards - Shrinkwrap pallet
    - Lululemon Outbound Cartons
* SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_TXN\_WIMS
  + Runs DB\_LOAD.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC
  + Charges calculated
    - Accent labour hours
    - Accent inbound units
    - Accent inbound surcharge units

### PostgreSQL Tables

* ACER\_SERIAL\_NUMBER\_TXN
  + Tracks which serial numbers are in the warehouse for Acer. Uses inbound data to determine which serial numbers have arrived, and outbound data to determine which serial numbers have left.tra
  + Not currently accurate. Needs data from SG\_MATRIX\_INVENTORY\_DETAILS because serial numbers can be instantiated on Matrix WMS through stock checks (serial number wasn’t input during inbound but was input during a stock check).
* COCO\_REP\_WMS
  + Was setup to be a mini-WMS for Coco Republic until they were put on Matrix WMS. Not used anymore.
* CS\_CONSUMABLES\_MASTER
  + Tracks consumable items that need to be billed to customers. Admin buys consumables then uses the Excel tool ‘Consumables Master.xlsm’ to input consumables to DWBT. Consumables are purchased for all customers, so we decide a percentage split that each customer receives of the total cost.
* DEPS\_SAVED\_DDL
  + Temporarily tracks DDL for objects that need to be dropped in order to replace an object they depend on. See [object\_dependencies](#object_dependencies).
* DIM\_ACCENT\_RATE\_THRESHOLD
  + Auxiliary table that stores Accent inbound and outbound calculation logic. Accent inbound and outbound charges have different rates depending on the number of units processed. This table stores these upper and lower limits so the table can be used for the calculation instead of hard coded in a function.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\ DIM\_ACCENT\_RATE\_THRESHOLD.xlsx
* DIM\_ACCT
  + Contains high level data about customers. Each row represents an entity that is billed.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\ CUSTOMER TABLES.xlsx on the DIM\_ACCT tab.
* DIM\_BU\_LBRHR\_CAT\_SPLITS
  + Auxiliary table used to calculate hours spent inbounding inventory vs hours spent outbounding orders. There’s no way to track how much time each employee spent on inbounding vs outbounding, so we approximated percentages. Inbounding is more time-intensive so generally takes 60% vs 40% outbounding.
* DIM\_BU\_LBRHR\_SPLITS
  + Auxiliary table used to calculate labour cost allocation to customers based on full time and temp labour stratifications. There’s no way to track how much time each employee spent on each customer, so we approximated percentages.
* DIM\_CARTON\_CONV
  + Auxiliary table used to calculate how many SKU units comprise a carton. We charge customers LSS and HKW by the carton for some charges.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\ DIM\_CARTON\_CONV.xlsx
* DIM\_CLIENT\_MAPPING
  + Auxiliary table used to relate source system customer identifiers to an account.
  + EX: The same customer, Kogan, is identified as CLIENT\_ID ‘KGN524’ in Matrix WMS, and ‘KOGAN’ in FMS. This table helps you relate tables from different source systems.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\ CUSTOMER TABLES.xlsx on the DIM\_CLIENT\_MAPPING tab.
* DIM\_DATE
  + Auxiliary table that serves as a basic calendar.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\DIM\_DATE.xlsx
* DIM\_DESTINATION
  + Auxiliary table that relates FMS destinations regions. Transport charges sometimes depend on the region.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\DIM\_DESTINATION.xlsx
* DIM\_DIFOT\_SLA
  + Auxiliary table that contains data used for calculating DIFOT customer SLA’s.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\ DIM\_DIFOT\_SLA.xlsx
* DIM\_DIFOT\_SLA\_COMMENTARY
  + Auxiliary table that contains data to accompany the SLA calculations in cases where an explanation is required.
* DIM\_FOOTPRINTS
  + Auxiliary table used to allocate property cost across business units.
* DIM\_FREIGHT\_SURCHARGE
  + Auxiliary table used to calculate transport surcharges. This table is updated by the Transport supervisor using the Excel tool ‘Freight Surcharge Manager.xlsm’
* DIM\_FULL\_PLT\_THRSHD
  + Auxiliary table used to calculate how many SKU units comprise a pallet. Some customers are charged by the pallet and don’t have accurate pack configs.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\DIM\_FULL\_PLT\_THRSHD.xlsx
* DIM\_GDT\_CAT
  + Auxiliary table used to relate tires to a charge category. The ‘material’ column functions like a SKU column.
  + Has an insert spreadsheet: \Supporting Code\SQL\Tables\Insert Spreadsheets\DIM\_GDT\_CAT.xlsx
* DIM\_GDT\_CHC\_FIXED\_COST
  + Auxiliary table that stores fixed GDT charges. Since GDT charges are cost plus a surcharge percentage, these charges couldn’t go in the normal fixed charge calculation. This table allows the charges to be calculated by multiplying the cost by the surcharge rate.
* DIM\_IB\_SLA\_COMMENTARY
  + Currently not used. Auxiliary table that stores commentary for IB SLA in cases where the SLA requires explanation.
* DIM\_INBOUND\_SLA
  + Currently not used. Auxiliary table that stores parameters for calculating the IB SLA for customers.
* DIM\_KOGAN\_PACK\_CONFIG\_TEMP
  + Auxiliary table that was setup as a temporary fix to a problem. To calculate some Kogan charges, we need to know how many units comprise a pallet for a given SKU. We use pack config data grabbed from TXN\_MATRIX\_CONSOL and stored in DIM\_PACK\_CONFIG (via SP\_PACK\_CONFIG). for the majority of SKUs. However, there were many incorrect pack configs and this table contains the data to override incorrect pack configs in DIM\_PACK\_CONFIG.
* DIM\_OUTBOUND\_SLA
  + An auxiliary table that stores outbound SLAs defined in customer contracts. Used to determine whether we are meeting our outbound KPI for each customer.
* DIM\_PACK\_CONFIG
  + Auxiliary table containing packing relationships for SKUs. To calculate some Data is grabbed from TXN\_MATRIX\_CONSOL and stored in DIM\_PACK\_CONFIG via SP\_PACK\_CONFIG. DIM\_KOGAN\_PACK\_CONFIG\_TEMP contains a subset of pack configs that are more accurate for Kogan.
  + TRACKING\_LEVEL\_<integer> represents a real-world packing configuration like units/cartons/pallets.
  + RATIO\_<integer\_x>\_TO\_<integer\_y> represents the number of TRACKING\_LEVEL\_X in TRACKING\_LEVEL\_Y.
  + EX:
    - TRACKING\_LEVEL\_1 = EA
    - RATIO\_1\_TO\_2 = 10
    - TRACKING\_LEVEL\_2 = CA
    - RATIO\_2\_TO\_3 = 473
    - TRACKING\_LEVEL\_3 = PL
    - 10 units (EA) in a carton (CA). 473 cartons (CA) in a pallet (PL)
  + The scope of a pack config is by CLIENT\_ID. This means that any site containing the same CLIENT\_ID can see pack configs of other sites containing the same CLIENT\_ID. EX: Kogan is a customer in NZ and a customer in AUS. AUS may use specific pack configs and NZ may use specific pack configs, but we can see their pack configs and they can see ours.
* DIM\_RATE
  + Contains all rates for all customers including history.
  + Columns
    - DIM\_RATE\_CK – unique row identifier in DIM\_RATE
    - DISPLAY\_ORDER – Contains a number used to order the rates in a convenient way. Usually orders so categories are grouped together.
    - DIM\_CLIENT\_MAPPING\_CK - Unique row identifier from DIM\_CLIENT\_MAPPING table.
    - ACTIVITY\_CAT – Allowable values: ‘N/A’, ‘Warehouse’, ‘Transport’
    - ACTIVITY\_SUB\_CAT – Further categorises the rate. EX: ‘Outbound’
    - RATE\_DESC – Description of the rate.
    - RATE\_GROUP – Groups related rates together if necessary. For instance, Accent IB and OB rates are stratified based on number of units for the week. They have the same rate group because they are the same rate just for different number of units.
    - RATE\_EXPLANATION – Column where the billing admin can put explanations for rates that are irregular
    - RATE – The rate to charge
    - UOM\_DESC – unit of measure for the units. Also used for transport charges to identify container size
    - CALCULATION\_DESC – Description of how the units are calculated.
    - SOURCE\_REPORT – Name of the report(s) sourcing the rate’s units calculation
    - PARTS\_CUST\_ID – Accounting column used by Australia. Included in case AUS and NZ billing systems need to merge.
    - PARTS\_CUST\_DESC - Accounting column used by Australia. Included in case AUS and NZ billing systems need to merge.
    - GL\_CODE - Accounting column used by Australia. Included in case AUS and NZ billing systems need to merge.
    - SUBLEDGER\_CODE - Accounting column used by Australia. Included in case AUS and NZ billing systems need to merge.
    - SUBLEDGER\_TYPE - Accounting column used by Australia. Included in case AUS and NZ billing systems need to merge.
    - FIXED\_IND – Indicates the rate is fixed and charged every week
    - AUTO\_CALCULATED\_IND – Indicates the rate is calculated automatically
    - REQUIRES\_ATTN\_DURING\_BILLING\_IND – Indicates the charge requires admin attention during billing. Usually the inverse of AUTO\_CALCULATED\_IND.
    - ACTIVE\_IND – Indicates the rate is active.
    - ACTIVE\_START – Indicates the rates active start timestamp
    - ACTIVE\_END – Indicates the rates active end timestamp
    - CUST\_PURCHASE\_ORDER - Column used by Australia. Included in case AUS and NZ billing systems need to merge. In AUS used to track the customer’s purchase order number used to pay for the rate.
* DIM\_SERVICE
  + Auxiliary table used to relate transport service abbreviations to descriptions.
* DIM\_TPT\_KPI\_ORDER
  + Auxiliary table used to order KPI’s
* DIM\_XWALK\_ACCENT\_PRODUCTS
  + Auxiliary table used to calculate Accent charges. We charge by the category Shoes vs Apparel and this table relates Accent products to those categories.
* DIM\_XWALK\_HOLIDAYS
  + Auxiliary table used to relate days to holidays. Use by left joining on a day. If DIM\_WXALK\_HOLIDAYS data is present, it’s a holiday.
* FR\_FMS\_CNM\_INFO
  + Report table see [data\_sources](#data_sources) section.
* FR\_FMS\_CONSOLAKL
  + Report table see [data\_sources](#data_sources) section.
* FR\_FMS\_PHANT
  + Report table see [data\_sources](#data_sources) section.
* FT\_CHARGES\_SUMMARY
  + The culmination of the billing project. Contains charges calculated by the SP\_CHARGES\_ functions.
* IB\_ACCENT\_FORECAST
  + Report table see [data\_sources](#data_sources) section.
* IB\_COCO\_REP\_COMM\_INV
  + Not used. Report table that was intended to store COCO Republic report data before they were set up on Matrix
* IB\_KOGAN\_FORECAST
  + Report table see [data\_sources](#data_sources) section.
* IB\_MATRIX\_RECEIPT\_ACTIVITY\_BY\_DATE\_RANGE
  + Report table see [data\_sources](#data_sources) section.
* IB\_SAP\_AKL
  + Report table see [data\_sources](#data_sources) section.
* IB\_SAP\_CHC
  + Report table see [data\_sources](#data_sources) section.
* LBR\_EMPLIVE\_HRSBREAKDOWN
  + Report table see [data\_sources](#data_sources) section.
* OB\_ACCENT\_PICKRUN
  + Report table see [data\_sources](#data_sources) section.
* OB\_FMS\_ACER\_PROD\_ENQ
  + Report table see [data\_sources](#data_sources) section.
* OB\_FMS\_ERR\_RPT\_KOGAN
  + Report table see [data\_sources](#data_sources) section.
* OB\_KOGAN\_CEVA\_BACKORDERS
  + Report table see [data\_sources](#data_sources) section.
* OB\_MATRIX\_DETAILED\_ORDER\_RECEIPT
  + Report table see [data\_sources](#data_sources) section.
* OB\_MATRIX\_PICKING\_ACTIVITY
  + Report table see [data\_sources](#data_sources) section.
* OB\_MATRIX\_SHIPPING\_ACTIVITY\_BY\_DATE\_RANGE
  + Report table see [data\_sources](#data_sources) section.
* OB\_SAP\_AKL
  + Report table see [data\_sources](#data_sources) section.
* OB\_SAP\_CHC
  + Not used. Mean to store OB data for GDT CHC
* OB\_WIMS\_SO\_ITEMS
  + Report table see [data\_sources](#data_sources) section.
* OB\_WIMS\_SO\_PICK
  + Report table see [data\_sources](#data_sources) section.
* OB\_WIMS\_SO\_RPT
  + Report table see [data\_sources](#data_sources) section.
* OPS\_WIMS\_SO\_ALLOC\_PICK\_KPI
  + Report table see [data\_sources](#data_sources) section.
* SG\_MATRIX\_INVENTORY\_DETAILS
  + Report table see [data\_sources](#data_sources) section.
* SG\_MATRIX\_WAREHOUSE\_OCCUPANCY
  + Report table see [data\_sources](#data_sources) section.
* SG\_WIMS\_STOCK\_SUMMARY
  + Report table see [data\_sources](#data_sources) section.
* TPT\_KPI\_POD
  + Report table see [data\_sources](#data_sources) section.
* TXN\_MATRIX\_CONSOL
  + Report table see [data\_sources](#data_sources) section.
* TXN\_WIMS\_TRANSACTION\_EXTRACT
  + Report table see [data\_sources](#data_sources) section.
* XWALK\_BILLING\_FREQUENCY
  + Relates days (DIM\_DATE\_CK key column for DIM\_DATE) to billing year and billing periods for different billing frequencies. Right now they are all ‘WEEKLY\_MONDAY’ frequency, but it’s possible we could get a customer that is on a different billing frequency.
* YARD\_MANAGEMENT\_TOOL
  + Report table see [data\_sources](#data_sources) section.

### PostgreSQL Views

Every auxiliary and report table should have a pass-thru view in the access layer (DB\_ACCESS). These views are named exactly like the table and have no transformations. There are also VW\_ prefixed views which are standalone views (not a pass-thru) that do transformations. There are sub-categories of VW\_ views. General-purpose views are solely prefixed with VW\_. Views used by the Excel front-end are prefixed VW\_EXCEL. Views that are used by the Python supporting data export are prefixed VW\_EXPORT. Report views are prefixed with VW\_RPT.

#### General-Purpose Views

* FR\_FMS\_CONSOLAKL\_W\_SURCH
  + A pass-thru view of FR\_FMS\_CONSOLAKL that also calculates surcharges.
* VW\_ACCENT\_DCM\_KPI
  + Calculates KPIs for Accent. Since Accent is on WIMS and not Matrix, it requires a separate view.
* VW\_ACCENT\_SCA\_DASHBOARD
  + Calculates SCA (site classification audit) metrics. Fraema uses this view to update a spreadsheet that tracks these values.
* VW\_ACER\_DCM\_KPI
  + Same as VW\_DCM\_KPI, but filtered for Acer customer
* VW\_ACTIVE\_BILLING\_PERIOD
  + Returns the billing year and billing period that contains the current day.
* VW\_BU\_LBRHR\_TOTALS
  + Calculates labour hours for each BU.
* VW\_BUSINESS\_DAYS\_SEQUENCE
  + Sequences business days so business for day arithmetic. Assigns a number to each day that increases if the day is a business day.
  + EX:
    - Wed – 1
    - Thurs – 2
    - Friday – 3
    - Saturday – 3 (not a business day, no increase)
    - Sunday – 3 (not a business day, no increase)
    - Monday – 4
    - To get the day 3 business days after Wed, just add 3 to the sequence. 1 + 3 = 4 and the day with a sequence 4 is the following Monday.
* VW\_CALC\_REFRESH\_PD
  + View that tells the charges functions how far back to do calculations. Helps restrict amount of data that is processed when charges are re-calculated.
* VW\_CHARGES\_3\_MO\_AVG
  + View that shows the 3 month average for charge, charge group, charge category. Will be used with approval tool to show how recent charges compare to their 3 month averages.
* VW\_CLIENT\_BILLING\_PERIOD
  + View that relates billing year and period to DIM\_CLIENT\_MAPPING\_CK’s and days. Useful for translating a day to a billing year/period for a given customer.
* VW\_DATE\_BREAKDOWN
  + View that gives a billing year, period, and month for every day. Assumes there is only one billing frequency in XWALK\_BILLING\_FREQUENCY. There is now, but could be others in future.
* VW\_IB\_ACCENT\_ADHOC\_LBRHRS
  + View used to calculate labour hours for Accent. Base view that is nested in others. Ultimately sent as a report using generateReport.py.
* VW\_IB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED
  + View used to get sum of stock for Accent. Ultimately sent as a report using generateReport.py.
* VW\_IB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED\_PREVWEEK
  + Same as VW\_IB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED but for previous week
* VW\_IB\_ACCENT\_ADHOC\_LBRHRS\_PREVWEEK
  + Same as VW\_IB\_ACCENT\_ADHOC\_LBRHRS but for previous week.
* VW\_IB\_MATRIX\_NZ\_KOGAN\_PALLET
  + Same transformations as VW\_SG\_MATRIX\_NZ\_KOGAN\_PALLET. Kogan pallet billing has a special clause that states we must charge for the number of theoretical pallets, not actual. This means the total units for a given SKU is divided by the units per pallet to get the theoretical number of pallets if we properly combined all the units. For example, if you had 2 pallets that were half full of the same SKU, you would charge for one pallet. This view calculates inbound pallets for Kogan billing.
* VW\_IB\_MATRIX\_NZ\_MUWH\_LOCATIONS
  + Looks at the transaction table to figure out the storage location of inbounded items. That is, where was an inbounded item put. Sometimes it’s necessary to know the location/sub zone that something was inbounded to for charges
* VW\_IB\_MATRIX\_RECEIPT\_ACTIVITY\_BY\_DATE\_RANGE\_MAX\_UPD\_TS
  + A view that returns most recently inserted data for IB\_MATRIX\_RECEIPT\_ACTIVITY\_BY\_DATE\_RANGE.
* VW\_KOGAN\_CONTAINER\_COUNT\_ACTUAL
  + Not sure this is actually used [Phillip]
* VW\_KOGAN\_CONTAINER\_COUNT\_FORECAST
  + Not sure this is actually used [Phillip]
* VW\_KOGAN\_CONTAINER\_REVENUE\_ACTUAL
  + Not sure this is actually used [Phillip]
* VW\_KOGAN\_CONTAINER\_REVENUE\_FORECAST
  + Not sure this is actually used [Phillip]
* VW\_KOGAN\_CONTAINER\_SKU\_REVENUE\_ACTUAL
  + Not sure this is actually used [Phillip]
* VW\_KOGAN\_CONTAINER\_SKU\_REVENUE\_FORECAST
* VW\_KOGAN\_CONTAINERS\_SKU\_QTY\_ACTUAL
* VW\_KOGAN\_CONTAINERS\_SKU\_QTY\_FORECAST
* VW\_KOGAN\_IB\_SKU\_QTY\_ACTUAL
* VW\_KOGAN\_DCM\_KPI
  + Same as VW\_DCM\_KPI, filtered for Kogan
* VW\_KOGAN\_LABOUR\_HOURS
  + Calculates extra ad hoc labour hours for Kogan based on inbounded containers. Categories are assigned by weight, and if the units in a weight category exceeds a threshold, ad hoc hours are charged.
* VW\_DCM\_KPI
  + Calculates KPIs submitted to Global for Matrix WMS customers.
* VW\_LBR\_COST\_TOTALS
  + Calculates labour cost allocation to customers based on full time and temp labour stratifications. There’s no way to track how much time each employee spent on each customer, so we approximated percentages.
* VW\_LBR\_COST\_VS\_REVENUE
  + Compares labour cost to revenue for a Qlik Sense BI dashboard.
* VW\_LBR\_DAILY\_COST\_TOTALS
  + Same as VW\_LBR\_COST\_TOTALS but daily grain. Improvement: should be the base view that VW\_LBR\_COST\_TOTALS pulls from.
* VW\_LBR\_EMPLIVE\_HRSBREAKDOWN\_NON\_NULL
  + Same as VW\_LBR\_EMPLIVE\_HRSBREAKDOWN but coalesces all the columns so they are not null. This is necessary for transformations that don’t work well with nulls.
* VW\_LBR\_VS\_COST\_CURR\_WEEK\_KPI
  + [Phillip]
* VW\_LSS\_DCM\_KPI
  + Same as VW\_DCM\_KPI, filtered for LSS
* VW\_MG\_DCM\_KPI
  + Same as VW\_DCM\_KPI, filtered for LSS
* VW\_MG\_STOCK\_CHECK\_PREVWEEK
  + A report that gets sent to Sanjhu to help her with a manual billing entry regarding the quantity of pieces counted for MG Motors in the previous week.
* VW\_MUWH\_DCM\_KPI
  + Same as VW\_DCM\_KPI, filtered for MUWH (multi-user warehouse)
* VW\_NZ\_MUWH\_DATA\_CHECKS
  + This view checks for data errors and is used by ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’ so the admin is aware of any data errors that need to be addressed. Examples: data not populated on a specific day, important column missing data, auxiliary table missing data, etc.
* VW\_OB\_ACCENT\_ADHOC\_LBRHRS
  + Calculates extra ad hoc labour hours for Accent based on pick quantity. If pick quantity for a day exceeds a threshold, take the remainder and divide by a constant calculated by Aman Sahai.
* VW\_OB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED
  + Related to VW\_OB\_ACCENT\_ADHOC\_LBRHRS, but contains order grain.
* VW\_OB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED\_PREVWEEK
  + Same as VW\_OB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED, but filtered for previous week.
* VW\_OB\_ACCENT\_ADHOC\_LBRHRS\_PREVWEEK
  + Same as VW\_OB\_ACCENT\_ADHOC\_LBRHRS\_DETAILED, but filtered for previous week.
* VW\_OB\_ACCENT\_PICKRUN\_RPT
  + Shows closed and in progress quantities for Accent pickruns. See glossary for pickrun definition.
* VW\_OB\_KOGAN\_BACKORDERS\_DESPATCHREF
  + Kogan often wants to know the status of orders that have been placed, but have not been shipped. They send us a report containing the orders and SKUs in questions (stored in OB\_ACCENT\_PICKRUN). From this report, we compare against FMS and Matrix WMS data to give them the shipped date and despatch reference of the orders in question.
* VW\_OB\_MATRIX\_SHIPPING\_ACTIVITY\_BY\_DATE\_RANGE\_MAX\_UPD\_TS
  + Same as table OB\_MATRIX\_SHIPPING\_ACTIVITY\_BY\_DATE\_RANGE, but filtered for the most recently updated rows.
* VW\_OPS\_DIFOT\_PERFORMANCE
  + Calculates DIFOT (delivery in full on time) SLA for Matrix WMS customers. Will be displayed on a Qlik Sense BI dashboard.
* VW\_OPS\_IB\_PERFORMANCE
  + Calculates DIFOT (delivery in full on time) SLA for Matrix WMS customers. Will be displayed on a BI dashboard. This is the time from order manifest date (the date the carrier was aware of the order) to the day the order was delivered.
* VW\_OPS\_OB\_PERFORMANCE
  + Calculates OB SLA for Matrix Customers. This is the time from order allocation to order shipment.
* VW\_RATE\_ACTIVE\_CUST
  + Filters DIM\_RATE for active DIM\_CLIENT\_MAPPING\_CK’s
* VW\_RATES\_W\_PERIODS
  + Assigns DIM\_RATE\_CK’s to possible billing years and billing periods.
* VW\_SG\_MATRIX\_INVENTORY\_DETAILS\_MAX\_UPD\_TS
  + Same as SG\_MATRIX\_INVENTORY\_DETAILS table, but filtered for rows containing the most recent update timestamp.
* VW\_SG\_MATRIX\_NZ\_GODDARD\_PALLET
  + Calculates Goddard customer pallet storage charges using SKU dimensions.
* VW\_SG\_MATRIX\_NZ\_HKW
  + Calculates storage charges for HKW customer. Charges are for pallet locations and pick face locations. Pick face locations are low to the ground and typically have loose units that are easy to grab for picking. A location is determined to be a pick face if it has more than 1 distinct SKU.
* VW\_SG\_MATRIX\_NZ\_KOGAN\_PALLET
  + Similar to VW\_IB\_MATRIX\_NZ\_KOGAN\_PALLET. Kogan pallet billing has a special clause that states we must charge for the number of theoretical pallets, not actual. This means the total units for a given SKU is divided by the units per pallet to get the theoretical number of pallets if we properly combined all the units. For example, if you had 2 pallets that were half full of the same SKU, you would charge for one pallet. This view calculates storage pallets for Kogan billing.
* VW\_SG\_MATRIX\_NZ\_MUWH\_BASE
  + View that serves as a base for storage calculations. CL charges customers for storage occupied weekly. However, the storage report is a point-in-time report. Therefore, inventory could be inbounded after the moment storage report was run. To capture and charge all storage, you must take the storage report at the beginning of the week and add all inventory inbounded during the week. The storage charge calculation therefore requires storage data and inbound data which is consolidated in this view.
* VW\_SG\_MATRIX\_OPENING\_TS
  + Since storage is point-in-time and comes daily, it’s necessary to use the data that corresponds to the first storage report of the billing week. This view gets the first storage report run timestamp of each billing week for each customer on Matrix WMS.
* VW\_SG\_WIMS\_OPENING\_TS
  + Since storage is point-in-time and comes daily, it’s necessary to use the data that corresponds to the first storage report of the billing week. This view gets the first storage report run timestamp of each billing week for each customer on WIMS.
* VW\_TABLE\_TS
  + Returns each report table and the most recent update timestamp. Helpful to determine the last time each table loaded new data.
* VW\_TXN\_MATRIX\_CONSOL\_MAX\_UPD\_TS
  + Same as TXN\_MATRIX\_CONSOL but filtered for the rows with the most recent update timestamp.
* VW\_TXN\_MATRIX\_CONSOL\_OB\_DETAIL\_MAX\_UPD\_TS
  + Same as TXN\_MATRIX\_CONSOL\_OB\_DETAIL but filtered for the rows with the most recent update timestamp.
* VW\_TXN\_PACK\_CONFIG
  + Gets a pack config list using transaction data. Since everything is done in a view in one step, it’s slow. An emergency alternative to DIM\_PACK\_CONFIG.
* VW\_WEEK\_MIN\_MAX\_DATE
  + Returns the max and min date for each billing year, period.

#### Excel Views

* VW\_EXCEL\_COCO\_REP\_SO
  + Not used. Was intended to help with the interim Coco Republic WMS until they were integrated with Matrix WMS.
* VW\_EXCEL\_COLLATE
  + Gets customer charges.
* VW\_EXCEL\_COLLATE\_TPT
  + Gets customer charges exclusively for transport.
* VW\_EXCEL\_CURRENT\_BILLING\_PERIOD
  + Gets the current billing year and billing period.
* VW\_EXCEL\_FMS\_VS\_MATRIX\_ORDERS
  + Compares Matrix WMS shipments to FMS despatches. The idea is that once an order is marked as shipped in WMS, it should be despatched by the carrier quickly. If not, there could be a problem. This view helps quantify the difference between shipped and despatched orders.
* VW\_EXCEL\_FMS\_VS\_MATRIX\_ORDERS\_SUMMARY
  + Same purpose as VW\_EXCEL\_FMS\_VS\_MATRIX\_ORDERS, but summarises the data.
* VW\_EXCEL\_FREIGHT\_SURCHARGE
  + Returns active freight surcharges. Used by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Freight Surcharge Manager.xlsm’
* VW\_EXCEL\_FREIGHT\_SURCHARGE\_CARRIER
  + Returns a list of carriers that can have freight surcharges. Used to populate a dropdown in ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Freight Surcharge Manager.xlsm’
* VW\_EXCEL\_FREIGHT\_SURCHARGE\_CHARGE\_DESC
  + Returns a list of possible freight surcharge descriptions. Used to populate a dropdown in ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Freight Surcharge Manager.xlsm’
* VW\_EXCEL\_IB\_COCO
  + Not used. Was intended to help with the interim Coco Republic WMS until they were integrated with Matrix WMS.
* VW\_EXCEL\_IB\_DUMP
  + Dump of Matrix WMS inbound data
* VW\_EXCEL\_NZ\_MUWH\_FR\_FMS\_DUMP
  + Used to populate supporting freight data in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_NZ\_MUWH\_IB\_DUMP
  + Used to populate supporting inbound data in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_NZ\_MUWH\_OB\_MATRIX\_PICK\_DUMP
  + Used to populate supporting outbound pick data in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_NZ\_MUWH\_OB\_TXN\_DUMP
  + Used to populate supporting outbound transaction-based data in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_NZ\_MUWH\_SG\_MATRIX\_DUMP
  + Used to populate supporting storage data in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_OB\_DUMP
  + Dump of Matrix WMS outbound data
* VW\_EXCEL\_OB\_SHIPPING\_DUMP
  + Dump of Matrix WMS outbound data from shipping activity reports.
* VW\_EXCEL\_RATE
  + Returns rates so the billing admin can see a complete list in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_BILLING\_PERIOD
  + Returns a list of possible billing periods. Used in a dropdown in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_BILLING\_YEAR
  + Returns a list of possible billing years. Used in a dropdown in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_CALENDAR
  + Returns the billing years and periods for each customer as reference in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_CHARGES\_SUMMARY
  + Returns charges displayed in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_CLIENT\_GROUP
  + Returns distinct list of clients for use as dropdown list in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’
* VW\_EXCEL\_COCO\_REP\_OB\_ITEMS
  + Not used. Was intended to help with the interim Coco Republic WMS until they were integrated with Matrix WMS.

#### Supporting Data Views

* VW\_EXPORT\_GDT\_AKL\_PATS\_ANNOYING\_REQUEST
  + You an tell by the name how Drew feels about this request. GDT requested that the supporting charges data be sent to them in a specific format that is similar to the way they received it when it was calculated in spreadsheets.
* VW\_EXPORT\_GET\_CURRENT\_PD\_CUST
  + Returns customer, billing year, billing period for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_HISTORICAL\_CHARGES
  + Returns customer, billing year, billing period for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_IB\_SAP\_AKL
  + Returns IB GDT supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_IB\_SHIPPING\_MATRIX
  + Returns IB Shipping Activity by date range supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_IB\_TXN\_WIMS
  + Returns WIMS IB transaction supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_OB\_DETAILED\_MATRIX
  + Returns Matrix OB detailed order receipt supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_OB\_PICK\_WIMS
  + Returns WIMS OB picking supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_OB\_SAP\_AKL
  + Returns GDT OB supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_PICKING
  + Returns Matrix OB Picking Activity supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_RATE
  + Returns rate supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_SG
  + Returns rate supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_SG\_STOCK\_WIMS
  + Returns WIMS stock summary supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_SG\_WAREHOUSE\_OCC\_MATRIX
  + Returns Matrix WMS storage supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_TPT\_KPI\_POD
  + Returns supporting data which is eventually sent in the body of an email for FM. (Incorrectly naming convention of TPT). Used by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\transportKPIObj.py’
* VW\_EXPORT\_CHARGES\_SUMMARY\_TOTALS
  + Returns charges summary totals supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_CONSUMABLES
  + Returns consumables supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_FR\_CONSOL
  + Returns FMS transport supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_FT\_CHARGES\_SUMMARY
  + Returns charges supporting data for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_FT\_CHARGES\_SUMMARY\_APPROVAL
  + Returns charges supporting data for the approval workbook for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’.
* VW\_EXPORT\_FT\_CHARGES\_SUMMARY\_CAT
  + Returns charge categories for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_FT\_CHARGES\_SUMMARY\_CAT\_ACCENT
  + Returns Accent customer charge categories for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’
* VW\_EXPORT\_FT\_CHARGES\_SUMMARY\_CAT\_ACER
  + Returns Acer charge categories for use by ‘\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’

### PostgreSQL Functions

**Import Functions SP\_IMPORT\_<report>:** See [import\_functions](#import_functions) section.

**Load Functions SP\_LOAD\_<report>:** See [load\_functions](#load_functions) section.

**Consumables Functions:**

* **DB\_LOAD.SP\_DELETE\_CONSUMABLES** (IN\_SRNO INTEGER) RETURNS VOID
  + Deletes a row from the table DB\_LOAD.CS\_CONSUMABLES\_MASTER that matches the serial number argument. Used by “\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Consumables Master.xlsm"
* **DB\_LOAD.SP\_INSERT\_CONSUMABLES** (IN\_BILLING\_YEAR INTEGER, IN\_BILLING\_PERIOD INTEGER, IN\_CUSTOMER\_NAME VARCHAR, IN\_VENDOR VARCHAR, IN\_WEB3\_NO VARCHAR, IN\_INVOICE\_NO VARCHAR, IN\_INVOICE\_AMT\_EX\_GST DECIMAL(18, 8), IN\_INVOICE\_DATE VARCHAR, IN\_SPLITS\_APPLICABLE VARCHAR, IN\_SPLIT\_RATE DECIMAL(18, 8), IN\_WNDWS\_UPDATE\_USR VARCHAR) RETURNS VOID
  + Inserts rows into DB\_LOAD.CS\_CONSUMABLES\_MASTER. This table is later used for billing. Used by “\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Consumables Master.xlsm"

**SLA Functions:**

* **DB\_LOAD.SP\_DELETE\_DIFOT\_EXCEPTION** (IN\_CLIENT\_GROUP VARCHAR, IN\_ORDER\_ID VARCHAR, IN\_DELIVERED\_DATE DATE) RETURNS VOID
  + Deletes from DB\_LOAD.SP\_DELETE\_DIFOT\_EXCEPTION given a match on the arguments.
* **DB\_LOAD.SP\_DELETE\_IB\_EXCEPTION** (IN\_CLIENT\_GROUP VARCHAR, IN\_PRE\_ADVICE\_ID VARCHAR, IN\_PUTAWAY\_DATE DATE) RETURNS VOID
  + Deletes from DB\_LOAD.DIM\_IB\_SLA\_COMMENTARY given a match on arguments.
* **DB\_LOAD.SP\_DELETE\_OB\_EXCEPTION** (IN\_CLIENT\_GROUP VARCHAR, IN\_REFERENCE VARCHAR, IN\_SHIP\_DATE DATE) RETURNS VOID
  + Deletes from DB\_LOAD.DIM\_OB\_SLA\_COMMENTARY given a match on arguments.
* **DB\_LOAD.SP\_INSERT\_DIFOT\_EXCEPTION** (IN\_CLIENT\_GROUP VARCHAR, IN\_ORDER\_ID VARCHAR, IN\_DELIVERED\_DATE DATE, IN\_REGION VARCHAR, IN\_REGION2 VARCHAR, IN\_IS\_EXCEPTION INT, IN\_COMMENTS VARCHAR, IN\_WNDWS\_INSERT\_USR VARCHAR, IN\_WNDWS\_UPDATE\_USR VARCHAR) RETURNS VOID
  + Inserts into DB\_LOAD.DIM\_DIFOT\_SLA\_COMMENTARY using arguments.
* **DB\_LOAD.SP\_INSERT\_IB\_EXCEPTION** (IN\_CLIENT\_GROUP VARCHAR, IN\_PRE\_ADVICE\_ID VARCHAR, IN\_PUTAWAY\_DATE DATE, IN\_IS\_EXCEPTION INT, IN\_COMMENTS VARCHAR, IN\_WNDWS\_INSERT\_USR VARCHAR, IN\_WNDWS\_UPDATE\_USR VARCHAR) RETURNS VOID
  + Inserts into DB\_LOAD.DIM\_IB\_SLA\_COMMENTARY using arguments.
* **DB\_LOAD.SP\_INSERT\_OB\_EXCEPTION** (IN\_CLIENT\_GROUP VARCHAR, IN\_REFERENCE VARCHAR, IN\_SHIP\_DATE DATE, IN\_IS\_EXCEPTION INT, IN\_COMMENTS VARCHAR, IN\_WNDWS\_INSERT\_USR VARCHAR, IN\_WNDWS\_UPDATE\_USR VARCHAR) RETURNS VOID
  + Inserts into DB\_LOAD.DIM\_OB\_SLA\_COMMENTARY using arguments.

**Charge Functions:** All charge functions insert charges into DB\_ACCESS.FT\_CHARGES\_SUMMARY. Each function does inserts for charges derived from a specific report. Sometimes functions depend on other functions. See [function\_dependencies](#function_dependencies) section.

* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_GDT\_IB\_SAP\_AKL** () RETURNS VOID
  + For charges derived from SAP report ‘IB Chch’.
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_GDT\_OB\_SAP\_AKL** () RETURNS VOID
  + For charges derived from SAP report ‘OB Chch’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_CONSUM** () RETURNS VOID
  + For charges derived from Excel tool ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Consumables Master.xlsm’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_FMS\_CONSOLAKL** () RETURNS VOID
* For charges derived from FMS report ‘Consolakl’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_IB\_MATRIX\_RECEIPT** () RETURNS VOID
  + For charges derived from Matrix WMS report ‘Receipt Activity by Date Range’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_IB\_YMT** () RETURNS VOID
  + For charges derived from Excel tool ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Yard Management Tool V5 - Shortcut.xlsm’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_MISC** () RETURNS VOID
  + For charges that can be auto-calculated but are not derived from a report EX: fixed charges. Also inserts dummy rows for charges (rows that have zero units/charge) so that Sanjhu can see them in the tool and update if necessary.
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_OB\_MATRIX\_PICK** () RETURNS VOID
  + For charges derived from Matrix WMS report ‘Picking Activity by Date Range’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_OB\_MATRIX\_SHIP** () RETURNS VOID
  + For charges derived from Matrix WMS report ‘Shipping Activity by Date Range’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_OB\_WIMS\_PICK** () RETURNS VOID
  + For charges derived from WIMS report ‘so\_pick\_rpt’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_SG\_MATRIX** () RETURNS VOID
  + For charges derived from Matrix WMS report ‘Warehouse Occupancy’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_SG\_MATRIX\_INV\_DET** () RETURNS VOID
  + For charges derived from Matrix WMS report ‘INVENTORY\_DETAILS\_by\_date\_range’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_SG\_WIMS\_STOCK\_SUMM** () RETURNS VOID
* For charges derived from WIMS report ‘Stock Summary’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_TXN\_MATRIX** () RETURNS VOID
* For charges derived from Matrix WMS report ‘Consolidated Transactions by Date Range’
* **DB\_ACCESS.SP\_CHARGES\_SUMMARY\_NZ\_MUWH\_TXN\_WIMS** () RETURNS VOID
  + For charges derived from WIMS report ‘transaction\_extract\_rpt’
* **DB\_ACCESS.SP\_UPSERT\_CHARGES\_SUMMARY** () RETURNS VOID
  + Inserts/updates charges that cannot be calculated from reports. Used by ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’

**Misc Functions:**

* **DB\_LOAD.SP\_INSERT\_FREIGHT\_SURCH** (IN\_CARRIER TEXT, IN\_CHARGE\_DESCRIPTION TEXT, IN\_CHARGE\_RATE DECIMAL(18, 8), IN\_NOTES TEXT, IN\_WNDWS\_UPDATE\_USR VARCHAR) RETURNS VOID
  + Inserts into DB\_LOAD.DIM\_FREIGHT\_SURCHARGE using arguments. Used in “\\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools\Freight Surcharge Manager.xlsm"
* **DB\_LOAD.SP\_PACK\_CONFIG** () RETURNS VOID
  + Inserts into DB\_LOAD.DIM\_PACK\_CONFIG whenever the Matrix Consolidated Transactions report arrives. The transaction report contains pack configurations
* **DB\_ACCESS.SP\_UPDATE\_BILLING\_FREQUENCY** (IN\_DD\_DATE DATE, IN\_BILLING\_YEAR INT, IN\_BILLING\_PERIOD INT, IN\_BU\_DESC VARCHAR, IN\_WNDWS\_UPDATE\_USR VARCHAR) RETURNS VOID
  + Deprecated. Was used by ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’ so the admin could update which days fell into which billing period.
* **DB\_LOAD.SP\_UPSERT\_YARD\_MANAGEMENT\_TOOL**
* **DB\_ACCESS.SP\_UPSERT\_IB\_COCO\_REP\_WMS**
  + Deprecated. Was used when we needed an interim WMS for Coco Republic customer.
* **DB\_ACCESS.SP\_UPSERT\_OB\_COCO\_REP\_WMS**
  + Deprecated. Was used when we needed an interim WMS for Coco Republic customer.
* **DB\_TOOLS. DEPS\_SAVE\_AND\_DROP\_DEPENDENCIES** (IN\_VIEW\_SCHEMA VARCHAR, IN\_VIEW\_NAME VARCHAR) RETURNS VOID
  + PostgreSQL does not allow you to drop an object if there are other objects dependent on it. It’s quite painful to update a table/view because you have to drop all the dependent objects before you can make the change. This procedure allows you to pass the schema and the object you wish to update/drop/replace, and it will automatically drop and save the DDL of dependent objects to the table DB\_TOOLS.DEPS\_SAVED\_DDL.
* **DB\_TOOLS. DEPS\_RESTORE\_DEPENDENCIES** (IN\_VIEW\_SCHEMA VARCHAR, IN\_VIEW\_NAME VARCHAR) RETURNS VOID
  + Related to DEPS\_SAVE\_AND\_DROP\_DEPENDENCIES. Responsible for restoring objects dependent on the one you’re replacing.
* **DB\_LOAD.SP\_DUMMY\_PLACEHOLDER** () RETURNS VOID
  + Procedure that does nothing. Used in “\Configs\importConfig.json” when you want to import the data but don’t want to calculate anything.

### PostgreSQL Users

* **postgres**
  + Password: popplenose
  + Description: All-powerful user for code promotion.
* **load\_user**
  + Password: caffeine
  + Description: Second-most powerful user. Can be used for loading data, calling functions, insert/deletes across the database.
* **access\_user**
  + Password: coffee
  + Description: User for presentation of data. Has insert, update, delete access to DB\_ACCESS schema.
* **read\_only\_user**
  + Password: r34d\_0n1y
  + Description: Has read-only access across the database.

### Related Columns

Generally, if 2 columns have the same name, they are joinable. The section below is meant to highlight columns that have different names but are joinable.

* **Matrix WMS Order Number**
  + DB\_LOAD.TXN\_MATRIX\_CONSOL.REFERENCE
    - Can be an order placed by a customer for outbound or pre advice id depending on the transaction.
  + DB\_LOAD.IB\_MATRIX\_RECEIPT\_ACTIVITY\_BY\_DATE\_RANGE.PRE\_ADVICE\_ID
    - See [glossary](#glossary) for definition. Can be joined to TXN\_MATRIX\_CONSOL.REFERENCE if transaction is related to inbounding process.
  + DB\_LOAD.OB\_MATRIX\_PICKING\_ACTIVITY.ORDER\_ID
  + DB\_LOAD.OB\_MATRIX\_SHIPPING\_ACTIVITY\_BY\_DATE\_RANGE.WMS\_REFERENCE
  + DB\_LOAD.OB\_MATRIX\_DETAILED\_ORDER\_RECEIPT.ORDER\_ID
  + DB\_LOAD.OB\_KOGAN\_CEVA\_BACKORDERS.ORDER\_REF
    - Must take the part of the string that comes before the ‘-‘ in Matrix WMS order columns.
    - EX: OB\_KOGAN\_CEVA\_BACKORDERS.ORDER\_REF = SPLIT\_PART(OB\_MATRIX\_DETAILED\_ORDER\_RECEIPT.ORDER\_ID, '-', 1)
  + DB\_LOAD.FR\_FMS\_CNM\_INFO.CNMITEMSHIPLP
  + DB\_LOAD.FR\_FMS\_CONSOLAKL.PSLIP\_NUMBER
* **Matrix WMS CLIENT\_ID**
  + DB\_LOAD.DIM\_CLIENT\_MAPPING.SOURCE\_SYSTEM\_CLIENT
    - When the ‘SOURCE\_SYSTEM’ column is ‘MATRIX’
  + Any other column named ‘CLIENT\_ID’
* **Matrix WMS SITE\_ID**
  + DB\_LOAD.DIM\_CLIENT\_MAPPING.SOURCE\_SYSTEM\_SITE\_ID
    - When the ‘SOURCE\_SYSTEM’ column is ‘MATRIX’
  + Any other column named ‘SITE\_ID’
* **Matrix WMS Pack Confg**
  + DB\_LOAD.DIM\_KOGAN\_PACK\_CONFIG\_TEMP.DIM\_PACK\_CONFIG\_CK
  + DB\_LOAD.DIM\_KOGAN\_PACK\_CONFIG\_TEMP.PACK\_CONFIG
  + DB\_LOAD.DIM\_PACK\_CONFIG.PACK\_CONFIG\_ID
  + DB\_LOAD.IB\_MATRIX\_RECEIPT\_ACTIVITY\_BY\_DATE\_RANGE.PACK\_CONFIG
  + DB\_LOAD.SG\_MATRIX\_WAREHOUSE\_OCCUPANCY.PACK\_CONFIG
  + DB\_LOAD.TXN\_MATRIX\_CONSOL.PACK\_CONFIG\_ID
  + DB\_LOAD.TXN\_MATRIX\_CONSOL\_OB\_DETAIL.PACK\_CONFIG\_ID

### Metadata columns

Every table in the database should have the following metadata columns:

* DB\_INSERT\_USR – The database user that inserted the row
* WNDWS\_INSERT\_USR – The Windows/CEVA user that inserted the row
* INSERT\_TS – The timestamp the row was inserted
* DB\_UPDATE\_USR – The database user that updated the row
* WNDWS\_UPDATE\_USR - The Windows/CEVA user that updated the row
* UPDATE\_TS – The timestamp the row was updated

## Output/User Interaction

Pre-requisite: In order to refresh PostgreSQL data in Excel, users need a driver installed. The driver that needs to be installed is dependent on which bit-version of Excel the user has. They can check this by opening Excel and navigating to File -> Account -> About Excel. Then the matching driver should be installed by contacting Ian and asking him to install the specific driver.

32 bit: \\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\\_Setup - Get Started\Install Files\psqlodbc\_x86.msi

64 bit: \\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\\_Setup - Get Started\Install Files\psqlodbc\_x64.msi

Graphical user interface, application

Description automatically generated

### Billing Tool

DWBT calculates charges from report data, but some charges are not derived from this data. These charges may come from word of mouth, and email, or a one-off spreadsheet. Additionally, some of the charges calculated by DWBT may need to be updated in special circumstances. To add/update charges manually, an Excel tool was created: ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Billing Tool - NZ v1.03.xlsm’. This tool uses VBA and forms to allow the billing admin to update/remove/create charges in the database.

This workbook is password protected with: ‘thereisnotry’ (which is Patrick’s ringtone).

**Tabs**

* Version Hist
  + Contains a log of changes throughout different versions of the workbook.
* Process
  + The control centre. Here the admin selects a single billing year and period and clicks ‘Refresh Data’ button to view data for that billing year and period. Once all the charges have been updated, the admin clicks ‘Create Output Files’ button to create the supporting data workbooks. Supporting data is stored in ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Customer Reports’
* Data Checks
  + Provides a list of data errors/warning related to the underlying data for the specified year/period. EX: a column that is used to calculate a charge is blank.
* Calendar
  + Shows the billing calendar by period for each customer. At the time of writing all customers have the same billing schedule and therefore the same calendar, but it’s possible that a customer could have a different billing schedule.
* Charges Summary
  + Lists all charges for the specified billing period. The ‘Manage’ button at the top allows the admin to update a specific charge. Charges are listed even if there aren’t any units so that the admin can see all potential charges.
* Storage (Prev Week)
  + Holds the underlying storage data that was used to calculate the charges on the ‘Charges Summary’ tab.
* Freight
  + Holds the underlying storage data that was used to calculate the charges on the ‘Charges Summary’ tab.
* Inbound
  + Holds the underlying inbound data that was used to calculate the charges on the ‘Charges Summary’ tab.
* Outbound - Detailed Receipt
  + Holds the underlying outbound data that was used to calculate the charges on the ‘Charges Summary’ tab.
* Outbound – Picking
  + Holds the underlying outbound picking data that was used to calculate the charges on the ‘Charges Summary’ tab.
* Rate Codes Reference
  + Holds the rates used in ‘Charges Summary’ tab for reference.
* ComboBoxes
  + Contains some queries that return values used elsewhere in the workbook.

### Excel Tools

This section refers to the Excel Workbooks contained in the ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Excel Tools’ folder. These workbooks provide a way for users to interact with data contained in DWBT.

* 1.Charges History.xlsm
  + Shows all historical charges. Shows data in DB\_ACCESS.VW\_EXCEL\_COLLATE.
* Accent SCA Dashboard Calculator.xlsm
  + Used by BPE. Someone copies the rows in this workbook and pastes them into another workbook for Accent metrics. If this workbook didn’t exist, someone would have to manually pull the data and calculate the metrics. Shows data in DB\_ACCESS.VW\_ACCENT\_SCA\_DASHBOARD.
* Charges History - Transport.xlsm
  + Same as ‘1.Charges History.xlsm’ but for transport charges. Data from DB\_ACCESS.VW\_EXCEL\_COLLATE\_TPT
* Charges History.xlsm
  + Same as ‘1.Charges History.xlsm’. I think someone made a copy for their own pivot table.
* Consumables Master.xlsm
  + Contains data related to consumables purchases. Hooked up to the database so that entries in this workbook are saved to the database (DB\_LOAD.CS\_CONSUMABLES\_MASTER). Consumables are on-charged to many customers.
* Freight Surcharge Manager.xlsm
  + A tool to enable the transport supervisor to create/update freight surcharges in DB\_LOAD.DIM\_FREIGHT\_SURCHARGE.
* KPI Tracker Tool.xlsm
  + Shows KPI data for many of our customers. Data from this workbook is copied into a workbook that is overseen by AUS/Global.
* Matrix and FMS Order Checks.xlsm
  + Data from DB\_ACCESS.VW\_EXCEL\_FMS\_VS\_MATRIX\_ORDERS and DB\_ACCESS.VW\_EXCEL\_FMS\_VS\_MATRIX\_ORDERS\_SUMMARY. Used as a sanity check for Matrix WMS activity against FMS activity. Matrix tracks in warehouse, FMS tracks as stock is transported. Ideally the activity in Matrix should match FMS because everything is transported somewhere during outbound.
* NZ GR Allocation.xlsx
  + Non-DWBT connected workbook. I believe it’s used by Amanda Bristow for accounting.
* Ops SLA Exceptions.xlsm
  + Used by warehouse floor manager to enter comments on Outbound orders that did not meet the SLA due to extenuating circumstances. Password: ceva123
* P2P Labour tracker.xlsx
  + Non-DWBT connected workbook. Created by Aman.
* Viewer - Accent Acticity.xlsm
  + Allows users to view IB transaction and picking data for Accent from WIMS. Users can enter filter parameters. Data from DB\_ACCESS.TXN\_WIMS\_TRANSACTION\_EXTRACT and DB\_ACCESS.OB\_WIMS\_SO\_PICK
* Viewer - Emplive.xlsm
  + Allows users to view labour data from EmpLive. Users can enter filter parameters. Data from DB\_ACCESS.LBR\_EMPLIVE\_HRSBREAKDOWN. Password: emplive123!
* Viewer - Rate.xlsm
  + Allows users to view labour data from EmpLive. Data from DB\_ACCESS.VW\_EXCEL\_RATE.
* Viewer - Storage - Accent.xlsm
  + Allows users to view storage data from WIMS. Users can enter filter parameters. Data from DB\_ACCESS.SG\_WIMS\_STOCK\_SUMMARY.
* Viewer - Storage.xlsm
  + Allows users to view storage data from Matrix WMS. Users can enter filter parameters. Data from DB\_ACCESS.SG\_MATRIX\_WAREHOUSE\_OCCUPANCY.
* Viewer - Transaction.xlsm
  + Allows users to view transaction data from Matrix WMS. Users can enter filter parameters. Data from DB\_ACCESS.TXN\_MATRIX\_CONSOL.
* Viewer - Transport.xlsm
  + Allows users to view transport orders from FMS. Users can enter filter parameters. Data from DB\_ACCESS.FR\_FMS\_CONSOLAKL\_W\_SURCH.
* Weekly Revenue tracker.xlsx
  + Non-DWBT connected workbook. Created by Aman.
* Yard Management Tool V5 - Shortcut.lnk
  + Shortcut to the Yard Management Tool. This is the tool used to manage containers that arrive with stock that needs to be inbounded. The tool inserts new data to DWBT DB\_LOAD.YARD\_MANAGEMENT\_TOOL.

### Freight Management Functionality

Freight Management (FM) had a use case that could be well-handled by DWBT. They wanted to provide a system-generated report as input and receive an email with formatted text in the body that is derived from the report.

The importConfig.json entry has the reportNameRegex: “HouseBill\_Air\_Destination”. An FM admin will pull the report and drop it on the server’s FTP site using WinSCP. It will be processed and loaded into DB\_LOAD.TPT\_KPI\_POD. Then, at the end of “\NZ Data Warehouse Working Directory\Supporting Code\Python\Import.py”, there is an instance of the class ‘transportKPIObj’ that handles the HTML generation. This html is mailed to a list of people.

# Tools Not Related to DWBT

Drew Hentz developed some tools that utilise VBA/python to help automate tasks for people around the office. See below:

## ETAFormatConverter

This project is to help Transport (Troy Wills <Troy.Wills@Cevalogistics.com>). He downloads 2 csv files from a system and needs to do some basic data transformations, then upload the transformed data as a csv to another system.

**Dev:** [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Sharing\FM\ETA Format Converter Tool](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/Sharing/FM/ETA%20Format%20Converter%20Tool).

**Production:** [\\AKL2WSFPS001\share$\AKL CUSTOMS\LYT Vessel Update](file://AKL2WSFPS001/share$/AKL%20CUSTOMS/LYT%20Vessel%20Update).

**ETAFormatConverter.py:** Python file that uses ‘LYT.csv’ and ‘Lloyds.csv’ as input, and outputs transformed data in ‘LYT Output.csv’

**ETAFormatConverter.exe:** A compiled version of the ETAFormatConverter.py Python file. This was created so that the user doesn’t need to have Python installed. To create executables, use pip to install the package auto-py-to-exe (pip install auto-py-to-exe). An executable should appear in your Python scripts directory (ex: Python\Python37\Scripts\autopytoexe.exe). Run this executable to compile you Python script to an executable.

**Lloyds.csv:** Input file generated by an FM system.

**LYT Output.csv:** Output file that is uploaded to an FM system.

**LYT.csv:** Input file generated by an FM system.

**README.txt:** Contains expected columns for each input csv file.

## ETAFormatConverter

This project is to help Transport (Troy Wills <Troy.Wills@Cevalogistics.com>). They receive an Excel workbook via email and need to format it and save it as a csv so it can be uploaded into another system. There is an Excel workbook that uses VBA to download the file from their email, import the data into a worksheet, format the data, and save the formatted data as a csv.

**Dev:** [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Sharing\FM\Daiken Invoices Tool](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/Sharing/FM/Daiken%20Invoices%20Tool)

**Production:** [\\AKL2WSFPS001\share$\AKL CUSTOMS\Daikin Invoice Tool](file://AKL2WSFPS001/share$/AKL%20CUSTOMS/Daikin%20Invoice%20Tool)

**Daikin Invoice Tool.xlsm:** This workbook contains buttons that allows the user to execute the tasks below:

* Download From Email Button
  + The 'Outlook Folder' needs to be a sub-folder of your inbox
  + The most recent .xlsx file found in the Outlook Folder will be saved to the 'Download Folder'
  + The 'Download Folder' and 'Output Folder' should not end with '\'
* Get Source Data Button
  + Copies data from 'Sheet1' in the 'File Found' from the 'Download Folder' to 'Original Data' in this workbook
* Format Data Button
  + Formats the data in 'Formatted Data' sheet
* Export CSV Button
  + Exports data in 'Formatted Data' sheet to the 'Output Folder' with the file name in 'File Found'

**\Input:** Folder where the Excel file from Outlook is saved to

**\Output:** Folder where the output csv is saved

# Database Risk Management

This section describes processes in place to mitigate risk related to the database.

## Checking for Source Reports

The data in DWBT is sourced in multiple ways (see [data\_sources](#data_sources)). It’s important that the data we expect arrives every day. To mitigate the risk of a report not arriving on a given day, there is a Python script that checks if reports have been received every day: \Supporting Code\Python\checkForMissingReports.py. It checks the directory “D:\Source Data\Prod\<current date>” for all the reports listed in “\Configs\importConfig.json”. If there are reports in the json file that are not in the source data folder, it emails the people listed in “\Configs\importConfig.json” key: reportCheckerEmailErrors.

This Python file is initiated by Windows Task Scheduler on the server. Job: Check For Missing Reports. This happens every day at 11 AM.

Note: We don’t receive every report every day, so don’t be alarmed if there’s an email with a lot of reports. The important reports are the Matrix WMS reports which at the time of writing are generated by RPA. If there are RPA reports that aren’t received, you need to email Steve Wong <[Steve.Wong@cevalogistics.com](mailto:Steve.Wong@cevalogistics.com)> and ask him to rerun the RPA jobs. The exception is if there wasn’t any activity (if there’s no activity, no report will be generated for that activity). There should be a storage report every day because regardless of activity, storage is being occupied in the warehouse. EX: if there is a storage report but no inbound report, it’s likely okay because RPA ran for storage but didn’t have inbound data.

## Database Backups

Database backups are taken daily at midnight on the server. Windows Task Scheduler kicks off the job ‘PostgreSQL PROD Backup’ which runs the batch script “D:\Backups\PostgreSQL\Batch Files\pgProdBackup.bat” This batch script dumps the database to a single file which is 7-zipped to reduce storage (D:\Backups\PostgreSQL\Backup Files).

## Database Syncs

Prior to syncing, you will need to place the postgresql folder contained here: [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\\_Setup - Get Started\In Order to Sync Database Environments](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/_Setup%20-%20Get%20Started/In%20Order%20to%20Sync%20Database%20Environments) in your C:\Users\<user> \AppData\Roaming folder on the server. This allows for programmatic login to drop and create objects in the lower environments.

In order to sync an environment with the most recent prod backup, run the following command:

"D:\Backups\PostgreSQL\Batch Files\pgEnvironSync.bat" <dev, test, or prod>

# DevOps

This section describes the development lifecycle and development environment of DWBT.

## Managed Materials

There are 2 sets of materials that need to be managed and have development areas:

* Front End
  + This is the set of Excel Workbooks and Python used to support front end/user interaction with DWBT. The front end lives her: “\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool”
* Back End
  + The code living on the server 10.76.156.24.

## Back End

The back end is a Windows server, IP: 10.76.156.24 HostName: NZMAE01VWAPP002. You need to request a separate user to access it (your normal username with ‘-adm’ appended). You can create a GetHelp ticket for this request. Use RemoteDesktop with the IP address and your credentials to log in.

### Git Ecosystem

Git is used to manage version control. The production code runs on the server (10.76.156.24). The production code is pulled from the master repository (also on the server) every time a push is made. Each developer has their own working instance of the repo in their documents folder on the server. The reason for having the development/working directories on the server is that it allows the developers to test their code. It would be difficult to run the Python ETL on their local machines for testing. Also, the use of Git hooks enables the push command to deploy the code instead of manually deploying. Diagram can be found here: [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\\_Documentation](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/_Documentation)\Git Ecosystem.vsdx. Be sure the change the port in \Configs\generalConfig.json so that you can test data ingestion in DEV/TEST.

**Setup on server:**

1. Grant full access to C:\Users\Public\Documents\NZ Data Warehouse GIT Repository
2. Clone C:\Users\Public\Documents\NZ Data Warehouse GIT Repository to your user’s documents folder
   1. Create a ‘Logs’ folder in your newly created repository (this is ignored by git, so wasn’t cloned).
   2. Update \Configs\generalConfig.json
      1. Change the port to be a lower environment
      2. Create a folder called ‘FTP’ in your documents folder
      3. Change the FTPWatchFolderPath value to be the path to the newly created FTP folder in previous step
      4. Change the email lists to only include your email
3. Copy the file [FILL] to your user’s documents folder
   1. Change the path to watchFolder.py to be the path to watchFolder.py in your user’s documents folder.
4. Grant full access to Anaconda folder C:\ProgramData\Anaconda3
5. Grant full access to Source data folder D:\Source Data

## Front End

The front end on the shared drive: [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/Prod). There is VBA and Python that runs with the front end. The Excel workbook ‘Billing Tool - NZ v1.03.xlsm’ calls the Python script ‘\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Prod\Supporting Code\Python\generateCustomerInvoices.py’ to create the supporting data workbooks. The Python instance lives here: [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\\_Python Program Files\Python37](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/_Python%20Program%20Files/Python37). To develop and test the front end, use the folder here: [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Test](file:///C:/akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/Test). The front end is not tracked by version control.

## Lifecycle

Patrick/Aman will sporadically ask for new features/improvements which are tracked in the Excel Workbook: [\\akl2wsfps001\clshare$\AA Customer Billing\Data Warehouse Billing Tool\Sharing](file://akl2wsfps001/clshare$/AA%20Customer%20Billing/Data%20Warehouse%20Billing%20Tool/Sharing)\Task Priorities.xlsx. There is a bi-weekly meeting with them to prioritize tasks. The high-priority tasks are assigned until the developers' bandwidth is full.

# Australia

The Australia system is nearly identical to the New Zealand system, but on a different server. The key differences are: different server, different site, different customers, different charge calculations, different method of creating the supporting data.

## Server Details

* Description: Microsoft Server OS.
* Application(s): Host for all DWBT technologies except for user interface.
* Technical details: IP: 10.68.1.15

## FTP Details

* Technical details:
  + Server: 10.68.1.15
  + User: ftpUser
  + Pwd: Tr4n5f3r!
  + Port: 21

## Front End Details

Located here: [\\syddvwdfs001\Groups$\Truganina\CL\Alliance\Admin\Billing\Data Warehouse Billing Tool](file://syddvwdfs001/Groups$/Truganina/CL/Alliance/Admin/Billing/Data%20Warehouse%20Billing%20Tool)

# Training Exercises

Below are some training exercises to help provide hands-on experience for common tasks. Complete the tasks in a non-production environment.

## Charges Maintenance

**Task:** The customer ‘All Good‘ is charged for outbound carton picks. Replace this charge with an ‘Outbound Lines’ charge. This charge is for $2 per line.

**Hints:** You will need to update the rate table and modify a function.