**Data Intelligence Platform  
  
  
  
  
  
  
  
  
Refined Layer – Party Model  
  
Transformation Rules**

|  |
| --- |
| Inland Revenue |

**Data Intelligence Platform**

**Version:** v2.18

**Version Date:** 27/02/2020

**Production Release**

**Afbout this document**

This document is designed to provide a set of outline principles, key information, standards and guidance for the clear and unambiguous definition of transmission and transformation, of operational data from core IR data and intelligence sources, into the DIP physical Party model.

It assumes the transformation logic will be that data will be transmitted from EDW and START as the data ‘**sources**’ to the DIP data model as a ‘**target**’.

This is a living document for the duration of the iterative development and will be used to guide the build and transformation logic of the full DIP data model. Therefore, this document will be reviewed and updated with each data model element update.

**Document control**

|  |  |
| --- | --- |
| **File name and path** | <https://irnz.sharepoint.com/:w:/s/DataIntelligenceTeam/ERU-bVotfW1NhAksbnSgjWwBbvMKsAbnQ3YGZbhe1AaRKg?e=08wAxI> |
|  |  |
| **Contact person** | Marc Pearce |
| **Status** | In Review |

**Document signoff (RACI)**

| **Formal Review Area** | **Name/Title** | **Signature** | **Date** |
| --- | --- | --- | --- |
| Responsible person | Marc Pearce |  |  |
| The following people and groups have been Consulted | * DIP Data Working Group * Platform & Data Team |  |  |

**Document review history**

| **No.** | **Date** | **Change Description** | **Contact** |
| --- | --- | --- | --- |
| 1.0 | 14/03/2019 | Document creation from PMO template. | Ally Rutherford |
| 2.0 | 17/03/2019 | Content table reviewed and aligned to Platform team requirements. | Ally Rutherford |
| 3.0 | 17/03/2019 | Document information details in progress. | Ally Rutherford |
| 4.0 | 17/03/2019 | Introduction completed | Ally Rutherford |
| 5.0 –15.0 | 24/03/2019 | Document formatting, Document purpose + Audience complete. | Ally Rutherford |
| 16.0 – 20.0 | 25/03/2019 | START query, transform logic included for PARTY table. Input tables from START/EDW updated. | Ally Rutherford |
| 21.0 – 22.0 | 26/03/2019 | START query changed for updated version (Party table only) | Ally Rutherford |
| 23.0 | 27/03/2019 | Initial review by Marc P, Max W (Platform team) | Ally Rutherford |
| 49.0 | 29/3/2019 | Added Queries and transform blocks for Party, Individual and Org | Greg McPheat |
| 50-398 | 01/04/2019 | Update Start mappings for Party, Individual and Organisation | Ray Duncan |
| 1.71 | 21/06/2019 | Tidied up & added Step 6 SQL statements | Marc Pearce |
| 1.72 | 2/07/2019 | Added more rules in Steps 1 - 4 | Marc Pearce |
| 1.73 | 19/07/2019 | Start transformation rules ready for review | Ray Duncan |
| 1.74 | 22/07/2019 | Added information to section 5 | Mark Openstein |
| 1.75 | 22/07/2019 | Post-review changes for Start | Ray Duncan |
| 1.76 | 24/07/2019 | Merge changes and add EDW updated transform rules formatting | Mark Openstein |
| 25/07/2019 | Further post-review changes for Start | Ray Duncan |
| 1.77 | 31/07/2019 | Ensure EDW & Start are consistent | Ray Duncan |
| 1.78 | 21/08/2019 | Miscellaneous corrections | Ray Duncan |
| 1.79 | 27/08/2019 | Some restructuring and culling | Marc Pearce |
| 1.80 | 13/09/2019 | Standardised de-duplication rules. Removed tax residency (separate table) and NZ Citizen cert/flag. | Ray Duncan |
| 1.90 | 3/10/2019 | Updated rules for Party\_Cease\_Flag and Insolvency\_Flag for FIRST. | Marc Pearce |
| 1.91 | 09/10/2019 | Included insolvency code and reason. Retain inactive flngver=0 records until step#6. New tables: tblNZ\_CustomerStd , tblNZ\_CustomerSubTypeHistory and tblNZ\_CustomerSegmentHistory. | Ray Duncan |
| 1.92 | 15/10/2019 | New table tblcustomerhistory. Changes to Steps#1&6 filters for business dates. Remove tax residency. Apply standards. | Ray Duncan |
| 1.93 | 18/10/2019 | Remove lancountry. Specify cut-off boundaries for single customer timeline. New derived tables app\_tblid\_ird and rfn\_ps\_bankruptcy. New Step#6b to compress the post-merge timelines. New logic for Step#5 and filters for Step#6a. | Ray Duncan |
| 1.94 | 21/10/2019 | Change defaults for Insolvency fields. | Ray Duncan |
| 1.95 | 24/10/2019 | Remove industry\_code\_type and include 5 new BIC description fields. | Ray Duncan |
| 1.96 | 29/10/2019 | Add Business Start Date and Business End Date to Step 4 to be added to change timestamp creation. | Marc Pearce |
| 1.97 | 29/10/2019 | Modified rule for Security\_Level to become a Type1 change so that current values are populated over all time for each Party | Marc Pearce |
| 1.98 | 30/10/2019 | Added ANZSIC\_level[1,2,3,4]\_code as well as descriptions to allow better grouping | Marc Pearce |
| 1.99 | 31/10/2019 | Changed Step#6a filtering rules for business active date ranges - also avoid high-date issue when timestamp offset is > minute. | Ray Duncan |
| 2.00 | 04/11/2019 | Corrected EDW usage of ‘U’ for flags. Retain nulls for org\_incorp\_date to differentiate from 01-01-1900 dates. | Ray Duncan |
| 2.02 | 20/11/2019 | Updated the logic for collecting business cease dates at step 4 so the ceased version doesn’t capture the value of the previous version. | Max Wang |
| 2.03 | 26/11/2019 | EDW customers with a client status of ‘S’ are deemed to have ceased. Customers who only exist in DIP as sourced from EDW and are ‘resurrected’ in Start, must use their existing dip\_party\_id. | Ray Duncan |
| 2.04 | 29/11/2019 | Add explanation of rounding business dates to nearest minute in Step4 | Marc Pearce |
| 2.05 | 03/12/2019 | Change fblnactive to 0 if business commence date equals to cease date for start tables. | Marc Pearce |
| 2.06 | 04/12/2019 | Corrected business dates for EDW. | Ray Duncan |
| 2.07 | 05/12/2019 | Create a derived working field called ActiveFlag for EDW tables passing through Step 2 and set to 0 if business commence date equals business cease date, else set to 1. Also modified ranking in Step 2 and Step 7 rules for EDW tables to align with START rules. | Marc Pearce |
| 2.08 | 11/12/2019 | New Step 2 tie-breaker rules for dss\_client\_codes. Reverse changes for crw\_pe\_client\_codes\_am (AMG/AMD). | Ray Duncan |
| 2.09 | 29/01/2020 | Changes to legal\_name and preferred names for individuals for both FIRST and START rules. Also change to date\_of\_birth rule for FIRST. | Marc Pearce |
| 2.10 | 31/01/2020 | Change to de-duplication logic when both business timeline and record\_effective/record\_expiry timelines are present | Marc Pearce |
| 2.11 | 5/02/2020 | Updated de-duping rules for all Step 2 tables having business commence and cease dates an dupdated Appendix A | Marc Pearce |
| 2.12 | 14/02/2020 | Updated rules around setting of security\_level\_code when level is NULL | Marc Pearce |
| 2.13 | 17/02/2020 | Update EDW Party\_Cease\_Date rule to use DATE\_APPLIED as Cease Date instead of the R3 Cut over date. Also add rule to reset all Record\_Expry\_Timestamps to be set to 1 time grain (1 minute) before the next record’s Record\_Effective\_Date to avoid gaps in a timeline where the first observed START fdtmWhen is after the R3 cut over date. | Marc Pearce |
| 2.14 | 19/02/2020 | Include standard Unknown and N/A DIP Ids for all dimensions. Rewrite step#7 rules for security\_level(\_code) to check Start before EDW. | Ray Duncan |
| 2.15 | 21/02/2020 | Include RecordExpiryTimestamp + 1 minute as a candidate timestamp for EDW timeline generation. | Marc Pearce |
| 2.16 | 24/02/2020 | No longer default Trading Name to Legal Name. | Ray Duncan |
| 2.17 | 26/02/2020 | Add a solution to fix dip\_party\_id for customer who is resurrected and goes bankrupt before the next party incremental load. | Max Wang |
| 2.18 | 27/02/2020 | Use a coalesce of CustomerClass and CustomerSubtypeClass to populate Party\_Class\_Code from START | Marc Pearce |

Contents

[1 Introduction 8](#_Toc33700434)

[1.1 Background 8](#_Toc33700435)

[1.2 Document Purpose 8](#_Toc33700436)

[1.3 Audience 8](#_Toc33700437)

[1.4 Reference Documents 8](#_Toc33700438)

[1.5 Acronyms, Abbreviations and Key Terms 8](#_Toc33700439)

[2 Data Model 10](#_Toc33700440)

[2.1 Party Model – Individual + Organisation Layer. 10](#_Toc33700441)

[2.2 Party Model - Conformed Dimensions 10](#_Toc33700442)

[2.3 DIP Party Level Table view 11](#_Toc33700443)

[3 Processing Overview 12](#_Toc33700444)

[3.1 Macro Load View – Combining data from START and EDW 12](#_Toc33700445)

[3.2 Transformation of each source 12](#_Toc33700446)

[4 Processing Sequence 13](#_Toc33700447)

[4.1 Processing Sequence 13](#_Toc33700448)

[4.2 EDW – Commence from DoB 13](#_Toc33700449)

[4.3 EDW-START Cut-Off 13](#_Toc33700450)

[4.4 Input Tables required for PARTY. 13](#_Toc33700451)

[5 Mapping Tables Required for Processing EDW & START 15](#_Toc33700452)

[5.1 EDW-START Code Map 15](#_Toc33700453)

[5.2 Start Customer Key - IRD Number Map 16](#_Toc33700454)

[5.3 Dedup Rankings for Start Indicators 19](#_Toc33700455)

[6 Transform Rules: EDW 21](#_Toc33700456)

[6.1 Step 1 – Selecting Valid Values from Source Tables 21](#_Toc33700457)

[6.2 Step 2 – De-duplication of Records 23](#_Toc33700458)

[6.3 Step 3 – Compressing the timeline 24](#_Toc33700459)

[6.5 Step 4 – Combining all change timeline 26](#_Toc33700460)

[6.6 Step 5 – Create a single timeline for each Party 27](#_Toc33700461)

[6.7 Step 6a – Merge change timelines with Party timeline 28](#_Toc33700462)

[6.8 Step 6b – Compress the Post-Merge Timelines 29](#_Toc33700463)

[6.9 Step 7 – Transform Rule Logic 30](#_Toc33700464)

[7 Transform Rules: START 47](#_Toc33700465)

[7.1 Step 1 – Selecting Valid Values from Source Tables 47](#_Toc33700466)

[7.2 Step 2 – De-duplication of Records 50](#_Toc33700467)

[7.3 Step 3 – Compressing the timeline 53](#_Toc33700468)

[7.4 Step 4 – Combining all change timelines 57](#_Toc33700469)

[7.5 Step 5 – Create a single timeline for each Party 59](#_Toc33700470)

[7.6 Step 6a – Merge change timelines with Party timeline 59](#_Toc33700471)

[7.7 Step 6b – Compress the Post-Merge Timelines 62](#_Toc33700472)

[7.8 Step 7 – Transform Rule Logic 68](#_Toc33700473)

[Appendix A – Standard De-Duplication Process 86](#_Toc33700474)

[Appendix B – SQL to Populate table mp\_source\_code 89](#_Toc33700475)

[Appendix C – EDW-START Party Map 92](#_Toc33700476)

[Appendix D – Populating Natural Key DIP\_PARTY\_ID for EDW 93](#_Toc33700477)

[Appendix E – Sample SQL for Context – EDW 95](#_Toc33700478)

[Appendix F – Sample SQL for Context – Start 101](#_Toc33700479)

# Introduction

## Background

Inland Revenue (IR) is delivering a new Data and Intelligence Platform (DIP) as part of the Business Transformation Programme and a move to becoming an Intelligence-led organisation.

The DIP will be pivotal in enabling IR to make better decisions faster, and to improve outcomes for customers.

The DIP will have a pivotal role in exposing consistent, cleansed and integrated data to IR staff to enable the concept of being “intel-led” to become embedded throughout the organisation.

## Document Purpose

To detail the information and processing required to populate the new IR DIP Data Model, starting at the Party level.

The core is a set of transformation instructions determining how to convert the structure and content of raw data layer generated by regular extracts from START and EDW (FIRST) to the required structure and content needed in the DIP (Target) refined data layer.

## Audience

This document is primarily intended for to be used by data integration developers within the Platform & Data team to develop the SAS Data Integration Studio packages that will process the data from the raw layer to the refined layer.

A secondary purpose is to allow any user including data scientists and data engineers to understand how the raw source system data was cleansed, integrated and transformed into the refined layer model.

## Reference Documents

The following deliverables, reviews and open source information can be considered when reading this document:

* IRD DIP - [Understanding and Navigating the START database](https://irnz.sharepoint.com/:w:/s/DataIntelligenceTeam/EfW98mcCJXBKrv5PhWKvmVcBJ7445kWMbwtqah-aKLcGIw?e=FcBCls)
* IRD DIP – Data Modelling Standards – ***To be confirmed***
* IRD DIP - DIP\_Party (DIP\_Data\_Model) Oracle SQL Developer | Data Modeller (Figure 2)
* IRD DIP - [Dimension Standards](https://teams.microsoft.com/l/file/C43DC83E-9B7D-4863-99A6-A1E2A5CFA3E1?tenantId=fb39e3e9-23a9-404e-93a2-b42a87d94f35&fileType=docx&objectUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam%2FShared%20Documents%2FRefined%20Layer%20Design%20and%20Build%2F9.%20%20Standards%2FDIP_Dimension_Standards.docx&baseUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam&serviceName=teams&threadId=19:c9ee9980de684e74badc5fe1d414bc56@thread.skype&groupId=7100c035-72cd-4625-aa78-807e71980487)

## Acronyms, Abbreviations and Key Terms

**Acronyms**

| **Acronym** | **Long Name** |
| --- | --- |
| START | New operational tax system |
| EDW | Enterprise Data Warehouse, based on FIRST data |
| TDW | Transitional Data Warehouse (A schema within EDW) |
| DIP | Data & Intelligence Platform |
| FIRST | ILegacy operational tax system |

**Abbreviations**

| **Abbreviation** | **Meaning** | **Definition** |
| --- | --- | --- |
| $$HWM | High-water mark | This is the timestamp of the point in time to which a DIP table has been populated. Just as a dam has a high-water mark which measure how much water it contains; each DIP table has a high-water mark that shows timestamp of the last record inserted in chronological order. |
| $$LODATE | System low date | 01-Jan-1900 |
| $$HIDATE | System high date | 31-Dec-9000 |
| $$EDWDT | EDW Cut-off date | The cut-off date between EDW and Start used for the initial load i.e. 05-Feb-2017 |
| $$EDWDT-1 | EDW Cut-off date – 1 second | One second prior to $$EDWDT |

**Key Terms**

| **Term** | **Definition** |
| --- | --- |
| Production quality data | An instance or subset of production data, or, an offline version of production data that is representative of the quality levels of current production data. The data must be sufficiently representative that it can be effectively used for data profiling or quality assessment. |
| Master Data | Any information that is relatively static and is considered to play a key role in the core operation of the business (i.e. reference data); includes data about customers, products, employees, partners etc. Master data is typically shared by multiple users and groups across the organisation. |
| Dimensional or Star Schema | Dimensional data modelling standard is most often used in data warehousing systems. This is different from the Relational or 3rd normal form, commonly used for transactional (OLTP) type systems. Dimensional modelling uses denormalisation steps to improve performance, which introduces redundancy of data |
| Subject Area | A logical grouping of Entities relating to a sub domain of interest within a data model. A subject area allows focus attention on key entities of interest. |
| Staging Layer | The table structures in staging layer reflect the table or file structure of the source extract without any transformation applied. |

# Data Model

The Refined Layer Data Model has been designed specifically for use with the DIP. It is aligned with the concept of Dimensional Modelling in that some tables are clearly dimensions and others clearly fact tables. However, where dimension to dimension relationships are held over time, the model generally uses a relationship table holding the natural keys rather than the dimensional surrogate keys to reduce processing needs and make the data more accessible.

The Data model is designed around a ‘Party Concept’. A Party being a ‘Legal-Entity’ in the context of an Individual or an Organisation.

## Party Model – Individual + Organisation Layer.

PARTY

Organisation

Individual

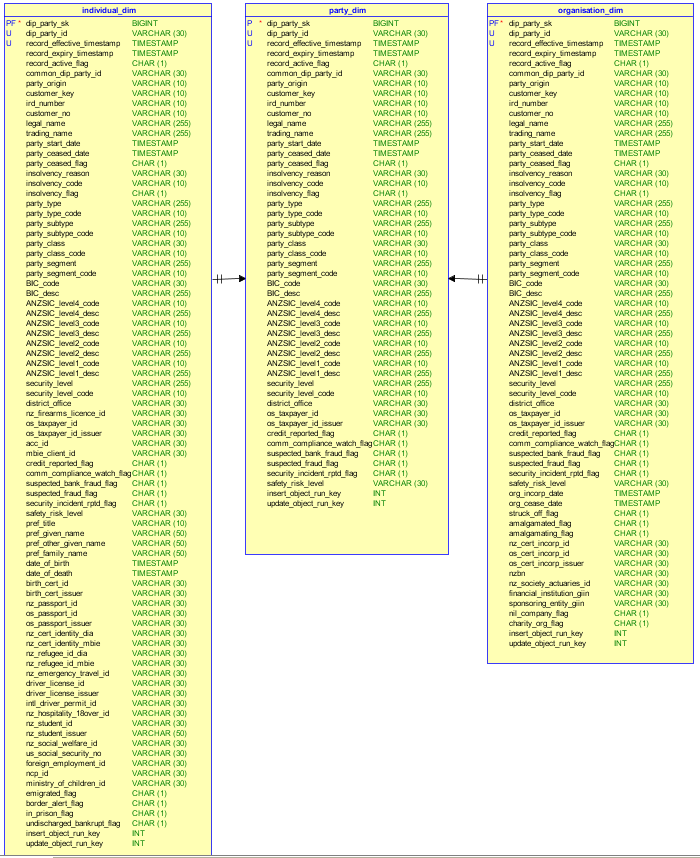
Figure 1: Initial Level - Party Model Design

Individuals and Organisations are the two sub-types of the gerneric Party entity. Individuals include Child Support customers (Start customer type of CHD) who were registered with one of the three Child Support account (tax) types i.e. NCP (Liable Parent) or CPR (Receiving Carer) or children (normally without accounts).

## Party Model - Conformed Dimensions

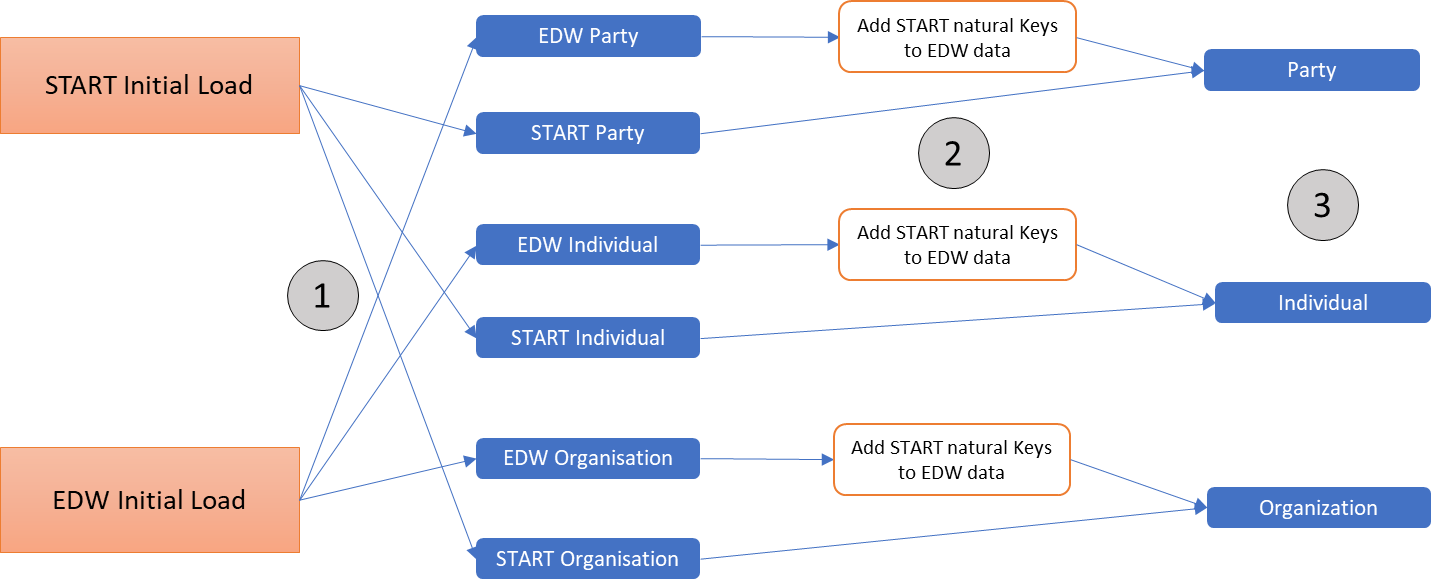
Tables PARTY\_DIM, INDIVIDUAL\_DIM and ORGANISATION\_DIM are “conformed dimensions” with ‘common’ attributes having identical column names and data type domains. Information may be combined in a single report by using conformed dimension attributes that are associated with each fact table. When a ‘conformed’ attribute is used as the row header (that is, the grouping column in the SQL query), the results can be aligned on the same rows in a drill-across report.

## DIP Party Level Table view



# Processing Overview

## Macro Load View – Combining data from START and EDW

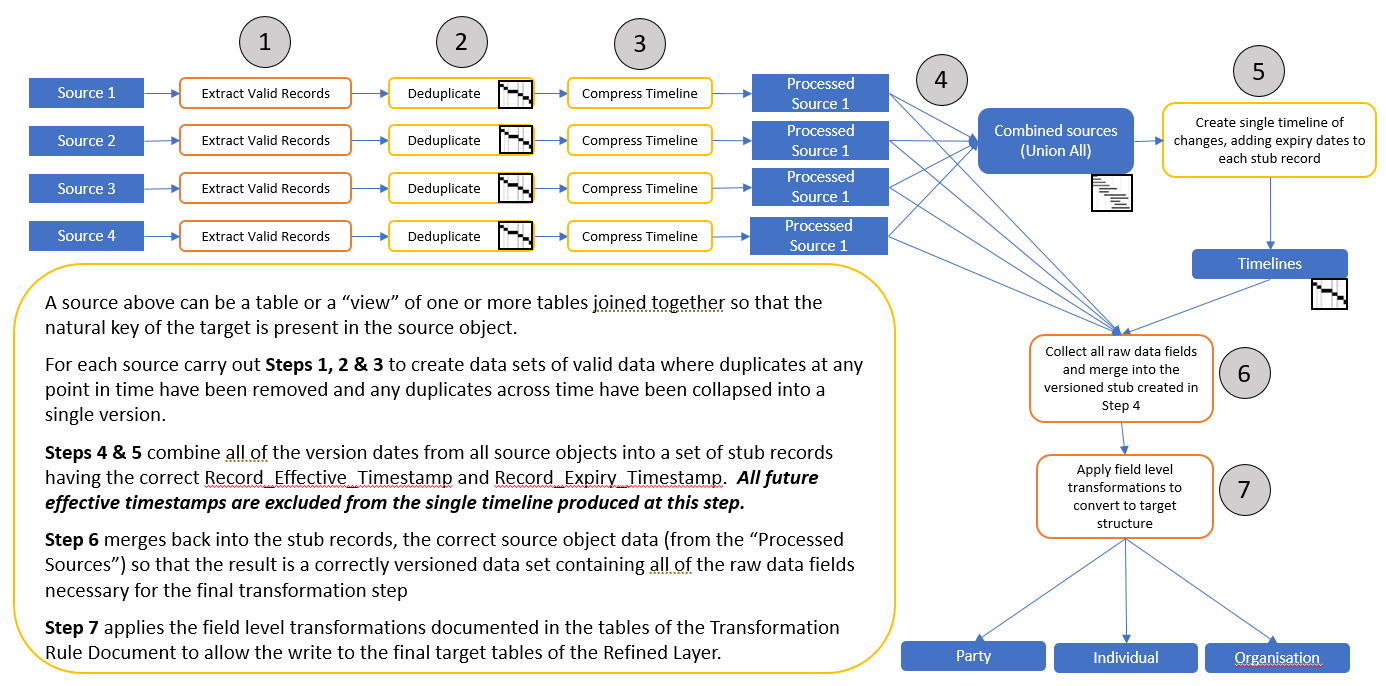


In **Step 1**, each source system produces a target data set with all transforms performed and clean correctly versioned timelines

In **Step 2**, the appropriate START keys and IDs are added to the EDW data to make the versioning over time seamless between the two source systems

In **Step 3** the historical data from EDW is combined with the START data according to the rules specified in this document to produce the final target tables of the Refined layer

## Transformation of each source



**NB**

After merging the EDW and START data for a customer using the rules relating to the cut-over date. Adjust all Record\_Expiry\_Timestamps to be set to 1 minute earlier than the Record\_Effective\_Timestamp of the following record for that Natural Key. (Do not adjust the latest record for the Natural Key)

# Processing Sequence

## Processing Sequence

The Party level model is the foundation layer of the DIP data model. It is generated by <40 common fields found in both the Individual and Organisational layers and so processing is required to follow in the below order:

| **#** | **Source System** | **Ingestion Details** |
| --- | --- | --- |
| 1 | EDW/START | Create mapping table *dip\_edw\_start\_party\_map* |
| 2 | EDW | One-off ingestion of Individuals, Organisations and Parties with cut-off at $$EDWDT |
| 3 | START | Initial ingestion of Individuals, Organisations and Parties from $$EDWDT |
| 4 | START | Ongoing incremental ingestion of Party domain after initial load |

## EDW – Commence from DoB

For EDW individuals, their first record will be deemed to commence from their date of birth. This is especially relevant for child customers (i.e. child support/custody).

## EDW-START Cut-Off

For EDW customers who also exist in Start (see Appendixes C and D) with records which have record\_expiry\_date after $$EDWDT, set the record\_active\_flag to ‘N’ and set the record\_expiry\_date to $$EDWDT-1. Otherwise, leave the record unchanged.

## Input Tables required for PARTY.

Prior to loading the tables into the **Party** area, the following raw tables are required:

Raw Layer tables from START

|  |
| --- |
| app\_tblcustomer |
| app\_tblcustomerinfo |
| app\_tblcustomerhistory |
| app\_tblcustomerlevel |
| app\_tblcustomerstd |
| app\_tblNameRecord |
| app\_tblid |
| app\_tblindicator |
| app\_tblnaics |
| app\_tblnz\_accidgovr |
| app\_tblNZ\_AccIDGEdu |
| app\_tblnz\_cstcominfo |
| app\_tblnz\_customerstd |
| app\_tblnz\_custsegmenthistory |
| ref\_lancustomertype |
| ref\_lancustomerlevel |
| ref\_lancustomersubtype |
| ref\_lannz\_customerclass |
| ref\_lannz\_customersegments |
| ref\_lanofficelocation |
| ref\_lanindicator |
| local\_start\_bankruptcy |

Note: Table local\_start\_bankruptcy is a one-off load from a file provided by Aaron Parker. The contents of this file will not change as bankruptcies are now managed in START.

Raw Layer tables from EDW (FIRST)

|  |
| --- |
| dss\_clients |
| dss\_entity\_types |
| dss\_ entity \_classes |
| dss\_customers |
| dss\_offices |
| dss\_old\_system\_numbers |
| dss\_agents |
| dss\_client\_names |
| dss\_client\_codes |
| dss\_external\_references |
| app\_tblid |
| dip\_edw\_start\_party\_map *(DIP table created to support mappings)* |

# Mapping Tables Required for Processing EDW & START

Some mapping tables are required to be built to allow the correct assignment of “START-like” codes to legacy data and to create a value for the DIP\_PARTY\_ID that maps to an IRD\_NUMBER or flngCustomer\_Key.

## EDW-START Code Map

Because columns in EDW tables have different sets of codes and/or descriptions when compared to the equivalent columns in START we need a mapping table to allow processing to lookup the correct values.

Table mp\_source\_code (mp is an abbreviation for mapping) will follow a pattern 1 design and will only function as an insert table using pre-set string values input by the user. The table itself will only be populated once for party and used initially for only mapping START to EDW value precedence where appropriate for consistency of data across EDW and START. e.g. where a value in EDW is ‘I’, START may have a value of ‘INDVL’ which will overwrite I in the load to the party domain.

The following table reflects the contents of mp\_source\_code. The basic SQL to populate table mp\_source\_code can be found in Appendix B.

| **Source System** | **Source Code Name** | **Source Code Value** | **Source Description** | **DIP Code Name** | **DIP Code Value** | **DIP description** |
| --- | --- | --- | --- | --- | --- | --- |
| FIRST | ENTITY\_TYPE | I | Individual | PARTY\_SUBTYPE | INDVDL | Individual |
| FIRST | ENTITY\_TYPE | C | Company | PARTY\_SUBTYPE | COMPNY | Company |
| FIRST | ENTITY\_TYPE | T | Trust | PARTY\_SUBTYPE | TRUST | Trust |
| FIRST | ENTITY\_TYPE | P | Partnership | PARTY\_SUBTYPE | PTNRSP | Partnership |
| FIRST | ENTITY\_TYPE | S | Society/Club | PARTY\_SUBTYPE | SOCITY | Society/Club |
| FIRST | ENTITY\_TYPE | M | Maori Auth. | PARTY\_SUBTYPE | MRIAUT | Maori Authority |
| FIRST | ENTITY\_TYPE | X | To be est'd | PARTY\_SUBTYPE | TBD | To Be Established |
| FIRST | ENTITY\_TYPE | U | Unit Trust | PARTY\_SUBTYPE | UNTTST | Unit Trust |
| FIRST | ENTITY\_TYPE | F | Super. Fund | PARTY\_SUBTYPE | SPRFND | Super Fund |
| FIRST | ENTITY\_TYPE | A | Holding Acct | PARTY\_SUBTYPE | HLDACC | Holding Account |
| FIRST | ENTITY\_TYPE | D | Diplomatic M | PARTY\_SUBTYPE | EMBASY | Embassy |
| FIRST | ENTITY\_TYPE | G | GOV'T DEP'T | PARTY\_SUBTYPE | GVMDPT | Government Department |
| FIRST | CUST\_TYPE | ORG | ORG | PARTY\_TYPE | COM | Non-Individual |
| FIRST | CUST\_TYPE | IND | IND | PARTY\_TYPE | IND | Individual |
| FIRST | OFFICE\_CODE | 00 |  | DISTRICT\_OFFICE | 0 | No access office |
| FIRST | OFFICE\_CODE | 02 | Auckland Corporate | DISTRICT\_OFFICE | 02 | Auckland Large Enterprises |
| FIRST | OFFICE\_CODE | 10 | SOUTH ISLAND SERVICE CENTRE | DISTRICT\_OFFICE | 10 | Russley Road. Christchurch |
| FIRST | OFFICE\_CODE | 15 | Dunedin | DISTRICT\_OFFICE | 15 | Dunedin |
| FIRST | OFFICE\_CODE | 19 | Gisborne | DISTRICT\_OFFICE | 19 | Gisborne |
| FIRST | OFFICE\_CODE | 23 | Greymouth | DISTRICT\_OFFICE | 23 | Greymouth |
| FIRST | OFFICE\_CODE | 27 | Hamilton | DISTRICT\_OFFICE | 27 | Hamilton |
| FIRST | OFFICE\_CODE | 34 | Invercargill | DISTRICT\_OFFICE | 34 | Invercargill |
| FIRST | OFFICE\_CODE | 38 | Auckland Service Centre | DISTRICT\_OFFICE | 38 | Manukau - Twin Towers |
| FIRST | OFFICE\_CODE | 47 | Napier | DISTRICT\_OFFICE | 47 | Napier |
| FIRST | OFFICE\_CODE | 48 | Nelson | DISTRICT\_OFFICE | 48 | Nelson |
| FIRST | OFFICE\_CODE | 49 | New Plymouth | DISTRICT\_OFFICE | 49 | New Plymouth |
| FIRST | OFFICE\_CODE | 58 | Palmerston North | DISTRICT\_OFFICE | 58 | Palmerston North |
| FIRST | OFFICE\_CODE | 62 | Rotorua | DISTRICT\_OFFICE | 62 | Rotorua |
| FIRST | OFFICE\_CODE | 67 | Takapuna | DISTRICT\_OFFICE | 67 | Takapuna - AIA House |
| FIRST | OFFICE\_CODE | 69 | Corporate - Tax Avoidance | DISTRICT\_OFFICE |  | Corporate - Tax Avoidance |
| FIRST | OFFICE\_CODE | 72 | Tauranga | DISTRICT\_OFFICE | 72 | Tauranga Regional House |
| FIRST | OFFICE\_CODE | 80 | Timaru | DISTRICT\_OFFICE | 80 | Timaru |
| FIRST | OFFICE\_CODE | 89 | Wellington Corporate | DISTRICT\_OFFICE | 89 | Wellington Large Enterprises |
| FIRST | OFFICE\_CODE | 90 | Wellington | DISTRICT\_OFFICE | 90 | Wellington Service Centre |
| FIRST | OFFICE\_CODE | 93 | Whangarei | DISTRICT\_OFFICE | 93 | Whangarei |
| FIRST | CLIENT\_STATUS | B | Bankrupt | INSOLVENCY\_CODE | INSCBF | Bankruptcy finalised |
| FIRST | CLIENT\_STATUS | R | Rec'ship | INSOLVENCY\_CODE | INSCRC | In Receivership |
| FIRST | CLIENT\_STATUS | L | Liq'dation | INSOLVENCY\_CODE | INSCLF | Liquidation finalised |
| FIRST | CLIENT\_STATUS | U | Und Bnkrpt | INSOLVENCY\_CODE | UNDSCH | Undischarged bankruptcy |
| FIRST | CLIENT\_STATUS | V | VOL ADMIN | INSOLVENCY\_CODE | INSCVA | Under Voluntary Administration |

## Start Customer Key - IRD Number Map

Mappings between start customer keys and IR numbers are required to populate the start\_customer\_key and dip\_party\_id fields in the Party table. This information is available in the rfn\_party\_sk\_map table which holds the generated dip party surrogate key for each version of the party changes for both Start and EDW.

During the Party load for Start, the map table is populated with each unique version of start customer key + record\_effective\_timestamp with an allocated DIP party surrogate key, the active associated ird\_number and transformed dip\_party\_id.

The following SQL used in the ETL process performs the following tasks:

1. Identify all unique versions of party changes from the timeline table for START.
2. Check if the start customer key has been loaded into the key map table before.
3. If the start key already exists in the key map, use the dip\_party\_id from the key map for this version
4. If the start key doesn’t exist, look up the active IRD number in the clean raw tblid table to find a valid IRD number and see if a dip\_party\_id value can be found based on the ird\_number (to resurrect converted customer from First)
5. If no values returned from both step 3 and step 4, generate a new dip\_party\_id by concatenating ‘DIP-C-’ with the start customer key.

select

cast(ifnull(key\_map.dip\_party\_id,ifnull(edw.dip\_party\_id , concat('DIP-C-',cast(timeline.flngcustomerkey as string)))) as varchar(20)) as dip\_party\_id,

timeline.record\_effective\_timestamp,

timeline.flngcustomerkey as start\_customer\_key,

cast( ird.fstrid as int) as ird\_no

from

rfn\_ps\_timeline as timeline left join

(

select distinct

rfn\_party\_sk\_map.start\_customer\_key,

rfn\_party\_sk\_map.dip\_party\_id

from

rfn\_party\_sk\_map as rfn\_party\_sk\_map

) as key\_map

on

(

timeline.flngcustomerkey = ifnull(key\_map.start\_customer\_key,-1000)

) left join

crw\_ps\_tblid\_ird as ird

on

(

timeline.flngcustomerkey = ird.flngcustomerkey

and timeline.record\_effective\_timestamp between ird.record\_effective\_timestamp and ird.record\_expiry\_timestamp

and timeline.record\_effective\_timestamp >= ird.fdtmcommence and days\_add(timeline.record\_effective\_timestamp,-1) < ird.fdtmcease

and ird.fblnactive = 1

) left join

(

select distinct

rfn\_party\_sk\_map.dip\_party\_id,

rfn\_party\_sk\_map.ird\_no

from

rfn\_party\_sk\_map as rfn\_party\_sk\_map

where

start\_customer\_key is null

) as edw

on

(

edw.ird\_no = cast( ird.fstrid as int)

)

After the Start Party load is completed, the EDW Party load uses this table to find the start customer key for its IRD numbers and populates the unique version to the sk\_map table. The following SQL used by the ETL process performs the following tasks:

1. Identify all the unique version of party changes from the all\_dates table for EDW
2. Check if the start customer key has been loaded into the key map table
3. Look up the IRD number in the key map to find the associated start customer key number.
4. If the process is not able to find a matched customer key from the Start side, it sets the upper cut off date to 9000-12-31 which is used to determine the application of the interleave operation for this EDW client.

select

all\_dates.ird\_no,

all\_dates.record\_effective\_timestamp,

case when key\_map.start\_customer\_key is not null then key\_map.dip\_party\_id

else cast(concat('DIP-I-',cast(all\_dates.ird\_no as varchar(16))) as varchar(30)) end as dip\_party\_id,

key\_map.start\_customer\_key,

case when c.cust\_type = 'ORG' and c.org\_commencement\_date between '1880-01-01' and now() then c.org\_commencement\_date

when c.cust\_type = 'IND' and c.date\_of\_birth between '1880-01-01' and now() then c.date\_of\_birth

when c.cust\_type = 'IND' and c.date\_created between '1880-01-01' and now() then c.date\_created

else '1900-01-01' end as lower\_cutoff,

case when key\_map.start\_customer\_key is not null then minutes\_add(%nrbquote('&start\_cutoff'),-1)

else '9000-12-31' end as upper\_cutoff

from

rfn\_pe\_all\_dates\_f as all\_dates left join

(

select distinct

rfn\_party\_sk\_map.dip\_party\_id,

rfn\_party\_sk\_map.start\_customer\_key,

rfn\_party\_sk\_map.ird\_no

from

rfn\_party\_sk\_map as rfn\_party\_sk\_map

) as key\_map

on

(

all\_dates.ird\_no = ifnull(key\_map.ird\_no,-1000)

) left join

(

select

crw\_pe\_clients.ird\_number,

crw\_pe\_clients.cust\_type,

min(crw\_pe\_clients.date\_created) as date\_created,

min(crw\_pe\_clients.date\_of\_birth) as date\_of\_birth,

min(crw\_pe\_clients.org\_commencement\_date) as org\_commencement\_date

from

crw\_pe\_clients as crw\_pe\_clients

group by

crw\_pe\_clients.ird\_number,

crw\_pe\_clients.cust\_type

) as c

on

(

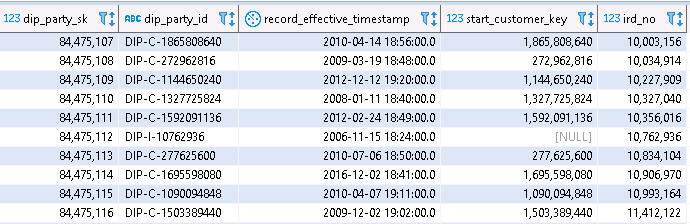
all\_dates.ird\_no = c.ird\_number

)

where

all\_dates.record\_effective\_timestamp < now()

The structure of the rfn\_party\_sk\_map table with some sample records is illustrated below:



There is a potential issue for finding the original dip\_party\_id in the key map for resurrected customers during an incremental load:

Let’s assume a customer is converted to Start from EDW, and then receives a new IRD number after being made bankrupt. If both activities happen before the next party incremental load, two version of the same customer will be generated in START with different IRD numbers.

In the next load, we can still find the original DIP\_PARYT\_ID for the first version by using the IRD number. However, the process will generate a new dip party id for the second version because it is not able to find a matched DIP\_PARTY\_ID by either the IRD number (because it is a new number) or the START customer key (because the first version is in the same batch and has not been loaded into the key map yet).

To resolve this problem, we use the following SQL to detect records for this special case and correct their DIP\_PARTY\_IDs before loading them into the key map.

select

cast( ifnull(rec.dip\_party\_id,keys.dip\_party\_id) as varchar(30)) as dip\_party\_id,

keys.record\_effective\_timestamp,

keys.start\_customer\_key,

keys.ird\_no

from

rfn\_ps\_keys as keys left join

(

select distinct

rfn\_ps\_keys.dip\_party\_id,

rfn\_ps\_keys.start\_customer\_key

from

rfn\_ps\_keys as rfn\_ps\_keys

where

dip\_party\_id like '%-I-%' and start\_customer\_key is not null

) as rec

on

(

keys.start\_customer\_key = rec.start\_customer\_key

and keys.dip\_party\_id <> rec.dip\_party\_id

)

## Dedup Rankings for Start Indicators

When ‘clustered’ Start indicators are de-duplicated, the indicators within the ‘cluster’ need to be ranked to determine their priority. For example, if boh INSCBF and UNDSCH indicators existed at the same point-in-time for a party, the INSCBF would be selected because it has a higher priority.

A derived table drd\_code\_family in the work database:

| **Code Family** | **origin** | **Indicator** | **Rank** |
| --- | --- | --- | --- |
| INSOLVENCY | START | UNDSCH | 13 |
| INSOLVENCY | START | INSCBF | 12 |
| INSOLVENCY | START | INSCLF | 11 |
| INSOLVENCY | START | INSFIN | 10 |
| INSOLVENCY | START | INSCBK | 9 |
| INSOLVENCY | START | INSCNA | 8 |
| INSOLVENCY | START | INSCLQ | 7 |
| INSOLVENCY | START | INSCRC | 6 |
| INSOLVENCY | START | INSCVA | 5 |
| INSOLVENCY | START | RECEIV | 4 |
| INSOLVENCY | START | INSMTR | 3 |
| INSOLVENCY | START | INSPRG | 2 |
| INSOLVENCY | START | PRVBNK | 1 |
| RISK | START | HGRSK | 1 | |
| RISK | START | MEDRSK | 2 | |
| RISK | START | LOWRSK | 3 | |

Notes:

1. Use the highest ranking in preference to lower rankings
2. For the ETL build, this could be implemented as a table and left-outer-joined to table app\_tblIndicator with nulls translated to zero.

# Transform Rules: EDW

## Step 1 – Selecting Valid Values from Source Tables

The purpose of this step is to reduce the number of records from the source object to those that are relevant to the processing. This may include only extracting a subset of rows as well as excluding records that are not deemed to be valid.

| **Source Table** | **Filtering Criteria** | **Transformation Rule / Logic** | **Change Capture Column** | **Business Active**  **Timestamps** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| dss\_clients | select cust\_type in ('ORG', 'IND')  and ird\_number not equal to old\_system\_number within dss\_old\_system\_numbers  and ird\_number not equal to old\_system\_number within dss\_agents  and date\_created < $$EDWDT |  | timestamp |  | We will not handle records created after this date. |
| dss\_customers | select location\_number=1  and ird\_number not equal to old\_system\_number within dss\_old\_system\_numbers  and ird\_number not equal to old\_system\_number within dss\_agents |  | date\_ceased |  |  |
| dss\_client\_names | select location\_number in (1,0)  and client\_name\_type in (‘P’,’T’)  and ird\_number not equal to old\_system\_number   within dss\_old\_system\_numbers  and ird\_number not equal to old\_system\_number within dss\_agents |  | timestamp |  | For preferred names, location\_number=0  For trading names, location\_number=1 |
| dss\_external\_references | select location\_number=1  and external\_org\_code in ('BIR', 'PAS', 'OPT', 'ETD', 'LIC', 'DSW', 'HNZ', 'NZC', 'RTD')  and ird\_number not equal to old\_system\_number   within dss\_old\_system\_numbers  and ird\_number not equal to old\_system\_number within dss\_agents | Derive ActiveFlag:  if date\_start = date\_end  then ActiveFlag = 0  else ActiveFlag = 1  If date\_start is null, set it to 9000-12-31 | timestamp | date\_start  date\_end |  |
| dss\_client\_codes | location\_number = 1  and ird\_number not equal to old\_system\_number   within dss\_old\_system\_numbers  and ird\_number not equal to old\_system\_number within dss\_agents  client\_code\_value in (‘AMG’,’AMD’) | Derive ActiveFlag:  if date\_start = date\_end  then ActiveFlag = 0  else ActiveFlag = 1  If date\_start is null, set it to 9000-12-31 | timestamp | date\_start  date\_end |  |
| dss\_entity\_types |  |  | timestamp |  | Lookup value table |
| dss\_ entity \_classes |  |  | timestamp |  | Lookup value table |
| dss\_offices |  |  | timestamp |  | Lookup value table |
| dss\_old\_system\_numbers |  |  | timestamp |  | Lookup value table (we will not process old system numbers, so we exclude them) |
| dss\_agents |  |  | timestamp |  | Lookup value table (we will not process old system numbers, so we exclude them) |
| dss\_special\_clients\_all | select ird\_number not equal to old\_system\_number within dss\_old\_system\_numbers |  |  |  |  |

**Derived Tables:**

| **Derived Table** | **Source Table(s)** | **Derivation Rules** | **Comments** |
| --- | --- | --- | --- |
| ~~crw\_pe\_client\_codes\_am~~ | ~~dss\_client\_codes~~ | **~~Filter from table dss\_client\_codes:~~**  ~~client\_code\_value in (‘AMG’,’AMD’)~~ |  |

## Step 2 – De-duplication of Records

The purpose of this step is to remove duplicates over time so that the resulting rows for each natural key are contiguous and non-overlapping in time. For these tables create a derived working field called “Active” to bet set to 1 unless the business active timestamps are equal, in which case set to 0. This field is then used in the Tie-Breaker logic for de-duping.

The default de-duplication process (see Appendix A) is to be following unless explicitly stated otherwise.

| **Source Table** | **De-duplication Rule** | **Natural Key** | **Business Active**  **Timestamps** | **Tie-Breaker(s)** |
| --- | --- | --- | --- | --- |
| dss\_clients |  | ird\_number |  |  |
| dss\_customers |  | ird\_number  location\_number |  |  |
| dss\_client\_names | Use the default de-duplication process (see Appendix A) | ird\_number  location\_number  client\_name\_type |  | Rank:   1. record\_expiry\_timetamp (latest first i.e. most recently in effect) 2. date\_applied(latest first i.e. current records first) 3. sequence\_number (highest first – final tie-breaker if needed) |
| dss\_external\_references | Use the default de-duplication process (see Appendix A) | ird\_number  location\_number  external\_org\_code | date\_start  date\_end | Rank:   1. ActiveFlag (1 before 0) 2. Number\_of\_valid\_seconds ( highest first. this is a derived column. Refer to Appendix 1) 3. Date\_applied (latest first) 4. Record\_expiry\_timestamp (latest first) 5. Date\_end (latest first) 6. Date\_start (earliest first) 7. externalid (highest first (alpha sort)) |
| dss\_client\_codes |  | ird\_number  client\_code\_value | date\_start  date\_end | Rank:   1. ActiveFlag (1 before 0) 2. Number\_of\_valid\_seconds ( highest first) 3. date\_applied (latest first – final tie-breaker) |
| dss\_entity\_types |  | entity\_type\_code  record\_effective\_timestamp |  |  |
| dss\_ entity \_classes |  | entity\_class\_code  record\_effective\_timestamp |  |  |
| dss\_offices |  | office\_code  record\_effective\_timestamp |  |  |
| dss\_special\_clients\_all |  | ird\_number |  |  |

## Step 3 – Compressing the timeline

After de-duplicating records in Step 2, one or more contiguous records sharing the same Natural Key may have the same values for all columns used for comparison. This step will ‘compress’ (i.e. merge) any such contiguous records using the minimum Record\_Effective\_Date and maximum Record\_Expiry\_Date for the resulting timeline.

NB. Business active timestamps/dates should always be included in the “columns to compare”.

| **Source Table** | **Natural Key** | **Business Active**  **Timestamps** | **Columns to Compare** | **Dates for Timelines** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| dss\_clients | ird\_number |  | client\_status  cust\_type  date\_of\_birth  entity\_class  entity\_type  nz\_citizen\_ind  org\_commencement\_date  resident\_ind | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_customers | ird\_number  location\_number |  | cust\_type  sic\_code  district\_office | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_client\_names | ird\_number  client\_name\_type |  | first\_names  organisation\_name  surname  title | Date\_applied (set the first record to low-date)  Record\_Expiry\_Timestamp |  |
| dss\_external\_references | ird\_number  external\_org\_code  external\_id | date\_start  date\_end | country\_code  date\_start  date\_end | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_client\_codes | ird\_number  client\_code\_value | date\_start  date\_end | client\_code\_value  date\_start  date\_end | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_entity\_types | entity\_type\_code  record\_effective\_timestamp |  | description | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_ entity \_classes | entity\_class\_code  record\_effective\_timestamp |  | description | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_offices | office\_code  record\_effective\_timestamp |  | description | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| dss\_special\_clients\_all | ird\_number |  | special\_classification | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ~~dss\_client\_codes\_am~~ | ~~ird\_number~~  ~~client\_code\_value~~ | ~~date\_start~~  ~~date\_end~~ | ~~client\_code\_value~~  ~~date\_start~~  ~~date\_end~~ | ~~Record\_Effective\_Timestamp Record\_Expiry\_Timestamp~~ |  |

## Step 4 – Combining all change timeline

The purpose of this step is to create a UNION of all change timestamps for the Party data that will become the basis for all change records created. These timelines are stored in the Refined Layer in table rfn\_pe\_all\_dates. Each record in the timeline is bounded in time from record\_effective\_timestamp to record\_expiry\_timestamp.

A change timestamp is generated for each Record\_Effective\_Timestamp at a 1-minute granularity, additionally where a Business Start Date and/or Business End date are listed below, a change timestamp should be created for each at 1-minute granularity. Where a Business\_Start\_Date contains sub-minute data the time should be **rounded up** to the next minute. A Business\_End\_Date should be rounded up to the next unit of time. (If an end date preseted at the day grain is shown as ’01-10-1989’ rounding up to the next unit of time means creating a change timestamp of ’02-01-1989’. This is done as we need to create a change timestamp immediately after the business end date, so that we show a Party record ***not*** having the “ceased” value.)

Additionally, a change timestamp is generated for each Record\_Expiry\_Timestamp rounded up to the next minute. This is done for the same reason as for the Business\_End\_Date to ensure that we have a version record in the target table showing the state of the Party after a record is ceased in the case where a ceased record is the last version for that Natural Key.

When there are 2 or more change timestamps each 1 minute apart, select the only the latest one. This is to prevent issues with the setting of Record\_Expiry\_Timestamp at a 1-minute grain and effectively takes the final “accumulated” record when multiple changes come through at the same time (or within a minute of each other).

e.g. A set of records having change timestamps of:

01-Mar-2019 10:52 A B NULL NULL NULL  
01-Mar-2019 10:53 A B X NULL NULL  
01-Mar-2019 10:54 A B X Y NULL  
01-Mar-2019 10:55 A B X Y Z

Will be treated as:

01-Mar-2019 10:55 A B X Y Z

This may result in additional change records being created, but they will be compressed in the final step of the Refined Layer build for each target table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Raw Table** | **Clean Raw Table** | **Business Start date** | **Business End date** | **Comments** |
| dss\_clients | crw\_pe\_clients |  |  |  |
| dss\_customers | crw\_pe\_customers |  |  |  |
| dss\_client\_names | crw\_pe\_client\_names |  | if date\_ceased is not >= $$HIDATE  then minutes\_add(date\_ceased,1) |  |
| dss\_external\_references | crw\_pe\_external\_references | date\_start | if date\_ceased is not >= $$HIDATE  then minutes\_add(date\_ceased,1) |  |
| dss\_client\_codes | crw\_pe\_client\_codes~~\_am~~ | date\_start | if date\_ceased is not >= $$HIDATE  then minutes\_add(date\_ceased,1) | Filtering Amalgamating/Amalgamated Codes |
| dss\_entity\_types | crw\_pe\_entity\_types |  |  |  |
| dss\_ entity \_classes | crw\_pe\_ entity \_classes |  |  |  |
| dss\_offices | crw\_pe\_offices |  |  |  |
| dss\_special\_clients\_all | crw\_pe\_special\_clients\_all |  |  |  |

## Step 5 – Create a single timeline for each Party

Create a single timeline for each Party. The party timeline is stored the Refined Layer in table rfn\_pe\_timeline. Each record in the timeline is bound in time using the following cut-off boundaries.

| **Condition** | **Cut-off Boundaries** | **Rule Logic** | **Comments** |
| --- | --- | --- | --- |
| EDW Customer has been migrated to Start | Lower cut-off date | Use the earliest EDW Party start date |  |
| Upper cut-off date | if a resurrected customer can be found in START during an incremental load  then min(record\_effective\_date) of the start version – 1 minute  else  R1 cutoff date - 1 minute  end |  |
| EDW Customer has not been migrated to Start | Lower cut-off date | Use the earliest EDW Party start date |  |
| Upper cut-off date | $$HIDATE | No cut-off is required for the upper boundary |

## Step 6a – Merge change timelines with Party timeline

Merge the change timelines from Step#4 with the single Party timeline from Step#5. Each record in the timeline has a natural key of flngcustomerkey, record\_effective\_timestamp and record\_expiry\_timestamp.

NB. When the offset for the business active timestamps (or dates) is greater than one minute, one day must be added to the ‘end’ timestamp when comparing to the timeline.record\_effective\_date.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clean Raw Table** | **Filtering Criteria** | **Business Active**  **Timestamps** | **Time-stamp Offset** | **Comments** |
| crw\_pe\_clients |  |  |  |  |
| crw\_pe\_customers |  |  |  |  |
| crw\_pe\_client\_names |  |  |  |  |
| crw\_pe\_external\_references | ***timeline.ird\_no*** = ird\_no  and ***timeline.record\_effective\_timestamp*** between   record\_effective\_timestamp and record\_expiry\_timestamp  and ***timeline.record\_effective\_timestamp***   between date\_start and date\_end | date\_start  date\_end | Day  Day |  |
| crw\_pe\_client\_codes~~\_am~~ | ***timeline.ird\_no*** = ird\_no  and ***timeline.record\_effective\_timestamp*** between   record\_effective\_timestamp and record\_expiry\_timestamp  and ***timeline.record\_effective\_timestamp***   between date\_start and date\_end | date\_start  date\_end | Day  Day |  |
| crw\_pe\_entity\_types |  |  |  |  |
| crw\_pe\_ entity \_classes |  |  |  |  |
| crw\_pe\_offices |  |  |  |  |
| crw\_pe\_special\_clients\_all |  |  |  |  |

## Step 6b – Compress the Post-Merge Timelines

After timelines are merged in Step#6a, one or more contiguous records sharing the same Natural Key may have the same values for all columns used for comparison. This step will ‘compress’ (i.e. merge) any such contiguous records using the minimum Record\_Effective\_Date and maximum Record\_Expiry\_Date for the resulting timeline.

See Step#6b for Start.

## Step 7 – Transform Rule Logic

Example SQL for Context

Example SQL is provided Appendix E for EDW Individuals, Organisations and Parties.

The “Select” column names in these SQL statements will be referred to in the “Source Table/Column” fields of the transformation rules and are intended to give context to those rules as well as a guide on how to merge all the source objects into the final Individual & Organisation data streams.

Overarching Rules/logic

1. Party, Organisation and Individual records are to be versioned to share the same keys. Therefore, any change to the versioning for a Party will result in an identical change to the versioning for either the corresponding Individual or Organisation.
2. Required date validation functions are:

| **Function Name** | **Purpose** | **Logic** |
| --- | --- | --- |
| $$is\_date\_valid$$ | Determine if the specified date (parameter: $in\_date$) is valid | FUNCTION $$check\_date\_valid$$ (in $in\_date$) return boolean  If $in\_date$ is not between ‘1880-01-01’ and $today$  then return FALSEelse return TRUE |

EDW Individuals

Filter for individuals: dss\_clients.cust\_type in (‘IND’).

See DIP - [Dimension Standards](https://teams.microsoft.com/l/file/C43DC83E-9B7D-4863-99A6-A1E2A5CFA3E1?tenantId=fb39e3e9-23a9-404e-93a2-b42a87d94f35&fileType=docx&objectUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam%2FShared%20Documents%2FRefined%20Layer%20Design%20and%20Build%2F9.%20%20Standards%2FDIP_Dimension_Standards.docx&baseUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam&serviceName=teams&threadId=19:c9ee9980de684e74badc5fe1d414bc56@thread.skype&groupId=7100c035-72cd-4625-aa78-807e71980487) for standard rows for Unknown and N/A DIP Ids.

| **Source Table/Column** | **Transform Rule/Logic** | **Target Column** | **Key** | **If Null** |
| --- | --- | --- | --- | --- |
|  | Increment by 1 for each record. This key is pervasive across Party, Organisation and Individual. | dip\_party\_sk | sk |  |
| dss\_clients.ird\_number | lookup rfn\_party\_sk\_map on ird\_number  if dip\_party\_id is returned  then dip\_party\_id  else concatenate (‘DIP-I-’, cast (ird\_number as varchar)) | dip\_party\_id | uk |  |
| dss\_clients.ird\_number  local\_start\_bankruptcy.individual\_party\_id  local\_start\_bankruptcy.ird\_number | lookup local\_start\_bankruptcy on dss\_clients.ird\_number if common\_dip\_party\_id is returned  then common\_dip\_party\_id else concatenate(‘DIP-I-’, cast(dss\_clients.ird\_number as varchar)) | common\_dip\_party\_id |  | Unknown |
|  | ‘FIRST’ | party\_origin |  |  |
| dss\_clients.ird\_number | lookup rfn\_party\_sk\_map on ird\_number  if start\_customer\_key is returned  then start\_customer\_key  else ‘Unknown’ | customer\_key |  | Unknown |
| dss\_clients.ird\_number | lpad(dss\_clients.ird\_number,9,’0’) | ird\_number |  | Unknown |
|  | ‘N/A’ | customer\_no |  | N/A |
| dss\_client\_names.client\_name\_type  dss\_client\_names.title  dss\_client\_names.first\_names  dss\_client\_names.surname | if client\_name\_type = ‘P’  then concatenate(title,’ ‘,first\_names, ‘ ’, surname)  */\* Omit spaces if a field is NULL) \*/*  else  ‘Unknown’ | legal\_name |  | Unknown |
| dss\_client\_names.client\_name\_type  dss\_client\_names.organisation\_name | If client\_name\_type = ‘T’  and organisation\_name is not null  then organisation\_name  else  NULL ~~same as Target Column legal\_name~~  (*Note: Trading name for individuals is stored in organisation\_name*) | trading\_name |  | ~~Unknown~~  NULL |
| dss\_clients.date\_created  dss\_clients.date\_of\_birth | if $$is\_date\_valid$$ (date\_of\_birth)  then date\_of\_birth  else if $$is\_date\_valid$$ (date\_created)  then date\_created  else $$LODATE | party\_start\_date |  | $$LODATE |
| dss\_clients.client\_status | if client\_status in ('C', 'D', 'I', 'M', ‘S’)  then DATE\_APPLIED  else $$HIDATE | party\_ceased\_date |  | $$HIDATE |
| dss\_clients.client\_status | if client\_status in ('C', 'D', 'I', 'M', ‘S’) then  set to “Y”  else  set to “N”  endif | party\_ceased\_flag |  | N |
| dss\_clients.client\_status  mp\_source\_code.dip\_description | lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CLIENT\_STATUS’  and source\_code\_value = client\_status if dip\_description is returned  then dip\_description | insolvency\_reason |  | N/A |
| dss\_clients.client\_status  mp\_source\_code.dip\_code\_value | lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CLIENT\_STATUS’  and source\_code\_value = client\_status if dip\_code\_value is returned  then dip\_code\_value | insolvency\_code |  | N/A |
| dss\_clients.client\_status | if client\_status in (‘B’, ’R’, ’L’, ’U’, ’V’) then  set to ‘Y’  elseif client\_status is NULL then  set to ‘N’  else  set to ‘N’  endif | insolvency\_flag |  | N |
| dss\_clients.cust\_type  mp\_source\_code.dip\_description | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CUST\_TYPE’  and code\_value = cust\_type if dip\_description is returned  then dip\_description | party\_type |  | Unknown |
| dss\_clients.cust\_type  mp\_source\_code.dip\_description | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CUST\_TYPE’  and code\_value = cust\_type if dip\_code\_value is returned  then dip\_code\_value | party\_type\_code |  | Unknown |
| dss\_clients.entity\_type  mp\_source\_code.dip\_code\_value | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘ENTITY\_TYPE’  and code\_value = entity\_type  if dip\_description is returned  then dip\_description | party\_subtype |  | Unknown |
| dss\_clients.entity\_type  mp\_source\_code.dip\_code\_value | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘ENTITY\_TYPE’  and code\_value = entity\_type if dip\_code\_value is returned  then dip\_code\_value | party\_subtype\_code |  | Unknown |
| dss\_clients.entity\_class  dss\_entity\_classes.description  ref\_lannz\_customerclass.fstrDecode2 | lookup ref\_lannz\_customerclass on fstrCustomerClass = entity\_class if fstrDecode2 is returned  then fstrDecode2  else lookup dss\_entity\_classes on entity\_class\_code = entity\_class if description is returned  then cast(description as varchar(30)) | party\_class |  | Unknown |
| dss\_clients.entity\_class | copy | party\_class\_code |  | Unknown |
|  | ‘Unknown’ | party\_segment |  | Unknown |
|  | ‘Unknown’ | party\_segment\_code |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else upper(sic\_code) | bic\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.BIC\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(bic\_code)  if BIC\_desc is returned  then BIC\_desc | BIC\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else concatenate(substr(sic\_code,1,5),’00’) | ANZSIC\_level4\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level4\_desc  *target column bic\_code* | Lookup snz\_BIC\_ANZSIC2006 using upper(*bic\_code*)  if ANZSIC\_level4\_desc is returned  then ANZSIC\_level4\_desc | ANZSIC\_level4\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else substr(sic\_code,1,4) | ANZSIC\_level3\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level3\_desc  *target column bic\_code* | Lookup snz\_BIC\_ANZSIC2006 using upper(*bic\_code*)  if ANZSIC\_level3\_desc is returned  then ANZSIC\_level3\_desc | ANZSIC\_level3\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else substr(sic\_code,1,3) | ANZSIC\_level2\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level2\_desc  *target column: bic\_code* | Lookup snz\_BIC\_ANZSIC2006 using upper(*bic\_code*)  if ANZSIC\_level2\_desc is returned  then ANZSIC\_level2\_desc | ANZSIC\_level2\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else substr(sic\_code,1,1) | ANZSIC\_level1\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level1\_desc  *target column bic\_code* | Lookup snz\_BIC\_ANZSIC2006 using upper(*bic\_code*)  if ANZSIC\_level1\_desc is returned  then ANZSIC\_level1\_desc | ANZSIC\_level1\_desc |  | Unknown |
| ref\_lancustomerlevel.fstrCustomerLevel  ref\_lancustomerlevel.fstrDecode2  *target column: security\_level\_code* | lookup ref\_lancustomerlevel   on fstrCustomerLevel = *security\_level\_code* if fstrDecode2 is returned then  update all records for this DIP\_PARTY\_ID  Set to fstrDecode2 **over all time** | security\_level |  | Unknown |
| dss\_special\_clients\_all.special\_classification  app\_tblCustomer.flngcustomerkey  app\_tblCustomer.flngDocKey  app\_tblcustomerlevel.fstrCustomerLevel  *target column: customer\_key*  *target column: ird\_number* | Lookup app\_tblcustomer on flngcustomerkey = *customer\_key* if flngDocKey is returned then  Lookup app\_tblcustomerlevel on flngDocKey  if fstrCustomerLevel is returned then  update all records for this DIP\_PARTY\_ID   Set to fstrCustomerLevel **over all time** else if special\_classification is populated then  update all records for this DIP\_PARTY\_ID   set to special\_classification **over all time**  ***NB. In the START load this field will be updated over all time with the latest value of Customer Level (becomes a SCD1 type operation over all time)*** | security\_level\_code |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.district\_office  mp\_source\_code.dip\_description | Lookup dss\_customers on ird\_number if district\_office is returned  then lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘OFFICE\_CODE’  and code\_value = district\_office  if dip\_description is returned  then dip\_description  else  dss\_customers.district\_office | district\_office |  | Unknown |
|  | Unknown | nz\_firearms\_licence\_id |  | Unknown |
|  | Unknown | os\_taxpayer\_id |  | Unknown |
|  | Unknown | os\_taxpayer\_id\_issuer |  | Unknown |
|  | Unknown | acc\_id |  | Unknown |
|  | Unknown | mbie\_client\_id |  | Unknown |
|  | ‘U’ | credit\_reported\_flag |  | U |
|  | ‘U’ | comm\_compliance\_watch\_flag |  | U |
|  | ‘U’ | suspected\_bank\_fraud\_flag |  | U |
|  | ‘U’ | suspected\_fraud\_flag |  | U |
|  | ‘U’ | security\_incident\_rptd\_flag |  | U |
|  | Unknown | safety\_risk\_level |  | Unknown |
| dss\_client\_names.title  dss\_client\_names.client\_name\_type  dss\_client\_names.location\_number | if client\_name\_type = ‘P’ and location\_number = 0 then  set to title | pref\_title |  |  |
| dss\_client\_names.first\_names  dss\_client\_names.client\_name\_type  dss\_client\_names.location\_number | if client\_name\_type = ‘P’ and location\_number = 0 then  set to first\_names | pref\_given\_name |  |  |
|  | Set to NULL  *(There are no separate middle names in FIRST)* | pref\_other\_given\_name |  |  |
| dss\_client\_names.surname  dss\_client\_names.client\_name\_type  dss\_client\_names.location\_number | if client\_name\_type = ‘P’ and location\_number = 0 then  set to surname | pref\_family\_name |  |  |
| dss\_clients.date\_of\_birth | if $$is\_date\_valid$$ (date\_of\_birth) then  set to date\_of\_birth  else  NULL | date\_of\_birth |  | NULL |
|  | NULL | date\_of\_death |  | NULL |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references.external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'BIR' if external\_id is returned  then external\_id  else ‘Unknown’ | birth\_cert\_id |  | Unknown |
|  | ‘Unknown’  (*Note: country\_code is always NULL for birth certificates*) | birth\_cert\_issuer |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'PAS' if external\_id is returned  then external\_id  else ‘Unknown’ | nz\_passport\_id |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'OPT' if external\_id is returned  then external\_id  else ‘Unknown’ | os\_passport\_id |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.country\_code  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'OPT' if country\_code is returned  then country\_code  else ‘Unknown’ | os\_passport\_issuer |  | Unknown |
|  | ‘Unknown’ | nz\_cert\_identity\_dia |  | Unknown |
|  | ‘Unknown’ | nz\_cert\_identity\_mbie |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'RTD' if external\_id is returned  then external\_id  else ‘Unknown’ | nz\_refugee\_id\_dia |  | Unknown |
|  | ‘Unknown’ | nz\_refugee\_id\_mbie |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'ETD' if external\_id is returned  then external\_id  else ‘Unknown’ | nz\_emergency\_travel\_id |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'LIC' if external\_id is returned  then external\_id  else ‘Unknown’ | drivers\_licence\_id |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.country\_code  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = 'LIC' if country\_code is returned  then country\_code  else ‘Unknown’  (*Note: country\_code is always null*) | driver\_license\_issuer |  | Unknown |
|  | ‘Unknown’ | intl\_driver\_permit\_id |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = ' HNZ ' if external\_id is returned  then external\_id  else ‘Unknown’ | nz\_hospitality\_18over\_id |  | Unknown |
|  | ‘Unknown’ | nz\_student\_id |  | Unknown |
|  | ‘Unknown’ | nz\_student\_issuer |  | Unknown |
| dss\_clients.ird\_number  dss\_external\_references.external\_id  dss\_external\_references. external\_org\_code | Lookup dss\_external\_references   on ird\_number = ird\_number  and external\_org\_code = ' DSW ' if external\_id is returned  then external\_id  else ‘Unknown’ | nz\_social\_welfare\_id |  | Unknown |
|  | ‘Unknown’ | us\_social\_security\_no |  | Unknown |
|  | ‘Unknown’ | foreign\_employment\_id |  | Unknown |
|  | ‘Unknown’ | ncp\_id |  | Unknown |
|  | ‘Unknown’ | ministry\_of\_children\_id |  | Unknown |
|  | ‘U’ | emigrated\_flag |  | U |
|  | ‘U’ | border\_alert\_flag |  | U |
|  | ‘U’ | in\_prison\_flag |  | U |
| dss\_clients.client\_status | if client\_status = ‘U’  then ‘Y’  else ‘N’ | undischarged\_bankrupt\_flag |  | N |

EDW Organisations

Filter for organisations: dss\_clients.cust\_type in (‘ORG’).

See DIP - [Dimension Standards](https://teams.microsoft.com/l/file/C43DC83E-9B7D-4863-99A6-A1E2A5CFA3E1?tenantId=fb39e3e9-23a9-404e-93a2-b42a87d94f35&fileType=docx&objectUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam%2FShared%20Documents%2FRefined%20Layer%20Design%20and%20Build%2F9.%20%20Standards%2FDIP_Dimension_Standards.docx&baseUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam&serviceName=teams&threadId=19:c9ee9980de684e74badc5fe1d414bc56@thread.skype&groupId=7100c035-72cd-4625-aa78-807e71980487) for standard rows for Unknown and N/A DIP Ids.

| **Source Table/Column** | **Transform Rule/Logic** | **Target Column** | **Key** | **Default value for Null** |
| --- | --- | --- | --- | --- |
|  | Increment by 1 for each record. This key is pervasive across Party, Organisation and Individual. | dip\_party\_sk | sk |  |
| dss\_clients.ird\_number | lookup rfn\_party\_sk\_map on ird\_number  if dip\_party\_id is returned  then dip\_party\_id  else concatenate (‘DIP-I-’, cast (ird\_number as varchar)) | dip\_party\_id | uk |  |
|  | Same as Target Column dip\_party\_id  *Note: bankruptcy only applies to Individuals* | common\_dip\_party\_id |  | Unknown |
|  | ‘FIRST’ | party\_origin |  |  |
| dss\_clients.ird\_number | lookup rfn\_party\_sk\_map on ird\_number  if start\_customer\_key is returned  then start\_customer\_key  else ‘Unknown’ | customer\_key |  | Unknown |
| dss\_clients.ird\_number | lpad(dss\_clients.ird\_number, 9, ’0’) | ird\_number |  | Unknown |
|  | ‘N/A’ | customer\_no |  | N/A |
| dss\_client\_names.client\_name\_type  dss\_client\_names.organisation\_name | if client\_name\_type = ‘P’  then organisation\_name  else ‘Unknown’ | legal\_name |  | Unknown |
| dss\_client\_names.client\_name\_type  dss\_client\_names.organisation\_name | If client\_name\_type = ‘T’  then organisation\_name  else   NULL ~~same as Target Column legal\_name~~ | trading\_name |  | ~~Unknown~~  NULL |
| dss\_clients.org\_commencement\_date | if $$is\_date\_valid$$ (org\_commencement\_date)  then org\_commencement\_date | party\_start\_date |  | $$LODATE |
| dss\_clients.client\_status | if client\_status in ('C', 'D', 'I', 'M', ‘S’)  then DATE\_APPLIED  else $$HIDATE | party\_ceased\_date |  | $$HIDATE |
| dss\_clients.client\_status | If client\_status in ('C', 'D', 'I', 'M', ‘S’) then  set to “Y”  else  set to “N”  endif | party\_ceased\_flag |  | N |
| dss\_clients.client\_status  mp\_source\_code.dip\_description | lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CLIENT\_STATUS’  and source\_code\_value = client\_status if dip\_description is returned  then dip\_description | insolvency\_reason |  | N/A |
| dss\_clients.client\_status  mp\_source\_code.dip\_code\_value | lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CLIENT\_STATUS’  and source\_code\_value = client\_status if dip\_code\_value is returned  then dip\_code\_value | insolvency\_code |  | N/A |
| dss\_clients.client\_status | if client\_status in (‘B’, ’R’, ’L’, ’U’, ’V’) then  set to ‘Y’  elseif client\_status is NULL then  set to ‘N’  else  set to ‘N’  endif | insolvency\_flag |  | N |
| dss\_clients.cust\_type  mp\_source\_code.dip\_description | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CUST\_TYPE’  and code\_value = cust\_type if dip\_description is returned  then dip\_description | party\_type |  | Unknown |
| dss\_clients.cust\_type  mp\_source\_code.dip\_description | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘CUST\_TYPE’  and code\_value = cust\_type if dip\_code\_value is returned  then dip\_code\_value | party\_type\_code |  | Unknown |
| dss\_clients.entity\_type  mp\_source\_code.dip\_code\_value | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘ENTITY\_TYPE’  and code\_value = entity\_type  if dip\_description is returned  then dip\_description | party\_subtype |  | Unknown |
| dss\_clients.entity\_type  mp\_source\_code.dip\_code\_value | Lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘ENTITY\_TYPE’  and code\_value = entity\_type if dip\_code\_value is returned  then dip\_code\_value | party\_subtype\_code |  | Unknown |
| dss\_clients.entity\_class  dss\_entity\_classes.description  ref\_lannz\_customerclass.fstrDecode2 | lookup ref\_lannz\_customerclass on fstrCustomerClass = entity\_class if fstrDecode2 is returned  then fstrDecode2  else lookup dss\_entity\_classes on entity\_class\_code = entity\_class if description is returned  then cast(description as varchar(30)) | party\_class |  | Unknown |
| dss\_clients.entity\_class | copy | party\_class\_code |  | Unknown |
|  | ‘Unknown’ | party\_segment |  | Unknown |
|  | ‘Unknown’ | party\_segment\_code |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else sic\_code | bic\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.BIC\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(bic\_code)  if BIC\_desc is returned  then BIC\_desc | BIC\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else concatenate(substr(sic\_code,1,5),’00’) | ANZSIC\_level4\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level4\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(bic\_code)  if ANZSIC\_level4\_desc is returned  then ANZSIC\_level4\_desc | ANZSIC\_level4\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else substr(sic\_code,1,4) | ANZSIC\_level3\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level3\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(bic\_code)  if ANZSIC\_level3\_desc is returned  then ANZSIC\_level3\_desc | ANZSIC\_level3\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else substr(sic\_code,1,3) | ANZSIC\_level2\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level2\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(bic\_code)  if ANZSIC\_level2\_desc is returned  then ANZSIC\_level2\_desc | ANZSIC\_level2\_desc |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.sic\_code | Lookup dss\_customers on ird\_number if sic\_code is returned  if sic\_code = ‘UNKNOWN’  then ‘Unknown’  else substr(sic\_code,1,1) | ANZSIC\_level1\_code |  | Unknown |
| snz\_BIC\_ANZSIC2006.ANZSIC\_level1\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(bic\_code)  if ANZSIC\_level1\_desc is returned  then ANZSIC\_level1\_desc | ANZSIC\_level1\_desc |  | Unknown |
| ref\_lancustomerlevel.fstrCustomerLevel  ref\_lancustomerlevel.fstrDecode2  *target column: security\_level\_code* | lookup ref\_lancustomerlevel on   fstrCustomerLevel = *security\_level\_code*  if fstrDecode2 is returned then  update all records for this DIP\_PARTY\_ID  Set to fstrDecode2 **over all time** | security\_level |  | Unknown |
| dss\_special\_clients\_all.special\_classification  app\_tblCustomer.flngcustomerkey  app\_tblCustomer.flngDocKey  app\_tblcustomerlevel.fstrCustomerLevel  *target column: customer\_key*  *target column: ird\_number* | Lookup app\_tblcustomer on flngcustomerkey = *customer\_key* if flngDocKey is returned then  Lookup app\_tblcustomerlevel on flngDocKey  if fstrCustomerLevel is returned then  update all records for this DIP\_PARTY\_ID   Set to fstrCustomerLevel **over all time** else if special\_classification is populated then  update all records for this DIP\_PARTY\_ID   set to special\_classification **over all time**  ***NB. In the START load this field will be updated over all time with the latest value of Customer Level (becomes a SCD1 type operation over all time)*** | security\_level\_code |  | Unknown |
| dss\_clients.ird\_number  dss\_customers.district\_office  mp\_source\_code.dip\_description | Lookup dss\_customers on ird\_number if district\_office is returned  then lookup mp\_source\_code   on source\_system = ‘FIRST’  and source\_code\_name = ‘OFFICE\_CODE’  and code\_value = district\_office  if dip\_description is returned  then dip\_description  else  dss\_customers.district\_office | district\_office |  | Unknown |
|  | ‘Unknown’ | os\_taxpayer\_id |  | Unknown |
|  | ‘Unknown’ | os\_taxpayer\_id\_issuer |  | Unknown |
|  | ‘U’ | credit\_reported\_flag |  | U |
|  | ‘U’ | comm\_compliance\_watch\_flag |  | U |
|  | ‘U’ | suspected\_bank\_fraud\_flag |  | U |
|  | ‘U’ | suspected\_fraud\_flag |  | U |
|  | ‘U’ | security\_incident\_rptd\_flag |  | U |
|  | ‘Unknown’ | safety\_risk\_level |  | Unknown |
| dss\_clients.org\_commencement\_date | Copy  ***Retain any Null value*** | org\_incorp\_date |  | Null |
|  | $$HIDATE | org\_cease\_date |  | $$HIDATE |
| dss\_clients.client\_status | if client\_status = ‘S’  then ‘Y’  else if client\_status is null  then ‘N’  else ‘N’ | struck\_off\_flag |  | N |
| dss\_clients.ird\_number  dss\_client\_codes.client\_code\_value | Lookup dss\_client\_codes on ird\_number if client\_code\_value is returned  if client\_code\_value =’AMD’  then ‘Y’  else if client\_code\_value is null  then ‘N’  else ‘N’  else ‘N’ | amalgamated\_flag |  | N |
| dss\_clients.ird\_number  dss\_client\_codes.client\_code\_value | Lookup dss\_client\_codes on ird\_number if client\_code\_value is returned  if client\_code\_value =’AMG’  then ‘Y’  else if client\_code\_value is null  then ‘N’  else ‘N’  else ‘N’ | amalgamating\_flag |  | N |
|  | ‘Unknown’ | nz\_cert\_incorp\_id |  | Unknown |
|  | ‘Unknown’ | os\_cert\_incorp\_id |  | Unknown |
|  | ‘Unknown’ | os\_cert\_incorp\_issuer |  | Unknown |
|  | ‘Unknown’ | nzbn |  | Unknown |
|  | ‘Unknown’ | nz\_society\_actuaries\_id |  | Unknown |
|  | ‘Unknown’ | financial\_institiution\_giin |  | Unknown |
|  | ‘Unknown’ | sponsoring\_entity\_giin |  | Unknown |
|  | ‘U’ | nil\_company\_flag |  | U |
|  | ‘U’ | charity\_org\_flag |  | U |

# Transform Rules: START

## Step 1 – Selecting Valid Values from Source Tables

The purpose of this step is to reduce the number of records from the source object to those that are relevant to the processing. This may include only extracting a subset of rows as well as excluding records that are not deemed to be valid.

The Filtering Criteria will apply to both the initial raw ingestion and subsequent raw ingestion for incremental loads. For an incremental load, changes for each source table will be identified by the corresponding change capture column.

| **Source Table** | **Filtering Criteria** | **Transform Rule/Logic** | **Change Capture Column** | **Business Active**  **Timestamps** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| app\_tblcustomer | record\_expiry\_timestamp > $$EDWDT and fstrcustomertype in (‘IND’, 'CHD',’COM’) |  | fdtmwhen\_info |  | See note 1 |
| app\_tblcustomerhistory | record\_expiry\_timestamp > $$EDWDT |  | fdtmwhen |  |  |
| app\_tblcustomerinfo | record\_expiry\_timestamp > $$EDWDT |  | fdtmwhen |  | See note 1 |
| app\_tblcustomerlevel | record\_expiry\_timestamp > $$EDWDT |  | fdtmwhen |  | See note 1 |
| app\_tblcustomerstd | record\_expiry\_timestamp > $$EDWDT |  | fdtmwhen |  | See note 1 |
| app\_tblNameRecord | record\_expiry\_timestamp > $$EDWDT  and fintProfilenumber = 1 | if fdtmeffectivefrom = fdtmeffectiveto  then fblnActive = 0 | fdtmwhen | fdtmeffectivefrom  fdtmeffectiveto | See note 1 |
| app\_tblid | record\_expiry\_timestamp > $$EDWDT and fintProfilenumber in (0,1) | if fdtmcommence = fdtmcease  then fblnActive = 0 | fdtmwhen | fdtmcommence  ftdmcease | See notes 4,5 |
| app\_tblindicator | record\_expiry\_timestamp > $$EDWDT and fintProfilenumber = 0 | if fdtmcommence = fdtmcease  then fblnActive = 0 | fdtmwhen | fdtmcommence  ftdmcease | See Derived Tables |
| app\_tblnaics | record\_expiry\_timestamp > $$EDWDT  and fintProfilenumber = 1 | if fdtmcommence = fdtmcease  then fblnActive = 0 | fdtmwhen | fdtmcommence  ftdmcease |  |
| app\_tblnz\_custsegmenthist | record\_expiry\_timestamp > $$EDWDT |  | fdtmwhen |  |  |
| app\_tblnz\_accidgovr |  |  | fdtmwhen |  | See note 2 |
| app\_tblNZ\_AccIDGEdu |  |  | fdtmwhen |  | See note 2 |
| app\_tblnz\_cstcominfo |  |  | fdtmwhen |  | See note 2 |
| app\_tblnz\_customerstd |  |  | fdtmwhen |  | See note 2 |
| ref\_lancustomertype | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| ref\_lancustomerlevel | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| ref\_lancustomersubtype | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| ref\_lanindicator | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| ref\_lannz\_customerclass | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| ref\_lannz\_customersegments | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| ref\_lanofficelocation | fstrLanguage = ‘ENG’ |  | record\_effective\_timestamp |  | See note 2 |
| local\_start\_bankruptcy |  |  |  |  | See note 2 |

**Notes:**

|  |  |
| --- | --- |
| 1 | This table only holds the current version (i.e. flngVer always 0) |
| 2 | This table is not versioned (i.e. there is no flngVer column) |
| 3 | Though records are not versioned, they have fintSeq |
| 4 | Changed 30/09/2019 to ignore future-dated IDs |

**Derived Tables:**

| **Derived Table** | **Source Table(s)** | **Derivation Rules** | **Change Capture Column** | **Comments** |
| --- | --- | --- | --- | --- |
| crw\_ps\_tblindicator\_insolv | app\_tblindicator | **Filter from table app\_tblindicator:**  record\_expiry\_timestamp > $$EDWDT and fintProfilenumber = 0 and fdtmcommence <= fdtmcease and fstrindicator in ('INSCNA','INSCBK', 'INSCBF', 'UNDSCH', 'PRVBNK', 'INSCLQ', 'INSCLF', 'INSPRG', 'INSFIN', 'INSMTR', 'INSCRC', 'INSCVA', 'RECEIV')  **Derive *insolvency\_level*:**  Lookup mp\_insolvency\_rankings  on indicator\_group = ‘INSOLVENCY’  and source\_system = ‘START’  and indicator = fstrIndicator if rank is returned  then rank else 0 (i.e. zero) | fdtmwhen |  |
| crw\_ps\_tblindicator\_sec | app\_tblindicator | **Filter from table app\_tblindicator:**  record\_expiry\_timestamp > $$EDWDT and fintProfilenumber = 0 and fdtmcommence <= fdtmcease and fstrindicator in ('LOWRSK','MEDRSK','HGHRSK')  **Derive *safety\_risk\_level*:**  If fstrindicator = 'HGHRSK'  then safety\_risk\_level = 1 else if fstrindicator = 'MEDRSK'  then safety\_risk\_level = 2 else (fstrindicator = 'LOWRSK')  then safety\_risk\_level = 3 | fdtmwhen |  |
| crw\_ps\_tblid\_ird | app\_tblid | **Filter from table app\_tblindicator:**  record\_expiry\_timestamp > $$EDWDT and fintProfilenumber in (0,1) and fdtmcommence <= fdtmcease and fstrtype = ‘IRD’ | fdtmwhen |  |
| rfn\_ps\_bankruptcy | local\_start\_bankruptcy  crw\_ps\_tblid\_ird | **Derive table rfn\_ps\_bankruptcy:**  See SQL in Appendix F |  |  |

## Step 2 – De-duplication of Records

The purpose of this step is to remove duplicates over time so that the resulting rows for each natural key are contiguous and non-overlapping in time. The default de-duplication process (see Appendix A) is to be following unless explicitly stated otherwise.

NB. Active flag should always be the first tie-breaker for de-duplication

| **Source Table** | **De-duplication Rule** | **Natural Key** | **Business Active**  **Timestamps** | **Tie-Breaker(s)** |
| --- | --- | --- | --- | --- |
| app\_tblcustomer | No de-duplication required | flngCustomerKey |  |  |
| app\_tblcustomerhistory | Use the default de-duplication process (see Appendix A) | flngCustomerKey |  |  |
| app\_tblcustomerinfo | No de-duplication required | flngCustomerKey |  |  |
| app\_tblcustomerlevel | No de-duplication required | flngDocKey |  |  |
| app\_tblcustomerstd | No de-duplication required | flngDocKey |  |  |
| app\_tblNameRecord | Use the default de-duplication process (see Appendix A) | flngCustomerKey fstrNameType | fdtmeffectivefrom  fdtmeffectiveto | Rank:   1. fblnactive (1 before 0) 2. record\_expiry\_timestamp (latest first) 3. fdtmeffectiveto (latest first) 4. fdtmeffectivefrom (earliest first) 5. fl64namekey (highest first) 6. flngverlast (highest first) |
| app\_tblid | Use the default de-duplication process (see Appendix A) | flngCustomerKey  fstridtype | fdtmcommence  ftdmcease | Rank:   1. fblnactive (1 before 0) 2. Number\_of\_valid\_seconds( highest first) 3. fdtmwhen(latest first) 4. fblnDefault (1 before 0) 5. flngidkey (highest first) 6. flngverlast (highest first) |
| app\_tblid\_ird | Use the default de-duplication process (see Appendix A) | flngCustomerKey  fstridtype | fdtmcommence  ftdmcease | Rank:   1. fblnactive (1 before 0) 2. Number\_of\_valid\_seconds (highest first) 3. Fdtmwhen (latest first) 4. fblnDefault (1 before 0) 5. flngidkey (highest first) 6. flngverlast (highest first) |
| app\_tblindicator | Use the default de-duplication process (see Appendix A) | flngCustomerKey fstrindicator | fdtmcommence  ftdmcease | Rank:   1. fblnactive (1 before 0) 2. Number\_of\_valid\_seconds (highest first) 3. Fdtmwhen (latest first) 4. flngindicatorkey (highest first) 5. flngverlast (highest first) |
| app\_tblindicator\_insolv | Use the default de-duplication process (see Appendix A) | flngCustomerKey | fdtmcommence  ftdmcease | Rank:   1. fblnactive (1 before 0) 2. Number\_of\_valid\_seconds (highest first) 3. insolvency\_level (highest first) 4. Fdtmwhen (latest first) 5. flngindicatorkey (highest first) 6. flngverlast (highest first) |
| app\_tblindicator\_sec | Use the default de-duplication process (see Appendix A) | flngCustomerKey | fdtmcommence  ftdmcease | Rank:   1. fblnactive (1 before 0) 2. Number\_of\_valid\_seconds (highest first) 3. safety\_risk\_level (1=HGHRSK first) 4. Fdtmwhen (latest first) 5. flngindicatorkey (highest first) 6. flngverlast (highest first) |
| app\_tblnaics | Use the default de-duplication process (see Appendix A) | flngCustomerKey | fdtmcommence  ftdmcease | Rank:   1. fblnactive (1 before 0) 2. Number\_of\_valid\_seconds (highest first) 3. fdtmwhen 4. fblnDefault (1 before 0) 5. flngNAICSkey (highest first) 6. flngverlast (highest first) |
| app\_tblnz\_custsegmenthist | No de-duplication required | flngDocKey |  |  |
| app\_tblnz\_accidgovr | No de-duplication required | flngDocKey |  |  |
| app\_tblNZ\_AccIDGEdu | No de-duplication required | flngDocKey |  |  |
| app\_tblnz\_cstcominfo | No de-duplication required | flngDocKey |  |  |
| app\_tblnz\_customerstd | No de-duplication required | flngDocKey |  |  |
| ref\_lancustomertype | No de-duplication required | fstrLanguage fstrCustomerType |  |  |
| ref\_lancustomerlevel | No de-duplication required | fstrLanguage fstrCustomerLevel |  |  |
| ref\_lancustomersubtype | No de-duplication required | fstrLanguage fstrCustomerSubType |  |  |
| ref\_lanindicator | No de-duplication required | fstrLanguage fstrIndicator |  |  |
| ref\_lannz\_customerclass | No de-duplication required | fstrLanguage fstrCustomerClass |  |  |
| ref\_lannz\_customersegments | No de-duplication required | fstrLanguage fstrSegment |  |  |
| ref\_lanofficelocation | No de-duplication required | fstrLanguage fstrSDistrictOffice |  |  |

## Step 3 – Compressing the timeline

After de-duplicating records in Step 2, one or more contiguous records sharing the same Natural Key may have the same values for all columns used for comparison. This step will ‘compress’ (i.e. merge) any such contiguous records using the minimum Record\_Effective\_Date and maximum Record\_Expiry\_Date for the resulting timeline.

NB. Business active timestamps/dates should always be included in the “columns to compare”.

| **Source Table** | **Natural Key** | **Business Active**  **Timestamps** | **Columns to Compare** | **Dates for Timelines** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| app\_tblcustomer | flngCustomerKey |  | fstrCustomerType  fdtmCommence  fdtmCease  fdtmCreated | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblcustomerhistory | flngCustomerKey |  | fstrCustomerTypeFrom  fstrCustomerTypeTo  flngDocKeyTo | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblcustomerinfo | flngCustomerKey |  | fstrDistrictOffice  fdtmCommence  fdtmCease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblcustomerlevel | flngDocKey |  | fstrCustomerLevel | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblcustomerstd | flngDocKey |  | fdtmdob  fdtmdod | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblNameRecord | flngCustomerKey fstrNameType | fdtmeffectivefrom  fdtmeffectiveto | fstrNameType  fstrlistformatname fstrtitle fstrfirstname fstrmiddlename fstrlastname  fblnactive fdtmeffectivefrom  fdtmeffectiveto | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblid | flngCustomerKey  fstridtype | fdtmcommence  ftdmcease | fstrId fblnactive fdtmcommence  ftdmcease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblid\_ird | flngCustomerKey  fstridtype | fdtmcommence  ftdmcease | fstrId fblnactive fdtmcommence  ftdmcease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblindicator | flngCustomerKey fstrindicator | fdtmcommence  ftdmcease | fstrindicator fblnactive fdtmcommence  ftdmcease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblindicator\_insolv | flngCustomerKey | fdtmcommence  ftdmcease | fstrindicator fblnactive fdtmcommence  ftdmcease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblindicator\_sec | flngCustomerKey | fdtmcommence  ftdmcease | fstrindicator fblnactive fdtmcommence  ftdmcease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblnaics | flngCustomerKey | fdtmcommence  ftdmcease | fstNAICS  fstrNAICSVersion  fblnactive fdtmcommence  ftdmcease | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblnz\_custsegmenthist | flngDocKey |  | fstrCustomerSegment | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblnz\_accidgovr | flngDocKey |  | fstrIssuingCountry | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblNZ\_AccIDGEdu | flngDocKey |  | fstrEducationProvider | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblnz\_cstcominfo | flngDocKey |  | fblnNilCompany | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| app\_tblnz\_customerstd | flngDocKey |  | fstrCustomerSubType  fblnSpecialTaxResidency  fstrCurrentTaxResidency  fstrCustomerClass  fstrCustomerSubtypeClass | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lancustomertype | fstrCustomerType |  | fstrDecode2 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lancustomerlevel | fstrCustomerLevel |  | fstrDecode2 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lancustomersubtype | fstrCustomerSubType |  | fstrDecode2 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lanindicator | fstrIndicator |  | fstrDecode1 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lannz\_customerclass | fstrCustomerClass |  | fstrDecode2 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lannz\_customersegments | fstrSegment |  | fstrDecode2 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |
| ref\_lanofficelocation | fstrSDistrictOffice |  | fstrDecode1 | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp |  |

## Step 4 – Combining all change timelines

The purpose of this step is to create a UNION of all change timestamps for the Party data that will become the basis for all change records created. These timelines are stored in the Refined Layer in table rfn\_pe\_all\_dates. Each record in the timeline is bounded in time from record\_effective\_timestamp to record\_expiry\_timestamp.

A change timestamp is generated for each Record\_Effective\_Timestamp at a 1-minute granularity, additionally where a Business Start Date and/or Business End date are listed below, a change timestamp should be created for each at 1-minute granularity. Where a Business\_Start\_Date contains sub-minute data the time should be **rounded up** to the next minute. A Business\_End\_Date should be rounded up to the next unit of time. (If an end date preseted at the day grain is shown as ’01-10-1989’ rounding up to the next unit of time means creating a change timestamp of ’02-01-1989’. This is done as we need to create a change timestamp immediately after the business end date, so that we show a Party record ***not*** having the “ceased” value.)

When there are 2 or more change timestamps each 1 minute apart, select the only the latest one. This is to prevent issues with the setting of Record\_Expiry\_Timestamp at a 1-minute grain and effectively takes the final “accumulated” record when multiple changes come through at the same time (or within a minute of each other).  
e.g. A set of records having change timestamps of:

01-Mar-2019 10:52 A B NULL NULL NULL  
01-Mar-2019 10:53 A B X NULL NULL  
01-Mar-2019 10:54 A B X Y NULL  
01-Mar-2019 10:55 A B X Y Z

Will be treated as:

01-Mar-2019 10:55 A B X Y Z

This may result in additional change records being created, but they will be compressed in the final step of the Refined Layer build for each target table.

NB. Only tables which are combined to the main timeline are shown here, reference data (lookup) are ignored.

| **Raw Table** | **Clean Raw Table** | **Business Start date** | **Business End date** | **Comments** |
| --- | --- | --- | --- | --- |
| app\_tblcustomer | crw\_ps\_tblcustomer | fdtmcreated |  | The customer timeline commences on fdtmCommence, however it is worth creating a change record on the date that the record was first added to START |
| app\_tblcustomerhistory | crw\_ps\_tblcustomerhistory |  |  |  |
| app\_tblcustomerinfo | crw\_ps\_tblcustomerinfo |  |  |  |
| app\_tblcustomerlevel | crw\_ps\_tblcustomerlevel |  |  |  |
| app\_tblcustomerstd | crw\_ps\_tblcustomerstd |  |  |  |
| app\_tblid | crw\_ps\_tblid | fdtmCommence | if fdtmCease is not = $$HIDATE  then days\_add(fdtmCease,1) |  |
| app\_tblid | crw\_ps\_tblid\_ird | fdtmCommence | if fdtmCease is not = $$HIDATE  then days\_add(fdtmCease,1) |  |
| app\_tblindicator | crw\_ps\_tblindicator | fdtmCommence | if fdtmCease is not = $$HIDATE  then minutes\_add(fdtmCease,1) |  |
| app\_tblindicator | crw\_ps\_tblindicator\_insolv | fdtmCommence | if fdtmCease is not = $$HIDATE  then minutes\_add(fdtmCease,1) |  |
| app\_tblindicator | crw\_ps\_tblindicator\_sec | fdtmCommence | if fdtmCease is not = $$HIDATE  then minutes\_add(fdtmCease,1) |  |
| app\_tblnaics | crw\_ps\_tblnaics | fdtmCommence | if fdtmCease is not = $$HIDATE  then minutes\_add(fdtmCease,1) |  |
| app\_tblNameRecord | crw\_ps\_tblnamerecord | fdtmeffectivefrom | if fdtmeffectiveto is not = $$HIDATE  then days\_add(fdtmeffectiveto,1) | fdtmEffectiveFrom and fdtmEffectiveTo are already driving the Record\_Effective\_Timestamp and Record\_Expiry\_Timestamp |
| app\_tblNZ\_AccIDGEdu | crw\_ps\_tblnz\_accidgedu |  |  |  |
| app\_tblnz\_accidgovr | crw\_ps\_tblnz\_accidgovr |  |  |  |
| app\_tblnz\_cstcominfo | crw\_ps\_tblnz\_cstcominfo |  |  |  |
| app\_tblnz\_customerstd | crw\_ps\_tblnz\_customerstd |  |  |  |
| app\_tblnz\_custsegmenthist | crw\_ps\_tblnz\_custsegmenthist |  |  |  |
| ref\_lancustomerlevel | crw\_ps\_lancustomerlevel |  |  |  |
| ref\_lancustomersubtype | crw\_ps\_lancustomersubtype |  |  |  |
| ref\_lancustomertype | crw\_ps\_lancustomertype |  |  |  |
| ref\_lanindicator | crw\_ps\_lan\_indicator |  |  |  |
| ref\_lannz\_customerclass | crw\_ps\_lannz\_customerclass |  |  |  |
| ref\_lannz\_customersegments | crw\_ps\_lannz\_customersegments |  |  |  |
| ref\_lanofficelocation | crw\_ps\_lanofficelocation |  |  |  |

## Step 5 – Create a single timeline for each Party

Create a single timeline for each Party. The party timeline is stored the Refined Layer in table rfn\_ps\_timeline. Each record in the timeline is bound in time using the following cut-off boundaries.

| **Condition** | **Cut-off Boundaries** | **Rule Logic** | **Comments** |
| --- | --- | --- | --- |
| All Start customers | Lower cut-off date | Use the later of:   1. the earliest Start party\_start\_date 2. default interleaving date |  |
| Upper cut-off date | $$HIDATE | No cut-off is required for the upper boundary |

## Step 6a – Merge change timelines with Party timeline

Merge the change timelines from Step#4 with the single Party timeline from Step#5. Each record in the timeline has a natural key of flngcustomerkey, record\_effective\_timestamp and record\_expiry\_timestamp.

NB. When the offset for the business active timestamps (or dates) is greater than one minute, one day must be added to the ‘end’ timestamp when comparing to the timeline.record\_effective\_date.

| **Clean Raw Table** | **Filtering Criteria** | **Business Active**  **Timestamps** | **Time-stamp Offset** | **Comments** |
| --- | --- | --- | --- | --- |
| crw\_ps\_tblcustomer |  |  |  |  |
| crw\_ps\_tblcustomerhistory |  |  |  |  |
| crw\_ps\_tblcustomerinfo |  |  |  |  |
| crw\_ps\_tblcustomerlevel |  |  |  |  |
| crw\_ps\_tblcustomerstd |  |  |  |  |
| crw\_ps\_tblid | ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  and ***timeline.record\_expiry\_timestamp*** >= fdtmcommence  and days\_add(***timeline.record\_effective\_timestamp***,-1) <= fdtmcease  and fblnactive = 1 | fdtmcommence  ftdmcease | Day | See note 1 |
| crw\_ps\_tblid\_ird | ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  and ***timeline.record\_expiry\_timestamp*** >= fdtmcommence  and days\_add(***timeline.record\_effective\_timestamp***,-1) <= fdtmcease  and fblnactive = 1 | fdtmcommence  ftdmcease | Day | See notes 1, 2 |
| crw\_ps\_tblindicator | ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  ***timeline.record\_effective\_timestamp***   between fdtmcommence and fdtmcease  and fblnactive = 1 | fdtmcommence  ftdmcease | Millisecond | See note 1 |
| crw\_ps\_tblindicator\_insolv | ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  ***timeline.record\_effective\_timestamp***   between fdtmcommence and fdtmcease  and fblnactive = 1 | fdtmcommence  ftdmcease | Millisecond | See note 1 |
| crw\_ps\_tblindicator\_sec | ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  ***timeline.record\_effective\_timestamp***   between fdtmcommence and fdtmcease  and fblnactive = 1 | fdtmcommence  ftdmcease | Millisecond | See note 1 |
| crw\_ps\_tblnaics | ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  ***timeline.record\_effective\_timestamp***   between fdtmcommence and fdtmcease  and fblnactive = 1 | fdtmcommence  ftdmcease | Millisecond |  |
| crw\_ps\_tblNameRecord | If fstrnametype = 'LGL' then  ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp else  ***timeline.flngcustomerkey*** = flngcustomerkey  and ***timeline.record\_effective\_timestamp*** between record\_effective\_timestamp and record\_expiry\_timestamp  and ***timeline.record\_effective\_timestamp*** >= fdtmeffectivefrom  and days\_add(***timeline.record\_effective\_timestamp***,-1) <= fdtmeffectiveto  and fblnactive = 1 | fdtmeffectivefrom  fdtmeffectiveto | Day | See note 2 |
| crw\_ps\_tblNZ\_AccIDGEdu |  |  |  |  |
| crw\_ps\_tblnz\_accidgovr |  |  |  |  |
| crw\_ps\_tblnz\_cstcominfo |  |  |  |  |
| crw\_ps\_tblnz\_customerstd |  |  |  |  |
| crw\_ps\_tblnz\_custsegmenthistory |  |  |  |  |
| crw\_ps\_lancustomerlevel |  |  |  |  |
| crw\_ps\_lancustomersubtype |  |  |  |  |
| crw\_ps\_lancustomertype |  |  |  |  |
| crw\_ps\_lanindicator |  |  |  |  |
| crw\_ps\_lannz\_customerclass |  |  |  |  |
| crw\_ps\_lannz\_customersegments |  |  |  |  |
| crw\_ps\_lanofficelocation |  |  |  |  |

**Notes:**

|  |  |
| --- | --- |
| 1 | ‘Inactive’ (i.e. fblnActive=0) records with flngver=0 are to be ignored in this step. They are required for Step#5 to ensure that the record\_expiry\_timestamp is set correctly for contiguous records and must now be removed. |
| 2 | When the ‘offset’ for the business active timestamps (or dates) is greater than one minute, one unit of the ‘offset’ value must be subtracted from the timeline.record\_effective\_date when compared to the record’s ‘end’ timestamp. |

## Step 6b – Compress the Post-Merge Timelines

After timelines are merged in Step#6a, one or more contiguous records sharing the same Natural Key may have the same values for all columns used for comparison. This step will ‘compress’ (i.e. merge) any such contiguous records using the minimum Record\_Effective\_Date and maximum Record\_Expiry\_Date for the resulting timeline.

| **Source Table** | **Natural Key** | **Columns to Compare** | **Columns for Timelines** | **Time-stamp Offset** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| rfn\_ps\_individual | dip\_party\_id | dip\_party\_id  common\_dip\_party\_id  party\_origin  customer\_key  ird\_number  customer\_no  legal\_name  trading\_name  party\_start\_date  party\_ceased\_date  party\_ceased\_flag  insolvency\_reason  insolvency\_code  insolvency\_flag  party\_type  party\_type\_code  party\_subtype  party\_subtype\_code  party\_class  party\_class\_code  party\_segment  party\_segment\_code  BIC\_code  BIC\_desc  ANZSIC\_level4\_code  ANZSIC\_level4\_desc  ANZSIC\_level3\_code  ANZSIC\_level3\_desc  ANZSIC\_level2\_code  ANZSIC\_level2\_desc  ANZSIC\_level1\_code  ANZSIC\_level1\_desc  security\_level  security\_level\_code  district\_office  nz\_firearms\_licence\_id  os\_taxpayer\_id  os\_taxpayer\_id\_issuer  acc\_id  mbie\_client\_id  credit\_reported\_flag  comm\_compliance\_watch\_flag  suspected\_bank\_fraud\_flag  suspected\_fraud\_flag  security\_incident\_rptd\_flag  safety\_risk\_level  pref\_title  pref\_given\_name  pref\_other\_given\_name  pref\_family\_name  date\_of\_birth  date\_of\_death  birth\_cert\_id  birth\_cert\_issuer  nz\_passport\_id  os\_passport\_id  os\_passport\_issuer  nz\_cert\_identity\_dia  nz\_cert\_identity\_mbie  nz\_refugee\_id\_dia  nz\_refugee\_id\_mbie  nz\_emergency\_travel\_id  driver\_license\_id  driver\_license\_issuer  intl\_driver\_permit\_id  nz\_hospitality\_18over\_id  nz\_student\_id  nz\_student\_issuer  nz\_social\_welfare\_id  us\_social\_security\_no  foreign\_employment\_id  ncp\_id  ministry\_of\_children\_id  emigrated\_flag  border\_alert\_flag  in\_prison\_flag  undischarged\_bankrupt\_flag | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp | Minute |  |
| rfn\_ps\_organisation | dip\_party\_id | dip\_party\_id  common\_dip\_party\_id  party\_origin  customer\_key  ird\_number  customer\_no  legal\_name  trading\_name  party\_start\_date  party\_ceased\_date  party\_ceased\_flag  insolvency\_reason  insolvency\_code  insolvency\_flag  party\_type  party\_type\_code  party\_subtype  party\_subtype\_code  party\_class  party\_class\_code  party\_segment  party\_segment\_code  BIC\_code  BIC\_desc  ANZSIC\_level4\_code  ANZSIC\_level4\_desc  ANZSIC\_level3\_code  ANZSIC\_level3\_desc  ANZSIC\_level2\_code  ANZSIC\_level2\_desc  ANZSIC\_level1\_code  ANZSIC\_level1\_desc  security\_level  security\_level\_code  district\_office  os\_taxpayer\_id  os\_taxpayer\_id\_issuer  credit\_reported\_flag  comm\_compliance\_watch\_flag  suspected\_bank\_fraud\_flag  suspected\_fraud\_flag  security\_incident\_rptd\_flag  safety\_risk\_level  org\_incorp\_date  org\_cease\_date  struck\_off\_flag  amalgamated\_flag  amalgamating\_flag  nz\_cert\_incorp\_id  os\_cert\_incorp\_id  os\_cert\_incorp\_issuer  nzbn  nz\_society\_actuaries\_id  financial\_institution\_giin  sponsoring\_entity\_giin  nil\_company\_flag  charity\_org\_flag | Record\_Effective\_Timestamp Record\_Expiry\_Timestamp | Minute |  |

## Step 7 – Transform Rule Logic

Example SQL for Context

Example SQL is provided Appendix F for Start Individuals, Organisations and Parties.

The “Select” column names in these SQL statements will be referred to in the “Source Table/Column” fields of the transformation rules and are intended to give context to those rules as well as a guide on how to merge all the source objects into the final Individual & Organisation data streams.

Overarching Rules/logic

1. Party, Organisation and Individual records are to be versioned to share the same keys. Therefore, any change to the versioning for a Party will result in an identical change to the versioning for either the corresponding Individual or Organisation.
2. Required date validation functions are:

| **Function Name** | **Purpose** | **Logic** |
| --- | --- | --- |
| $$is\_date\_valid$$ | Determine if the specified date (parameter: $in\_date$) is valid | FUNCTION $$check\_date\_valid$$ (in $in\_date$) return boolean  If $in\_date$ is not between ‘1880-01-01’ and $today$  then return FALSEelse return TRUE |

Start Individuals

Filter for individuals: app\_tblCustomer.fstrcustomertype in (‘IND’, 'CHD').

See DIP - [Dimension Standards](https://teams.microsoft.com/l/file/C43DC83E-9B7D-4863-99A6-A1E2A5CFA3E1?tenantId=fb39e3e9-23a9-404e-93a2-b42a87d94f35&fileType=docx&objectUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam%2FShared%20Documents%2FRefined%20Layer%20Design%20and%20Build%2F9.%20%20Standards%2FDIP_Dimension_Standards.docx&baseUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam&serviceName=teams&threadId=19:c9ee9980de684e74badc5fe1d414bc56@thread.skype&groupId=7100c035-72cd-4625-aa78-807e71980487) for standard rows for Unknown and N/A DIP Ids.

| **Source Table/Column** | **Transform Rule/Logic** | **Target Column** | **Key** | **Default value for Null** |
| --- | --- | --- | --- | --- |
|  | Increment by 1 for each record. Starting value must exceed the highest corresponding value for EDW. This key is pervasive across Party, Organisation and Individual. | dip\_party\_sk | sk |  |
| app\_tblcustomer.flngCustomerKey  rfn\_party\_sk\_map.dip\_party\_id | rfn\_party\_sk\_map.dip\_party\_id | dip\_party\_id | uk |  |
| app\_tblid.fstrId  app\_tblid.fstrIdType  local\_start\_bankruptcy.ird\_no  local\_start\_bankruptcy.common\_dip\_party\_id | If fstrIdType = ‘IRD’  then lookup rfn\_ps\_bankruptcy on ird\_no = fstrId  if common\_dip\_party\_id is returned  then common\_dip\_party\_id  else concatenate(‘DIP-C-’, flngCustomerKey) else concatenate(‘DIP-C-’, flngCustomerKey) | common\_dip\_party\_id |  |  |
|  | ‘START’ | party\_origin |  |  |
| app\_tblcustomer.flngCustomerKey | Copy | customer\_key |  |  |
| app\_tblid.fstrId  app\_tblid.fstrIdType  app\_tblid.record\_active\_flag | If fstrIdType = ‘IRD’  then lpad(fstrId, 9, ’0’) else ‘Unknown’ | ird\_number |  | Unknown |
| app\_tblid.fstrId  app\_tblid.fstrIdType | If fstrIdType = ‘CST’  then fstrId else ‘N/A’ | customer\_no |  | N/A |
| app\_tblNameRecord.fstrtitle  app\_tblNameRecord.fstrfirstname  app\_tblNameRecord.fstrmiddlename  app\_tblNameRecord.fstrlastname  app\_tblNameRecord.fstrNameType | if exists fstrNameType = 'LGL'   then concatenate(fstrtitle, ‘ ’, fstrfirstname, ‘ ‘, fstrmiddlename, ‘ ‘, fstrlastname)  Eliminate spurious spaces if some fields are NULL  else ‘Unknown’ | legal\_name |  | Unknown |
| app\_tblNameRecord.fstrListFormatName  app\_tblNameRecord.fstrNameType | if fstrNameType = 'DBACST'   then fstrListFormatName else  NULL | trading\_name |  | NULL |
| app\_tblcustomerstd.fdtmdob  app\_tblCustomer.fdtmcreated  sapp\_tblCustomer.fdtmcommence | if $$is\_date\_valid$$ (fdtmcommence)   then trunc(fdtmcommence,’DDD’) else if $$is\_date\_valid$$ (fdtmdob)   then trunc(fdtmdob,’DDD’)  else if $$is\_date\_valid$$ (fdtmcreated)   then trunc(fdtmcreated,’DDD’) else $$LODATE | party\_start\_date |  | $$LODATE |
| app\_tblCustomer.fdtmcease | if $$is\_date\_valid$$ (fdtmcease)   then trunc(fdtmcease,’DDD’) else $$HIDATE | party\_ceased\_date |  | $$HIDATE |
|  | if (target column) party\_ceased\_date < $$HIDATE  then ‘Y’ else ‘N’ | party\_ceased\_flag |  |  |
| app\_tblindicator\_insolv.fstrIndicator  ref\_lanindicator.fstrdecode1 | lookup ref\_lanindicator on fstrIndicator if fstrdecode1 is returned  then fstrdecode1 | insolvency\_reason |  | N/A |
| app\_tblindicator\_insolv.fstrIndicator | Copy | insolvency\_code |  | N/A |
| app\_tblindicator\_insolv.fstrindicator | if fstrIndicator is populated  then ‘Y’ else ‘N’ | insolvency\_flag |  | N |
| app\_tblCustomer.fstrCustomerType  app\_tblCusttomerHistory.fstrCustomerTypeTo ref\_lancustomertype.fstrdecode2 | Lookup ref\_lancustomertype   on coalesce(fstrCustomerTypeTo, fstrCustomerType) if fstrdecode2 is returned  then fstrdecode2 | party\_type |  | Unknown |
| app\_tblCustomer.fstrCustomerType  app\_tblCusttomerHistory.fstrCustomerTypeTo | coalesce(fstrCustomerTypeTo, fstrCustomerType) | party\_type\_code |  | Unknown |
| app\_tblCustomer.flngDocKey  app\_tblnz\_custsubtypehistory.fstrCustomerSubType ref\_lancustomersubtype.fstrddecode2 | Lookup app\_tblnz\_customerstd on flngDocKey if fstrcustomersubtype is returned  then lookup ref\_lancustomersubtype on fstrcustomersubtype  if fstrddecode2 is returned  then fstrddecode2 | party\_subtype |  | Unknown |
| app\_tblCustomer.flngDocKey  app\_tblnz\_custsubtypehistory.fstrCustomerSubType | Lookup app\_tblnz\_customerstd on flngDocKey if fstrcustomersubtype is returned  then fstrCustomerSubType | party\_subtype\_code |  | Unknown |
| app\_tblCustomer.flngDocKey  app\_tblnz\_customerstd.fstrCustomerClass  app\_tblnz\_customerstd. fstrCustomerSubtypeClass ref\_lannz\_customerclass.fstrDecode2 | Select fstrDecode2  From ref\_lannz\_customerclass  where derived value of Party\_Class\_Code = fstrCustomerClass | party\_class |  | Unknown |
| app\_tblCustomer.flngDocKey  app\_tblnz\_customerstd.fstrCustomerClass  app\_tblnz\_customerstd. fstrCustomerSubtypeClass | Select Coalesce(fstrCustomerSubtypeClass, fstrCustomerClass) as Party\_Class\_Code  From app\_tblnz\_customerstd  where app\_tblCustomer.flngDocKey = app\_tblnz\_customerstd flngDocKey | party\_class\_code |  | Unknown |
| app\_tblnz\_custsegmenthistory.fstrCustomerSegment  ref\_lannz\_customersegments.fstrSegment  ref\_lannz\_customersegments.fstrDecode2 | if fstrCustomerSegment is null then  ‘Unknown’ else lookup ref\_lannz\_customersegments   on fstrSegment = fstrCustomerSegment  if fstrDecode2 is returned   then fstrDecode2  else ‘Unknown’  else ‘Unknown’ | party\_segment |  | Unknown |
| app\_tblnz\_custsegmenthistory.fstrCustomerSegment | if fstrCustomerSegment is null then  ‘Unknown’ else concatenate(‘SEG’, fstrCustomerSegment) | party\_segment\_code |  | Unknown |
| app\_tblnaics.fstrnaics | Set to UPPER(fstrnaics) | BIC\_code |  | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.BIC\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if BIC\_desc is returned  then BIC\_desc | BIC\_desc |  | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Concatenate(Substr(fstrnaics,1,5), ‘00’)  Else  Set to “Unknown” | ANZSIC\_level4\_code |  | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level4\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level4\_desc is returned  then ANZSIC\_level4\_desc | ANZSIC\_level4\_desc |  | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Substr(fstrnaics,1,4)  Else  Set to “Unknown” | ANZSIC\_level3\_code |  | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level3\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level3\_desc is returned  then ANZSIC\_level3\_desc | ANZSIC\_level3\_desc |  | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Substr(fstrnaics,1,3)  Else  Set to “Unknown” | ANZSIC\_level2\_code |  | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level2\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level2\_desc is returned  then ANZSIC\_level2\_desc | ANZSIC\_level2\_desc |  | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Substr(fstrnaics,1,1)  Else  Set to “Unknown” | ANZSIC\_level1\_code |  | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level1\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level1\_desc is returned  then ANZSIC\_level1\_desc | ANZSIC\_level1\_desc |  | Unknown |
| app\_tblcustomerlevel.fstrCustomerLevel  ref\_lancustomerlevel.fstrDecode2  *target column: security\_level\_code* | lookup ref\_lancustomerlevel   on fstrCustomerLevel = *security\_level\_code*  if fstrDecode2 is returned then  update all records for this DIP\_PARTY\_ID  Set to fstrDecode2 **over all time** | security\_level |  | Unknown |
| app\_tblCustomer.flngDocKey  app\_tblcustomerlevel.fstrCustomerLevel  dss\_special\_clients\_all.ird\_number  dss\_special\_clients\_all.special\_classification  *target column: ird\_number* | Lookup app\_tblcustomerlevel on flngDocKey if fstrCustomerLevel is returned then  update all records for this DIP\_PARTY\_ID   Set to fstrCustomerLevel **over all time**  else lookup dss\_special\_clients\_all on *ird\_number*  if special\_classification is returned then  update all records for this DIP\_PARTY\_ID   set to special\_classification **over all time** | security\_level\_code |  | Unknown |
| app\_tblcustomer.flngCustomerKey  app\_tblcustomerinfo.fstrDistrictOffice  ref\_lanofficelocation.fstrDecode1 | Lookup app\_tblcustomerinfo on flngCustomerKey if fstrDistrictOffice is returned  then lookup ref\_lanofficelocation on fstrDistrictOffice  if fstrDecode1 is returned   then fstrDecode1 | district\_office |  | Unknown |
| app\_tblid.fstrId  app\_tblid.fstrIdType | if fstrIdType = ' NZFRAM '  then fstrId else ‘Unknown’ | nz\_firearms\_licence\_id |  | Unknown |
| app\_tblid.fstrId  app\_tblid.fstrIdType | if fstrIdType = ' TXIDNM '  then fstrId else ‘Unknown’ | os\_taxpayer\_id |  | Unknown |
| app\_tblid.fstrIdType  app\_tblid.flngdockey  app\_tblnz\_accidgovr.fstrIssuingCountry | if fstrIdType = ' TXIDNM '  lookup app\_tblnz\_accidgovr on flngdockey  if fstrIssuingCountry is returned   then fstrIssuingCountry | os\_taxpayer\_id\_issuer |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = ' ACOMPC'  then fstrId else ‘Unknown’ | acc\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = ' MBCLNM '  then fstrId else ‘Unknown’ | mbie\_client\_id |  | Unknown |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' CRDRPD '  then ‘Y’ else ‘N’ | credit\_reported\_flag |  |  |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' CSCCOM '  then ‘Y’ else ‘N’ | comm\_compliance\_watch\_flag |  |  |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' FRDBNK '  then ‘Y’ else ‘N’ | suspected\_bank\_fraud\_flag |  |  |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' KNWFRD '  then ‘Y’ else ‘N’ | suspected\_fraud\_flag |  |  |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' DNGCUS '  then ‘Y’ else ‘N’ | security\_incident\_rptd\_flag |  |  |
| app\_tblindcator\_sec.safety\_risk\_level | if safety\_risk\_level = 3  then ‘Low risk’ else if safety\_risk\_level = 2  then ‘Medium risk’  else if safety\_risk\_level = 1  then ‘High risk’  else ‘No known risk’ | safety\_risk\_level |  | No known risk |
| app\_tblNameRecord.fstrtitle app\_tblNameRecord.fstrNameType | If exists fstrNameType = 'PREFER'  then set to fstrtitle from that record else if exists fstrNameType = 'LGL'  then set to fstrtitle from that record  else  NULL | pref\_title |  |  |
| app\_tblNameRecord.fstrfirstname app\_tblNameRecord.fstrNameType | If exists fstrNameType = 'PREFER'  then set to fstrfirstname from that record else if exists fstrNameType = 'LGL'  then set to fstrfirstname from that record  else  NULL | pref\_given\_name |  |  |
| app\_tblNameRecord.fstrmiddlename  app\_tblNameRecord.fstrNameType | If exists fstrNameType = 'PREFER'  then set to fstrmiddlename from that record else if exists fstrNameType = 'LGL'  then set to fstrmiddlename from that record  else  NULL | pref\_other\_given\_name |  |  |
| app\_tblNameRecord.fstrlastname  app\_tblNameRecord.fstrNameType | If exists fstrNameType = 'PREFER'  then set to fstrlastname from that record else if exists fstrNameType = 'LGL'  then set to fstrlastname from that record  else  NULL | pref\_family\_name |  |  |
| app\_tblCustomerStd.fdtmdob | if $$is\_date\_valid$$ (fdtmdob)   then fdtmdob else NULL | date\_of\_birth |  | NULL |
| app\_tblCustomerStd.fdtmdod | if $$is\_date\_valid$$ (fdtmdod)   then fdtmdod else NULL | date\_of\_death |  | NULL |
| pp\_tblId.fstrid (nzbcid)  app\_tblId.fstrid (nzobcid)  app\_tblId.fstrid (ovbrth)  app\_tblId.fstrIdType (nzbcid) app\_tblId.fstrIdType (nzobcid) app\_tblId.fstrIdType (ovbrth) | if fstrIdType (nzbcid) = ‘NZBRTH’ )  then fstrid (nzbcid) else if fstrIdType (nzobcid) = 'OLBRTH'  then fstrid (nzobcid) else if fstrIdType (ovbrth) = 'OVBRTH'  then fstrid (ovbrth) else ‘Unknown’ | birth\_cert\_id |  | Unknown |
| app\_tblNZ\_AccIDGovr.fstrissuingcountry  app\_tblId. fstrIdType (ovbrth)  app\_tblId. fstrIdType (nzbcid)  app\_tblId. fstrIdType (nzobcid) | if fstrIdType (ovbrth) = 'OVBRTH'   then fstrissuingcountry  else if fstrIdType (nzbcid) = ‘NZBRTH’  then ‘NEZ’  else if fstrIdType (nzobcid) = 'OLBRTH'  then ‘NEZ’  else ‘Unknown’ | birth\_cert\_issuer |  | Unknown |
| app\_tblId.fstrid  app\_tblId.fstrIdType | if fstrIdType = 'NZPPRT'  then fstrId  else ‘Unknown’ | nz\_passport\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblId.fstrIdType | if fstrIdType = 'OSPPRT'  then fstrId else ‘Unknown’ | os\_passport\_id |  | Unknown |
| app\_tblId.fstrIdType  app\_tblId.flngdockey  app\_tblNZ\_AccIDGovr.fstrissuingcountry | if fstrIdType = 'OSPPRT'  then lookup app\_tblNZ\_AccIDGovr on flngdockey  if fstrissuingcountry returned  then fstrissuingcountry | os\_passport\_issuer |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZCIDD'  then fstrId else ‘Unknown’ | nz\_cert\_identity\_dia |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZCIDM'   then fstrId  else ‘Unknown’ | nz\_cert\_identity\_mbie |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZRFGT'  then fstrId  else ‘Unknown’ | nz\_refugee\_id\_dia |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZRFMB'  then fstrId  else ‘Unknown’ | nz\_refugee\_id\_mbie |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZETDC'  then fstrId  else ‘Unknown’ | nz\_emergency\_travel\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZDLIC'  then fstrId  else if fstrIdType = 'OVDLIC'  then fstrId  else ‘Unknown’ | drivers\_licence\_id |  | Unknown |
| app\_tblId.fstrIdType  app\_tblid.flngdockey  app\_tblNZ\_AccIDGovr.fstrissuingcountry | if fstrIdType = 'NZDLIC'  then ‘NEZ’ else if fstrIdType = ' OVDLIC '  lookup app\_tblNZ\_AccIDGovr on flngdockey  if fstrissuingcountry is returned  then fstrissuingcountry | driver\_license\_issuer |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'INTDRV'  then fstrId else ‘Unknown’ | intl\_driver\_permit\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'HANZ18'  then fstrId  else ‘Unknown’ | nz\_hospitality\_18over\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NZSTDN'  then fstrId  else ‘Unknown’ | nz\_student\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType  app\_tblNZ\_AccIDGEdu.fstreducationprovider | if fstrIdType = 'NZSTDN'  then lookup app\_tblNZ\_AccIDGEdu on flngdockey  if fstreducationprovider is returned  then fstreducationprovider | nz\_student\_issuer |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'SWN'  then fstrId  else ‘Unknown’ | nz\_social\_welfare\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'SSN'   then fstrId  else ‘Unknown’ | us\_social\_security\_no |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'FEIN'   then fstrId  else ‘Unknown’ | foreign\_employment\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'NCP'   then fstrId  else ‘Unknown’ | ncp\_id |  | Unknown |
| app\_tblId.fstrid  app\_tblid.fstrIdType | if fstrIdType = 'ORTAI'   then fstrId  else ‘Unknown’ | ministry\_of\_children\_id |  | Unknown |
| app\_tblIndicator.fstrindicator | if fstrindicator = 'EMGRTD',   then ‘Y’  else ’N’ | emigrated\_flag |  |  |
| app\_tblIndicator.fstrindicator | if fstrindicator = 'BDRALT'  then ‘Y’  else ’N’ | border\_alert\_flag |  |  |
| app\_tblIndicator.fstrindicator | if fstrindicator = 'PRISON'  then ‘Y’  else ’N’ | in\_prison\_flag |  |  |
| app\_tblIndicator.fstrindicator | if fstrindicator = 'UNDSCH'  then ‘Y’  else ’N’ | undischarged\_bankrupt\_flag |  |  |

Start Organisations

Filter for organisations: app\_tblCustomer.fstrcustomertype = ‘COM’ .

See DIP - [Dimension Standards](https://teams.microsoft.com/l/file/C43DC83E-9B7D-4863-99A6-A1E2A5CFA3E1?tenantId=fb39e3e9-23a9-404e-93a2-b42a87d94f35&fileType=docx&objectUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam%2FShared%20Documents%2FRefined%20Layer%20Design%20and%20Build%2F9.%20%20Standards%2FDIP_Dimension_Standards.docx&baseUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataPlatformTeam&serviceName=teams&threadId=19:c9ee9980de684e74badc5fe1d414bc56@thread.skype&groupId=7100c035-72cd-4625-aa78-807e71980487) for standard rows for Unknown and N/A DIP Ids.

| **Source Table/Column** | **Transform Rule/Logic** | **Target Column** | **Key** | **Default value for Null** | |
| --- | --- | --- | --- | --- | --- |
|  | Increment by 1 for each record. Starting value must exceed the highest corresponding value for EDW. This key is pervasive across Party, Organisation and Individual. | dip\_party\_sk | sk |  | |
| app\_tblcustomer.flngCustomerKey  rfn\_party\_sk\_map.dip\_party\_id | rfn\_party\_sk\_map.dip\_party\_id | dip\_party\_id | uk |  | |
|  | Same as Target Column dip\_party\_id | common\_dip\_party\_id |  |  | |
|  | ‘START’ | party\_origin |  |  | |
| app\_tblcustomer.flngCustomerKey | Copy | customer\_key |  |  | |
| app\_tblid.fstrId  app\_tblid.fstrIdType | If fstrIdType = ‘IRD’   then lpad(fstrId, 9, ’0’) else ‘Unknown’ | ird\_number |  | Unknown | |
| app\_tblid.fstrId  app\_tblid.fstrIdType | If fstrIdType = ‘CST’  then fstrId else ‘N/A’ | customer\_no |  | N/A | |
| app\_ tblNameRecord.fstrListFormatName  app\_tblNameRecord.fstrNameType | if fstrNameType = 'LGL'   then fstrListFormatName else ‘Unknown’ | legal\_name |  | Unknown | |
| app\_tblNameRecord.fstrListFormatName  app\_tblNameRecord.fstrNameType | if fstrNameType = 'DBACST'   then fstrListFormatName else  NULL | trading\_name |  | NULL | |
| app\_tblCustomer.fdtmCommence  app\_tblCustomer.fdtmcreated | if $$is\_date\_valid$$ (fdtmCommence)  then trunc(fdtmcommence,’DDD’)  else if $$is\_date\_valid$$ (fdtmcreated)  then trunc(fdtmcreated,’DDD’) else $$LODATE | party\_start\_date |  | $$LODATE | |
| app\_tblCustomer.fdtmCease | if $$is\_date\_valid$$ (fdtmCease)  then trunc(fdtmcease,’DDD’)  else $$HIDATE | party\_ceased\_date |  | $$HIDATE | |
|  | if (target column) party\_ceased\_date < $$HIDATE  then ‘Y’ else ‘N’ | party\_ceased\_flag |  |  | |
| app\_tblindicator\_insolv.fstrIndicator  ref\_lanindicator.fstrdecode1 | lookup ref\_lanindicator on fstrIndicator if fstrdecode1 is returned  then fstrdecode1 | insolvency\_reason |  | N/A | |
| app\_tblindicator\_insolv.fstrIndicator | Copy | insolvency\_code |  | N/A | |
| app\_tblindicator\_insolv.fstrindicator | if fstrIndicator is populated  then ‘Y’ else ‘N’ | insolvency\_flag |  | N | |
| app\_tblCustomer.fstrCustomerType  ref\_lancustomertype.fstrdecode2 | Lookup ref\_lancustomertype on fstrCustomerType if fstrdecode2 is returned  then fstrdecode2 | party\_type |  | Unknown | |
|  | ‘COM’ | party\_type\_code |  | Unknown | |
| app\_tblCustomer.flngDocKey  app\_tblnz\_customerstd.fstrCustomerSubType ref\_lancustomersubtype.fstrddecode2 | Lookup app\_tblnz\_customerstd on flngDocKey if fstrcustomersubtype is returned  then lookup ref\_lancustomersubtype on fstrcustomersubtype  if fstrddecode2 is returned  then fstrddecode2 | party\_subtype |  | Unknown | |
| app\_tblCustomer.flngDocKey  app\_tblnz\_customerstd.fstrCustomerSubType | Lookup app\_tblnz\_customerstd on flngDocKey if fstrcustomersubtype is returned  then fstrCustomerSubType | party\_subtype\_code |  | Unknown | |
| app\_tblCustomer.flngDocKey  app\_tblnz\_customerstd.fstrCustomerClass  app\_tblnz\_customerstd. fstrCustomerSubtypeClass ref\_lannz\_customerclass.fstrDecode2 | Select fstrDecode2  From ref\_lannz\_customerclass  where derived value of Party\_Class\_Code = fstrCustomerClass | party\_class |  | | Unknown |
| app\_tblCustomer.flngDocKey  app\_tblnz\_customerstd.fstrCustomerClass  app\_tblnz\_customerstd. fstrCustomerSubtypeClass | Select Coalesce(fstrCustomerSubtypeClass, fstrCustomerClass) as Party\_Class\_Code  From app\_tblnz\_customerstd  where app\_tblCustomer.flngDocKey = app\_tblnz\_customerstd flngDocKey | party\_class\_code |  | | Unknown |
| app\_tblnz\_custsegmenthistory.fstrCustomerSegment  ref\_lannz\_customersegments.fstrSegment  ref\_lannz\_customersegments.fstrDecode2 | if fstrCustomerSegment is null then  ‘Unknown’ else lookup ref\_lannz\_customersegments   on fstrSegment =fstrCustomerSegment  if fstrDecode2 is returned   then fstrDecode2  else ‘Unknown’  else ‘Unknown’ | party\_segment |  | Unknown | |
| ~~a~~app\_tblnz\_custsegmenthistory.fstrCustomerSegment | if fstrCustomerSegment is null then  ‘Unknown’ else concatenate(‘SEG’, fstrCustomerSegment) | party\_segment\_code |  | Unknown | |
| app\_tblnaics.fstrnaics | ~~Copy~~ Set to UPPER(app\_tblnaics.fstrnaics) | BIC\_code |  | | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.BIC\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if BIC\_desc is returned  then BIC\_desc | BIC\_desc |  | | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Concatenate(Substr(fstrnaics,1,5), ‘00’)  Else  Set to “Unknown” | ANZSIC\_level4\_code |  | | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level4\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level4\_desc is returned  then ANZSIC\_level4\_desc | ANZSIC\_level4\_desc |  | | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Substr(fstrnaics,1,4)  Else  Set to “Unknown” | ANZSIC\_level3\_code |  | | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level3\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level3\_desc is returned  then ANZSIC\_level3\_desc | ANZSIC\_level3\_desc |  | | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Substr(fstrnaics,1,3)  Else  Set to “Unknown” | ANZSIC\_level2\_code |  | | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level2\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level2\_desc is returned  then ANZSIC\_level2\_desc | ANZSIC\_level2\_desc |  | | Unknown |
| app\_tblnaics.fstrnaics | If fstrnaics is not NULL then  Set to Substr(fstrnaics,1,1)  Else  Set to “Unknown” | ANZSIC\_level1\_code |  | | Unknown |
| app\_tblnaics.fstrnaics  snz\_BIC\_ANZSIC2006.ANZSIC\_level1\_desc | Lookup snz\_BIC\_ANZSIC2006 using upper(fstrnaics)  if ANZSIC\_level1\_desc is returned  then ANZSIC\_level1\_desc | ANZSIC\_level1\_desc |  | | Unknown |
| ref\_lancustomerlevel.fstrCustomerLevel  ref\_lancustomerlevel.fstrDecode2  dss\_special\_clients\_all.special\_classification  *target column: security\_level\_code* | lookup ref\_lancustomerlevel on   fstrCustomerLevel = *security\_level\_code*  if fstrDecode2 is returned then  update all records for this DIP\_PARTY\_ID  Set to fstrDecode2 **over all time** | security\_level |  | Unknown | |
| app\_tblCustomer.flngDocKey  app\_tblcustomerlevel.fstrCustomerLevel  dss\_special\_clients\_all.special\_classification  *target column: ird\_number* | Lookup app\_tblcustomerlevel on flngDocKey if fstrCustomerLevel is returned then  update all records for this DIP\_PARTY\_ID   Set to fstrCustomerLevel **over all time** else lookup dss\_special\_clients\_all on *ird\_number*  if special\_classification is returned then  update all records for this DIP\_PARTY\_ID   set to special\_classification **over all time** | security\_level\_code |  | Unknown | |
| app\_tblcustomer.flngCustomerKey  app\_tblcustomerinfo.fstrDistrictOffice  ref\_lanofficelocation.fstrDecode1 | Lookup app\_tblcustomerinfo on flngCustomerKey if fstrDistrictOffice is returned  then lookup ref\_lanofficelocation on fstrDistrictOffice  if fstrDecode1 is returned   then fstrDecode1 | district\_office |  | Unknown | |
| app\_tblid.fstrId  app\_tblid.fstrIdType | if fstrIdType = ' TXIDNM '  then fstrId else ‘Unknown’ | os\_taxpayer\_id |  | Unknown | |
| app\_tblid.fstrIdType  app\_tblid.flngdockey  app\_tblnz\_accidgovr.fstrIssuingCountry | if fstrIdType = ' TXIDNM '  lookup app\_tblnz\_accidgovr on flngdockey  if fstrIssuingCountry is returned   then fstrIssuingCountry | os\_taxpayer\_id\_issuer |  | Unknown | |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' CRDRPD '  then ‘Y’ else ‘N’ | credit\_reported\_flag |  |  | |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' CSCCOM '  then ‘Y’ else ‘N’ | comm\_compliance\_watch\_flag |  |  | |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' FRDBNK '  then ‘Y’ else ‘N’ | suspected\_bank\_fraud\_flag |  |  | |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' KNWFRD '  then ‘Y’ else ‘N’ | suspected\_fraud\_flag |  |  | |
| app\_tblindcator.fstrIndicator | if fstrIndicator = ' DNGCUS '  then ‘Y’ else ‘N’ | security\_incident\_rptd\_flag |  |  | |
| app\_tblindcator\_sec.safety\_risk\_level | if safety\_risk\_level = 3  then ‘Low risk’ else if safety\_risk\_level = 2  then ‘Medium risk’  else if safety\_risk\_level = 1  then ‘High risk’  else ‘No known risk’ | safety\_risk\_level |  | No known risk | |
| app\_tblCustomer.fdtmCommence | if $$is\_date\_valid$$ (fdtmCommence)  then trunc(fdtmCommence,’DDD’) | org\_incorp\_date |  | Null | |
| app\_tblCustomer.fdtmCease | if $$is\_date\_valid$$ (fdtmCease)  then trunc(fdtmCease,’DDD’)  else $$HIDATE | org\_cease\_date |  | $$HIDATE | |
| app\_tblIndicator.fstrIndicator | if fstrIndicator = 'STROFF'   then 'Y'  else 'N' | struck\_off\_flag |  |  | |
| app\_tblIndicator.fstrIndicator | if fstrIndicator = ‘AMLGMD’  then 'Y'  else 'N' | amalgamated\_flag |  |  | |
| app\_tblIndicator.fstrIndicator | if fstrIndicator = 'AMLGMG'   then 'Y'  else 'N' | amalgamating\_flag |  |  | |
| app\_tblId.fstrId  app\_tblid.fstrIdType | if fstrIdType = 'NZCINC'   then fstrId else ‘Unknown’ | nz\_cert\_incorp\_id |  | Unknown | |
| app\_tblId.fstrId  app\_tblid.fstrIdType | if fstrIdType = 'OSCINC'   then fstrId  else ‘Unknown’ | os\_cert\_incorp\_id |  | Unknown | |
| app\_tblId.fstrIdType  app\_tblId.flngdockey  app\_tblNZ\_AccIDGovr.fstrissuingcountry | if fstrIdType = 'OSCINC'  lookup app\_tblNZ\_AccIDGovr on flngdockey  if fstrissuingcountry is returned  then fstrissuingcountry | os\_cert\_incorp\_issuer |  | Unknown | |
| app\_tblId.fstrId  app\_tblid.fstrIdType | if fstrIdType = 'NZBN'   then fstrId else ‘Unknown’ | nzbn |  | Unknown | |
| app\_tblId.fstrId  app\_tblid.fstrIdType | if fstrIdType = 'GOVACT'   then fstrId  else ‘Unknown’ | nz\_society\_actuaries\_id |  | Unknown | |
| app\_tblId.fstrId  app\_tblid.fstrIdType | if fstrIdType = 'GIIN'  then fstrId else ‘Unknown’ | financial\_institiution\_giin |  | Unknown | |
| app\_tblId.fstrId  app\_tblid.fstrIdType | if fstrIdType = 'SEGIIN'  then fstrId  else ‘Unknown’ | sponsoring\_entity\_giin |  | Unknown | |
| app\_tblNz\_cstcominfo.fblnNilCompany | if fblnNilCompany = 1  then ~~else~~ ‘Y’  else ‘N’ | nil\_company\_flag |  | N | |
| app\_tblIndicator.fstrIndicator | if fstrIndicator = 'CHRITY'  then 'Y'  else 'N' | charity\_org\_flag |  |  | |

# Appendix A – Standard De-Duplication Process

The purpose of step 2 is to remove duplicates over time so that the resulting rows for each natural key are contiguous and non-overlapping in time. The default de-duplication process is to be following unless explicitly stated otherwise.

**De-duplication Process where Business Dates Exist**

We use a slightly different process where the data has business start and end dates which are often completely independent of the RecordEffective and RecordExpiry dates. The former shows a history of when something was valid, and the latter are showing what the source system looked like at a point in time.

We can take both of these timelines into account when looking to resolve duplicates, by creating some derived columns during processing and then using those columns in our tie-breaker logic.

For each record we create the following derived columns:

|  |  |
| --- | --- |
| **Derived Column** | **Definition** |
| **Timeslice\_From** | **Business\_Start\_Date or Record\_Effective\_Timestamp** |
| **Timeslice\_To** | **Business\_End\_Date or Record\_Expiry\_Timestamp** |
| **Number\_of\_valid\_seconds** | **Min(Business\_End\_Date or Record\_Expiry\_Timestamp) – Max(Business\_Start\_Date or Record\_Effective\_Timestamp)** |

Number\_of\_valid\_seconds can then be used as a tie-breaker field in determining the “correct” record for each timeslice.

**Default De-duplication Process**

De-duplicate each source table over its ***Natural Key*** as follows:

1. Establish a single timeline from all unique ***business active timestamps***. Gaps may occur but there must be no overlaps.
2. Determine all time-periods between two contiguous timestamps on the timeline from (1) regardless of whether they are ‘start’ or ‘end’ timestamps
3. For each time-period from (2):
   1. Determine the record(s) which ‘exist’ during this time-period
   2. Identify the record in (3a) with the highest priority using the specified ***Tie-breakers***
4. Concatenate the ‘priority’ records from 3b. Gaps may occur but there must be no overlaps. The following ‘compress’ step with ‘merge’ contiguous duplicate values.

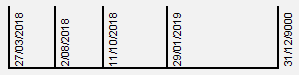
**Example**

This example uses the following source data.



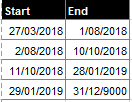
Step 2.1: Establish a single timeline

Establish a single timeline from all unique ***business active timestamps***. Gaps may occur but there must be no overlaps.



Step 2.2: Determine all time-periods

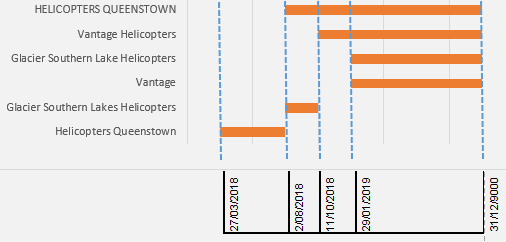
Determine all time-periods between two contiguous timestamps on the timeline from Step 2.1 regardless of whether they are ‘start’ or ‘end’ timestamps.



Step 2.3: Identify highest-priority records

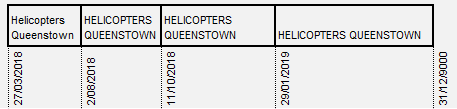
For each time-period from Step 2.2:

1. Determine the record(s) which ‘exist’ during this time-period.
   1. For 27/03/2018-02/08/2018, there is only one record.
   2. For 02/08/2018-11/10/2018, there are 2 records.
   3. For 11/10/2018-29/01/2019, there are also 2 records. Again, the top-most record has the highest priority.
   4. For 29/01/2019-31/12/9000, there are 4 records and the top-most record has the highest priority.
2. Identify the record in Step2.3a with the highest priority using the specified ***Tie-breakers***
   1. For 27/03/2018-02/08/2018, there is only one record, therefore, it is automatically the highest priority.
   2. For 02/08/2018-11/10/2018, the top-most record has the highest priority.
   3. For 11/10/2018-29/01/2019, the top-most record has the highest priority.
   4. For 29/01/2019-31/12/9000, the top-most record has the highest priority.



Step 2.4: Concatenate

Concatenate the ‘priority’ records from 3b. Gaps may occur but there must be no overlaps.



Step 3: Compress

After de-duplicating records in Step 2, one or more contiguous records sharing the same Natural Key may have the same values for all columns used for comparison. This step will ‘compress’ (i.e. merge) any such contiguous records using the minimum Record\_Effective\_Date and maximum Record\_Expiry\_Date for the resulting timeline.

In this example, records 2-4 from Step 2.4 are merged into record 5. Record 1 remains unchanged.



# Appendix B – SQL to Populate table mp\_source\_code

Because columns in EDW tables have different sets of codes and/or descriptions when compared to the equivalent columns in START we need a mapping table to allow processing to lookup the correct values.

Table mp\_source\_code will follow a pattern 1 design and will only function as an insert table using pre-set string values input by the user. The table itself will only be populated once for party and used initially for only mapping START to EDW value precedence where appropriate for consistency of data across EDW and START. e.g. where a value in EDW is ‘I’, START may have a value of ‘INDVL’ which will overwrite I in the load to the party domain.

The basic SQL to populate table mp\_source\_code follows.

CREATE TABLE dev\_refined.mp\_source\_code (

source\_system VARCHAR(16),

source\_code\_name VARCHAR(32),

source\_code\_value VARCHAR(8),

source\_description VARCHAR(120),

dip\_code\_name VARCHAR(32),

dip\_code\_value VARCHAR(12),

dip\_description VARCHAR(510),

record\_effective\_timestamp TIMESTAMP,

record\_expiry\_timestamp TIMESTAMP

) STORED AS PARQUET;

INSERT INTO dev\_refined.mp\_source\_code

VALUES

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('I' AS VARCHAR(8)), CAST('Individual' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('INDVDL' AS VARCHAR(12)), CAST('Individual' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('C' AS VARCHAR(8)), CAST('Company' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('COMPNY' AS VARCHAR(12)), CAST('Company' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('T' AS VARCHAR(8)), CAST('Trust' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('TRUST' AS VARCHAR(12)), CAST('Trust' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('P' AS VARCHAR(8)), CAST('Partnership' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('PTNRSP' AS VARCHAR(12)), CAST('Partnership' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('S' AS VARCHAR(8)), CAST('Society/Club' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('SOCITY' AS VARCHAR(12)), CAST('Society/Club' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('M' AS VARCHAR(8)), CAST('Maori Auth.' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('MRIAUT' AS VARCHAR(12)), CAST('Maori Authority' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('X' AS VARCHAR(8)), CAST("To be est'd" AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('TBD' AS VARCHAR(12)), CAST('To Be Established' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('U' AS VARCHAR(8)), CAST('Unit Trust' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('UNTTST' AS VARCHAR(12)), CAST('Unit Trust' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('F' AS VARCHAR(8)), CAST('Super. Fund' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('SPRFND' AS VARCHAR(12)), CAST('Super Fund' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('A' AS VARCHAR(8)), CAST('Holding Acct' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('HLDACC' AS VARCHAR(12)), CAST('Holding Account' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('D' AS VARCHAR(8)), CAST('Diplomatic M' AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('EMBASY' AS VARCHAR(12)), CAST('Embassy' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('ENTITY\_TYPE' AS VARCHAR(32)), CAST('G' AS VARCHAR(8)), CAST("GOV'T DEP'T" AS VARCHAR(120)), CAST('PARTY\_SUBTYPE' AS VARCHAR(32)), CAST('GVMDPT' AS VARCHAR(12)), CAST('Government Department' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CUST\_TYPE' AS VARCHAR(32)), CAST('ORG' AS VARCHAR(8)), CAST('ORG' AS VARCHAR(120)), CAST('PARTY\_TYPE' AS VARCHAR(32)), CAST('COM' AS VARCHAR(12)), CAST('Non-Individual' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CUST\_TYPE' AS VARCHAR(32)), CAST('IND' AS VARCHAR(8)), CAST('IND' AS VARCHAR(120)), CAST('PARTY\_TYPE' AS VARCHAR(32)), CAST('IND' AS VARCHAR(12)), CAST('Individual' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('00' AS VARCHAR(8)), CAST('Unknown' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('0' AS VARCHAR(12)), CAST('No access office' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('02' AS VARCHAR(8)), CAST('Auckland Corporate' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('02' AS VARCHAR(12)), CAST('Auckland Large Enterprises' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('10' AS VARCHAR(8)), CAST('SOUTH ISLAND SERVICE CENTRE' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('10' AS VARCHAR(12)), CAST('Russley Road. Christchurch' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('15' AS VARCHAR(8)), CAST('Dunedin' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('15' AS VARCHAR(12)), CAST('Dunedin' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('19' AS VARCHAR(8)), CAST('Gisborne' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('19' AS VARCHAR(12)), CAST('Gisborne' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('23' AS VARCHAR(8)), CAST('Greymouth' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('23' AS VARCHAR(12)), CAST('Greymouth' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('27' AS VARCHAR(8)), CAST('Hamilton' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('27' AS VARCHAR(12)), CAST('Hamilton' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('34' AS VARCHAR(8)), CAST('Invercargill' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('34' AS VARCHAR(12)), CAST('Invercargill' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('38' AS VARCHAR(8)), CAST('Auckland Service Centre' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('38' AS VARCHAR(12)), CAST('Manukau - Twin Towers' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('47' AS VARCHAR(8)), CAST('Napier' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('47' AS VARCHAR(12)), CAST('Napier' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('48' AS VARCHAR(8)), CAST('Nelson' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('48' AS VARCHAR(12)), CAST('Nelson' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('49' AS VARCHAR(8)), CAST('New Plymouth' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('49' AS VARCHAR(12)), CAST('New Plymouth' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('58' AS VARCHAR(8)), CAST('Palmerston North' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('58' AS VARCHAR(12)), CAST('Palmerston North' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('62' AS VARCHAR(8)), CAST('Rotorua' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('62' AS VARCHAR(12)), CAST('Rotorua' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('67' AS VARCHAR(8)), CAST('Takapuna' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('67' AS VARCHAR(12)), CAST('Takapuna - AIA House' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('69' AS VARCHAR(8)), CAST('Corporate - Tax Avoidance' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('69' AS VARCHAR(12)), CAST('Corporate - Tax Avoidance' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('72' AS VARCHAR(8)), CAST('Tauranga' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('72' AS VARCHAR(12)), CAST('Tauranga Regional House' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('80' AS VARCHAR(8)), CAST('Timaru' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('80' AS VARCHAR(12)), CAST('Timaru' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('89' AS VARCHAR(8)), CAST('Wellington Corporate' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('89' AS VARCHAR(12)), CAST('Wellington Large Enterprises' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('90' AS VARCHAR(8)), CAST('Wellington' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('90' AS VARCHAR(12)), CAST('Wellington Service Centre' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('OFFICE\_CODE' AS VARCHAR(32)), CAST('93' AS VARCHAR(8)), CAST('Whangarei' AS VARCHAR(120)), CAST('DISTRICT\_OFFICE' AS VARCHAR(32)), CAST('93' AS VARCHAR(12)), CAST('Whangarei' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CLIENT\_STATUS' AS VARCHAR(32)), CAST('B' AS VARCHAR(8)), CAST('Bankrupt' AS VARCHAR(120)), CAST('INSOLVENCY\_CODE ' AS VARCHAR(32)), CAST('INSCBF' AS VARCHAR(12)), CAST('Bankruptcy finalised ' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CLIENT\_STATUS' AS VARCHAR(32)), CAST('R' AS VARCHAR(8)), CAST("Rec'ship" AS VARCHAR(120)), CAST('INSOLVENCY\_CODE ' AS VARCHAR(32)), CAST('INSCRC' AS VARCHAR(12)), CAST('In Receivership ' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CLIENT\_STATUS' AS VARCHAR(32)), CAST('L' AS VARCHAR(8)), CAST("Liq'dation" AS VARCHAR(120)), CAST('INSOLVENCY\_CODE ' AS VARCHAR(32)), CAST('INSCLF' AS VARCHAR(12)), CAST('Liquidation finalised ' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CLIENT\_STATUS' AS VARCHAR(32)), CAST('U' AS VARCHAR(8)), CAST('Und Bnkrpt ' AS VARCHAR(120)), CAST('INSOLVENCY\_CODE ' AS VARCHAR(32)), CAST('UNDSCH' AS VARCHAR(12)), CAST('Undischarged bankruptcy ' AS VARCHAR(510)), '1800-01-01', '9000-12-31'),

(CAST('FIRST' AS VARCHAR(16)), CAST('CLIENT\_STATUS' AS VARCHAR(32)), CAST('V' AS VARCHAR(8)), CAST('VOL ADMIN ' AS VARCHAR(120)), CAST('INSOLVENCY\_CODE ' AS VARCHAR(32)), CAST('INSCVA' AS VARCHAR(12)), CAST('Under Voluntary Administration ' AS VARCHAR(510)), '1800-01-01', '9000-12-31')

# Appendix C – EDW-START Party Map

A new lookup table will be created as part of the ingestion process called DIP\_EDW\_START\_PARTY\_MAP. This table will follow a pattern 1 design and will only function as an insert table. The table itself will only be populated once in order to create the initial DIPI and DIPC unique keys, however if job is run to populate the table a second time, there may be a scenario that new mappings become available.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **datatype** | **Comments** |
| ird\_number | bigint | Primary key / natural key (max of 9 digits) – cast as bigint |
| start\_customer\_key | bigint | Primary key / natural key (max of 10 digits) |
| dip\_party\_id | varchar (20) | Derived value (current max is 16 using DIP-C-CK and DIP-I- IRD#) |

**Processing**

The activity to populate the DIP\_EDW\_START\_PARTY\_MAP mapping table will be a one-off job, however the processing will be able to be run a second time ad-hoc at a future date if required.

In the case when this occurs, new mapping values will be inserted for the following reasons (a) we have no date filter applied (so new mapping values will be added) and (b) customer records in START that previously didn’t have an IRD# and now have one in tblid, will be inserted –

for scenario (b) this will be handled by a DIP\_PARTY\_ID lookup in the party table, to see if a key value was already generated and inserted.

**Basic SQL**

Select

flngcustomerkey

, cast(fstrid as bigint) as fstrid

from app\_tblid

where fstrIdType = 'IRD'

<<see Section 6.1 for filtering criteria for table app\_tblid >>  
<<see Section 6.2 for de-duplication rules for app\_tblid>>

**Field level transformations**

The below columns will be updated in the dip\_EDW\_START\_PARTY\_MAP table along with any standard processing columns.

|  |  |  |
| --- | --- | --- |
| **Source Table/ Column** | **Transformation Rule / Logic** | **Column** |
| app\_tblid.flngcustomerkey | copy | dip\_edw\_start\_party\_map.start\_customer\_key |
| app\_tblid. fstrid | copy | dip\_edw\_start\_party\_map.ird\_number |
| app\_tblid.flngcustomerkey | Concatenate(‘DIP-C-‘, cast(flngcustomerkey as string) | dip\_edw\_start\_party\_map.dip\_party\_id |

# Appendix D – Populating Natural Key DIP\_PARTY\_ID for EDW

The transformations below will attempt to get the DIP-C unique key value from the lookup table, but if it is not found, the DIP\_PARTY\_ID value will default to a DIP-I value, indicating a match wasn’t found between START and EDW for that IRD number.

**Processing**

The activity to enrich the EDW data with DIP\_PARTY\_ID’s and start customer keys, will be physically implemented within the same process to populate the party model from EDW data. This will form part of the one of EDW data load to parties.

**Basic SQL**

Select

  pm.start\_customer\_key

, pm.dip\_party\_id

, pm.ird\_number as map\_ird\_number

, p.dip\_party\_id as chk\_dip\_party\_id *-- table at refined layer used for the check*

, clr.ird\_number as ird\_number

from dss\_clients clr

left outer join party p on p.ird\_number = clr.ird\_number

left outer join tmp\_dip\_edw\_start\_party\_map pm on pm.ird\_number= clr.ird\_number

**Field level transformations**

The transformations below will attempt to get the DIP-C value from the lookup table in the party load, but if it is not found, the DIP\_PARTY\_ID value will default to a DIP-I value, indicating a match wasn’t found between START.

|  |  |  |
| --- | --- | --- |
| **Source Table/ Column** | **Transformation Rule / Logic** | **Target Table/ Column** |
| **dss\_clients (cleanraw)**  ird\_number  **dip\_edw\_start\_party\_map** map\_ird\_number  start\_customer\_key | If ird\_number = map\_ird\_number  then  start\_customer\_key  else  NULL end if | party.start\_customer\_key |
| **party**  chk\_dip\_party\_id  **dip\_edw\_start\_party\_map**  map\_ird\_number  dip\_party\_id  **dss\_clients (cleanraw)**  ird\_number | If chk\_dip\_party\_id is null  then  coalesce (dip\_party\_id, concatenate (‘DIP-I-’, cast (ird\_number as varchar))) as dip\_party\_id  else  NULL end if | party.dip\_party\_id |

**Scenario handling**

The following scenario may occur and will be treated as an “orphaned” EDW party.

| **Scenario#** | **Scenario** | **Transformation Rule** |
| --- | --- | --- |
| [SNO-1](https://teams.microsoft.com/l/file/AB48FFE5-D4EA-4013-830F-312C2E309C3F?tenantId=fb39e3e9-23a9-404e-93a2-b42a87d94f35&fileType=xlsx&objectUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataIntelligenceTeam%2FShared%20Documents%2FCo-Design%2FKYC%20Customer%20Modelling%2F00%20Data%20Models%2FKYC%2FTransformation%2Ftblid%20ird%20number%20analysis.xlsx&baseUrl=https%3A%2F%2Firnz.sharepoint.com%2Fsites%2FDataIntelligenceTeam&serviceName=teams&threadId=19:92db607d340148959dff070fae0da288@thread.skype&groupId=f7694694-0cef-4fcb-b517-db5fc86723e0) | Of the 13 000 unique IRD#’s found in EDW that are *Active (1 845 305 regardless of status)* and are not in START; Records will be populated using DIP-I keys. If START in the future creates these records using DIP-C logic, running the process to populate dip\_EDW\_START\_PARTY\_MAP a second time will result in a new mapping record created. It will be treated as an orphaned party “divorced” from the new DIP-C key created for the START customer equivalent for that IRD#. | Checking if a dip\_party\_id is already populated will handle this scenario by not overwriting an already inserted DIP-I value. At a minimum a log should be kept of these orphaned records *to be processed via a bulk cascade update process of the DIPI to DIPC in the future, if required, to re-couple the EDW orphaned party and set the EDW record active flag to N eg. there was no DIP-C match and loaded with a DIP-I indicating an orphaned record.* |

# Appendix E – Sample SQL for Context – EDW

The ‘Select’ column names in these SQL statements will be referred to in the “Source Table/Column” fields of the transformation rules and are intended to give context to those rules as well as a guide on how to merge all the source objects into the final Individual & Organisation data streams.

EDW Individual

select

pm.start\_customer\_key

,pm.dip\_party\_id

,brt.individual\_party\_id -- common\_party\_id

,$$HIDATE as party\_ceased\_date -- Just use a default for now

,c.ird\_number

,c.resident\_ind -- tax\_residency\_status

,c.cust\_type

,vmt.target\_column -- party\_type

,vmtc.target\_column -- party\_type\_code

,vmst.target\_column -- party\_subtype

,vmstc.target\_column -- party\_subtype\_code

,et.description -- party\_subtype unmapped values

,legn.first\_names -- makes up legal name

,legn.surname -- makes up legal name

,legn.organisation\_name -- makes up legal name

,tradn.organisation\_name -- makes up trading name

,ec.description -- party\_class

,cla.fstrDecode2 -- used to derive party\_class

,c.entity\_class -- party\_class\_code

,o.description -- district\_office

,cu.district\_office -- district\_office\_code

,cu.sic\_code -- BIC\_code

,anzsic.BIC\_desc -- BIC\_DESC

,anzsic.ANZSIC\_level4\_code -- ANZSIC\_level4\_code

,anzsic.ANZSIC\_level2\_code -- ANZSIC\_level3\_code

,anzsic.ANZSIC\_level3\_code -- ANZSIC\_level2\_code

,anzsic.ANZSIC\_level1\_code -- ANZSIC\_level1\_code

,anzsic.ANZSIC\_level4\_desc -- ANZSIC\_level4\_desc

,anzsic.ANZSIC\_level2\_desc -- ANZSIC\_level3\_desc

,anzsic.ANZSIC\_level3\_desc -- ANZSIC\_level2\_desc

,anzsic.ANZSIC\_level1\_desc -- ANZSIC\_level1\_desc

,'FIRST' as party\_origin

,'N/A' as nz\_firearms\_licence\_id

,'N/A' as os\_taxpayer\_id

,'N/A' as os\_taxpayer\_id\_issuer

,'N/A' as credit\_reported\_flag

,'N/A' as comm\_compliance\_watch\_flag

,'N/A' as suspected\_bank\_fraud\_flag

,'N/A' as suspected\_fraud\_flag

,'N/A' as safety\_risk\_level

,'N/A' as acc\_id

,'N/A' as mbie\_client\_id

,'N/A' as security\_incident\_rptd\_flag

,'N/A' as party\_segment

,'N/A' as party\_segment\_code

,'N/A' as security\_level

,'N/A' as security\_level\_code

,'WIP' as industry -- No logic for this value yet but will be a lookup to start static table

,c.org\_commencement\_date -- forms part of PARTY\_START\_DATE

,c.date\_of\_birth -- also forms part of PARTY\_START\_DATE

,c.client\_status -- undischarged\_bankrupt\_flag and for party ceased flag

,c.loan\_ind -- student\_loan\_flag

,c.nz\_citizen\_ind -- nz\_citizen\_flag

,cn.first\_names -- pref\_given\_name

,cn.surname -- pref\_family\_name

,cn.title -- pref\_title

,cn.first\_names -- given\_name

,cn.surname -- family\_name

,cn.title -- title

,erbir.external\_id -- birth\_cert

,erpas.external\_id -- nz\_passport\_id

,eropt.external\_id -- os\_passport\_id

,eretd.external\_id -- nz\_emergency\_travel\_id

,erlic.external\_id -- driver\_licence\_id

,erpasi.country\_code -- os\_passport\_issuer

,erdiri.country\_code -- driver\_license\_issuer

,erdsw.external\_id -- nz\_social\_welfare\_id

,erhnz.external\_id -- nz\_hospitality\_18over\_id

,ernzc.external\_id -- nz\_citizen\_cert\_id

,errtd.external\_id -- nz\_refugee\_id and mbie refugee cannot be split….

,'N/A' as nz\_refugee\_id\_mbie

,'N/A' as in\_prison\_flag

,'N/A' as us\_social\_security\_no

,'N/A' as foreign\_employment\_id

,'N/A' as birth\_cert\_issuer

,'N/A' as other\_given\_name

,'N/A' as pref\_other\_given\_name

,'N/A' as border\_alert\_flag

,'N/A' as emigrated\_flag

,'N/A' as ncp\_id

from dss\_clients c

LEFT OUTER JOIN dss\_client\_names cn

on cn.ird\_number = c.ird\_number

and cn.client\_name\_type = 'P'

LEFT OUTER JOIN dss\_external\_references erbir

on erbir.ird\_number = c.ird\_number

and erbir.external\_org\_code ='BIR'

LEFT OUTER JOIN dss\_external\_references erpas

on erpas.ird\_number = c.ird\_number

and erpas.external\_org\_code ='PAS'

LEFT OUTER JOIN dss\_external\_references eropt

on eropt.ird\_number = c.ird\_number

and eropt.external\_org\_code ='OPT'

LEFT OUTER JOIN dss\_external\_references eretd

on eretd.ird\_number = c.ird\_number

and eretd.external\_org\_code ='ETD'

LEFT OUTER JOIN dss\_external\_references erlic

on erlic.ird\_number = c.ird\_number

and erlic.external\_org\_code ='LIC'

LEFT OUTER JOIN dss\_external\_references erdsw

on erdsw.ird\_number = c.ird\_number

and erdsw.external\_org\_code ='DSW'

LEFT OUTER JOIN dss\_external\_references erhnz

on erhnz.ird\_number = c.ird\_number

and erhnz.external\_org\_code = 'HNZ'

LEFT OUTER JOIN dss\_external\_references ernzc

on ernzc.ird\_number = c.ird\_number

and ernzc.external\_org\_code = 'NZC'

LEFT OUTER JOIN dss\_external\_references errtd

on errtd.ird\_number = c.ird\_number

and errtd.external\_org\_code = 'RTD'

LEFT OUTER JOIN dss\_external\_references erpasi

on erpasi.ird\_number = c.ird\_number

and erpasi.external\_org\_code = 'OPT'

LEFT OUTER JOIN dss\_external\_references erdiri

on erpasi.ird\_number = c.ird\_number

and erpasi.external\_org\_code = 'LIC'

LEFT OUTER JOIN dss\_client\_names legn

on legn.ird\_number = c.ird\_number

and legn.location\_number = 0

and legn.client\_name\_type = 'P'

LEFT OUTER JOIN dss\_client\_names tradn

on tradn.ird\_number = c.ird\_number

and tradn.location\_number = 1

and tradn.client\_name\_type = 'T'

LEFT OUTER JOIN dss\_customers cu

on cu.ird\_number = c.ird\_number

and cu.location\_number = 1

LEFT OUTER JOIN dss\_entity\_types et

on et.entity\_type\_code = c.entity\_type

LEFT OUTER JOIN dss\_entity\_classes ec

on ec.entity\_class\_code = c.entity\_class

LEFT OUTER JOIN ref\_lannz\_customerclass cla

on cla.fstrCustomerClass = c.entity\_class

LEFT OUTER JOIN dss\_offices o

on o.office\_code = cu.district\_office

LEFT OUTER JOIN tmp\_dip\_edw\_start\_party\_map pm

on pm.ird\_number = c.ird\_number

LEFT OUTER JOIN mp\_source\_code mpt

on mpt.dip\_code\_name = ‘PARTY\_TYPE’

and mpt.source\_code\_value = c.cust\_type

LEFT OUTER JOIN mp\_source\_code mpst

on mpst.dip\_code\_name = ‘PARTY\_SUBTYPE’

and mpst.source\_code\_value = c.entity\_type

LEFT OUTER JOIN local\_start\_bankruptcy brt

on brt.ird\_number = c.ird\_number

LEFT OUTER JOIN snz\_BIC\_ANZSIC2006 anzsic

on anzsic.BIC\_code = upper(cu.sic\_code)

where c.cust\_type = 'IND'

EDW Organisation

select

pm.start\_customer\_key

,pm.dip\_party\_id

,brt.individual\_party\_id -- common\_party\_id

,$$HIDATE as party\_ceased\_date -- Just use a default for now

,c.ird\_number

,c.resident\_ind -- tax\_residency\_status

,c.cust\_type

,vmt.target\_column -- party\_type

,vmtc.target\_column -- party\_type\_code

,vmst.target\_column -- party\_subtype

,vmstc.target\_column -- party\_subtype\_code

,et.description -- party\_subtype unmapped values

,legn.first\_names -- makes up legal name

,legn.surname -- makes up legal name

,legn.organisation\_name -- makes up legal name

,tradn.organisation\_name -- makes up trading name

,ec.description -- party\_class

,cla.fstrDecode2 -- used to derive party\_class

,c.entity\_class -- party\_class\_code

,o.description -- district\_office

,cu.district\_office -- district\_office\_code

,cu.sic\_code -- BIC\_code

,anzsic.BIC\_desc -- BIC\_DESC

,anzsic.ANZSIC\_level4\_code -- ANZSIC\_level4\_code

,anzsic.ANZSIC\_level2\_code -- ANZSIC\_level3\_code

,anzsic.ANZSIC\_level3\_code -- ANZSIC\_level2\_code

,anzsic.ANZSIC\_level1\_code -- ANZSIC\_level1\_code

,anzsic.ANZSIC\_level4\_desc -- ANZSIC\_level4\_desc

,anzsic.ANZSIC\_level2\_desc -- ANZSIC\_level3\_desc

,anzsic.ANZSIC\_level3\_desc -- ANZSIC\_level2\_desc

,anzsic.ANZSIC\_level1\_desc -- ANZSIC\_level1\_desc

,'FIRST' as party\_origin

,'N/A' as os\_taxpayer\_id

,'N/A' as os\_taxpayer\_id\_issuer

,'N/A' as credit\_reported\_flag

,'N/A' as comm\_compliance\_watch\_flag

,'N/A' as suspected\_bank\_fraud\_flag

,'N/A' as suspected\_fraud\_flag

,'N/A' as safety\_risk\_level

,'N/A' as security\_incident\_rptd\_flag

,'N/A' as party\_segment

,'N/A' as party\_segment\_code

,'N/A' as security\_level

,'N/A' as security\_level\_code

,'WIP' as industry -- No logic for this value yet but will be a lookup to start static table

,c.org\_commencement\_date -- forms part of PARTY\_START\_DATE

,c.client\_status -- struck\_off\_flag and for party ceased flag

,c.org\_commencement\_date -- org\_incorp\_date

,ccamd.client\_code\_value as amalgamated\_flag

,ccamg.client\_code\_value as amalgamating\_flag

, 'Not Applicable' as org\_cease\_date

, 'Not Applicable' as nz\_cert\_incorp\_id

, 'Not Applicable' as os\_cert\_incorp\_id

, 'Not Applicable' as os\_cert\_incorp\_issuer

, 'Not Applicable' as nzbn

, 'Not Applicable' as nz\_society\_actuaries\_id

, 'Not Applicable' as financial\_institution\_giin

, 'Not Applicable' as sponsoring\_entity\_giin

, 'Not Applicable' as nil\_company\_flag

, 'U' as charity\_org\_flag

from dss\_clients c

LEFT OUTER JOIN dss\_client\_names legn

on legn.ird\_number = c.ird\_number

and legn.location\_number = 0

and legn.client\_name\_type = 'P'

LEFT OUTER JOIN dss\_client\_names tradn

on tradn.ird\_number = c.ird\_number

and tradn.location\_number = 1

and tradn.client\_name\_type = 'T'

LEFT OUTER JOIN dss\_customers cu

on cu.ird\_number = c.ird\_number

and cu.location\_number = 1

LEFT OUTER JOIN dss\_entity\_types et

on et.entity\_type\_code = c.entity\_type

LEFT OUTER JOIN dss\_entity\_classes ec

on ec.entity\_class\_code = c.entity\_class

LEFT OUTER JOIN ref\_lannz\_customerclass cla

on cla.fstrCustomerClass = c.entity\_class

LEFT OUTER JOIN dss\_offices o

on o.office\_code = cu.district\_office

LEFT OUTER JOIN dss\_client\_codes ccamd

on cc.ird\_number = c.ird\_number

and client\_code\_value in ('AMD')

LEFT OUTER JOIN dss\_client\_codes ccamg

on cc.ird\_number = c.ird\_number

and client\_code\_value in ('AMG')

LEFT OUTER JOIN dip\_edw\_start\_party\_map pm

on pm.ird\_number = c.ird\_number

LEFT OUTER JOIN local\_start\_bankruptcy brt

on brt.ird\_number = c.ird\_number  
LEFT OUTER JOIN mp\_source\_code mpt

on mpt.dip\_code\_name = ‘PARTY\_TYPE’

and mpt.source\_code\_value = c.cust\_type

LEFT OUTER JOIN mp\_source\_code mpst

on mpst.dip\_code\_name = ‘PARTY\_SUBTYPE’

and mpst.source\_code\_value = c.entity\_type

LEFT OUTER JOIN snz\_BIC\_ANZSIC2006 anzsic

on anzsic.BIC\_code = upper(cu.sic\_code)

where c.cust\_type = 'COM'

EDW Party

select

dip\_party\_sk

,dip\_party\_id

,record\_effective\_timestamp

,record\_expiry\_timestamp

,record\_active\_flag

,common\_party\_id

,party\_origin

,start\_customer\_key

,ird\_no

,customer\_no

,legal\_name

,trading\_name

,party\_start\_date

,party\_ceased\_date

,party\_ceased\_flag

,insolvency\_flag

,party\_type

,party\_type\_code

,party\_subtype

,party\_subtype\_code

,party\_class

,party\_class\_code

,party\_segment

,party\_segment\_code

,BIC\_code

,BIC\_desc

,ANZSIC\_level4\_code

,ANZSIC\_level4\_desc

,ANZSIC\_level3\_code

,ANZSIC\_level3\_desc

,ANZSIC\_level2\_code

,ANZSIC\_level2\_desc

,ANZSIC\_level1\_code

,ANZSIC\_level1\_desc

,security\_level

,security\_level\_code

,district\_office

,os\_taxpayer\_id

,os\_taxpayer\_id\_issuer

,credit\_reported\_flag

,comm\_compliance\_watch\_flag

,suspected\_bank\_fraud\_flag

,suspected\_fraud\_flag

,security\_incident\_rptd\_flag

,safety\_risk\_level

from individual

union

select

dip\_party\_sk

,dip\_party\_id

,record\_effective\_timestamp

,record\_expiry\_timestamp

,record\_active\_flag

,common\_party\_id

,party\_origin

,start\_customer\_key

,ird\_no

,customer\_no

,legal\_name

,trading\_name

,party\_start\_date

,party\_ceased\_date

,party\_ceased\_flag

,insolvency\_flag

,party\_type

,party\_type\_code

,party\_subtype

,party\_subtype\_code

,party\_class

,party\_class\_code

,party\_segment

,party\_segment\_code

,BIC\_code

,BIC\_desc

,ANZSIC\_level4\_code

,ANZSIC\_level4\_desc

,ANZSIC\_level3\_code

,ANZSIC\_level3\_desc

,ANZSIC\_level2\_code

,ANZSIC\_level2\_desc

,ANZSIC\_level1\_code

,ANZSIC\_level1\_desc

,security\_level

,security\_level\_code

,district\_office

,os\_taxpayer\_id

,os\_taxpayer\_id\_issuer

,credit\_reported\_flag

,comm\_compliance\_watch\_flag

,suspected\_bank\_fraud\_flag

,suspected\_fraud\_flag

,security\_incident\_rptd\_flag

,safety\_risk\_level

from organisation

# Appendix F – Sample SQL for Context – Start

The ‘Select’ column names in these SQL statements will be referred to in the “Source Table/Column” fields of the transformation rules and are intended to give context to those rules as well as a guide on how to merge all the source objects into the final Individual & Organisation data streams.

Start Individual

select

1000001 -- DIP\_PARTY\_SK (generated surrogate key)

,'xxx' -- DIP\_PARTY\_ID (generated as natural key of the Party)

,brpt.individual\_party\_id -- COMMON\_PARTY\_ID (use to generate another DIP\_PARTY\_ID)

,'START' -- PARTY\_ORIGIN

,c.flngCustomerKey -- START\_CUSTOMER\_KEY

,ird.fstrid -- IRD\_NO

,cusnum.fstrid -- CUSTOMER\_NO

,lglnm.fstrtitle -- Use to construct LEGAL\_NAME

,lglnm.fstrfirstname -- Use to construct LEGAL\_NAME

,lglnm.fstrmiddlename -- Use to construct LEGAL\_NAME

,lglnm.fstrlastname -- Use to construct LEGAL\_NAME

,trading.fstrlistformatname -- TRADING\_NAME

,c.fdtmcommence -- Use to generate PARTY\_START\_DATE

,c.fdtmcreated -- Use to generate PARTY\_START\_DATE

,c.fdtmcease -- Use to generate PARTY\_CEASED\_DATE and PARTY\_CEASED\_FLAG

,cstd.fdtmdob -- Use to generate PARTY\_START\_DATE and DATE\_OF\_BIRTH

,cstd.fdtmdod -- Use to generate DATE\_OF\_DEATH

,inslv.fstrindicator -- Use codes to generate INSOLVENCY\_FLAG

,ct1.fstrdecode2 -- Use to generate PARTY\_TYPE

,ct2.fstrdecode2 -- Use to generate PARTY\_TYPE

,c.fstrcustomertype -- Use to generate PARTY\_TYPE\_CODE

,ch.fstrCustomerTypeTo -- Use to generate PARTY\_TYPE\_CODE

,st.fstrDecode2 -- PARTY\_SUBTYPE

,cz.fstrCustomerSubType -- PARTY\_SUBTYPE\_CODE

,cla.fstrDecode2 -- PARTY\_CLASS

,cz.fstrCustomerClass -- PARTY\_CLASS\_CODE

,seg.fstrdecode2 -- PARTY\_SEGMENT

,sgmt.fstrCustomerSegment -- PARTY\_SEGMENT\_CODE

,naics.fstrnaics -- BIC\_CODE

,anzsic.BIC\_desc -- BIC\_DESC

,anzsic.ANZSIC\_level4\_code -- ANZSIC\_level4\_code

,anzsic.ANZSIC\_level4\_desc -- ANZSIC\_level4\_desc

,anzsic.ANZSIC\_level3\_code -- ANZSIC\_level3\_code

,anzsic.ANZSIC\_level3\_desc -- ANZSIC\_level3\_desc

,anzsic.ANZSIC\_level2\_code -- ANZSIC\_level2\_code

,anzsic.ANZSIC\_level2\_desc -- ANZSIC\_level2\_desc

,anzsic.ANZSIC\_level1\_code -- ANZSIC\_level1\_code

,anzsic.ANZSIC\_level1\_desc -- ANZSIC\_level1\_desc

,cldesc.fstrdecode2 -- SECURITY\_LEVEL

,cuslvl.fstrcustomerlevel -- SECURITY\_LEVEL\_CODE

,doff.fstrdecode1 -- DISTRICT\_OFFICE

,fa.fstrid -- NZ\_FIREARMS\_LICENCE\_ID

,txidnm.fstrid -- OS\_TAXPAYER\_ID

,txidnmi.fstrissuingcountry -- OS\_TAXPAYER\_ID\_ISSUER

,acompc.fstrid -- ACC\_ID

,mbclnm.fstrid -- MBIE\_CLIENT\_ID

,credt.fstrindicator -- CREDIT\_REPORTED\_FLAG

,comcom.fstrindicator -- COMM\_COMPLIANCE\_WATCH\_FLAG

,bfrd.fstrindicator -- SUSPECTED\_BANK\_FRAUD\_FLAG

,frd.fstrindicator -- SUSPECTED\_FRAUD\_FLAG

,seci.fstrindicator -- SECURITY\_INCIDENT\_RPTD\_FLAG

,safe.fstrindicator -- SAFETY\_RISK\_LEVEL

,pref.fstrtitle -- PREF\_TITLE

,pref.fstrfirstname -- PREF\_GIVEN\_NAME

,pref.fstrmiddlename -- PREF\_OTHER\_GIVEN\_NAME

,pref.fstrlastname -- PREF\_FAMILY\_NAME

,nzbcid.fstrid -- used to derive BIRTH\_CERT\_ID

,nzobcid.fstrid -- used to derive BIRTH\_CERT\_ID

,ovbrth.fstrid -- used to derive BIRTH\_CERT\_ID

,ovbrthi.fstrissuingcountry -- OS\_BIRTH\_CERT\_ISSUER

,nzpprt.fstrid -- NZ\_PASSPORT\_ID

,ospprt.fstrid -- OS\_PASSPORT\_ID

,ospprti.fstrissuingcountry -- OS\_PASSPORT\_ISSUER

,nzcidd.fstrid -- NZ\_CERT\_IDENTITY\_DIA

,nzcidm.fstrid -- NZ\_CERT\_IDENTITY\_MBIE

,nzrfgdia.fstrid -- NZ\_REFUGEE\_ID\_DIA

,nzrfgmb.fstrid -- NZ\_REFUGEE\_ID\_MBIE

,nzetdc.fstrid -- NZ\_EMERGENCY\_TRAVEL\_ID

,nzdlic.fstrid -- used to derive DRIVER\_LICENCE\_ID

,osdlic.fstrid -- used to derive DRIVER\_LICENCE\_ID

,osdlici.fstrissuingcountry -- DRIVER\_LICENCE\_ISSUER

,intdrv.fstrid -- INTL\_DRIVER\_PERMIT\_ID

,hanz18.fstrid -- NZ\_HOSPITALITY\_18OVER\_ID

,nzstdn.fstrid -- NZ\_STUDENT\_ID

,nzstdne.fstreducationprovider -- NZ\_STUDENT\_ISSUER

,swn.fstrid -- NZ\_SOCIAL\_WELFARE\_ID

,ssn.fstrid -- US\_SOCIAL\_SECURITY\_NO

,fein.fstrid -- FOREIGN\_EMPLOYMENT\_ID

,ncp.fstrid -- NCP\_ID

,ortai.fstrid -- MINISTRY\_OF\_CHILDREN\_ID

,emgrtd.fstrindicator -- EMIGRATED\_FLAG

,bdralt.fstrindicator -- BORDER\_ALERT\_FLAG

,prison.fstrindicator -- IN\_PRISON\_FLAG

,undsch.fstrindicator -- UNDISCHARGED\_BANKRUPT\_FLAG

--Customer--------------------------

from app\_tblCustomer c

LEFT OUTER JOIN app\_tblcustomerinfo ci

on ci.flngcustomerkey = c.flngcustomerkey

LEFT OUTER JOIN app\_tblnz\_customerstd cz

on cz.flngDocKey = c.flngDocKey

LEFT OUTER JOIN app\_tblcustomerlevel cuslvl

on cuslvl.flngdockey = c.flngdockey

LEFT OUTER JOIN app\_tblcustomerhistory ch

on ch.flngcustomerkey = c.flngcustomerkey

--Name Records--------------------

LEFT OUTER JOIN app\_tblnamerecord pref

on pref.flngcustomerkey = c.flngcustomerkey

and pref.fstrnametype = 'PREFER'

LEFT OUTER JOIN app\_tblnamerecord lglnm

on lglnm.flngcustomerkey = c.flngcustomerkey

and lglnm.fstrnametype = 'LGL'

LEFT OUTER JOIN app\_tblnamerecord trading

on trading.flngcustomerKey = c.flngCustomerKey

and trading.fstrnametype = 'DBACST'

--IDs-----------------------------------

LEFT OUTER JOIN app\_tblid\_ird ird

on ird.flngcustomerkey = c.flngCustomerKey

and ird.fstridtype = 'IRD'

LEFT OUTER JOIN app\_tblid nzbcid

on nzbcid.flngcustomerkey = c.flngCustomerKey

and nzbcid.fstridtype = ‘NZBRTH’

LEFT OUTER JOIN app\_tblid nzobcid

on nzobcid.flngcustomerkey = c.flngCustomerKey

and nzobcid.fstridtype = 'OLBRTH'

LEFT OUTER JOIN app\_tblid ovbrth

on ovbrth.flngcustomerkey = c.flngCustomerKey

and ovbrth.fstridtype = 'OVBRTH'

LEFT OUTER JOIN app\_tblid nzpprt

on nzpprt.flngcustomerkey = c.flngCustomerKey

and nzpprt.fstridtype = 'NZPPRT'

LEFT OUTER JOIN app\_tblid ospprt

on ospprt.flngcustomerkey = c.flngCustomerKey

and ospprt.fstridtype = 'OSPPRT'

LEFT OUTER JOIN app\_tblid nzcidd

on nzcidd.flngcustomerkey = c.flngCustomerKey

and nzciddd.fstridtype = 'NZCIDD'

LEFT OUTER JOIN app\_tblid nzcidm

on nzcidm.flngcustomerkey = c.flngCustomerKey

and nzcidm.fstridtype = 'NZCIDM'

LEFT OUTER JOIN app\_tblid nzrfgdia

on nzrfgdia.flngcustomerkey = c.flngCustomerKey

and nzrfgdia.fstridtype = 'NZRFGT'

LEFT OUTER JOIN app\_tblid nzrfgmb

on nzrfgmb.flngcustomerkey = c.flngCustomerKey

and nzrfgmb.fstridtype = 'NZRFMB'

LEFT OUTER JOIN app\_tblid nzetdc

on nzetdc.flngcustomerkey = c.flngCustomerKey

and nzetdc.fstridtype = 'NZETDC'

LEFT OUTER JOIN app\_tblid nzdlic

on nzdlic.flngcustomerkey = c.flngCustomerKey

and nzdlic.fstridtype = 'NZDLIC'

LEFT OUTER JOIN app\_tblid osdlic

on osdlic.flngcustomerkey = c.flngCustomerKey

and osdlic.fstridtype = 'OVDLIC'

LEFT OUTER JOIN app\_tblid intdrv

on intdrv.flngcustomerkey = c.flngCustomerKey

and intdrv.fstridtype = 'INTDRV'

LEFT OUTER JOIN app\_tblid hanz18

on hanz18.flngcustomerkey = c.flngCustomerKey

and hanz18.fstridtype = 'HANZ18'

LEFT OUTER JOIN app\_tblid nzstdn

on nzstdn.flngcustomerkey = c.flngCustomerKey

and nzstdn.fstridtype = 'NZSTDN'

LEFT OUTER JOIN app\_tblid swn

on swn.flngcustomerkey = c.flngCustomerKey

and swn.fstridtype = 'SWN'

LEFT OUTER JOIN app\_tblid ssn

on ssn.flngcustomerkey = c.flngCustomerKey

and ssn.fstridtype = 'SSN'

LEFT OUTER JOIN app\_tblid fein

on fein.flngcustomerkey = c.flngCustomerKey

and fein.fstridtype = 'FEIN'

LEFT OUTER JOIN app\_tblid ncp

on ncp.flngcustomerkey = c.flngCustomerKey

and ncp.fstridtype = 'NCP'

LEFT OUTER JOIN app\_tblid ortai

on ortai.flngcustomerkey = c.flngCustomerKey

and ortai.fstridtype = 'ORTAI'

LEFT OUTER JOIN app\_tblid fa

on fa.flngcustomerkey = c.flngcustomerkey

and fa.fstridtype = 'NZFRAM'

LEFT OUTER JOIN app\_tblid txidnm

on txidnm.flngcustomerkey = c.flngCustomerKey

and txidnm.fstridtype = 'TXIDNM'

LEFT OUTER JOIN app\_tblid acompc

on acompc.flngcustomerkey = c.flngCustomerKey

and acompc.fstridtype = 'ACOMPC'

LEFT OUTER JOIN app\_tblid mbclnm

on mbclnm.flngcustomerkey = c.flngCustomerKey

and mbclnm.fstridtype = 'MBCLNM'

LEFT OUTER JOIN app\_tblid cusnum

on cusnum.flngcustomerkey = c.flngCustomerKey

and cusnum.fstridtype = 'CST'

--Indicators--------------------------

LEFT OUTER JOIN app\_tblindicator emgrtd

on emgrtd.flngcustomerkey = c.flngcustomerkey

and emgrtd.fstrindicator = 'EMGRTD'

LEFT OUTER JOIN app\_tblindicator bdralt

on bdralt.flngcustomerkey = c.flngcustomerkey

and bdralt.fstrindicator = 'BDRALT'

LEFT OUTER JOIN app\_tblindicator prison

on prison.flngcustomerkey = c.flngcustomerkey

and prison.fstrindicator = 'PRISON'

LEFT OUTER JOIN app\_tblindicator undsch

on undsch.flngcustomerkey = c.flngcustomerkey

and undsch.fstrindicator = 'UNDSCH'

LEFT OUTER JOIN app\_tblindicator credt

on credt.flngcustomerkey = c.flngCustomerKey

and credt.fstrindicator = 'CRDRPD'

LEFT OUTER JOIN app\_tblindicator comcom

on comcom.flngcustomerkey = c.flngCustomerKey

and comcom.fstrindicator = 'CSCCOM'

LEFT OUTER JOIN app\_tblindicator bfrd

on bfrd.flngcustomerkey = c.flngCustomerKey

and bfrd.fstrindicator = 'FRDBNK'

LEFT OUTER JOIN app\_tblindicator frd

on frd.flngcustomerkey = c.flngCustomerKey

and frd.fstrindicator = 'KNWFRD'

LEFT OUTER JOIN app\_tblindicator seci

on seci.flngcustomerkey = c.flngCustomerKey

and seci.fstrindicator = 'DNGCUS'

LEFT OUTER JOIN app\_tblindicator\_sec safe

on safe.flngcustomerkey = c.flngCustomerKey

LEFT OUTER JOIN app\_tblindicator\_inslv inslv

on inslv.flngcustomerkey = c.flngCustomerKey

--DocKeys----------------------------

LEFT OUTER JOIN app\_tblcustomerstd cstd

on c.flngdockey = cstd.flngdockey

LEFT OUTER JOIN app\_tblNZ\_AccIDGovr ovbrthi

on ovbrthi.flngDocKey = ovbrth.flngdockey

LEFT OUTER JOIN app\_tblNZ\_AccIDGovr nzdlici

on nzdlici.flngDocKey = nzdlic.flngdockey

LEFT OUTER JOIN app\_tblNZ\_AccIDGovr ospprti

on ospprti.flngDocKey = ospprt.flngdockey

LEFT OUTER JOIN app\_tblNZ\_AccIDGovr osdlici

on osdlici.flngDocKey = osdlic.flngdockey

LEFT OUTER JOIN app\_tblNZ\_AccIDGEdu nzstdne

on nzstdne.flngDocKey = nzstdn.flngdockey

LEFT OUTER JOIN app\_tblnz\_accidgovr txidnmi

on txidnmi.flngdockey = txidnm.flngdockey

LEFT OUTER JOIN app\_tblnz\_custsegmenthistory sgmt  
on sgmt.flngdockey = c.flngdockey

--Other--------------------------------

LEFT OUTER JOIN app\_tblnaics naics

on naics.flngcustomerkey = c.flngCustomerKey

LEFT OUTER JOIN snz\_BIC\_ANZSIC2006 anzsic

on anzsic.BIC\_code = upper(naics.fstrnaics)

LEFT OUTER JOIN dev\_raw.local\_start\_bankruptcy brpt

on ird\_number = ird.fstrid

--Reference--------------------------

LEFT OUTER JOIN ref\_lancustomertype ct1

on ct1.fstrcustomertype = c.fstrCustomerType

LEFT OUTER JOIN ref\_lancustomertype ct2

on ct2.fstrcustomertype = ch.fstrCustomerTypeTo

LEFT OUTER JOIN ref\_lanCustomerLevel cldesc

on cuslvl.fstrcustomerlevel = cldesc.fstrcustomerlevel

LEFT OUTER JOIN ref\_lancustomersubtype st

on st.fstrcustomersubtype = cz.fstrcustomersubtype

LEFT OUTER JOIN ref\_lannz\_customerclass cla

on cla.fstrCustomerClass = cz.fstrCustomerClass

LEFT OUTER JOIN ref\_lannz\_customersegments seg

on seg.fstrSegment = substring(sgmt.fstrindicator,4,3)

LEFT OUTER JOIN ref\_lanofficelocation doff

on doff.fstrdistrictoffice = ci.fstrDistrictOffice

;

Start Organisation

select

1000001 -- DIP\_PARTY\_SK (generated surrogate key)

,'xxx' -- DIP\_PARTY\_ID (generated as natural key of the Party)

,NULL -- COMMON\_PARTY\_ID (use to generate another DIP\_PARTY\_ID)

,'START' -- PARTY\_ORIGIN

,c.flngCustomerKey -- START\_CUSTOMER\_KEY

,ird.fstrid -- IRD\_NO

,cusnum.fstrid -- CUSTOMER\_NO

,lglnm.fstrlistformatname -- LEGAL\_NAME

,trading.fstrlistformatname -- TRADING\_NAME

,c.fdtmcommence -- Use to generate PARTY\_START\_DATE & ORG\_INCORP\_DATE

,c.fdtmcreated -- Use to generate PARTY\_START\_DATE

,c.fdtmcease -- Use to generate PARTY\_CEASED\_DATE + PARTY\_CEASED\_FLAG + ORG\_CEASE\_DATE

,inslv.fstrindicator -- Use codes to generate INSOLVENCY\_FLAG

,ct1.fstrdecode2 -- Use to generate PARTY\_TYPE

,ct2.fstrdecode2 -- Use to generate PARTY\_TYPE

,c.fstrcustomertype -- Use to generate PARTY\_TYPE\_CODE

,ch.fstrCustomerTypeTo -- Use to generate PARTY\_TYPE\_CODE

,st.fstrDecode2 -- PARTY\_SUBTYPE

,cz.fstrCustomerSubType -- PARTY\_SUBTYPE\_CODE

,cla.fstrDecode2 -- PARTY\_CLASS

,cz.fstrCustomerClass -- PARTY\_CLASS\_CODE

,seg.fstrdecode2 -- PARTY\_SEGMENT

,sgmt.fstrCustomerSegment -- PARTY\_SEGMENT\_CODE

,naics.fstrnaics -- BIC\_CODE

,anzsic.BIC\_desc -- BIC\_DESC

,anzsic.ANZSIC\_level4\_code -- ANZSIC\_level4\_code

,anzsic.ANZSIC\_level4\_desc -- ANZSIC\_level4\_desc

,anzsic.ANZSIC\_level3\_code -- ANZSIC\_level3\_code

,anzsic.ANZSIC\_level3\_desc -- ANZSIC\_level3\_desc

,anzsic.ANZSIC\_level2\_code -- ANZSIC\_level2\_code

,anzsic.ANZSIC\_level2\_desc -- ANZSIC\_level2\_desc

,anzsic.ANZSIC\_level1\_code -- ANZSIC\_level1\_code

,anzsic.ANZSIC\_level1\_desc -- ANZSIC\_level1\_desc

,cldesc.fstrdecode2 -- SECURITY\_LEVEL

,cuslvl.fstrcustomerlevel -- SECURITY\_LEVEL\_CODE

,doff.fstrdecode1 -- DISTRICT\_OFFICE

,fa.fstrid -- NZ\_FIREARMS\_LICENCE\_ID

,txidnm.fstrid -- OS\_TAXPAYER\_ID

,txidnmi.fstrissuingcountry -- OS\_TAXPAYER\_ID\_ISSUER

,acompc.fstrid -- ACC\_ID

,mbclnm.fstrid -- MBIE\_CLIENT\_ID

,credt.fstrindicator -- CREDIT\_REPORTED\_FLAG

,comcom.fstrindicator -- COMM\_COMPLIANCE\_WATCH\_FLAG

,bfrd.fstrindicator -- SUSPECTED\_BANK\_FRAUD\_FLAG

,frd.fstrindicator -- SUSPECTED\_FRAUD\_FLAG

,seci.fstrindicator -- SECURITY\_INCIDENT\_RPTD\_FLAG

,safe.fstrindicator -- SAFETY\_RISK\_LEVEL

,stroff.fstrindicator -- STRUCK\_OFF\_FLAG

,amlgmd.fstrindicator -- AMALGAMATED\_FLAG

,amlgmg.fstrindicator -- AMALGAMATING\_FLAG

,nzcinc.fstrid -- NZ\_CERT\_INCORP\_ID

,oscinc.fstrid -- OS\_CERT\_INCORP\_ID

,oscinci.fstrissuingcountry -- OS\_CERT\_INCORP\_ISSUER

,nzbn.fstrid -- NZBN

,govact.fstrid -- NZ\_SOCIETY\_ACTUARIES\_ID

,giin.fstrid -- FINANCIAL\_INSTITUTION\_GIIN

,segiin.fstrid -- SPONSORING\_ENTITY\_GIIN

,com.fblnnilcompany -- NIL\_COMPANY\_FLAG

,chrity.fstrindicator -- CHARITY\_ORG\_FLAG

--Customer--------------------------

from app\_tblCustomer c

LEFT OUTER JOIN app\_tblcustomerinfo ci

on ci.flngcustomerkey = c.flngcustomerkey

LEFT OUTER JOIN app\_tblnz\_customerstd cz

on cz.flngDocKey = c.flngDocKey

LEFT OUTER JOIN app\_tblcustomerlevel cuslvl

on cuslvl.flngdockey = c.flngdockey

LEFT OUTER JOIN app\_tblcustomerhistory ch

on ch.flngcustomerkey = c.flngcustomerkey

--Name Records--------------------

LEFT OUTER JOIN app\_tblnamerecord lglnm

on lglnm.flngcustomerkey = c.flngcustomerkey

and lglnm.fstrnametype = 'LGL'

LEFT OUTER JOIN app\_tblnamerecord trading

on trading.flngcustomerKey = c.flngCustomerKey

and trading.fstrnametype = 'DBACST'

--IDs-----------------------------------

LEFT OUTER JOIN app\_tblid ird

on ird.flngcustomerkey = c.flngCustomerKey

and ird.fstridtype = 'IRD'

LEFT OUTER JOIN app\_tblid cusnum

on cusnum.flngcustomerkey = c.flngCustomerKey

and cusnum.fstridtype = 'CST'

LEFT OUTER JOIN app\_tblid fa

on fa.flngcustomerkey = c.flngcustomerkey

and fa.fstridtype = 'NZFRAM'

LEFT OUTER JOIN app\_tblid txidnm

on txidnm.flngcustomerkey = c.flngCustomerKey

and txidnm.fstridtype = 'TXIDNM'

LEFT OUTER JOIN app\_tblid acompc

on acompc.flngcustomerkey = c.flngCustomerKey

and acompc.fstridtype = 'ACOMPC'

LEFT OUTER JOIN app\_tblid mbclnm

on mbclnm.flngcustomerkey = c.flngCustomerKey

and mbclnm.fstridtype = 'MBCLNM'

LEFT OUTER JOIN app\_tblid nzcinc

on nzcinc.flngcustomerkey = c.flngCustomerKey

and nzcinc.fstridtype = 'NZCINC'

LEFT OUTER JOIN app\_tblid oscinc

on oscinc.flngcustomerkey = c.flngCustomerKey

and oscinc.fstridtype = 'OSCINC'

LEFT OUTER JOIN app\_tblid nzbn

on nzbn.flngcustomerkey = c.flngCustomerKey

and nzbn.fstridtype = 'NZBN'

LEFT OUTER JOIN app\_tblid govact

on govact.flngcustomerkey = c.flngCustomerKey

and govact.fstridtype = 'GOVACT'

LEFT OUTER JOIN app\_tblid giin

on giin.flngcustomerkey = c.flngCustomerKey

and giin.fstridtype = 'GIIN'

LEFT OUTER JOIN app\_tblid segiin

on segiin.flngcustomerkey = c.flngCustomerKey

and segiin.fstridtype = 'SEGIIN'

--Indicators--------------------------

LEFT OUTER JOIN app\_tblindicator credt

on credt.flngcustomerkey = c.flngCustomerKey

and credt.fstrindicator = 'CRDRPD'

LEFT OUTER JOIN app\_tblindicator comcom

on comcom.flngcustomerkey = c.flngCustomerKey

and comcom.fstrindicator = 'CSCCOM'

LEFT OUTER JOIN app\_tblindicator bfrd

on bfrd.flngcustomerkey = c.flngCustomerKey

and bfrd.fstrindicator = 'FRDBNK'

LEFT OUTER JOIN app\_tblindicator frd

on frd.flngcustomerkey = c.flngCustomerKey

and frd.fstrindicator = 'KNWFRD'

LEFT OUTER JOIN app\_tblindicator seci

on seci.flngcustomerkey = c.flngCustomerKey

and seci.fstrindicator = 'DNGCUS'

LEFT OUTER JOIN app\_tblindicator stroff

on stroff.flngcustomerkey = c.flngcustomerkey

and stroff.fstrindicator = 'STROFF'

LEFT OUTER JOIN app\_tblindicator amlgmd

on amlgmd.flngcustomerkey = c.flngcustomerkey

and amlgmd.fstrindicator = 'AMLGMD'

LEFT OUTER JOIN app\_tblindicator amlgmg

on amlgmg.flngcustomerkey = c.flngcustomerkey

and amlgmg.fstrindicator = 'AMLGMG'

LEFT OUTER JOIN app\_tblindicator chrity

on chrity.flngcustomerkey = c.flngcustomerkey

and chrity.fstrindicator = 'CHRITY'

LEFT OUTER JOIN app\_tblindicator\_inslv inslv

on inslv.flngcustomerkey = c.flngCustomerKey

LEFT OUTER JOIN app\_tblindicator\_sec safe

on safe.flngcustomerkey = c.flngCustomerKey

--DocKeys----------------------------

LEFT OUTER JOIN app\_tblcustomerstd cstd

on c.flngdockey = cstd.flngdockey

LEFT OUTER JOIN app\_tblnz\_accidgovr txidnmi

on txidnmi.flngdockey = txidnm.flngdockey

LEFT OUTER JOIN app\_tblNZ\_AccIDGovr oscinci

on oscinci.flngDocKey = oscinc.flngdockey

LEFT OUTER JOIN app\_tblNz\_cstcominfo com

on com.flngDocKey = c.flngDocKey

LEFT OUTER JOIN app\_tblnz\_custsegmenthistory sgmt  
on sgmt.flngdockey = c.flngdockey

--Others------------------------------

LEFT OUTER JOIN app\_tblnaics naics

on naics.flngcustomerkey = c.flngCustomerKey

LEFT OUTER JOIN snz\_BIC\_ANZSIC2006 anzsic

on anzsic.BIC\_code = upper(naics.fstrnaics)

--Reference--------------------------

LEFT OUTER JOIN ref\_lancustomertype ct1

on ct1.fstrcustomertype = c.fstrCustomerType

LEFT OUTER JOIN ref\_lancustomertype ct2

on ct2.fstrcustomertype = ch.fstrCustomerTypeTo

LEFT OUTER JOIN app\_lanCustomerLevel cldesc

on cldesc.fstrcustomerlevel = cuslvl.fstrcustomerlevel

LEFT OUTER JOIN ref\_lancustomersubtype st

on st.fstrcustomersubtype = cz.fstrcustomersubtype

LEFT OUTER JOIN ref\_lannz\_customerclass cla

on cla.fstrCustomerClass = cz.fstrCustomerClass

LEFT OUTER JOIN ref\_lannz\_customersegments seg

on seg.fstrSegment = substring(sgmt.fstrindicator,4,3)

LEFT OUTER JOIN ref\_lanofficelocation doff

on doff.fstrdistrictoffice = ci.fstrDistrictOffice

;

Start Party

select

dip\_party\_sk

,dip\_party\_id

,record\_effective\_timestamp

,record\_expiry\_timestamp

,record\_active\_flag

,common\_party\_id

,party\_origin

,customer\_key

,ird\_number

,customer\_no

,legal\_name

,trading\_name

,party\_start\_date

,party\_ceased\_date

,party\_ceased\_flag

,insolvency\_flag

,party\_type

,party\_type\_code

,party\_subtype

,party\_subtype\_code

,party\_class

,party\_class\_code

,party\_segment

,party\_segment\_code

,BIC\_code

,BIC\_desc

,ANZSIC\_level4\_code

,ANZSIC\_level4\_desc

,ANZSIC\_level3\_code

,ANZSIC\_level3\_desc

,ANZSIC\_level2\_code

,ANZSIC\_level2\_desc

,ANZSIC\_level1\_code

,ANZSIC\_level1\_desc

,security\_level

,security\_level\_code

,district\_office

,os\_taxpayer\_id

,os\_taxpayer\_id\_issuer

,credit\_reported\_flag

,comm\_compliance\_watch\_flag

,suspected\_bank\_fraud\_flag

,suspected\_fraud\_flag

,security\_incident\_rptd\_flag

,safety\_risk\_level

FROM individual

UNION

select

dip\_party\_sk

,dip\_party\_id

,record\_effective\_timestamp

,record\_expiry\_timestamp

,record\_active\_flag

,common\_party\_id

,party\_origin

,customer\_key

,ird\_number

,customer\_no

,legal\_name

,trading\_name

,party\_start\_date

,party\_ceased\_date

,party\_ceased\_flag

,insolvency\_flag

,party\_type

,party\_type\_code

,party\_subtype

,party\_subtype\_code

,party\_class

,party\_class\_code

,party\_segment

,party\_segment\_code

,BIC\_code

,BIC\_desc

,ANZSIC\_level4\_code

,ANZSIC\_level4\_desc

,ANZSIC\_level3\_code

,ANZSIC\_level3\_desc

,ANZSIC\_level2\_code

,ANZSIC\_level2\_desc

,ANZSIC\_level1\_code

,ANZSIC\_level1\_desc

,security\_level

,security\_level\_code

,district\_office

,os\_taxpayer\_id

,os\_taxpayer\_id\_issuer

,credit\_reported\_flag

,comm\_compliance\_watch\_flag

,suspected\_bank\_fraud\_flag

,suspected\_fraud\_flag

,security\_incident\_rptd\_flag

,safety\_risk\_level

FROM organisation

Derived table: rfn\_ps\_bankruptcy

SELECT distinct

cast(a.fstrid as int) as ird\_no, -- IRD\_NO

c.common\_dip\_party\_id -- COMMON\_DIP\_PARTY\_ID

FROM crw\_ps\_tblid\_ird as a

INNER JOIN crw\_ps\_tblid\_ird as b

on a.flngcustomerkey = b.flngcustomerkey

and a.fstrid <> b.fstrid

INNER JOIN local\_start\_bankruptcy as c

on cast(b.fstrid as int) = c.ird\_no

UNION

SELECT lsb.ird\_no, -- IRD\_NO

lsb.common\_dip\_party\_id -- COMMON\_DIP\_PARTY\_ID

FROM local\_start\_bankruptcy lsb