# 

|  | | |
| --- | --- | --- |
|  | |  |
|  |  | ODS Low Level Design |
|  |  | TSB |
| November 9, 2020 |
|  |  | | **Proposal No.** |  | | --- | --- | | Version No. | 1.0 | | Authorized by |  | |

Review History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Reviewed by** | **Comments** |
| 1.0 |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Data Origin 5](#_Toc55913421)

[1.1 Source tables information 5](#_Toc55913422)

[1.2 Source tables relationship view 6](#_Toc55913423)

[1.3 Sample Data 6](#_Toc55913424)

[1.4 Source Volumetric 6](#_Toc55913425)

[2 ODS PCA SAVINGS Data Model 7](#_Toc55913426)

[2.1 ODS PCA SAVINGS Raw Layer Data Model 7](#_Toc55913427)

[2.1.1 RAW LAYER TABLES 7](#_Toc55913428)

[2.1.2 PCA SAVINGS RAW DATA MODEL 7](#_Toc55913429)

[2.2 PCA SAVINGS Conceptual Model 9](#_Toc55913430)

[2.3 Data Access/Query Pattern 10](#_Toc55913431)

[2.4 PCA SAVINGS Curated Logical Data Model 10](#_Toc55913432)

[2.4.1 Version 1 10](#_Toc55913433)

[2.4.2 Version 2 11](#_Toc55913434)

[2.5 Proposed Physical Curated Data model 11](#_Toc55913435)

[2.5.1 CURATED LAYER TABLES: - 11](#_Toc55913436)

[3 ODS PCA SAVINGS Data Catalogue/Data Mapping 13](#_Toc55913437)

[3.1 High Level -Raw and Curated Source -Target Model table mapping relation 13](#_Toc55913438)

[3.2 Low Level Data Mapping 13](#_Toc55913439)

[3.2.1 TRANSACTIONS TABLES: - 13](#_Toc55913440)

[3.2.2 ACCOUNT TABLES: - 13](#_Toc55913441)

[3.2.3 CUSTOMER TABLES: - 13](#_Toc55913442)

[3.2.4 Data Model Design Decisions 13](#_Toc55913443)

[4 Application Design: 14](#_Toc55913444)

[4.1 High level Transaction Data Flow 14](#_Toc55913445)

[4.2 Low Level Data Flow (Transaction) 14](#_Toc55913446)

[4.2.1 Full Transactions Data Flow Diagram: 14](#_Toc55913447)

[4.2.2 Topology Design 14](#_Toc55913448)

[4.2.3 Data Processor Details 16](#_Toc55913449)

[4.2.4 Kafka Topics 16](#_Toc55913450)

[4.3 ODS ScyllaDB Application Configuration 17](#_Toc55913451)

[4.3.1 Keys space configuration 17](#_Toc55913452)

[4.3.2 ODS Target Table Names with DDL 18](#_Toc55913453)

[4.3.3 ScyllaDB SSL 18](#_Toc55913454)

[4.3.4 ScyllaDB Current Setup 20](#_Toc55913455)

[4.4 Kafka Configuration 20](#_Toc55913456)

[4.4.1 Current Kafka Setup 20](#_Toc55913457)

[4.4.2 Kafka Topic configuration 21](#_Toc55913458)

[4.4.3 Kafka SSL Connectivity and authentication 21](#_Toc55913459)

[4.5 AWS EKS Configuration 22](#_Toc55913460)

[4.5.1 Deployment process 22](#_Toc55913461)

[4.5.2 AWS EKS – Application Topology configuration 22](#_Toc55913462)

[4.6 Security Module: 23](#_Toc55913463)

[4.7 One time/Historical load for Target tables (Raw/Curated) 23](#_Toc55913464)

[4.8 Data Reconciliation 24](#_Toc55913465)

[4.9 Refresh of ODS tables on Load errors 24](#_Toc55913466)

[4.10 Resiliency in Data ingestion Pipeline 24](#_Toc55913467)

[4.10.1 Fault Tolerance 24](#_Toc55913468)

[4.11 Data Quality 26](#_Toc55913469)

[4.12 Monitoring/Logging 26](#_Toc55913470)

[5 Ciso Integration /Certification 28](#_Toc55913471)

[6 DevOps 29](#_Toc55913472)

[7 Appendix 30](#_Toc55913473)

[7.1.1 Kafka topics Name: 30](#_Toc55913474)

[7.1.2 Consume exactly Once: 30](#_Toc55913475)

[7.1.3 Commit current Offset 30](#_Toc55913476)

# Data Origin

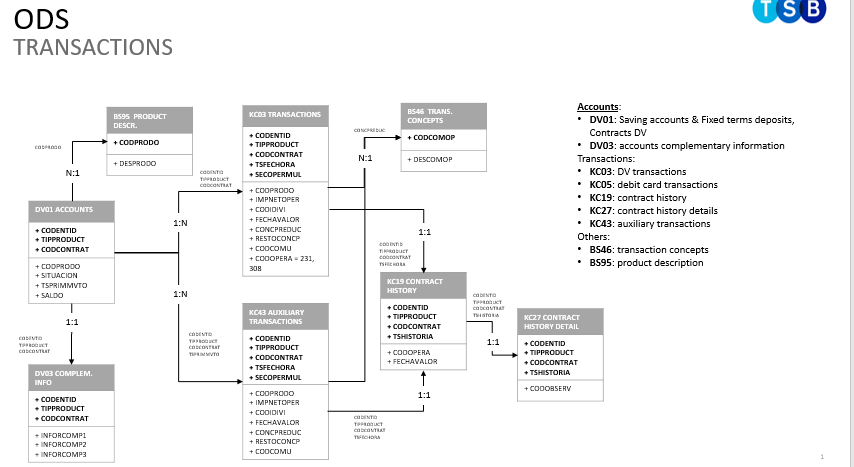
## Source tables information

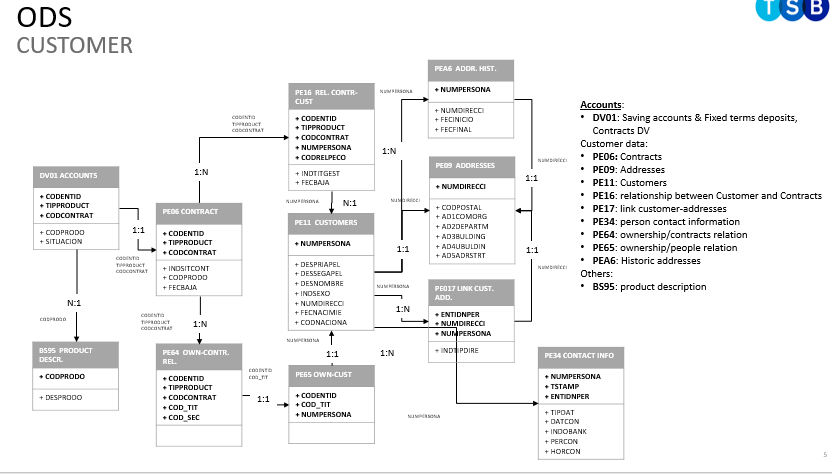
Below is the list of source tables in considered for ODS from Proteo

Below mentioned table represents DB2 source table name and details

|  |  |  |  |
| --- | --- | --- | --- |
| **DB2 Source Table** | **Source Description** | **Primary Keys** | **Remarks** |
| DV01 | Saving Accounts & Fixed Term Deposits, Contracts DV (Accounts) | CODENTID,  TIPPRODUCT,  CODCONTRAT | **CODENTID/TIPPRODUCT/CODCONTRAT:** Unique Id of the Customer account |
| BS95 | Product Description Table | CODPRODO | CODPRODO: **Code of product. This link this table with KC03 or DV01** |
| BS01 | Account Extra Information Table | CODENTID |  |
| KC03 | Current Transaction Table | CODENTID,  TIPPRODUCT,  CODCONTRAT,  TSFECHORA,  SECOPERMUL | **CODENTID/TIPPRODUCT/CODCONTRAT:** Unique Id of the Customer account  TSFECHORA Timestamp when transaction record gets created in KC03 |
| KC43 | History Transaction Table | CODENTID,  TIPPRODUCT,  CODCONTRAT,  TRANSACTION\_MONTH,  TSFECHORA,  SECOPERMUL | Transaction Month: - Format: - MMYYYY  Where MMYYYYY=FECHAVALOR column's sub strings for getting Month and Year in mentioned format  TSFECHORA Time when transaction record is created in KC43 |
| BS46 | Transaction Type Table | CODCOMOP |  |
| PE11 | Customer Table | NUMPERSONA | NUMPERSONA**: Internal Id of the customer** |
| PE34 | Customer Contact Table | NUMPERSONA, TSTAMP,  ENTIDNPER |  |
| PE09 | Customer Address Table | NUMDIRECCI |  |
| PEA6 | Customer Address History Table | NUMPERSONA |  |
| PE17 | Customer Address Link Table | NUMDIRECCI,  NUMPERSONA,  ENTIDNPER |  |
| PE06 | Contract Table | CODENTID,  TIPPRODUCT,  CODCONTRAT | **CODENTID/TIPPRODUCT/CODCONTRAT:** Unique Id of the Customer account |
| PE16 | Relationship between Customer and Contracts | CODENTID,  TIPPRODUCT,  CODCONTRAT,  NUMPERSONA,  CODRELPECO: | CODENTID/TIPPRODUCT/CODCONTRAT: **Unique Id of the account**  NUMPERSONA**: Internal Id of the customer** |
| PE10 | Business Customer Table | NUMPERSONA,  ENTIDNPER | NUMPERSONA: Internal Id of the customer |

## Source tables relationship view





## Sample Data



## Source Volumetric

Production count (2 days data and daily increase) for different source table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **# reg. 20/10** | **# reg. 21/10** | **Daily increase** |
| DV01 | 11.169.552 | 11.171.551 | 1.999 |
| DV03 | 11.169.552 | 11.171.551 | 1.999 |
| KC03 | 92.132.951 | 95.665.415 | 3.532.464 |
| KC43 | 5.759.588.436 | 5.759.649.947 | 61.511 |
| PE06 | 32.343.966 | 32.350.946 | 6.98 |
| PE16 | 37.255.258 | 37.263.923 | 8.665 |
| PE11 | 10.334.977 | 10.337.737 | 2.76 |
| PE09 | 20.348.520 | 20.355.903 | 7.383 |
| PE17 | 19.162.142 | 19.167.194 | 5.052 |
| PE34 | 25.579.106 | 25.585.941 | 6.835 |
| PEA6 | 37.817.690 | 37.831.997 | 14.307 |
| PE10 | 585.819 | 586.019 | 200 |

# ODS PCA SAVINGS Data Model

## ODS PCA SAVINGS Raw Layer Data Model

### RAW LAYER TABLES

Raw layer Table Naming Standard

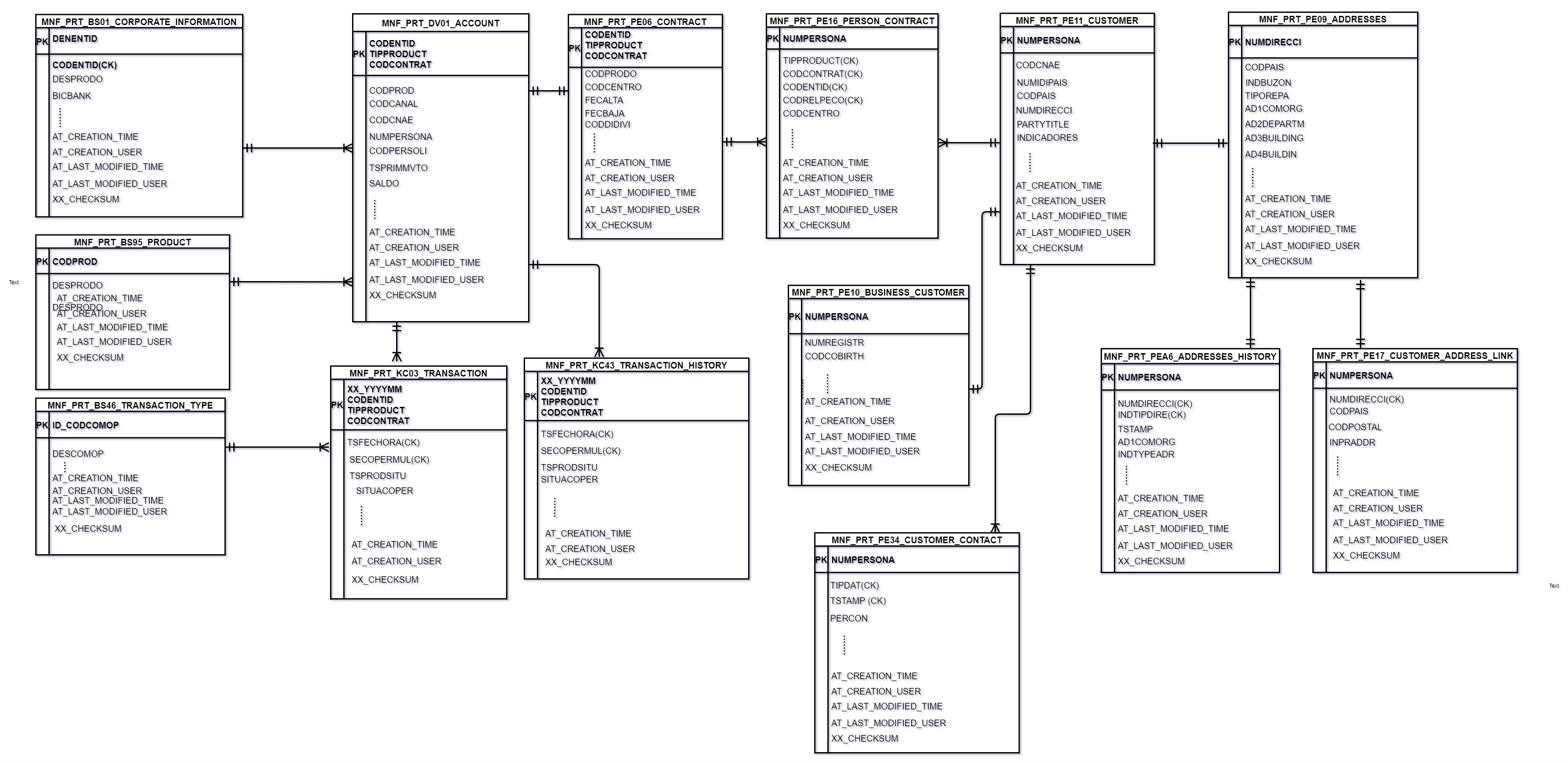
SourcePlatformName\_SourceSystemName\_SourceTableName\_BusinessArea

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Section** | **Field Format** | **Fields Value examples** | **Remarks** |
| Source Platform Name | 3 Characters | Mainframe=MNF | If we bring in data from Teradata/Oracle sources values will be Teradata=TER Oracle=ORA |
| Source System Name | 3 Characters | Proteo=PRT |  |
| Source Table Name | Variable Length | DV01 KC03 | DV01 is source table for accounts under PCA category KC03 is source table for transaction under PCA category  PE11 is source table for customer under PCA category |
| Business Area | Variable Length | ACCOUNT | Business area of table |

Example: - MNF\_PRT\_DV01\_ACCOUNT

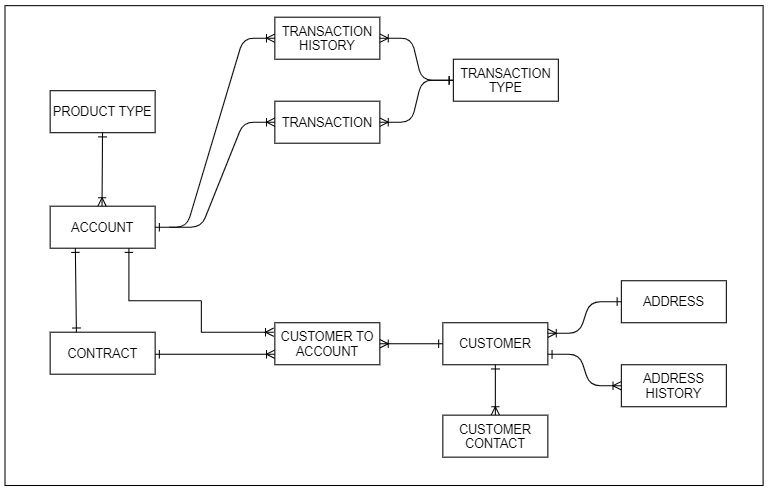
|  |  |  |
| --- | --- | --- |
| **Keyspace name** | **Scylla DB Raw Tables** | **CDC status** |
| TSB\_ODS\_RAW | MNF\_PRT\_DV01\_ACCOUNT | Exists |
| TSB\_ODS\_RAW | MNF\_PRT\_BS95\_PRODUCT | New |
| TSB\_ODS\_RAW | MNF\_PRT\_BS01\_CORPORATE\_INFORMATION | New |
| TSB\_ODS\_RAW | MNF\_PRT\_KC03\_TRANSACTION | Exists |
| TSB\_ODS\_RAW | MNF\_PRT\_KC43\_TRANSACTION | New |
| TSB\_ODS\_RAW | MNF\_PRT\_BS46\_TRANSACTION\_TYPE | Exists |
| TSB\_ODS\_RAW | MNF\_PRT\_PE11\_CUSTOMER | Exists |
| TSB\_ODS\_RAW | MNF\_PRT\_PE34\_CUSTOMER\_CONTACT | Exists (Not included in PUSH, but SCA is bring in this CDC subscription) |
| TSB\_ODS\_RAW | MNF\_PRT\_PE09\_ADDRESSES | New |
| TSB\_ODS\_RAW | MNF\_PRT\_PEA6\_ADDRESSES\_HISTORY | New |
| TSB\_ODS\_RAW | MNF\_PRT\_PE17\_CUSTOMER\_ADDRESS\_LINK | New |
| TSB\_ODS\_RAW | MNF\_PRT\_PE06\_CONTRACT | Exists |
| TSB\_ODS\_RAW | MNF\_PRT\_PE16\_PERSON\_CONTRACT | Exists |
| TSB\_ODS\_RAW | MNF\_PRT\_PE10\_BUSINESS\_CUSTOMER | New |

### PCA SAVINGS RAW DATA MODEL



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Partition Key** | **Cluster Key** | **Primary Key** | **Rational** | **Assumption** |
| MNF\_PRT\_DV01\_ACCOUNT | CODENTID, TIPPRODUCT, CONCONTRAT |  | It is combination of Partition key and cluster key. If there is no cluster keys then partition key is primary key of table. For e.g. here CODENTID, TIPPRODUCT, CONCONTRAT composite partition key is primary key for Account table | This 3-part key makes the account table unique and distribute the data evenly | Account table will be accessed with Account number (3- part key) |
| MNF\_PRT\_BS95\_PRODUCT | CODPROD |  | CODPROD is primary key (Partition key +Cluster keys) | Product code is the primary key on the source table | Access will be by Product code |
| MNF\_PRT\_BS01\_CORPORATE\_INFORMATION | DENENTID | CODENTID(CK) | DENENTID+CODENTID (Partition key + Cluster keys) | Corporate information can be stored more data based on DENENTID as partition key |  |
| MNF\_PRT\_KC03\_TRANSACTION | XX\_YYYYMM, CODENTID, TIPPRODUCT, CONCONTRAT, |  |  | 3-part account key along with month& year partition will distribute the data evenly across all the nodes. Month & year will be a derived column in the ODS table |  |
| TSFECHORA | XX\_YYYYMM, CODENTID, TIPPRODUCT, CONCONTRAT+TSFECHORA, SECOPERMUL (PARTITION KEY + CLUSTER KEYS) | Cluster key column will make SQL queries faster for a given date range for getting account statements | Transaction data will be accessed based on the Account number and monthly date range. With the ordering based on date and time. |
| SECOPERMUL |  |  |
|  |  | Assume most of the consumption use case for transaction will prefer monthly range. We need to know if there is any need for weekly range access |
|  |  | Need to decide name of the derived partition column of the partition columns |  |
| MNF\_PRT\_KC43\_TRANSACTION | XX\_YYYYMM, CODENTID, TIPPRODUCT, CONCONTRAT, |  |  | 3-part account key along with month& year partition will distribute the data evenly across all the nodes. Month & year will be a derived column in the ODS table |  |
|  | TSFECHORA | XX\_YYYYMM, CODENTID, TIPPRODUCT, CONCONTRAT+TSFECHORA, SECOPERMUL (PARTITION KEY + CLUSTER KEYS) | These cluster key columns will make sql queries faster by date range for account statements | Transaction data will be accessed based on the Account number and monthly date range. With the ordering based on date and time. |
| SECOPERMUL |  |  |
|  |  | Assume most of the consumption use case for transaction will prefer monthly range. We need to know if there is any need for weekly range access |
|  |  |  | Need to decide name of the derived partition column of the partition columns |  |
| MNF\_PRT\_BS46\_TRANSACTION\_TYPE | CODCOMOP |  | CODCOMP (PARTITION KEY + CLUSTER KEYS) | Query on Composite operation that make it partition key |  |
| MNF\_PRT\_PE11\_CUSTOMER | NUMPERSONA |  | NUMPERSONA (PARTITION KEY + CLUSTER KEYS) | Customer details will be fetched on customer number as NUMPERSONA is customer id as partition key |  |
| MNF\_PRT\_PE34\_CUSTOMER\_CONTACT | NUMPERSONA | TIPDAT | NUMPERSONA+TIPDAT+TSTAMP (PARTITION KEY + CLUSTER KEYS) | Customer Contact details query will fetch data based on customer number. |  |
| TSTAMP |  |
| MNF\_PRT\_PE09\_ADDRESSES | NUMDERECCI |  | NUMDERECCI (PARTITION KEY + CLUSTER KEYS) | Address code for customer as partition key |  |
| MNF\_PRT\_PEA6\_ADDRESSES\_HISTORY | NUMPERSONA | NUMDIRECCI | NUMPERSONA+NUMDIRECCI+INDTIPDIRE (PARTITION KEY + CLUSTER KEYS) | Customer id for address history to fetch all previous records as partition key |  |
| INDTIPDIRE |  |
| MNF\_PRT\_PE17\_CUSTOMER\_ADDRESS\_LINK | NUMPERSONA |  | NUMPERSONA (PARTITION KEY + CLUSTER KEYS) | Customer id for getting address type for that customer from PE17 table so made Customer id as Partition key |  |
| MNF\_PRT\_PE06\_CONTRACT | CODENTID, TIPPRODUCT, CONCONTRAT |  | CODENTID, TIPPRODUCT, CONCONTRAT (PARTITION KEY + CLUSTER KEYS) | Contract details faster access kept product wise contract column as partition key |  |
| MNF\_PRT\_PE16\_PERSON\_CONTRACT | NUMPERSONA |  | NUMPERSONA (PARTITION KEY + CLUSTER KEYS) | Customer id will be faster in accessing it’s Person contract details |  |
| MNF\_PRT\_PE10\_BUSINESS\_CUSTOMER | NUMPERSONA, ENTIDNPER |  | NUMPERSONA, ENTIDNPER (PARTITION KEY + CLUSTER KEYS) | Partition key same as DB2 Source table PE10 |  |

## PCA SAVINGS Conceptual Model

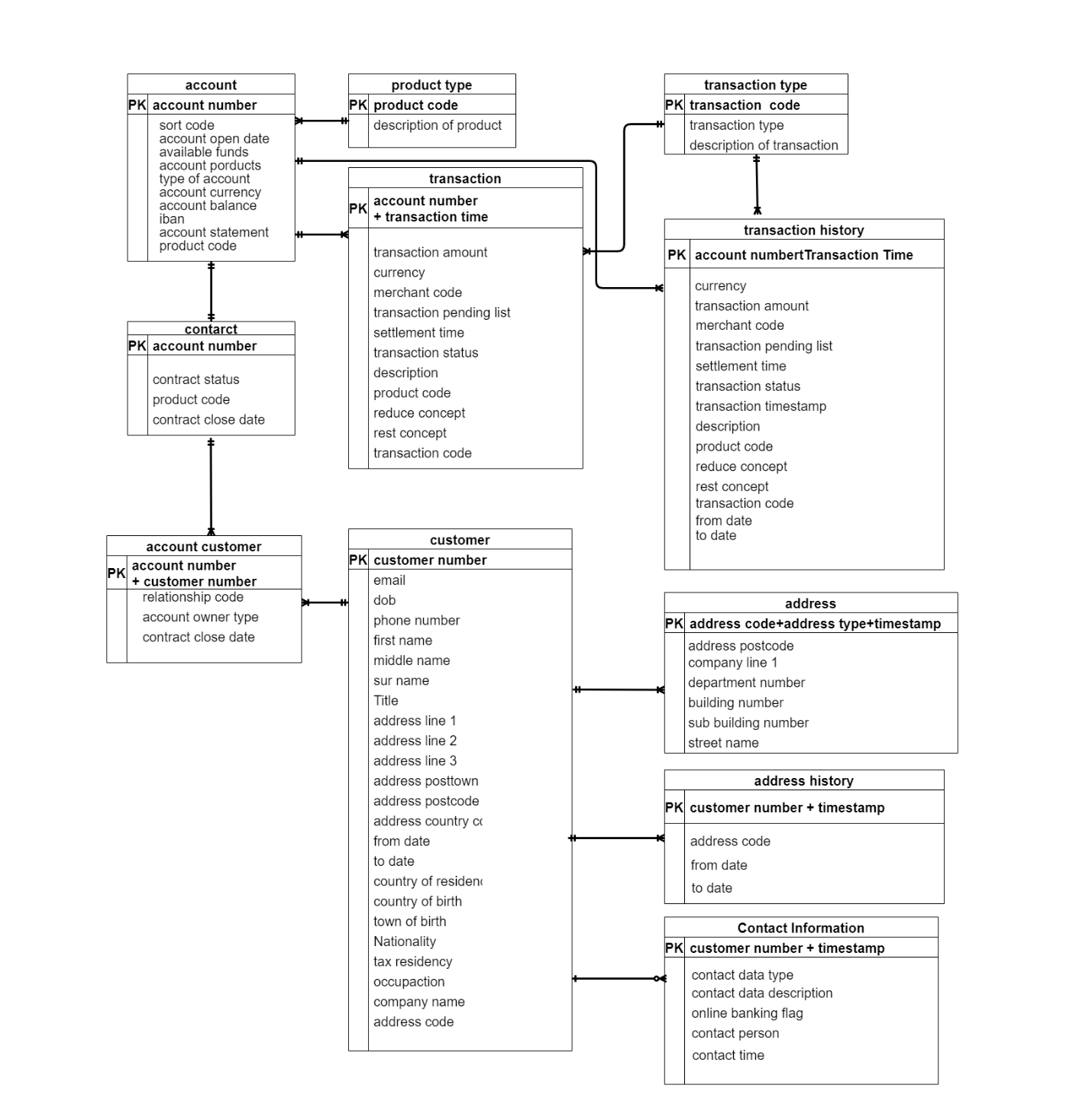


## Data Access/Query Pattern

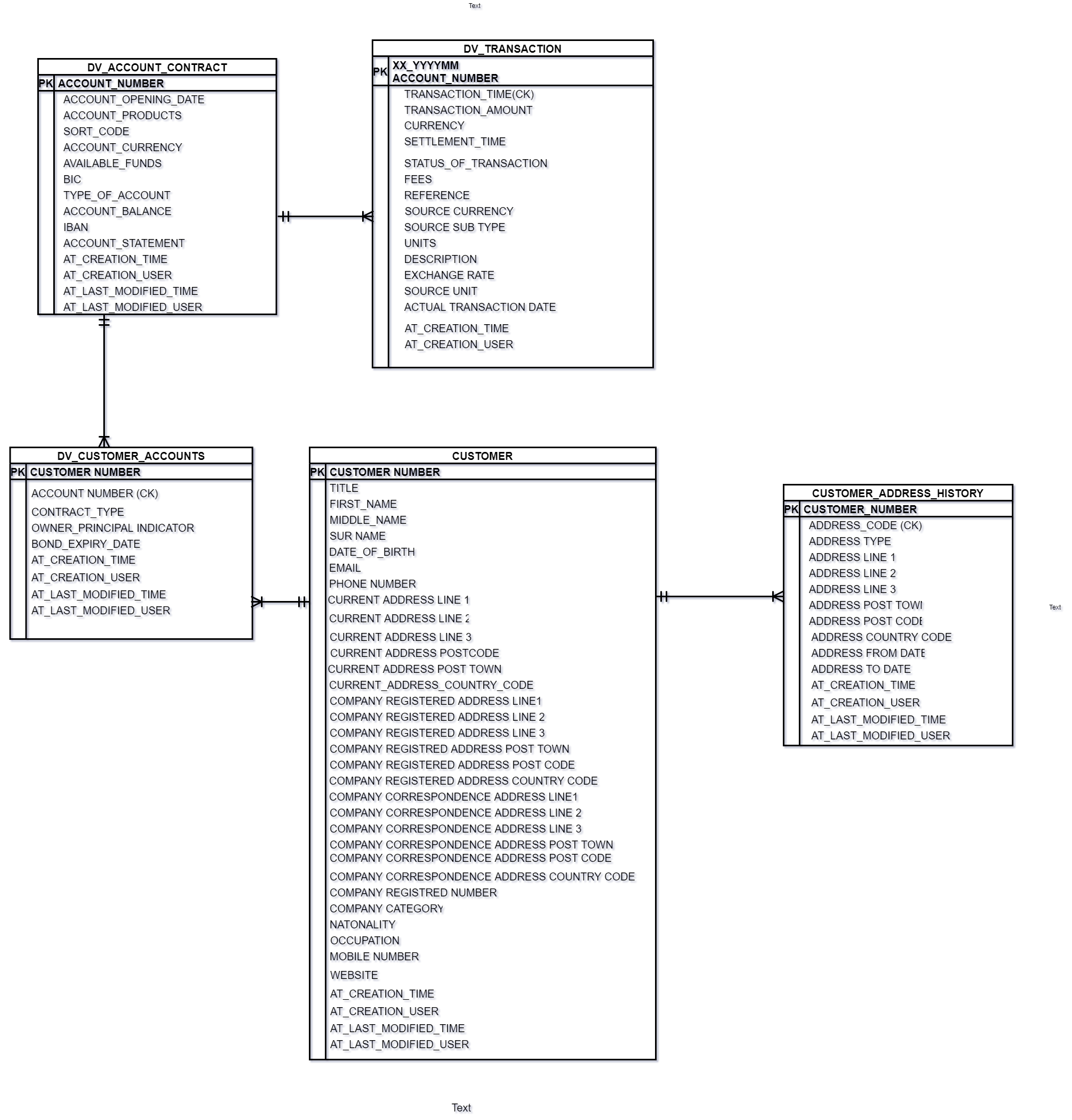
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Purpose** | **Inputs** | **Outputs** |
| getRetailCustomerDetails | Fetch latest retail customer details  based on a specific customer ID (Numpersonas) | Customer ID | Name, Primary Address, Date of Birth,  Marketing Preferences etc. |
| getCustomerAccounts | Fetch list of accounts held by customer based on a specific customer ID (Numpersonas) | Customer ID | Product Type, Sort Code, Account Number etc. |
| getCAAccountDetails | Fetch account level attributes for  Current Account based on unique account identifier | PCA Account ID | Sort Code, Account Number, Balance, Overdraft Limit,  Available Balance, Product, Rates, Account Level Address etc. |
| getSavingsAccountDetails | Fetch account level attributes  for Savings based on unique account identifier | Savings Account ID | Sort Code, Account Number, Balance, Product, Rates,  Account Level Address etc. |
| getCATransactionsDetails | Fetch transactions and relevant attributes for Current Account based on unique account identifier | PCA Account ID, Transactions Date Range | Transaction Date/Time, Amount, Transaction Narrative,  Merchant Details, Transaction Category |
| getSavingsTransactionsDetails | Fetch transactions and relevant attributes for Savings based on unique account identifier | Savings Account ID, Transactions Date Range | Transaction Date/Time, Amount, Transaction Narrative,  Transaction Category |

## PCA SAVINGS Curated Logical Data Model

### Version 1



### Version 2



## Proposed Physical Curated Data model

### CURATED LAYER TABLES

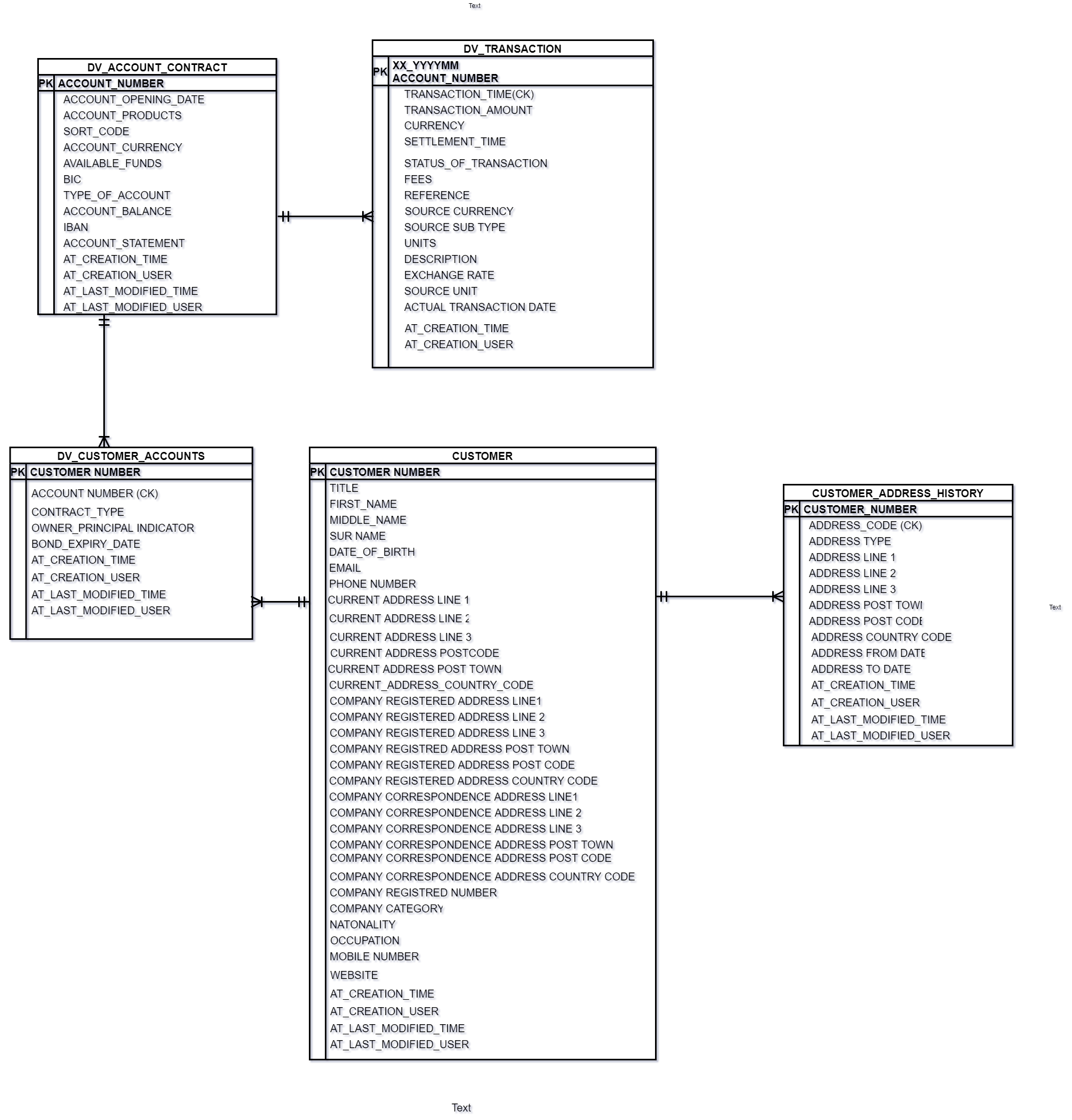
Curated Layer table Naming Standard

Product\_BusinessArea

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Section** | **Field Format** | **Fields Value** | **Remarks** |
| Product | 2 characters | DV | DV for PCA Products, TA for cards  If Table is related to any specific product then this field is required. |
| Business Area | Variable Length | ACCOUNT | Business area of table Like Account or Transaction |

Example DV\_TRANSACTION or DV\_ACCOUNT\_CONTRACT for PCA Product

|  |  |
| --- | --- |
| **Keyspace** | **Scylla DB Target table** |
| TSB\_ODS\_CURATED | DV\_ACCOUNT\_CONTRACT |
| TSB\_ODS\_CURATED | DV\_TRANSACTION |
| TSB\_ODS\_CURATED | CUSTOMER |
| TSB\_ODS\_CURATED | CUSTOMER\_ADDRESS\_HISTORY |
| TSB\_ODS\_CURATED | DV\_CUSTOMER\_ACCOUNT |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Partition Key** | **Cluster Key** | **Primary Key** | **Rational** | **Assumption** |
| DV\_ACCOUNT\_CONTRACT | ACCOUNT NUMBER |  | It is combination of Partition key and cluster keys.  If there are no cluster keys then partition key is primary key of table. For e.g. here ACCOUNT NUMBER partition key is primary key for Account table | Access pattern – Account level details for a given account number |  |
| DV\_TRANSACTION | XX\_YYYYMM+ACCOUNT NUMBER | TRANSACTION\_TIME | XX\_YYYYMM+ACCOUNT NUMBER+TRANSACTION\_TIME | Account number along with month& year partition will distribute the data evenly across all the nodes. Month and year will be a derived column in the ODS table.  Cluster key will make faster query retrieval for getting specific timestamp range data Access pattern – Get list of transactions for a given Account number by month   Need to decide name of the derived partition key | Transaction data will be accessed based on the Account number and monthly date range. with ordering by transaction date /time Assume most of the consumption use case for transaction will prefer monthly range. We need to know if there is any need for weekly range access |
| DV\_CUSTOMER\_ACCOUNTS | CUSTOMER\_NUMBER |  | CUSTOMER NUMBER | Access pattern – Get list of accounts linked to given customer number |  |
| CUSTOMER | CUSTOMER\_NUMBER |  | CUSTOMER NUMBER | Access pattern – Get customer data for a given customer number |  |
| CUSTOMER\_ADDRESS\_HISTORY | CUSTOMER\_NUMBER | ADDRESS\_CODE | CUSTOMER NUMBER+ADDRESS\_CODE | Access pattern – Get customer address history data for a given customer number search the previous addresses |  |

# ODS PCA SAVINGS Data Catalogue/Data Mapping

## High Level -Raw and Curated Source -Target Model table mapping relation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Source Table** | **Domain** | **Scylla DB Raw Tables** | **Scylla DB Curated Tables** | **Remark** |
| DV01 | Accounts | MNF\_PRT\_DV01 \_ACCOUNT | DV\_ACCOUNT\_CONTRACT | Refer Data Catalog: Accounts\_table\_mappings.xlsx |
| BS95 | Accounts | MNF\_PRT\_BS95\_PRODUCT | DV\_ACCOUNT\_CONTRACT | Refer Data Catalog: Accounts\_table\_mappings.xlsx |
| BS01 | Accounts | MNF\_PRT\_BS01 | DV\_ACCOUNT\_CONTRACT | Refer Data Catalog: Accounts\_table\_mappings.xlsx |
| KC03 | Transactions | MNF\_PRT\_KC03\_TRANSACTION | DV\_TRANSACTION | Refer Data Catalog: Transaction\_table\_mappings.xlsx |
| KC43 | Transactions | MNF\_PRT\_KC43\_TRANSACTION | DV\_TRANSACTION | Refer Data Catalog: Transaction\_table\_mappings.xlsx |
| BS46 | Transactions | MNF\_PRT\_BS46\_TRANSACTION\_TYPE | DV\_TRANSACTION | Refer Data Catalog: Transaction\_table\_mappings.xlsx |
| PE11 | Customer | MNF\_PRT\_PE11\_CUSTOMER | CUSTOMER | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PE34 | Customer | MNF\_PRT\_PE34\_CUSTOMER\_CONTACT | CUSTOMER | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PE09 | Customer | MNF\_PRT\_PE09\_ADDRESSES | CUSTOMER | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PEA6 | Customer | MNF\_PRT\_PEA6\_ADDRESSES\_HISTORY | DV\_CUSTOMER\_ADDRESS\_HISTORY | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PE17 | Customer | MNF\_PRT\_PE17\_CUSTOMER\_ADDRESS\_LINK | CUSTOMER | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PE06 | Customer | MNF\_PRT\_PE06\_CONTRACT | DV\_ACCOUNT\_CONTRACT | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PE16 | Customer | MNF\_PRT\_PE16\_PERSON\_CONTRACT | DV\_CUSTOMER\_ACCOUNT | Refer Data Catalog: Customer\_table\_mappings.xlsx |
| PE10 | Customer | MNF\_PRT\_PE10\_BUSINESS\_CUSTOMER | CUSTOMER | Refer Data Catalog: Customer\_table\_mappings.xlsx |

## Low Level Data Mapping

### TRANSACTIONS TABLES

Column Level – Transaction Data Mapping (Raw/Curated)



### ACCOUNT TABLES

Column Level – Account Data Mapping (Raw/Curated)



### CUSTOMER TABLES

Column Level – Customer Data Mapping (Raw/Curated)

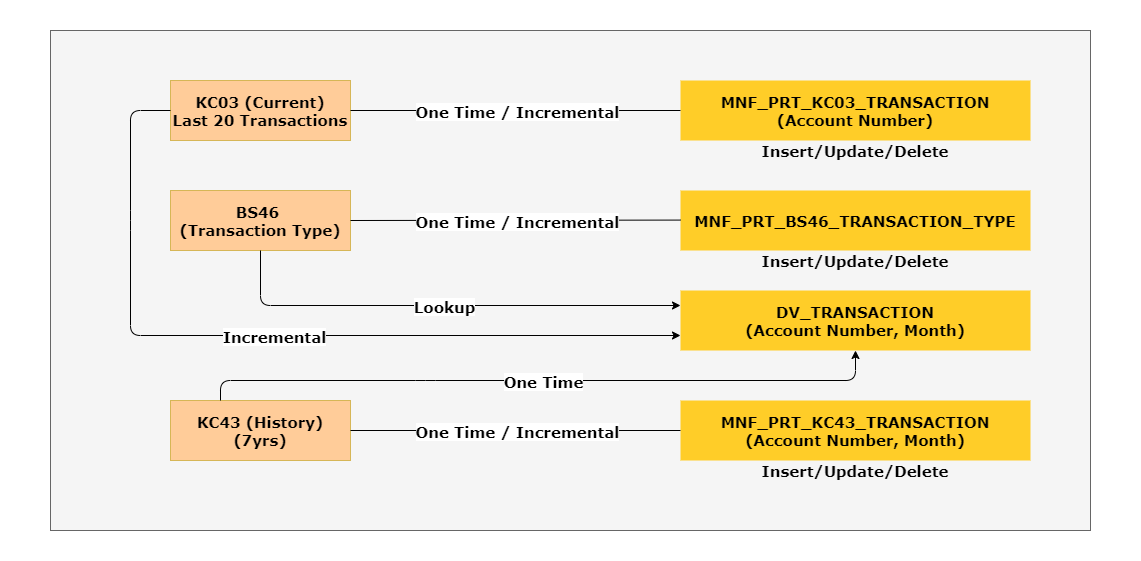


### Data Model Design Decisions

|  |  |
| --- | --- |
| No | **Data Model Design Decision** |
| 1 | Transaction tables are partitioned on Transaction Month  and Account Number where we have introduced derived transaction month in YYYYMM format so that accounts monthly data can get store in one partition and help in Monthly statements for accounts or other  read queries minimal partition read happen along with evenly distribute data across the nodes |
| 2 | Address History will be a separate table since it has multiple address and historical address changes data. |
| 3 | We have added Audit ETL/BI column like Create record time or  create user Id along with checksum field in raw layer to validate data transfer from source system  ETL columns are prefixed with AT\_Column Name |
| 4 | Data Cleansing and Data Standardization will not be part of ODS data processing |

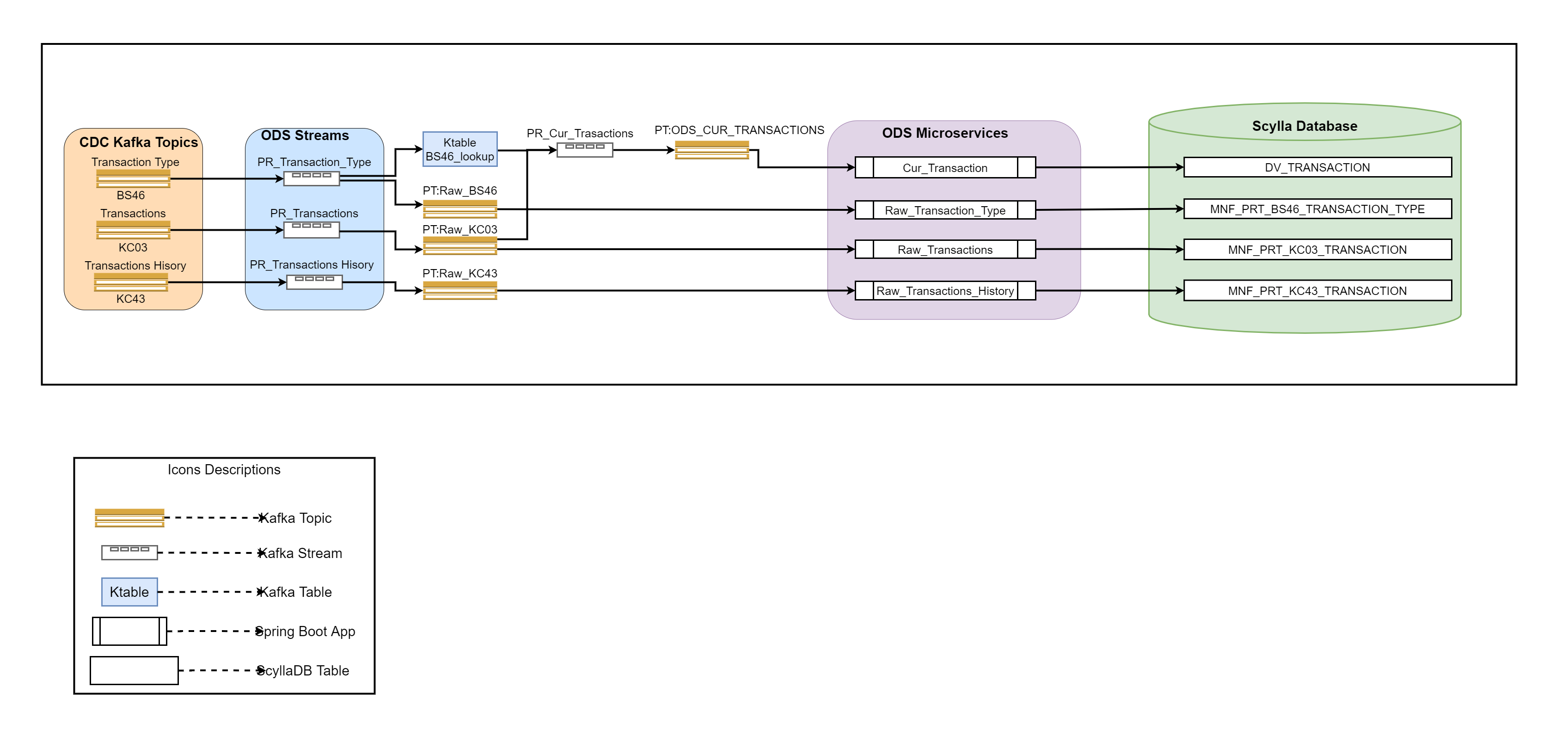
# Application Design:

## High level Transaction Data Flow



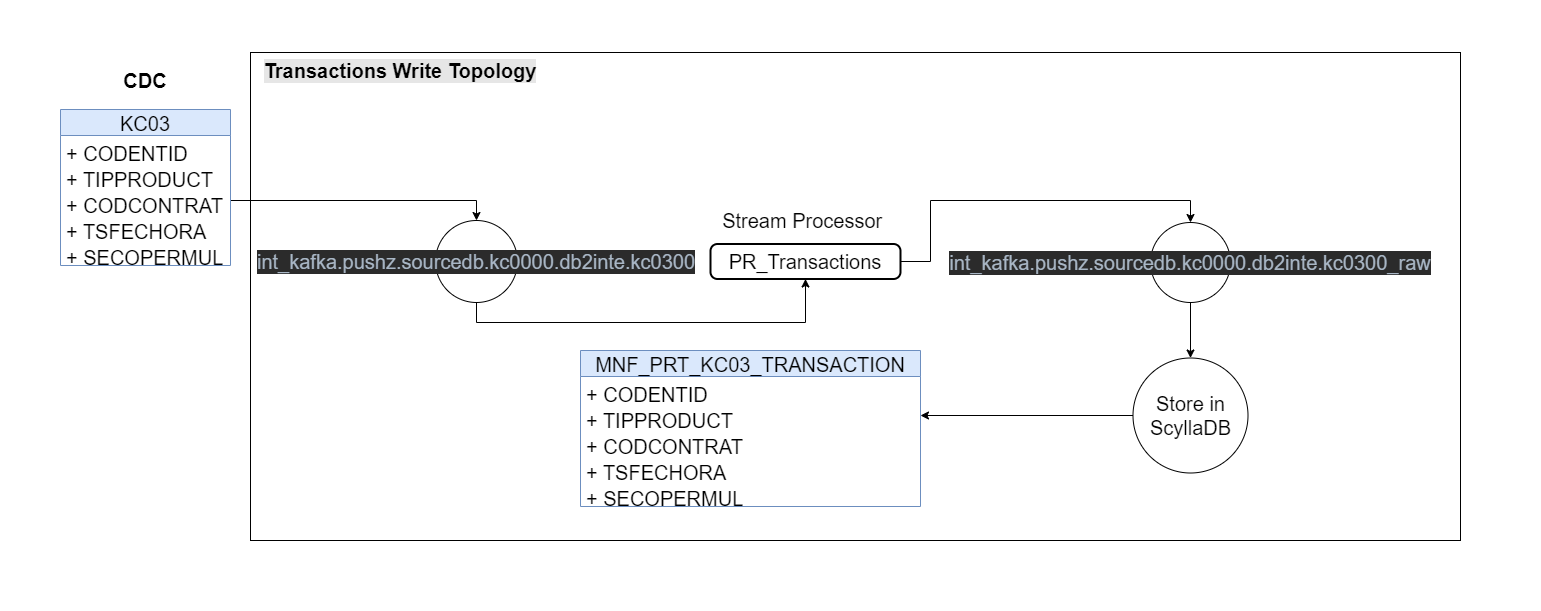
## Low Level Data Flow (Transaction)

### Full Transactions Data Flow Diagram

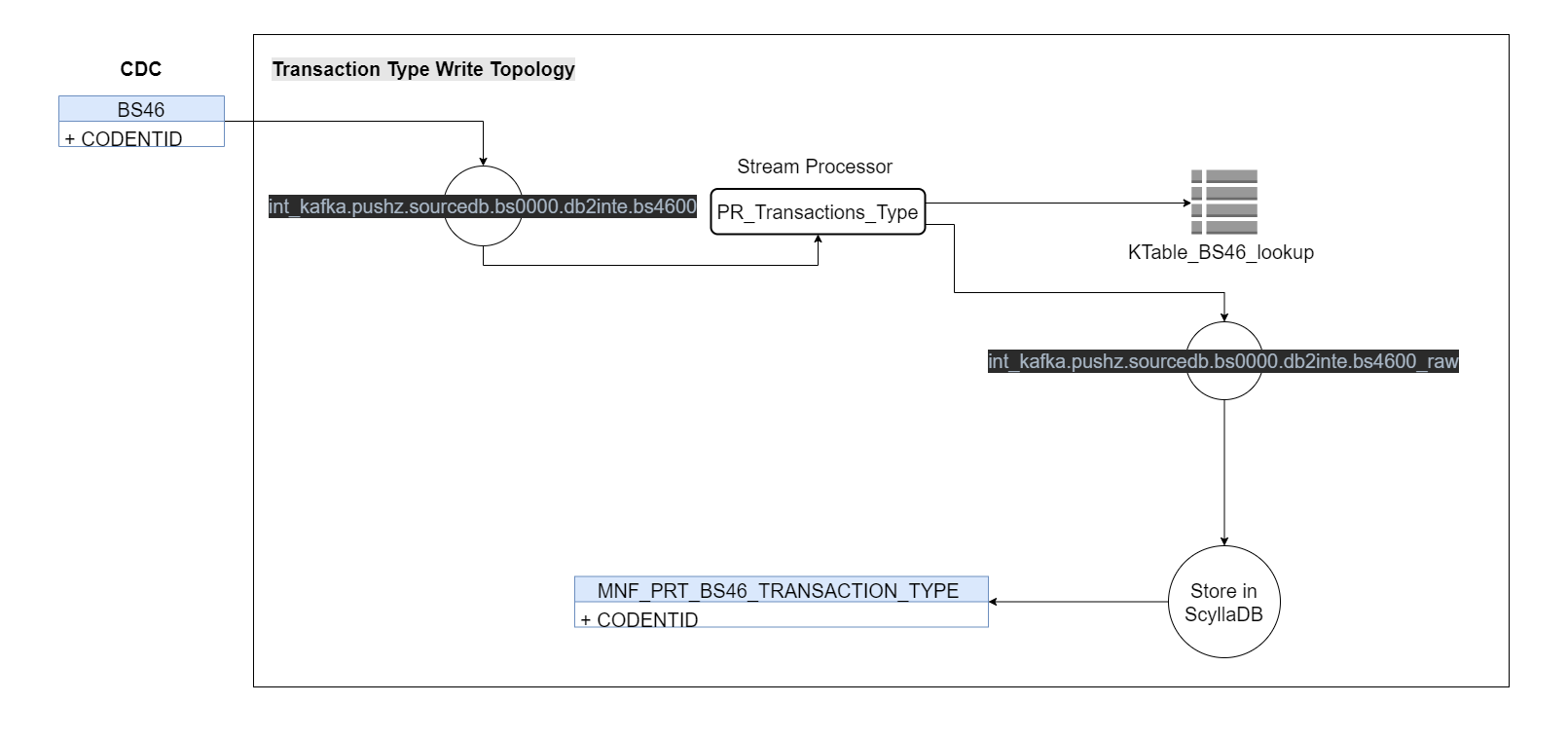


### Topology Design

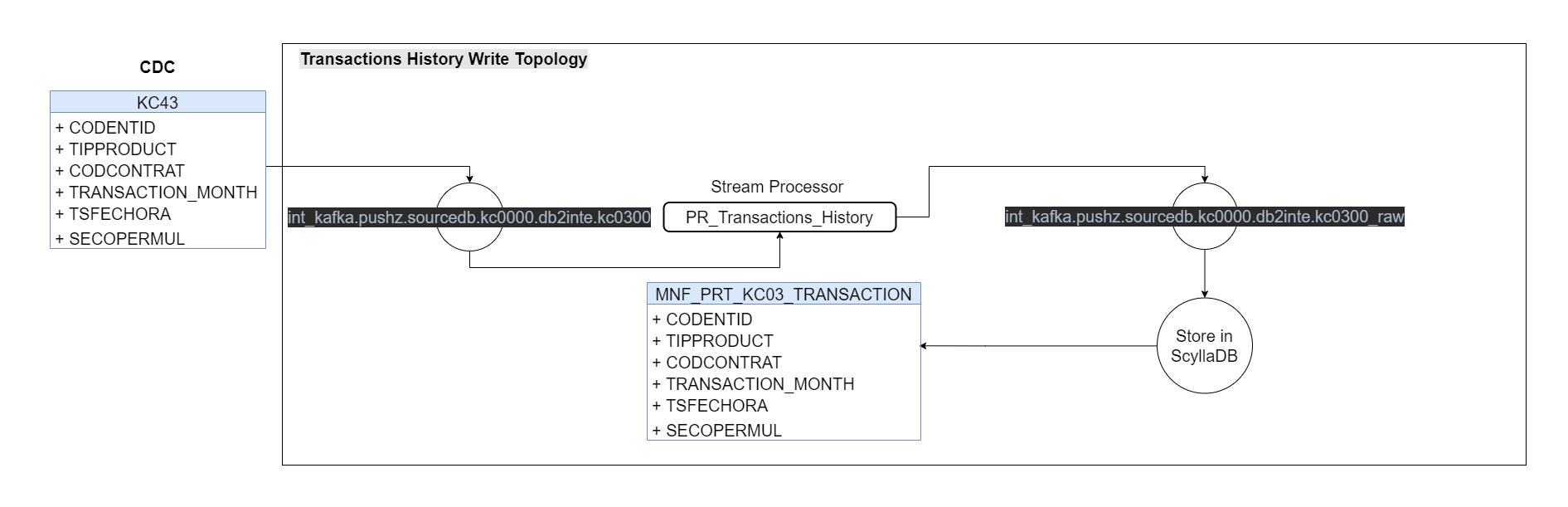
#### Transactions Write Topology



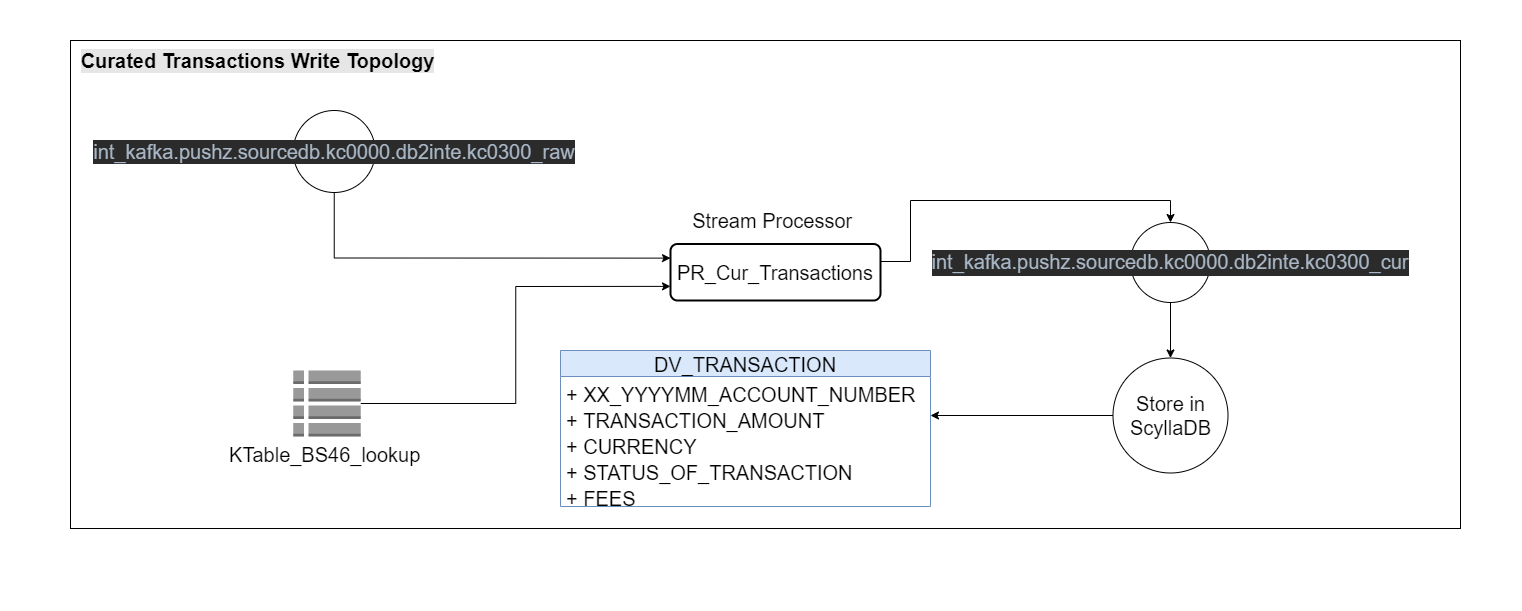
#### Transaction Type Write Topology



#### Transactions History Write Topology



#### Curated Transactions Write Topology



### Data Processor Details

| **SN0** | **Processor (logical name)** | **Processor (stream component name)** | **Interfaces** | **Description** | **Avro Schema** |
| --- | --- | --- | --- | --- | --- |
| 1 | Processor for Stream transaction events | PR\_Transactions | In: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300  Processing: write on to the private topic Out: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_raw | Creating private topic from cdc Kafka topic | TBD |
| 2 | Processor for Stream transaction type events | PR\_Transaction\_Type | In: int\_kafka.pushz.sourcedb.bs0000.db2inte.bs4600  Processing: write on to the private topic and create lookup Kafka table for internal processing  Out: int\_kafka.pushz.sourcedb.bs0000.db2inte.bs4600\_raw and create kTable\_BS46\_lookup | Creating private topic and deriving KTable lookup from cdc Kafka topic | TBD |
| 3 | Processor for Stream transaction history events | PR\_Transactions\_History | In: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc4300  Processing: write on to the private topic  Out: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc4300 | Creating private topic from cdc Kafka topic | TBD |
| 4 | Processor for Stream curated transaction events | PR\_Cur\_Transaction | In: kTable\_BS46\_lookup and int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_raw  Processing: write on to the private topic Out: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_cur | Create private topic using transformation logic with lookup table | TBD |
| 5 | Processor for ODS transaction events | ODS\_Transaction\_Processor | In: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_raw  Processing: write to nosql db from private topic  Out: ScyllaDB ods\_raw\_KC03\_transaction table | Stream application populating data in the Raw(Kc03) table in ScyllaDB. | TBD |
| 6 | Processor for ODS transaction type events | ODS\_Transaction\_Type\_Processor | In: int\_kafka.pushz.sourcedb.bs0000.db2inte.bs4600\_raw  Processing: write to nosql db from private topic Out: ScyllaDB ods\_raw\_BS46\_transaction table | Stream application  populating data in the  Raw BS46 transactions table in ScyllaDB | TBD |
| 7 | Processor for ODS transaction history events | ODS\_Transactions\_History\_Processor | In: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc4300  Processing: write to nosql db from private topic Out: ScyllaDB ods\_raw\_KC43\_trans\_history table | Stream application populating the transaction history data into history raw table KC43 in ScyllaDB | TBD |
| 8 | Processor for ODS curated transaction events | ODS\_Curated\_Transaction\_Processor | In: int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_cur  Processing: write to nosql db from private topic  Out: ScyllaDB ods\_curated\_transaction table | Stream application populating curated transaction table in ScyllaDB. | TBD |

### Kafka Topics

#### Kafka Topic Naming Convention

[https://tsbchange.atlassian.net/wiki/spaces/ITARCH/pages/231571457/Policy+KAFKA+Topics+Naming](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Ftsbchange.atlassian.net%2Fwiki%2Fspaces%2FITARCH%2Fpages%2F231571457%2FPolicy%2BKAFKA%2BTopics%2BNaming&data=04%7C01%7CSaravanan_D01%40infosys.com%7Cf1b407c237a547c0363008d87f427a3f%7C63ce7d592f3e42cda8ccbe764cff5eb6%7C1%7C0%7C637399270801634226%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=5CGTVxdl7X9HGzRvQaY0JZGmI45H2FWRqxuyHBt1DcA%3D&reserved=0)

#### Public Kafka Topics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Public** | **Physical Name** | **Replication factor** | **Number of partitions** | **Retention time (in days)** | **Topic compaction** | **Creation command config** |
| Public topic for BS46 source table data | int\_kafka.pushz.sourcedb.bs0000.db2inte.bs4600 | 3 | = Number of brokers | 90 | log.cleanup.policy=compact |  |
| Public topic for KC03 source table data | int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300 | 3 | = Number of brokers | 90 | log.cleanup.policy=compact |  |
| Public topic for KC43 source table data | TBD |  |  |  |  |  |

#### Private Kafka Topics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Private Topic** | **Physical Name** | **Replication factor** | **Number of partitions** | **Retention time (in days)** | **Topic compaction** | **Topic Type** | **Creation command config** |
| ODS transaction topic for curated | int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_cur | 3 | = Number of brokers | 90 | log.cleanup.policy=compact |  |  |
| ODS transaction topic for raw | int\_kafka.pushz.sourcedb.kc0000.db2inte.kc0300\_raw | 3 | = Number of brokers | Never expires | log.cleanup.policy=compact | kstream | --replication-factor 3 --partitions 18 --config ret ention.ms=7776000000 --config cleanup. policy=delete |
| ODS transaction type topic for raw | int\_kafka.pushz.sourcedb.bs0000.db2inte.bs4600\_raw | 3 | = Number of brokers | Never expires | log.cleanup.policy=compact | Ktable( Global) | --replication-factor 3 --partitions 18 --config ret ention.ms=315569520000 --config cleanup. policy=compact --config segment.ms=100 -- config min.cleanable.dirty.ratio=0.01 |
| ODS transaction history topic for raw | int\_kafka.pushz.sourcedb.kc0000.db2inte.kc4300\_raw | 3 | = Number of brokers | Never expires | log.cleanup.policy=compact | Kstream | --replication-factor 3 --partitions 18 --config ret ention.ms=7776000000 --config cleanup. policy=delete |

#### Source table Avro Schema/Sample JSON Details

(Below information need to be validated and schema new source tables identified need to be added)

* [Accounts Information](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-AccountsInformation)
  + - [DV01: Saving Accounts, Fixed Term Deposits & Contracts DV (Accounts)](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-DV01:SavingAccounts,FixedTermDeposits&ContractsDV(Accounts))
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema)
  + Transaction
    - [KC03 Saving Accounts, Fixed Term Deposits & Contracts DV (Accounts) transactions](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-KC03SavingAccounts,FixedTermDeposits&ContractsDV(Accounts)transactions)
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema.1)
    - [BS46: Reduced concept](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-BS46:Reducedconcept)
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema.2)
* [Customer Data](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-CustomerData)
  + - [PE06: Contracts](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-PE06:Contracts)
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema.3)
    - [PE11: Customers](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-PE11:Customers)
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema.4)
    - [PE34: Customer Contact Data](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-PE34:CustomerContactData)
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema.5)
    - [PE16: Relationship between Customer and Contracts](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-PE16:RelationshipbetweenCustomerandContracts)
      * [Suggested AVRO schema](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-SuggestedAVROschema.6)
* [EXAMPLES WITH DATA FOR DEV/TESTING](https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-EXAMPLESWITHDATAFORDEV/TESTING)

Avro Schemas and Sample data for above source table is available in below confluence link

<https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/538017807/Data+examples+Proteo#Dataexamples(Proteo)-EXAMPLESWITHDATAFORDEV/TESTING>

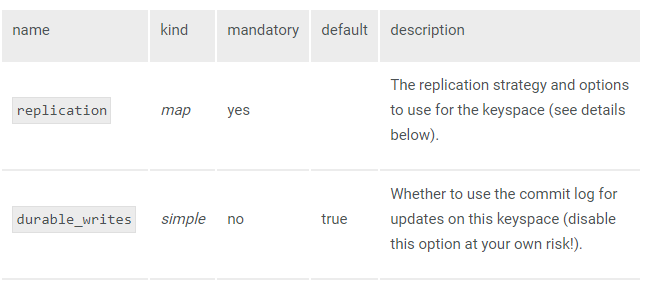
Actual AVRO Schema’s used are available in the GITHUB

[https://github.com/TSB-Bank/carlos/tree/master/tsb\_cdc\_ingestors/libs/PublicObjects/src/main/avro](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FTSB-Bank%2Fcarlos%2Ftree%2Fmaster%2Ftsb_cdc_ingestors%2Flibs%2FPublicObjects%2Fsrc%2Fmain%2Favro&data=04%7C01%7CSaravanan_D01%40infosys.com%7C3e0ac9e6e8104c0bc08608d87c251b04%7C63ce7d592f3e42cda8ccbe764cff5eb6%7C1%7C0%7C637395846104313595%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=b7vwwloPbBDgpAoAEwsY0D%2FDjsMl2eTKqzoyoQjCo6k%3D&reserved=0)

## ODS ScyllaDB Application Configuration

### Keys space configuration

**Keys space Name: TSB\_ODS**



For example:

*CREATE KEYSPACE TSB\_ODS*

*WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1’: 1, 'DC2’: 3}*

*AND durable\_writes = true; [Recommened]*

*Or,*

*CREATE KEYSPACE Ods*

*WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3};*

As per Scylla DB official documentation:

'NetworkTopologyStrategy': A replication strategy that lets you set the replication factor independently for each data-center. The rest of the sub-options are key-value pairs where a key is a data-center name and its value is the associated replication factor, or a single key value with [Auto-Expand Replication Factor](https://docs.scylladb.com/getting-started/ddl/#id3). **Recommended for production**.

'SimpleStrategy': defines a replication factor for the whole cluster. The only sub-options supported is 'replication\_factor' to define that replication factor and is mandatory.

NOTE

Using NetworkTopologyStrategy is recommended. Using SimpleStrategy will make it harder to add Data Center in the future.

### ODS Target Table Names with DDL

MNF\_PRT\_KC03\_TRANSACTION



MNF\_PRT\_KC43\_TRANSACTION



### ScyllaDB SSL

Enabling SSL with the self-signed CA certificate for Scylla DB’s cluster is required to make the data in transit secure. When connecting, the clients need to be able to validate the CA. This change is applicable to various entities like Push, CB (Conversational Banking), etc. who have to interact with Scylla as it is part of Shared Platform Services. Hence, the task is spread across teams for implementation.

**References:**  
<https://docs.scylladb.com/operating-scylla/security/>  
<https://docs.scylladb.com/operating-scylla/security/generate_certificate/>  
<https://docs.scylladb.com/operating-scylla/security/client_node_encryption/>

#### ScyllaDB Connectivity with client encryption SSL

As part of security hardening, SSL shall be enabled for the ScyllaDB database connections.

The code level modification and verification can be introduced to handle SSL connection with Scylla.

How to enable SSL connection?

When the database has the client encryption option set to true or enabled, all the connection done to the database will be verified via the trust store certificate and the trust store password. If the client\_authentication is enabled to true, verification is done via Keystore certificate as well.

Enabling the client encryption option in Scylla.yaml file:

|  |  |
| --- | --- |
| item | Key: value |
| Client encryption options | Enabled: true |
|  |  |

zookeeper configuration should be configured and the values must be passed through the pipeline. Below are the zookeeper configs created in addition to the existing variables when enabling SSL in scylla.

|  |  |
| --- | --- |
| Items | Command/script |
| SCYLLA\_SSL\_ENABLED | create /uk/co/tsb/cb/application::development/spring.data.cassandra.ssl ${SCYLLA\_SSL\_ENABLED} |
| SCYLLA\_TRUSTSTORE\_LOCATION | create /uk/co/tsb/cb/application::development/scylla.ssl.truststore.location ${SCYLLA\_TRUSTSTORE\_LOCATION} |
| SCYLLA\_TRUSTSTORE\_PASSWORD | create /uk/co/tsb/cb/application::development/scylla.ssl.truststore.password ${SCYLLA\_TRUSTSTORE\_PASSWORD} |

**DevOps (Cloud Admins):**

1. Configure and enable ScyllaDB SSL connection.
2. Create a Java keystore file named **scylladb-truststore.jks** which contains the necessary certificates to access Scylla via SSL. The file is usually protected with a password.
3. Update the existing **scylladb-admin** Kubernetes secret to include:
   1. the **scylladb-truststore.jks** (the java keystore file which contains the certificates to communicate via ssl) as created from 1.
   2. **truststore-password** (the password which allows access to 3a above.
4. On higher environments like **prd**, the following properties need to be added to Zookeeper to support the Orchestrator services:

create /uk/co/tsb/cb/application::development/spring.data.cassandra.ssl ${SCYLLA\_SSL\_ENABLED}

create /uk/co/tsb/cb/application::development/scylla.ssl.truststore.location ${SCYLLA\_TRUSTSTORE\_LOCATION}

create /uk/co/tsb/cb/application::development/scylla.ssl.truststore.password ${SCYLLA\_TRUSTSTORE\_PASSWORD}

**DevOps (Project Features):**

1. Update the Jenkins deployment pipeline code to include the following environment variable property: **SCYLLA\_SSL\_ENABLED.** This is a true / false value indicating when to use an SSL connection. It will be set as follows depending on the environment:
   1. dev = false
   2. sit = false
   3. pre = false
   4. prd = true
2. Update the Jenkins deployment pipeline code to include the following environment variable property: **SCYLLA\_TRUSTSTORE\_LOCATION** which is set to **/scylla/scylladb-truststore.jks**. This is only set for the **prd** environment. It is set empty for **dev**, **sit**, **pre** environments.
3. Update the Jenkins deployment pipeline code to include the following environment variable property: **SCYLLA\_TRUSTSTORE\_PASSWORD** which is set to the **trustore-password** value in the Kubernetes secret called **scylladb-admin**. This is only set for the **prd** environment. It is set empty for **dev**, **sit**, **pre** environments.
4. Update the Jenkins deployment pipeline code to mount the Kubernetes secret called **scylladb-admin** to the folder of **/scylla.** This is only set for the **prd** environment. For **dev**, **sit**, **pre** environments this is not mounted.

**Testing:**  
To validate that encrypted connection to the node is enabled, check the logs using journalctl \_COMM=scylla. The following message should be seen storage\_service - Enabling encrypted CQL connections between client and node.

<https://tsbchange.atlassian.net/wiki/spaces/CB/pages/1381728675/Scylla+SSL>

#### ScyllaDB Encryption at Rest

Scylla Enterprise protects your sensitive data with data-at-rest encryption. It protects the privacy of your user’s data, reduces the risk of data breaches, and helps meet regulatory requirements. In particular, it provides an additional level of protection for your data persisted in storage or its backups.

When Scylla Enterprise Encryption at Rest is used together with Encryption in Transit ([Node to Node](https://docs.scylladb.com/operating-scylla/security/node_node_encryption/) and [Client to Node](https://docs.scylladb.com/operating-scylla/security/client_node_encryption/)), you benefit from end to end data encryption.

NOTE

Encryption at Rest is available for Scylla Enterprise customers version 2019.1.1 and later. KMIP support is available for Scylla Enterprise customers version 2019.1.3 and later

[**About Encryption at Rest**](https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/#id6)

The following can be encrypted:

* Scylla persistent tables (SSTables)
* System level data, such as:
  + Commit logs
  + Batches
  + hints logs
  + KMIP Password (part of scylla.yaml)

Encryption at Rest works at table level granularity, so you can choose to encrypt only sensitive tables. For both system and table data, you can use different algorithms that are supported by [OpenSSL](https://www.openssl.org/) in a file block encryption scheme.

NOTE

SSTables of a particular table can have different encryption keys, use different encryption algorithms, or not be encrypted at all - at the same time.

[**When is Data Encrypted?**](https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/#id7)

As SSTables are immutable, tables are encrypted only once, as a result of memtable flush, compaction, or upgrade (with [Nodetool upgradesstables](https://docs.scylladb.com/operating-scylla/nodetool-commands/upgradesstables" \t "_blank)).

Once a table is encrypted, all resulting SSTables are encrypted using the most current key and algorithm. When you encrypt an existing table, the new SSTables are encrypted. The old SSTables which existed before the encryption are not updated. These tables are encrypted according to the same actions as described previously.

[**When is Data Decrypted?**](https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/#id8)

When Scylla reads an encrypted SSTable from disk, it fetches the encryption key’s ID from the SSTable and uses it to extract the key and decrypt the data. When Scylla reads an encrypted system table, it fetches the system table encryption key location from the scylla.yaml file. It locates the key and uses it to extract the key and decrypt the data.

[**Encryption Keys**](https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/#id9)

Two types of encryption keys are available: System Keys and Table Keys.

[**System Keys**](https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/#id10)

System keys are used for encrypting system data, such as commit logs, hints, and/or other user table keys. When a Replicated Key Provider is used for encrypting SSTables, the table keys are stored in the encrypted\_keys table, and the system key is used to encrypt the encrypted\_keys table. The system key is stored as the contents of a local file and is encrypted with a single key that you provide. The default location of system keys is /etc/scylla/resources/system\_keys/ and can be changed with the system\_key\_directory option in scylla.yaml file. When a Local Key Provider is used for encrypting system info, you can provide your own key, or Scylla can make one for you.

[**Table Keys**](https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/#id11)

Table keys are used for encrypting SSTables. Depending on your key provider, this key is stored in different locations:

* Replicated Key Provider - encrypted\_keys table
* KMIP Key Provider - KMIP server
* Local Key Provider - in a local file with multiple keys. You can provide your own key or Scylla can make one for you.

References:

<https://docs.scylladb.com/operating-scylla/security/encryption-at-rest/>

### ScyllaDB Current Setup

https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/554696770/ScyllaDB

https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/565280800/ScyllaDB+Backups+Proposal

https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/599228455/ScyllaDB+Backup+Process

## Kafka Configuration

### Current Kafka Setup

Kafka clusters are running and operated on AWS in all the environments (dev, sit, pre-pod, prod) following a similar architecture and design. The dev team is responsible for defining topics, partition assignment, and number of threads based on data model and non-functional requirements, resulting in improved performance, availability, and reliability.

Here are some documents related to the current setup of Kafka:

 Users and roles:

[https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/580124673/Users+and+roles](https://urldefense.proofpoint.com/v2/url?u=https-3A__tsbchange.atlassian.net_wiki_spaces_PNP_pages_580124673_Users-2Band-2Broles&d=DwMFAg&c=eIGjsITfXP_y-DLLX0uEHXJvU8nOHrUK8IrwNKOtkVU&r=IlDDFz3BNplQHYn9M-MUzf-vi3yQjVg6p7h-G2ZHSL8&m=lTIQyfh2LdQm9ENQqneDQLTX0rmvoHIJHZmXhYSlyAM&s=hDD8S9_2uzTN9Qk-LJ6WEgLHsjQqS-J8ucQlJmK1LAc&e=)

Kafka configuration:

[https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/584581697/Kafka+configuration](https://urldefense.proofpoint.com/v2/url?u=https-3A__tsbchange.atlassian.net_wiki_spaces_PNP_pages_584581697_Kafka-2Bconfiguration&d=DwMFAg&c=eIGjsITfXP_y-DLLX0uEHXJvU8nOHrUK8IrwNKOtkVU&r=IlDDFz3BNplQHYn9M-MUzf-vi3yQjVg6p7h-G2ZHSL8&m=lTIQyfh2LdQm9ENQqneDQLTX0rmvoHIJHZmXhYSlyAM&s=ntaoWt0dBFnSY98Ah7xJed0pxRYbbGGJfLfRl9paB6M&e=)

<https://tsbchange.atlassian.net/wiki/spaces/TCO/pages/1227260163/CDP+Kafka+Deployment+Configuration>

HA & DR & Backups

[https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/571080713/Disaster+Recovery](https://urldefense.proofpoint.com/v2/url?u=https-3A__tsbchange.atlassian.net_wiki_spaces_PNP_pages_571080713_Disaster-2BRecovery&d=DwMFAg&c=eIGjsITfXP_y-DLLX0uEHXJvU8nOHrUK8IrwNKOtkVU&r=IlDDFz3BNplQHYn9M-MUzf-vi3yQjVg6p7h-G2ZHSL8&m=lTIQyfh2LdQm9ENQqneDQLTX0rmvoHIJHZmXhYSlyAM&s=SsOpQ7gockPvpDDK5lYaev4jqvxFcdg1aKNhEU_cQek&e=)

Sizing:

[https://tsbchange.atlassian.net/wiki/spaces/PNP/pages/572653850/Push+Notifications+MVP+Infrastructure+Sizing](https://urldefense.proofpoint.com/v2/url?u=https-3A__tsbchange.atlassian.net_wiki_spaces_PNP_pages_572653850_Push-2BNotifications-2BMVP-2BInfrastructure-2BSizing&d=DwMFAg&c=eIGjsITfXP_y-DLLX0uEHXJvU8nOHrUK8IrwNKOtkVU&r=IlDDFz3BNplQHYn9M-MUzf-vi3yQjVg6p7h-G2ZHSL8&m=lTIQyfh2LdQm9ENQqneDQLTX0rmvoHIJHZmXhYSlyAM&s=x33F_a1y4Hh1SnYF3pqsIRWc3VX81Vtmna0ZgRIaeqM&e=)

### Kafka Topic configuration

< Private Kafka Topics configuration required for ODS will be defined during the development stage and this section will be updated accordingly>

Partition of topic = 3(for each topic)

Create Script-

*Kafka-topics.bat --create --zookeeper <ip\_address\_of\_kafka\_machine>:2181> --replication-factor 3 --partitions 3 --topic <new\_topic\_identified\_for\_ods>*

*eg-:*

*script -> Kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 3 --partitions 3 --topic transaction\_ods*

* *Created topic test*

Replication factor = 3 (for each topic)

Note:

It is assumed that the same configuration as in Push, will be used for ODS as well. This would be verified further and can be changed accordingly.

#### Process to create private topics

<https://tsbchange.atlassian.net/wiki/spaces/CB/pages/973406607/CDC+-+private+topics+changes>

### Kafka SSL Connectivity and authentication

* Kafka authentication-

The Kafka secret contains three items and is used for connecting to the Kafka broker instance within the environment.

It would contain the username and password plus a location of the truststore file. It should be created for each project namespace.

|  |  |
| --- | --- |
| Items | value |
| User Name | username |
| Password | password |
| Location of Truststore File | <../truststore\_location> |

command to connect Kafka with auth:

*kubectl create secret generic Kafka-auth --from-literal=username=<username> --from-literal=password=<password> --from-literal=truststore-location=/home/client.truststore.jks -n <project>-<environment>*

where,

<username> - the username credential that has access to the Kafka Broker.

<password> - the credential’s password that has access to the Kafka Broker.

<project> forms the namespace prefix. this is the project code. CB, COP, COB

<environment> forms the namespace suffix. this is the environment code. DEV, SIT, PRE, PRD

* Kafka cacert

[https://tsbchange.atlassian.net/wiki/spaces/CB/pages/1005060446/Kafka-cacert](https://tsbchange.atlassian.net/wiki/spaces/CB/pages/1005060446/kafka-cacert)

## AWS EKS Configuration

### Deployment process

Deployment of components is done via an automated Jenkins pipeline. The pipeline basically does a Helm Install of a localized Helm chart. Jenkins will need access to the target EKS environment with sufficient permissions to perform a Helm install within the environment. It would need to be able to create deployments, services, configmaps, etc.

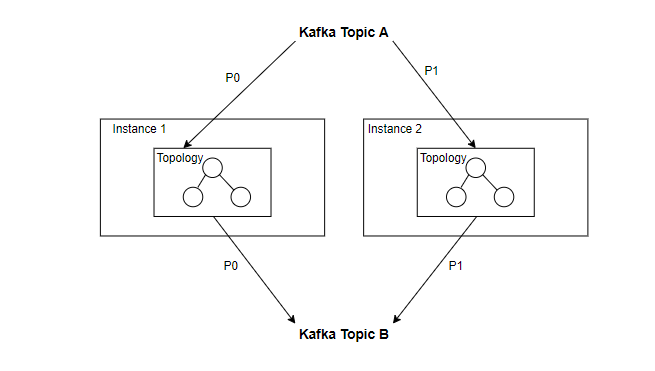
### AWS EKS – Application Topology configuration

**Multi-instance Model**

A topology defines the stream processing computational logic.

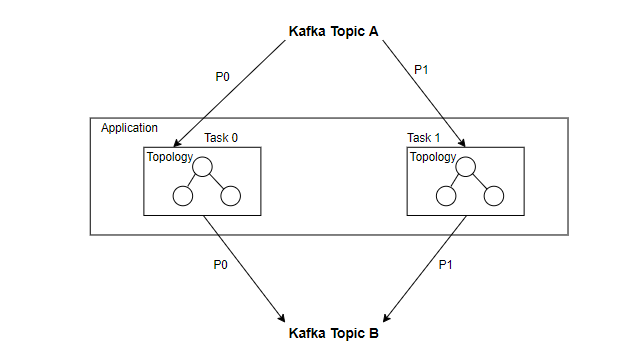
A stream processing application (or simply an application) defines one or more typologies.

In Kafka, parallelism is achieved by using **partitions**. The maximum parallelism at which an application may run is limited by the maximum number of the **partitions of the input topic(s)** the topology is reading from. For example, if an input topic (Topic A) has 2 partitions (P0, P1), then we can run up to 2 instances of the application (see the diagram below).



**Multi-thread Model**

In addition to instance parallelism, Kafka Stream allows to configure the number of threads that the library can use to parallelize processing within an application instance.

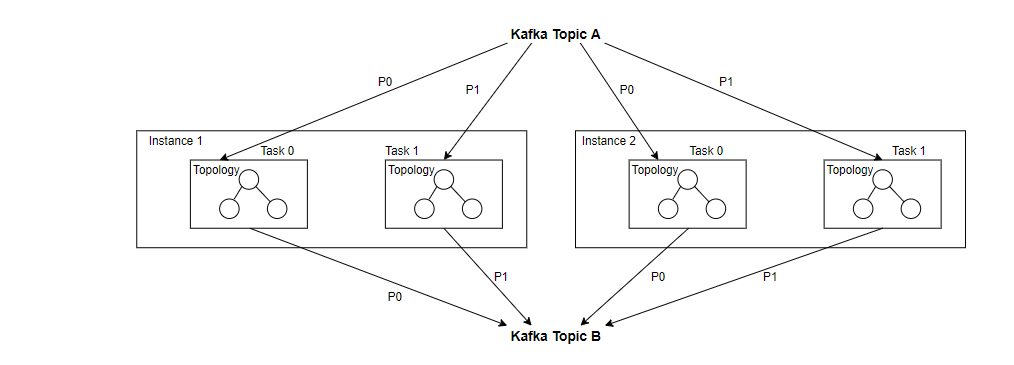


**Sub-topologies**

In addition to multi-threading and multi-instances, decomposing topologies into smaller sub-topologies (that call each other) will further scale-out the workload to multiple tasks.

**Combining Multi-instance, multi-thread and sub-topologies to Achieve Scalability**

Scaling an application with Kafka Streams is achieved by merely starting additional instances of an application, and Kafka Streams takes care of distributing partitions across sub-topologies that run in the application instances. We need to start as many threads of the application as there are input Kafka topic partitions so that, across all running instances of an application, every thread has at least one input partition to process (see the diagram below).

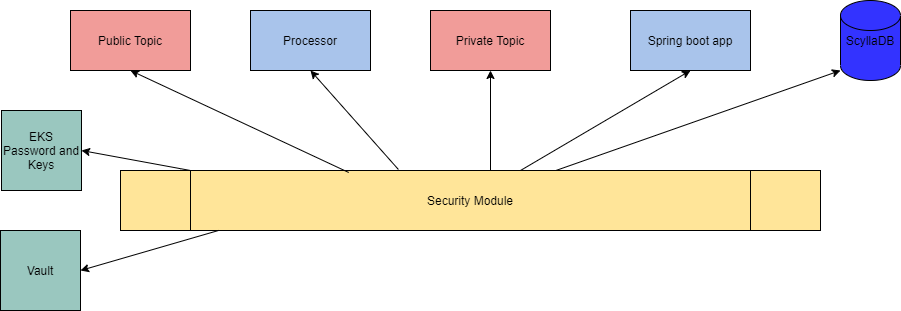


To achieve scalability and performance, we should decompose topologies into smaller sub-topologies and run one application per each topology. we can configure the number of partitions and threads for the topics and applications to achieve scalability.

For now, we can only assign an arbitrary number for the partitions and threads. This number then needs to be tuned (for each topic and topology) based on proper benchmarking and performance analysis.

## Security Module:

There will be common module for security across the application to be accessed by all components as below.





## One time/Historical load for Target tables (Raw/Curated)

Sri: can we use warehouse tables for historical load?

Design Will be updated based on inputs from CDC team

Option 1 – History data processed through IBM CDC/Kafka/ kstream process

* + - Need to check with IBM CDC team if history can be pushed one time into Kafka topic
    - Check Kafka configuration if can hold high volume history transaction data
    - Can we create a separate topic to process history data?

Gareth: This would be my preferred approach.  I’ve requested a session with the CDC SME to confirm whether this is possible.

Option 2 – Create separate batch process jobs (DataStage / Spring boot)

* + - Need to check the possibility of direct connectivity to Db2 tables

Option 3- Can Proteo team create a file Dump and transfer to AWS, Data team to load the file in to target table before starting the incremental load through CDC

**In- progress Design Approach as on 06-Nov-2020**

**Option1 is explored further and discussed with IBM CDC team .**

* + - We can do 1 time refresh of tables and historical data can be written to CDC/kafka topic
    - If we need to use the same public topic we need to collaborate with other teams , so that this time refresh doesn’t impact the inflight project like PUSH,CB, etc. – Need to check with Architecture forum teams to confirm usage of existing topics.
    - IBM CDC team suggested to create new subscription for ODS source tables so that the process is clean and it doesn’t impact other project.
    - On performance – CDC team mentioned it take 40 Minutes to refresh 40 million records.
    - One time refresh of KC03 transaction table need to be planned during the less mainframe load period
    - KC43 Historical Transaction table will be a new CDC subscription (as this table is not currently used by other team) and historical refresh need to be planned such that it spanned for multiple days ( sub incremental historical load) so that it doesn’t impact
    - Create new subscription /new topic for one time and switch to other topic. (cons -timestamp of the messages).
    - Need further Inputs from IBM on the options – other solution options like Sqoop also to be explored.
  1. Sri suggested if we can use Warehouse as source for history load process, also check if the input source files are SSL can be used – Need to check with Warehouse team , cost of doing this high, trust on the historical data in warehouse

Input from Mobile team on how they expose the current historical data one session/multiple session. – Sanjay to check with Mobile team

Final Approach :

* Option 1 CDC/Kafka is decided as recommended approach for historical data refresh
* CDC team will create a subscription and refresh for the new table/topic subscription ( KC43 transaction)
* ODS will use public topic for the tables which is already CDC subscribed by other projects ( KC03,BS46, other account ,customer tables)
* ODS project will target to get the ODS deployment to production with MVP configurations so that we meet the same timelines of other dependent project and there is no gap in history data processing.
* If there is any delay then ODS can leverage the Kafka retention duration of 90 days to process the historical data. ( Assumption ODS will be put in production before this 90 day period)

## Data Reconciliation

Design Will be updated based on inputs from CDC team (warehouse feed can be used for recon)

* + - Perform Aggregated Technical reconciliation check (like count, sum of trans amount) on both source (Db2) and Target (ScyllaDB)
    - Recon can be a separate batch process using spark/Spring run daily or weekly
    - Need to confirm if we can run aggregated queries on source DB2 tables. If not, can IBM CDC scheduled process run aggregated queries and put result in Kafka topic

Gareth: Interested to see what we can get from the CDC tool, as this may help here.  Failing that, we would get SABIS to produce some reconciliation counts, hash totals per table (probably a COBOL job).  If they ran a count at a point in time, we’d want them to record the count, hash totals and a run timestamp.  When we complete the rec on the ODS side, if we held a short history in the raw layer, we should be able to ‘rewind’ to the point in time the rec was completed to ensure we had a match.  Does that make sense?

**In- progress Design Approach as on 06-Nov-2020**

* + - On Discussion with the IBM CDC team inputs , We may not be able to trigger a scheduled Aggregated queries against source DB2 tables.
    - We need to check for alternate options
    - Sri suggested if we can use Warehouse table as source to do reconciliation and reduce the dependency on source DB2 tables

Engaging teams to achieve this solution approach what Gareth Suggested.

Final Approach :

* Run aggregated query for T-1 date in Db2 side ( Count, Sum of amount, Hash value) and expose it as API
* ODS to invoke the API and get the aggregated counts of DB2
* Run same aggregated query on ODS and compare the results
* Need discussion with Proteo team to decide on the process and approach to build the Cobol/Db2 stored procedure ( to run aggregated queries) and Expose it as API.

## Refresh of ODS tables on Load errors

Design options

* + - Based on the reconciliation results if there are any errors, we can reprocess the data b/w 2 reconciliation dates
    - E.g. if we do recon weekly and we see errors in 2nd week. We need to delete data loaded after last recon and reload 1-week data again
    - Kafka retention policy can be set to hold 30 to 90 days data, we can create separate reload jobs to process only week/month data from Kafka topic

Gareth: That makes sense.  I assume we’d have to consider at a database level or a table level.  What happens when this is running?  Is the data still accessible?  We need to conscious on availability in this scenario.

Detailing of the scenarios and data , detailed approach- validate process detects erros and publish in to a topic .

From business input- Transaction updates and SLA window for cancellation.

## Resiliency in Data ingestion Pipeline

Design options

* + - Assume Kafka infra level there is fail over (Kafka node failures) configuration enabled and ensure continuous availability of Kafka topic
    - Kafka will be configured for atleast once delivery option, if there are any application failures when we restart the application, we start reading from the point of failure
    - If there are any duplicates occur due to atleast once delivery setting, we will use the offset setting to avoid reading duplicate messages. Additionally, we can also set up insert/update load strategy for ODS tables

### Fault Tolerance

#### Kafka Application failure

In TSB, Kafka server is a 9 Node cluster and is resilient. Any failure is automatically handled by the server itself.

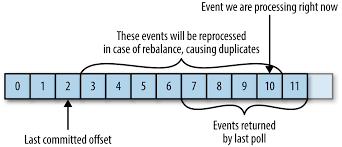
#### EKS failure

In TSB, EKS servers have auto recovery, if any node fails system automatically recover and

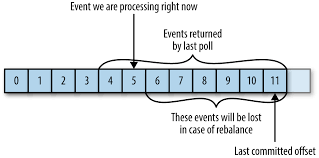
#### Consumer/ScyllaDB failure

When a consumer crashes it will trigger a rebalance. After a rebalance, each consumer may be assigned a new set of partitions than the one it processed before. In order to know where to pick up the work, the consumer will read the latest committed offset of each partition and continue from there. There are two scenarios:

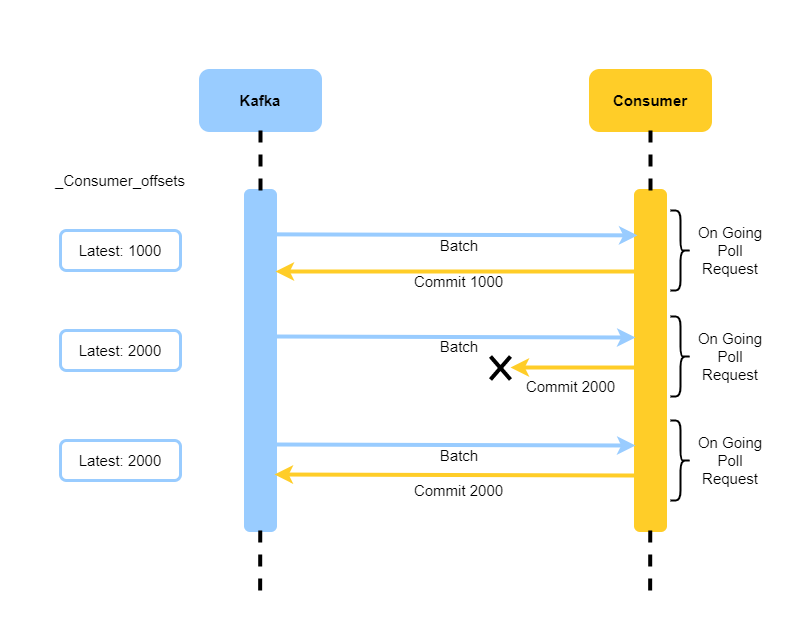
* If the committed offset is smaller than the offset of the last message the client processed, the messages between the last processed offset and the committed offset will be processed twice.



* If the committed offset is larger than the offset of the last message the client actually processed, all messages between the last processed offset and the committed offset will be missed by the consumer group



Proposed Solution: To ensure the data consistency, we choose commitSync() with Commit Current Offset because it will make sure that, before doing any further actions, we will know whether the offset commit is successful or failed. But because it is sync and blocking, it will spend more time on waiting for the commit to be finished, which leads to higher latency. (Check appendix for commitSync and Commit Current Offset)



## Data Quality

* **Source Data Quality –** SourceData Quality will be checked on key partition columns, any errors will be logged and notified.

## Monitoring/Logging

(Log 4J, Prometheus (central monitoring module), Grafana, ELK etc.)

Logging

All the logs are sent to Console in JSON format. Logs are sent to ELK module for monitoring.

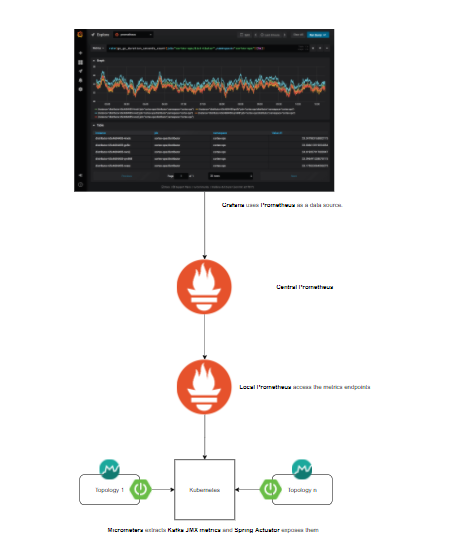
<https://www.elastic.co/guide/en/beats/filebeat/master/running-on-kubernetes.html>

What is logged

* All incoming Kafka message will be logged at DEBUG level
* All exceptions will be logged at ERROR level
* All incoming request to the Authentication API (access logs) will be logged.

<pattern>%date{yyyy-MM-dd HH:mm:ss.SSS} [${springAppName},%responseHeader{X-Request-Id}] - remote IP: %remoteIP | x-forwarded-for: %header{X-Forwarded-For} | logging uri: %requestMethod %requestURL | status code: %statusCode | elapsed time: %elapsedTime</pattern>

**Prometheus Monitoring:**In order to expose the metrics to the Prometheus operator, a ServiceMonitor object is created. It indicates where the topology is exposing the metrics.



Below are some metrics list for dashboard for Kubernetes, SpringBoot, Kafka client & broker and JVM. JVM metrics are automatically exported by the Spring Boot Actuator module (Ref: <https://grafana.com/grafana/dashboards/4701> )

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Source** | **Graph Type** |
| Instances | Number of Pods running show one Single stat per application | Kubernetes | Single stat & Graphic by time |
| Number of errors | Number of log back errors reported per minute per application | SpringBoot | Graphic by time |
| % Heap Used | % of Heap Memory Used per application | SpringBoot | Graphic by time |
| Pod Memory Usage | Pod Memory Used per application | Kubernetes | Graphic by time |
| % CPU | % Pod CPU per application | Kubernetes | Graphic by time |
| % Disk | % Disk space for mounted volumes per application | Kubernetes | Graphic by time |
| record-error-rate | The average per-second number of records sends that resulted in errors for a topic (Only for Kafka Processors) per application | Kafka Client | Graphic by time |
| Consumer message rate | Messages consumed per second breakdown per application | Kafka Client | Graphic by time |
| Consumer Lag | Number of messages by which the consumer lags behind the producer | Kafka Client | Graphic by time |
| Uptime | Time since application start | JVM (Auto Exported) | Graphic by time |
| Start time | Last start time | JVM (Auto Exported) | Graphic by time |
| % Heap Used | % of Heap Memory Used | JVM (Auto Exported) | Graphic by time |
| % Non-Heap Used | % of Non-Heap Memory Used | JVM (Auto Exported) | Graphic by time |
| Number of errors | Number of log back errors reported per minute | JVM (Auto Exported) | Graphic by time |
| Number of Garbage collections | Number of garbage collections fired | JVM (Auto Exported) | Graphic by time |
| ProcessCpuLoad | ProcessCpuLoad represents the CPU load in this process. | JVM (Auto Exported) | Graphic by time |
| ProcessCpuTime | This is the time the CPU has spent running this process. | JVM (Auto Exported) | Graphic by time |
| Rate | The number of requests across all connectors | JVM (Auto Exported) | Graphic by time |
| Duration | The total time to process all incoming requests, in milliseconds | JVM (Auto Exported) | Graphic by time |
| errorCount | The number of errors generated by server requests | JVM (Auto Exported) | Graphic by time |
| % CPU | % CPU of the process for each broker | Kafka Broker | Graphic by time |
| JVM Heap vs Max JVM Heap | JVM Heap Memory Usage For each JVM | Kafka Broker | Graphic by Time |
| Disk Size vs Max Disk Size | This metric is not included in the provided Kafka dashboard, but should be added | Node Exporter | Graphic by Time |
| Server Free memory | Node Free memory for each broker | Node Exporter | Graphic by Time |
| Swap space | Swap space for each broker | Node Exporter | Graphic by Time |
| Under-replicated Partitions | Number of under-replicated partitions | Kafka Broker | Single stat & Graphic by time |
| Offline Partitions | Number of partitions that don't have an active leader | Kafka Broker | Single stat & Graphic by time |
| ActiveControllerCount | Number of active controllers in the cluster. | Kafka Broker | Single stat & Graphic by time |
| Fetcher Lag | Lag in the number of messages per follower replica. | Kafka Broker |  |
| BytesInPerSec | Aggregate incoming byte rate. | Kafka Broker |  |
| BytesOutPerSec | Aggregate outgoing byte rate. | Kafka Broker |  |
| MessagesInPerSe | Aggregate incoming message rate | Kafka Broker |  |
| RequestsPerSec | Request rate. | Kafka Broker |  |
| *Network Errors Rate* | Network error | Kafka Broker |  |
| TotalProduceRequestsPerSec | Produce request rate. | Kafka Broker |  |
| TotalFetchRequestsPerSec | Fetch request rate. | Kafka Broker |  |
| FailedProduceRequestsPerSec | Produce request rate for requests that failed. | Kafka Broker |  |
| FailedFetchRequestsPerSec | Fetch request rate for requests that failed. | Kafka Broker |  |
| UncleanLeaderElectionsPerSec | Unclean leader election rate. | Kafka Broker | Single stat & Graphic by time |
| PartitionCount | Number of partitions on this broker. | Kafka Broker | Single stat & Graphic by time |
| Brokers Online | Number of leaders on this broker. | Kafka Broker | Single stat & Graphic by time |
| MaxLag | Maximum lag in messages between the follower and leader replicas. | Kafka Broker | Graphic by Time |
| Avg Request handler Idle Percent | Average fraction of time the request handler threads are idle. | Kafka Broker | Graphic by Time |
| Avg Network Processor Idle Percent | Average fraction of time the network processor threads are idle. | Kafka Broker | Graphic by Time |
| CPU | % of CPU Usage per server | Node Exporter | Graphic by Time |
| Free memory | Free memory per server | Node Exporter | Graphic by Time |
| Swap memory | Used swap memory per server | Node Exporter | Graphic by Time |
| Disk space | Available disk space vs Total disk space per mounted volume | Node Exporter | Graphic by Time |
| Network traffic | Input byte rate, output byte rate, | Node Exporter | Graphic by Time |
| Number of servers | Number of online servers | Node Exporter | Graphic by Time |

Ref: <https://tsbchange.atlassian.net/wiki/spaces/CB/pages/1051689365/Push+Notification+Metrics>

# Ciso Integration /Certification

# DevOps

**The Dev Ops tools and pipeline must be introduced for ODS application and overview to be captured here.**

# Appendix

**The confluence link for the contents are below:**

### Kafka topics Name:

<https://tsbchange.atlassian.net/wiki/spaces/ITARCH/pages/231571457/Policy+KAFKA+Topics+Naming>

### Consume exactly Once

[https://dzone.com/articles/Kafka-internals-consumer#:~:text=Consume%20Exactly%20Once](https://dzone.com/articles/kafka-internals-consumer#:~:text=Consume%20Exactly%20Once)

### Commit current Offset

[https://www.oreilly.com/library/view/Kafka-the-definitive/9781491936153/ch04.html#:~:text=Commit%20Current%20Offset](https://www.oreilly.com/library/view/kafka-the-definitive/9781491936153/ch04.html#:~:text=Commit%20Current%20Offset)