Prometheus - Customer Matching Engine - [Solution Overview]

The customer matching engine provides the capacity to match customers from different sources against the health insurance member list.

Key requirements for the matching engine include:

* Ability to take data from a variety of sources
* Ability to perform both deterministic (equality) and fuzzy matches
* Ability to determine best match for each incoming source record
* Ability to capture best matches into a table along with match confidence for use by other systems and user communities.

Further details of the engine can be reviewed via the below links:

* [CME Notebook Flow](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183378481/CME+Notebook+Flow)
* [CME Source Standardisation](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183083691/CME+Source+Standardisation)
* [CME Source Reporting](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183083725/CME+Source+Reporting)
* [CME Match Rules](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183116473/CME+Match+Rules)
* [CME Matching Process](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183804732/CME+Matching+Process)
* [CME BDP4 Migration](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3195602002/CME+BDP4+Migration)
* [CME POC Synthetic Data](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3196453108/CME+POC+Synthetic+Data)
* [BDP4 Rehosted CME Solution Design](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286368348/BDP4+Rehosted+CME+Solution+Design)

Further details of the epic and feature breakdowns relating to the CME can be found at [MVP of Customer Matching Engine using existing capability to provide single view of customer to connected care and HI use cases.](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3091595271)

Implementation

At current the SSAA platform has been used to prepare a matching capability. This is a tactical implementation that is expected to be supplanted by the strategic MDM tooling currently in the process of being specified/selected by the “Data Management Strategic Tooling” initiative.

The SSAA implementation is being used to prove the concept of fuzzy customer matching, and also to inform the MDM tooling selection process.

Please see the SSAA Implementation page for details of this tactical implementation.

CME Notebook Flow

The SSAA implementation of the customer matching engine is implemented as a series of notebooks within the [SSAA-customerandstrategyoffice Customer-Matching-Engine](https://adb-6334213679475485.5.azuredatabricks.net/browse/folders/4235815684247994?o=6334213679475485) worspace.

The flow of notebooks is as per the below:

The notebooks and their function is described in the below table:

|  |  |  |
| --- | --- | --- |
| **Notebook** | **Function** | **Output** |
| **Notebook** | **Function** | **Output** |
| CME Master Standardisation | Standardises HI\_MEMBER data as required by matching process | cme\_fuzzy.hi\_member\_std |
| CME Optical Standardisation | Standardises BOP\_CLIENT data as required by matching process | cme\_fuzzy.bop\_client\_std |
| CME Dental Standardisation | Standardises DC\_PATIENT data as required by matching process | cme.fuzzy.dc\_patient\_std cme.fuzzy.dc\_patient\_txn |
| CME Optical Matching | Performs matching of optical against the member master | cme\_fuzzy: bop\_match\_target bop\_fuzzy\_confidence bop\_match |
| CME Dental Matching | Performs matching of dental customer against the member master | cme\_fuzzy: dc\_match\_target dc\_fuzzy\_confidence dc\_match |
| CME Optical Reporting | Reports results of matching rules for optical data |  |
| CME Dental Reporting | Reports results of matching rules for dental data |  |
| CME Consolidated Master | Takes matches and creates combined master | cme\_fuzzy.hi\_master\_matched |

# CME Source Standardisation

Standardisation must be applied to data being matched by the CME so that the rules that are being applied may be reused across different sources. This minimises the amount of work required to integrate a new source into the engine.

## Standardisation Notebook

Each source has its own standardisation notebook that effects the standardisations. This notebook takes the data from the source, standardises it and persists it to the CME\_FUZZY schema in SSAA. The name of the standardised source table is <source\_name>\_std

## Standardisations Applied

Details of the standardisations are discussed in the sections below:

### Field Name Standardisation

Rules require a set of fields to have standardised names/types so they can be run across all data sets.

Current fields and their types that require standardisation are shown in the below table:

|  |  |  |
| --- | --- | --- |
| **Description** | **Standardised Name** | **Standardised Type** |
| First Name | FIRST\_NAME |  |
| Last Name | LAST\_NAME |  |
| Gender | GENDER | CHAR(1): M or F or Null  This is the standard in the HI\_Membership master source |
| Health Fund Number | HEALTH\_FUND\_NUMBER |  |
| Email Addresss | EMAIL\_ADDRESS |  |
| Mobile Number | MOBILE\_NUMBER |  |
| Source Key (Natural key or composite for the customer data source) | SOURCE\_NATIVE\_KEY |  |
| Date of Birth | DATE\_OF\_BIRTH | Date |
| Address | ADDRESS | String containing full address in format it comes from HI Master  for example “Flat 75, 73 Russell Street, HAYWARDS BAY, NSW 2530, AU” |

Objects containing the standardised fields are suffixed by \_STD to denote this.

### Domain Standardisation

As the master information is originally sourced from the HI\_Membership object, this defines the form (domain) that standardised attributes are to take. For example, HI\_Membership has only three gender values - M, F and Null. Accordingly all sources being standardised should have their Gender field populated with one of these three values.

### These Attributes Will Change

The list if fields above is the first cut of the standardised attributes. This enables the initial matching that was performed in the fuzzy matching prototype, as well as Address and Date Of Birth. It is expected additional attributes will be brought into the standardised list (e.g. landline number) such that rules that require them will be able to execute. These rules should increase match performance significantly.

Anticipated changes include:

* Introduction of additional phone numbers (over mobile already present)
  + Landline is present within dental but not in the HI Master data supplied
* Service Delivery Location
  + Service delivery location (e.g. location of dental or optical practice) should provide an ability to perform geo matching where customer address is not present
  + This would require dental/optical service location to be added to the source dataset and could leverage cso\_analytics\_feature\_store.closest\_locations\_bupa\_dental and similar objects to provide additional match data

## Source Internal Duplicates

Sources may not provide a unique customer list, since the data is often sourced from transactional systems that re-enter patient data each time they present. This is the case for both the dental and optical sources used in the CME SSAA processing.

Where multiple records in a source refer to the same entity (customer) it is highly desirable that this data be matched internally to the source such that only one customer record from that source is matched against the master.

Sources that contain multiple records which match a single master record (as would be expected if there are multiple records in a source for a single entity) are problematic, since they introduce product joins to any attempt to create a consolidated master record, containing the master key and the keys from each source as separate columns.

For example, If dental has 3 records that match a master record (because a single customer attended dental 3 times) and optical has 4 records that match (single customer 4 presentations) then we would see 12 records in a consolidate master.

There are numerous instances of this circumstance captured in the [Duplicate Match Investigation](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/1915044030454767/command/1915044030454768) notebook. In the [CME Consolidated Master](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/3873140696955074/command/1915044030454779) notebook the consolidated master is rendered into a customer cohort that accounts for this, with a columns specifying if a member is a dental customer and/or an optical customer., but not the actual records.

For the approach to address internal duplicates please see [CME Source Internal Matching](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3194685783)

## Historical Attribute Standardisation

At the current time only current HI Member information is being used to perform matching. This means that historical contact and address details are opaque to the matching engine and some matches that could easily qualify based on historical contact details will be omitted.

In order for historical attributes to be employed a mechanism for standardising prior values of contact (phone/email/address) would be required. This would be held against the master, and would be able to be augmented with data obtained from source systems once matched.

For example, if a source had a record which matches with high confidence to the master, and that record has a mobile number that is hitherto unknown to the master, then the master would be augmented with that number for the purposes of matching in future.

Similarly for name, where a customer has a changed last name (for example due to marriage) the name data for the master record would be augmented such that the new last name is included in future source matching.

CME Source Internal Matching

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Nov 21, 2023

5 min read3 people viewed

Problem Statement

As mentioned on the source standardisation page, sources may not provide a unique customer list, since the data is often sourced from transactional systems that re-enter patient data each time they present. This is the case for both the dental and optical sources used in the CME SSAA processing.

Where multiple records in a source refer to the same entity (customer) it is highly desirable that this data be matched internally to the source such that only one customer record from that source is matched against the master.

Sources that contain multiple records which match a single master record (as would be expected if there are multiple records in a source for a single entity) are problematic, since they introduce product joins to any attempt to create a consolidated master record, containing the master key and the keys from each source as separate columns.

For example, If dental has 3 records that match a master record (because a single customer attended dental 3 times) and optical has 4 records that match (single customer 4 presentations) then we would see 12 records in a consolidate master.

There are numerous instances of this circumstance captured in the [Duplicate Match Investigation](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/1915044030454767/command/1915044030454768) notebook. In the [CME Consolidated Master](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/3873140696955074/command/1915044030454779) notebook the consolidated master is rendered into a customer cohort that accounts for this, with a columns specifying if a member is a dental customer and/or an optical customer., but not the actual records.

Internal Matching Process

As part of standardisation internal matching should be performed. This internal matching is done in exactly the same fashion that the source-master matching is done, using the same rules and infrastructure, but instead of matching the master, the source itself is used as the master.

Here is an example of the match code that illustrates this:

# First we get the dental standard table as both master and source

source\_std = spark.table("cme\_fuzzy.dc\_patient\_std")

master\_std = spark.table("cme\_fuzzy.dc\_patient\_std")

# Setup the rule objects with the configuration needed to perform internal match

# NOTE: We do not persist the results to global match\_target as they are

# local to the source, and we manage the match set explicitly within

# this notebook

# Deterministic Rules

dcn\_rule = DeterministicContactNameRule(persist=False)

dfn\_rule = DeterministicFundNumberNameRule(persist=False)

# Fuzzy Rules

fcn\_rule = FuzzyContactNameRule(persist=False)

fdbn\_rule = FuzzyDOBNameRule(persist=False, minimum\_confidence=0.3)

dcp\_dcn\_results = dcn\_rule.run(

"dc\_patient",

source\_std,

"dc\_patient\_internal",

master\_std

)

# Get all source keys that were not matched in the last operation

dc\_patient\_dcn\_unmatched\_keys = (

dcp\_dcn\_results

.filter(col("Target\_Key").isNull())

.select("Source\_Key")

)

dc\_patient\_std\_dcn\_unmatched = (

dc\_patient\_std

.alias("std")

.join(

dc\_patient\_dcn\_unmatched\_keys,

on=(dc\_patient\_std.SOURCE\_NATIVE\_KEY == dc\_patient\_dcn\_unmatched\_keys.Source\_Key),

how='inner'

)

)

... other rules run here ...

The output of the matching process is a map table which has for each internally matched record a primary record, and the keys of the matched records.

The primary record is simply the one with lowest source natural key in the match group.

Here is code that generates the map:

# Pick the survivor...

survivor\_keys = (

spark.sql(

"""

SELECT

Source\_Key as Deduped\_Source\_Key,

max(Source\_Match\_Rank) as Max\_Rank,

count(\*) as Num\_Maps

FROM

(

SELECT

Target\_Key,

Source\_Key,

ROW\_NUMBER() OVER(PARTITION BY Target\_Key ORDER BY Source\_Key) as Source\_Match\_Rank

FROM

cme\_fuzzy\_tmp.dc\_internal\_det\_matches

) src

GROUP BY

Source\_Key

HAVING

max(Source\_Match\_Rank) = 1

;

"""

)

.select(

"Deduped\_Source\_Key",

"Num\_Maps"

)

)

# Create the survivor map

matches = spark.table('cme\_fuzzy\_tmp.dc\_internal\_det\_matches')

survivor\_map = (

matches

.alias('m')

.join(

survivor\_keys

.alias('keys'),

on=(matches.Target\_Key == survivor\_keys.Deduped\_Source\_Key)

)

.withColumn(

"Original\_Source\_Key",

col("m.Source\_Key")

)

.select(

"Original\_Source\_Key",

"Deduped\_Source\_Key"

)

)

# Cache the map in a table

survivor\_map.write.mode('overwrite').saveAsTable('cme\_fuzzy.dc\_patient\_std\_map')

Match Group Standardised Source

After the map is prepared, there needs to be another version of the standardised table created that has all combinations of values from each record in the match group. This is the table that is ultimately matched to master.

This ensures that the matching can be performed against the group as a whole, not just the primary record (the record with the SOURCE\_NATURAL\_KEY = Deduped\_Source\_Key in the map).

To create this table we must “shred the record” normalising it into separate dataframes for:

* Name
* DOB
* Mobile Number
* Email Address
* Health Fund Number
* Address

Only records where these attributes are not null are included in the normalised data.

Code that implements this is shown below:

# OK - let's shred the standardised table

source\_std = spark.table("cme\_fuzzy.dc\_patient\_std")

source\_name\_std = (

source\_std

.withColumn(

"ORIGINAL\_NATIVE\_KEY",

col("SOURCE\_NATIVE\_KEY")

)

.select(

"ORIGINAL\_NATIVE\_KEY",

"FIRST\_NAME",

"LAST\_NAME",

"GENDER"

)

)

source\_hf\_std = (

source\_std

.filter(col("HEALTH\_FUND\_NUMBER").isNotNull())

.withColumn(

"ORIGINAL\_NATIVE\_KEY",

col("SOURCE\_NATIVE\_KEY")

)

.select(

"ORIGINAL\_NATIVE\_KEY",

"HEALTH\_FUND\_NUMBER"

)

)

source\_email\_std = (

source\_std

.filter(col("EMAIL\_ADDRESS").isNotNull())

.withColumn(

"ORIGINAL\_NATIVE\_KEY",

col("SOURCE\_NATIVE\_KEY")

)

.select(

"ORIGINAL\_NATIVE\_KEY",

"EMAIL\_ADDRESS"

)

)

source\_mobile\_std = (

source\_std

.filter(col("MOBILE\_NUMBER").isNotNull())

.withColumn(

"ORIGINAL\_NATIVE\_KEY",

col("SOURCE\_NATIVE\_KEY")

)

.select(

"ORIGINAL\_NATIVE\_KEY",

"MOBILE\_NUMBER"

)

)

source\_dob\_std = (

source\_std

.filter(col("DATE\_OF\_BIRTH").isNotNull())

.withColumn(

"ORIGINAL\_NATIVE\_KEY",

col("SOURCE\_NATIVE\_KEY")

)

.select(

"ORIGINAL\_NATIVE\_KEY",

"DATE\_OF\_BIRTH"

)

)

source\_address\_std = (

source\_std

.filter(col("ADDRESS").isNotNull())

.withColumn(

"ORIGINAL\_NATIVE\_KEY",

col("SOURCE\_NATIVE\_KEY")

)

.select(

"ORIGINAL\_NATIVE\_KEY",

"ADDRESS"

)

)

We then take each of these normalised structures, join them to the standardised table and the map to get a product of all the attribute values mapped to the primary record key.

Code implementing this is shown below (note that only DOB and health fund number are included for berevity):

tmp = (

source\_map

.alias('map')

.withColumn(

"SOURCE\_NATIVE\_KEY",

col("Deduped\_Source\_Key")

)

.join(

source\_name\_std

.alias('name'),

on=(

(source\_name\_std.ORIGINAL\_NATIVE\_KEY == source\_map.Original\_Source\_Key) |

(source\_name\_std.ORIGINAL\_NATIVE\_KEY == source\_map.Deduped\_Source\_Key)

),

how="leftouter"

)

.join(

source\_hf\_std

.alias('hf'),

on=(

(source\_hf\_std.ORIGINAL\_NATIVE\_KEY == source\_map.Original\_Source\_Key) |

(source\_hf\_std.ORIGINAL\_NATIVE\_KEY == source\_map.Deduped\_Source\_Key)

),

how="leftouter"

)

.select(

"SOURCE\_NATIVE\_KEY",

"FIRST\_NAME",

"LAST\_NAME",

"GENDER",

"HEALTH\_FUND\_NUMBER",

)

).distinct()

CME Source Reporting

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Nov 14, 2023](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3183083725&selectedPageVersions=2&selectedPageVersions=3)

2 min read4 people viewed

Each source matched by the Customer Matching Engine has its own reporting notebook that permits users of the CME to observe the performance of the matching algorithms for that source.

These notebooks are named “<source\_name> Reporting” and can be found in the [Customer-Matching-Engine workspace](https://adb-6334213679475485.5.azuredatabricks.net/browse/folders/4235815684247994?o=6334213679475485).

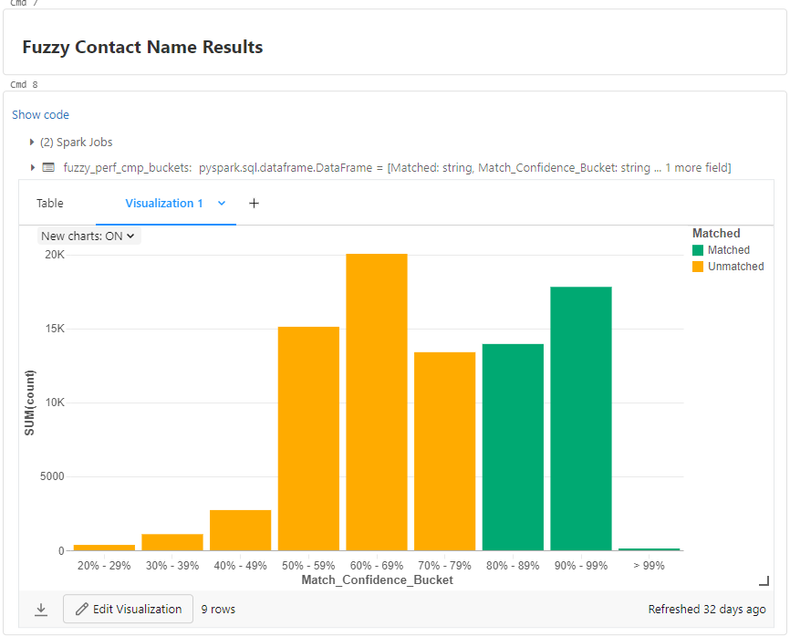
Each notebook takes data from the match result tables that were populated by the matching notebook for each system. It presents visualisations for the performance of each fuzzy rule and a sankey that illustrates the performance of the matches as each source record moves through the matching process.

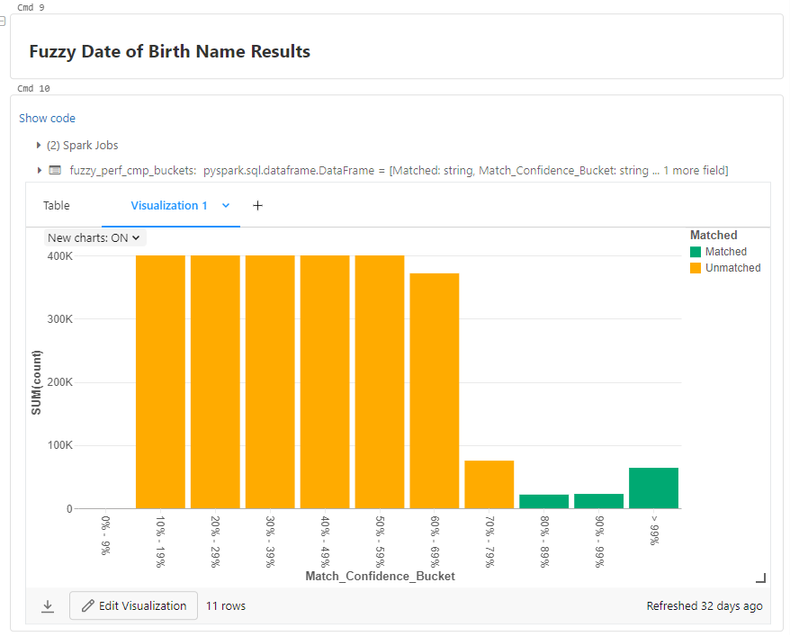
Example Reporting Output

The first visualisation provides the confidence distribution for the fuzzy matching algorithms, with the number of matches in each 10% range.

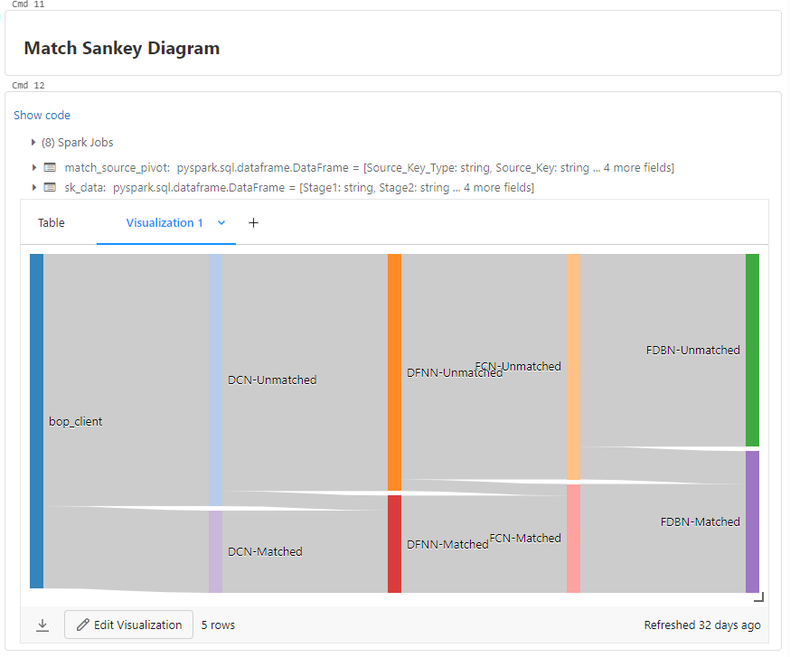


The individual fuzzy rule performance is then shown for each rule in turn. The matches considered valid are shown in green…



Open Screenshot (20).png

Finally a sankey diagram is shown - this shows how many matches were added by each rule in turn. The vertical bars correspond to each rule. The leftmost bar shows the incoming data records. Hovering over any of the bands supplies the number of records matched or otherwise by each rule.



CME Match Rules

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Nov 14, 2023](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3183116473&selectedPageVersions=2&selectedPageVersions=3)

7 min read4 people viewed

Matching rules are employed to identify match candidates and score them. This page describes the rules and the infrastructure under which they are run.

Defined Rules

The rules defined include the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rule ID** | **Description** | **Cluster** | **Confidence Algo** | **Source/Targets** |
| Deterministic-Contact-Name | Match of Client First, Last, Gender with matching Email or Mobile | email or mobile identical | 100% Hardcoded (i.e. 1.0) | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Deterministic-FundNumber-Name | Match of Client First, Last, Gender with matching Health Fund | Health Fund Number Identical | 100% Hardcoded (i.e. 1.0) | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Fuzzy-Contact-Name | Fuzzy Match on Concatenated First, Last, Gender, Email, Mobile with matching email or mobile | email or mobile identical | Fuzzy Match on Concatenated First, Last, Gender, Email, Mobile | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Fuzzy-DOB-Name | Fuzzy match on Concatenated First/Last/Gender with matching DOB. This is used as the base calculator for Name. Address, Mobile, Fund Number etc. fuzzy match calculations used in compound rules. | Date of Birth Identical | Fuzzy Match on Concatenated First, Last, Gender | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Combined-Weighted | Final weighted combination of other rules | N/A | Subject to tuning | * All matches present |

Clustering

Matching cannot be applied scalably between every record and every other record as this results in a quadratic explosion of the match candidates considered. This leads to performance and resource consumption issues with all but the smallest data volumes. As we are considering in excess of 4M records, we must first cluster the groups of entites that are matched.

In the table above cluster is used to pre-qualify records to match prior to obtaining any fuzzy matching score. Different rules have different clusters applied to them as required. In MVP1 this takes the form of a join, however in MVP3 clustering will likely be performed via a ML clustering algorithm.

Rule Classes

The matching rules are implemented as classes in the [CME Matching Infrastructure](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/4235815684248147/command/4235815684248148) notebook.

This class provides a standard way to perform, log and return results of match rules. Invoking the rule involves:

1. Creating dataframes for the source and master
2. Creating a Rule object with any parameters needed (e.g. persist/don’t persist results, persist only those results having greater than specified confidence etc. - see MatchRule class definition for details).
3. Calling the run method of the rule object with the required source/master dataframes

# Get the source and master standardised dataframes

hi\_master = spark.table('cme\_fuzzy.hi\_membership\_std')

optical\_patient = spark.table('cme\_fuzzy.bop\_client\_std')

# Create a rul object

fdbn\_rule = FuzzyDOBNameRule()

# Get match results

match\_results = fdbn\_rule.run(

'bop\_client',

optical\_patient

'hi\_master',

hi\_master

)

The above populates a match\_results dataframe which has the following structure:

* Rule\_ID: string (nullable = false)
* Source\_Key\_Type: string (nullable = false)
* Source\_Key: string (nullable = false)
* Target\_Key\_Type: string (nullable = false)
* Target\_Key: string (nullable = false)
* Match\_Confidence: decimal(10,0) (nullable = false

Here is the class definition:

# Class definition for matching rules

# This class requires only that the rule\_id and the get\_rule\_dataframe be defined

class MatchRule():

rule\_id = None

def \_\_init\_\_(self, \*args, \*\*kwargs):

# When establishing the rule object you may set

# default values for unmatched and persist flags

# These may be overridden when calling the run method

# Unmatched determines if the records that were not matched at all

# are included in the results (with a null Target Key)

self.include\_unmatched = kwargs.get('unmatched', True)

# Persist determines if the results of the rule are merged

# into the match target. If they are not merged (persist=False)

# then the user is responsible for handling the output

self.persist = kwargs.get('persist', True)

self.minimum\_confidence = kwargs.get('minimum\_confidence', None)

def get\_rule\_dataframe(

self,

match\_source\_name: str,

match\_source: DataFrame,

master\_source\_name: str,

master\_source: DataFrame,

\*\*kwargs

) -> DataFrame:

raise NotImplementedError("")

def run(

self,

match\_source\_name: str,

match\_source: DataFrame,

master\_source\_name: str="hi\_membership",

master\_source: DataFrame=hi\_membership\_std,

\*\*kwargs

) -> DataFrame:

global match\_target

global runnable\_rules

if self.rule\_id not in runnable\_rules:

print(f"WARNING: {self.rule\_id} rule is not being run as it is not present in runnable rules")

return

print(f"Running rule {self.rule\_id} for {match\_source\_name} against {master\_source\_name}...")

raw\_results = self.get\_rule\_dataframe(

match\_source\_name,

match\_source,

master\_source\_name,

master\_source,

\*\*kwargs

)

if self.include\_unmatched:

print("Both matched and unmatched records will be returned.")

match\_results = raw\_results

else:

print("Only matched records will be returned")

match\_results = raw\_results.filter(col("Target\_Key").isNotNull())

if self.minimum\_confidence:

print(f"Only records with a match confidence of at least {self.minimum\_confidence} will be returned.")

results = match\_results.filter(col("Match\_Confidence") >= self.minimum\_confidence)

else:

results = match\_results

if self.persist:

print(f"Merging {self.rule\_id} results to match target dataframe...")

match\_target = match\_target.union(results)

else:

print(f"Merge of {self.rule\_id} results into match target is disabled by persist=False")

print(f"Rule {self.rule\_id} run is complete")

return results

Each rule is a subclass of the MatchRule class and implements its own instance of the “get\_rule\_dataframe” method. This method is responsible for comparing the two supplied dataframes (the standardised master and the source datafraomes) and returning results in the standard match\_table format.

Here is an example rule that fuzzily matches based on name and gender after joining on date of birth:

class FuzzyDOBNameRule(MatchRule):

rule\_id = "Fuzzy-DOB-Name"

def get\_rule\_dataframe(

self,

match\_source\_name: str,

match\_source: DataFrame,

master\_source\_name: str,

master\_source: DataFrame,

\*\*kwargs

) -> DataFrame:

# Define The Join Condition

fuzzy\_join\_condition = (

(match\_source.DATE\_OF\_BIRTH == master\_source.DATE\_OF\_BIRTH)

)

# Get the matches as per the above

FCN\_rule\_matches = (

match\_source

.select(

"SOURCE\_NATIVE\_KEY",

"FIRST\_NAME",

"LAST\_NAME",

"GENDER",

"HEALTH\_FUND\_NUMBER",

"EMAIL\_ADDRESS",

"MOBILE\_NUMBER",

"DATE\_OF\_BIRTH"

)

.withColumn(

"concat\_string",

concat(

col("FIRST\_NAME"),

lit(" "),

col("LAST\_NAME"),

lit(" "),

col("GENDER"),

)

)

.alias("source")

.join(

master\_source

.withColumn(

"concat\_string",

concat(

col("FIRST\_NAME"),

lit(" "),

col("LAST\_NAME"),

lit(" "),

col("GENDER"),

)

)

.alias("master"),

on=fuzzy\_join\_condition,

how="leftouter"

)

.withColumn("Rule\_ID", lit(self.rule\_id))

.withColumn("Source\_Key\_Type", lit(match\_source\_name))

.withColumn("Source\_Key", col("source.SOURCE\_NATIVE\_KEY"))

.withColumn("Target\_Key\_Type", lit(master\_source\_name))

.withColumn("Target\_Key", col("master.SOURCE\_NATIVE\_KEY"))

.withColumn(

"Match\_Confidence",

when(col("master.SOURCE\_NATIVE\_KEY").isNotNull(), fuzz\_udf(col("source.concat\_string"), col("master.concat\_string")) / 100)

.otherwise(lit(0.0))

)

.select(

"Rule\_ID",

"Source\_Key\_Type",

"Source\_Key",

"Target\_Key\_Type",

"Target\_Key",

"Match\_Confidence",

)

)

return FCN\_rule\_matches

# CME Matching Process

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Dec 19, 2023](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3183804732&selectedPageVersions=5&selectedPageVersions=6) by [Hemanth Parasaram](https://bupateam.atlassian.net/wiki/people/5c623fd2cefe97640e699a8b?ref=confluence&src=profilecard)

6 min read4 people viewed

The matching processing performed by CME is located in a standalone workbook for each source (separate to the standardisation and reporting notebooks).

These notebooks are named “<source\_name> Matching” and can be found in the [Customer-Matching-Engine](https://adb-6334213679475485.5.azuredatabricks.net/browse/folders/4235815684247994?o=6334213679475485) workspace.

These notebooks contain the following high level steps:

1. Run the CommonSetup notebook (sets up the python libraries and matching configuration parameters)
2. Run the CME Matching Infrastructure notebook (sets up the target tables and rule reference and rule classes)
3. Source the standardised source and master data (this requires that the master standardisation and the source standardisation notebooks have already been run to prepare the data)
4. Execute each of the rules in turn
5. Persist the match results to a table

Each of these steps will be discussed in further detail in the sections below.

Upon completion of the matching process the reporting notebook can be run to review results, and to obtain a consolidated view of all the matches the CME Consolidated Master notebook can be run.

## Run Common Setup

The common setup notebook installs the matching libraries and sets up the matching configuration.

### Matching Library Installation

Matching libraries employed include:

* TheFuzz (Levenshtein Distance Algorithm)
* jaro-winkler (Jaro-Winkler Algorithm)

These libraries are installed using %pip install

### Matching Configuration

Matching parameters are setup using python code as follows:

# Matching Parameter Definition

# =========================================================================================

# Setup standardisation driver vars

# =========================================================================================

# This causes the gender domains to be standardised to the HI\_Membership domain for each source

matchable\_genders = ['M','F']

standardise\_genders = True

standardise\_mobile = True

# This causes the master only to consider the active members (otherwise inactive members are matched too)

master\_active\_only = True

# =========================================================================================

# Setup match confidence threshold

# =========================================================================================

# This is the confidence level at which a match candidate is considered to be accepted

# This is in the range of 0..1 - e.g. 0.8 is an 80% confidence level

match\_confidence\_threshold = 0.80

# =========================================================================================

# Setup sources that are to be matched

# =========================================================================================

# This is now superceded by the separate reporting notebook for each source

matchable\_sources = [

"bop\_customer",

"dc\_patient"

]

# =========================================================================================

# Setup rules to be run

# =========================================================================================

runnable\_rules = [

"Deterministic-Contact-Name",

"Deterministic-FundNumber-Name",

"Fuzzy-Contact-Name",

"Combined-Weighted"

# The below fuzzy matches require tuning of the clustering used as they result in > 5bn matches (for now)

# "Fuzzy-Name-Details",

# "Fuzzy-LastName-Details",

]

skip\_dc\_fuzzy = True

When this is run key settings are reported to the output to document the settings in force when the matching was performed. See below for an example of this output:

Only active members will be included in the master against which matching is performed.

Matched datasets will have gender values standardised to match master gender domain.

Matched datasets will have mobile number values standardised to match master mobile number domain.

Match minimum confidence threshold is set to 80.0%

Rules that will be run include:

- Deterministic-Contact-Name

- Deterministic-FundNumber-Name

- Fuzzy-Contact-Name

- Combined-Weighted

## Run CME Matching Infrastructure

The CME Matching Infrastructure notebook:

* Creates the rule reference dataframe and persists it to the cme\_fuzzy.rules table
* Defines the match target dataframe structure into which all match results are populated
* Defines the base class for Matching Rules (see [CME Match Rules](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183116473) for details)
* Defines classes for each of the Matching Rules (see [CME Match Rules](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183116473) for details)

This content is included in the matching infrastructure notebook instead of the source matching workbook to permit reuse across multiple source systems.

Any additional rules or matching functionality that is desired should be included in the matching infrastructure notebook so that all sources may benefit without a need to duplicate anything.

## Obtain Standardised DataFrames

Each source matching workbook has its own source dataframe which should be read from the standardised table that is populated at the end of the source standardisation notebook.

It is important that this be read from a table (vs pulling direct from the standardised dataframe) as SSAA has not got caching activated, and the standardisation should ideally be performed once, not repeatedly any time the \_std object is referenced.

For the optical matching the below code acquires the master and source dataframes:

bop\_client\_std = spark.table('cme\_fuzzy.bop\_client\_std')

hi\_membership\_std = spark.table('cme\_fuzzy.hi\_membership\_std')

In actuality this is performed within the CME Matching Infrastructure notebook, but here we are listing it separately owing to its significance in the process…

## Rule Execution

### Rule Object Setup

The first step in the rule execution is the instantiation of rule objects for each rule to be run against the source. Here is the code that sets these up:

# Setup the rule objects that we will be using to execute against each source

# Each of these rules may be tuned using the below keyword arguments during declaration:

#   unmatched=True (include unmatched records - this is default),

#   persist=False (write the results to the match\_target dataframe),

#   minimum\_confidence=0.5 (only save matches to match\_target if > 50% confidence )

# Deterministic Rules

dcn\_rule = DeterministicContactNameRule()

dfn\_rule = DeterministicFundNumberNameRule()

# Fuzzy Rules

fcn\_rule = FuzzyContactNameRule()

fdbn\_rule = FuzzyDOBNameRule()

fnd\_rule = FuzzyNameDetailsRule()

flnd\_rule = FuzzyLastNameDetailsRule()

### Running the Rules

Rules are executed sequentially, with the unmatched output of the prior rule being used as the source input to the next rule. The first rule of course runs against the entire standadised source dataframe . The sequence the rules should be run is captured in the Rule table by the “Weighting Order” column.

Sequentially running the rules ensures that items which have already been matched to a high confidence (100% for deterministic rules and the value set in the CommonSetup match\_confidence\_threshold for the fuzzy rules) are not repeatedly matched when they are already qualified.

This process can be seen in the below diagram:

As the rules are run the rule class captures the output to the match\_target table automatically. The intermediate rule result dataframes are simply outer joined to the source standardised table to get the input records for the subsequent step (see the “Get Unmatched Records” process in the above flowchart).

## Persisting Match Results

The final fuzzy match operation (FuzzyDOBNameRule) provides many match candidates that are worthy of further investigation. To facilitate this the FuzzyCalculator class is used. This class is defined in the Matching Infrastructure notebook, but is used to generate a dataframe that has a number of fuzzy match scores for various attributes or combinations thereof. These scores are useful when identifying clusters of records that can be used to tune which matches are considered valid.

The FuzzyCalculator provides a dataframe with the following structure:

1. Source\_Key:integer
2. Target\_Key:integer
3. DATE\_OF\_BIRTH\_CONF:integer
4. S\_DATE\_OF\_BIRTH:date
5. M\_DATE\_OF\_BIRTH:date
6. NAME\_GENDER\_CONF:integer
7. S\_FIRST\_NAME:string
8. M\_FIRST\_NAME:string
9. S\_LAST\_NAME:string
10. M\_LAST\_NAME:string
11. S\_GENDER:string
12. M\_GENDER:string
13. NAME\_CONF:integer
14. S\_FULL\_NAME:string
15. M\_FULL\_NAME:string
16. ADDRESS\_CONF:integer
17. S\_ADDRESS:string
18. M\_ADDRESS:string
19. EMAIL\_CONF:integer
20. S\_EMAIL\_ADDRESS:string
21. M\_EMAIL\_ADDRESS:string
22. MOBILE\_CONF:integer
23. S\_MOBILE\_NUMBER:string
24. M\_MOBILE\_NUMBER:string
25. FUND\_NBR\_CONF:integer
26. S\_HEALTH\_FUND\_NUMBER:string
27. M\_HEALTH\_FUND\_NUMBER:integer

This provides visibility of the following fuzzy match confidence figures and the attributes of the master and the source side by side to make it easier to visually confirm matches:

* Date of Birth Fuzzy Match Confidence
* First Name, Last Name, Gender Fuzzy Match Confidence
* First Name, Last Name Fuzzy Match Confidence
* Address Fuzzy Match Confidence
* Email Fuzzy Match Confidence
* Mobile Fuzzy Match Confidence
* Health Fund number Fuzzy Match Confidence

This data is written to the table cme\_fuzzy.<source\_name>\_fuzzy\_confidence and can be readily queried using sql queries such as:

SELECT

Source\_Key,

Target\_Key,

(

DOB\_SCORE +

NAME\_SCORE +

ADDRESS\_SCORE +

EMAIL\_SCORE +

MOBILE\_SCORE +

FUND\_NBR\_SCORE

) AS TOTAL\_SCORE,

DATE\_OF\_BIRTH\_CONF,

DOB\_SCORE,

NAME\_GENDER\_CONF,

NAME\_CONF,

NAME\_SCORE,

ADDRESS\_CONF,

ADDRESS\_SCORE,

EMAIL\_CONF,

EMAIL\_SCORE,

MOBILE\_CONF,

MOBILE\_SCORE,

FUND\_NBR\_CONF,

FUND\_NBR\_SCORE,

S\_DATE\_OF\_BIRTH,

M\_DATE\_OF\_BIRTH,

S\_FIRST\_NAME,

M\_FIRST\_NAME,

S\_LAST\_NAME,

M\_LAST\_NAME,

S\_GENDER,

M\_GENDER,

S\_FULL\_NAME,

M\_FULL\_NAME,

S\_ADDRESS,

M\_ADDRESS,

S\_EMAIL\_ADDRESS,

M\_EMAIL\_ADDRESS,

S\_MOBILE\_NUMBER,

M\_MOBILE\_NUMBER,

S\_HEALTH\_FUND\_NUMBER,

M\_HEALTH\_FUND\_NUMBER

FROM

(

SELECT

CASE WHEN DATE\_OF\_BIRTH\_CONF = 100 THEN 200 ELSE 0 END as DOB\_SCORE

,CASE WHEN FUND\_NBR\_CONF = 100 THEN 200 ELSE 0 END as FUND\_NBR\_SCORE

,CASE WHEN NAME\_CONF > 30 THEN (NAME\_CONF \* 2) END as NAME\_SCORE

,CASE WHEN EMAIL\_CONF = 100 THEN 50 ELSE 0 END as EMAIL\_SCORE

,CASE WHEN MOBILE\_CONF = 100 THEN 50 ELSE 0 END as MOBILE\_SCORE

,CASE WHEN ADDRESS\_CONF > 80 THEN 100 + ADDRESS\_CONF ELSE 0 END

as ADDRESS\_SCORE

,FCP.\*

FROM

cme

In the query above we see an aggregation of scores across the various fuzzy matches that can be used to find the best match candidate for an incoming source record.

CME BDP4 Migration

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Jan 23, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3195602002&selectedPageVersions=1&selectedPageVersions=2)

1 min read5 people viewed

At current CME is implemented in the SSAA environment. This can be regarded as a prototype only, as it uses SSAA (which is not hardened as one would expect of a production matching capability), and the data used was only authorised for prototyping purposes by the data owners.

In order for the CME capability to be exposed to end users, some or all of the functionality and structures needs to be migrated to the BDP4 platform.

This may involve either:

1. Migrating the entire processing and data to BDP4  
   OR
2. Migrating only the results obtained on SSAA to BDP4

This page will capture the decision taken and how it is to be implemented. Please subscribe to this page if you wish to be informed of updates to the decision and approach.

Please see the [BDP4 Rehosted CME Solution Design](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286368348) page for current details of the design of this migration.



Add label

CME POC Synthetic Data

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Nov 22, 2023

1 min read8 people viewed

The MDM Tool selection process includes a short engagement with a limited selection of tools to understand the strengths and weaknesses of their customer matching capabilities.

This engagement comprises giving vendors a selection of data for two sources and a customer master for them to perform matching against. The initial specification of this data can be found here: [MDM Customer Data Synthesis Approach](https://bupateam.atlassian.net/wiki/spaces/CSOCDA/pages/3169878606)

The mechanism and assets used to generate this data will be documented here so that it may be reused as needed.



Add label

# CME Matching Approach

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Dec 19, 2023](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3199992229&selectedPageVersions=1&selectedPageVersions=2) by [Hemanth Parasaram](https://bupateam.atlassian.net/wiki/people/5c623fd2cefe97640e699a8b?ref=confluence&src=profilecard)

3 min read3 people viewed

The matching approach is as follows:

1. Standardise Data
2. Run Rules
3. Select best match for each source/target record
4. Assemble consolidated master joining master to each of the sources

This is implemented in DataBricks notebooks as per the below flow:

The notebooks and their function is described in the below table:

|  |  |  |
| --- | --- | --- |
| **Notebook** | **Function** | **Output** |
| CME Master Standardisation | Standardises HI\_MEMBER data as required by matching process | cme\_fuzzy.hi\_member\_std |
| CME Optical Standardisation | Standardises BOP\_CLIENT data as required by matching process | cme\_fuzzy.bop\_client\_std |
| CME Dental Standardisation | Standardises DC\_PATIENT data as required by matching process | cme.fuzzy.dc\_patient\_std cme.fuzzy.dc\_patient\_txn |
| CME Optical Matching | Performs matching of optical against the member master | cme\_fuzzy: bop\_match\_target bop\_fuzzy\_confidence bop\_match |
| CME Dental Matching | Performs matching of dental customer against the member master | cme\_fuzzy: dc\_match\_target dc\_fuzzy\_confidence dc\_match |
| CME Optical Reporting | Reports results of matching rules for optical data |  |
| CME Dental Reporting | Reports results of matching rules for dental data |  |
| CME Consolidated Master | Takes matches and creates combined master | cme\_fuzzy.hi\_master\_matched |

# Standardisation of Data

Each source has its own standardisation notebook that effects the standardisations.

## Standardisations Applied

Details of the standardisations are discussed in the sections below:

### Field Name Standardisation

Rules require a set of fields to have standardised names/types so they can be run across all data sets.

Current fields and their types that require standardisation are shown in the below table:

|  |  |  |
| --- | --- | --- |
| **Description** | **Standardised Name** | **Standardised Type** |
| First Name | FIRST\_NAME |  |
| Last Name | LAST\_NAME |  |
| Gender | GENDER | CHAR(1): M or F or Null  This is the standard in the HI\_Membership master source |
| Health Fund Number | HEALTH\_FUND\_NUMBER |  |
| Email Addresss | EMAIL\_ADDRESS |  |
| Mobile Number | MOBILE\_NUMBER |  |
| Source Key (Natural key or composite for the customer data source) | SOURCE\_NATIVE\_KEY |  |
| Date of Birth | DATE\_OF\_BIRTH | Date |
| Address | ADDRESS | String containing full address in format it comes from HI Master  for example “Flat 75, 73 Russell Street, HAYWARDS BAY, NSW 2530, AU” |

Objects containing the standardised fields are suffixed by \_STD to denote this.

### Domain Standardisation

As the master information is originally sourced from the HI\_Membership object, this defines the form (domain) that standardised attributes are to take. For example, HI\_Membership has only three gender values - M, F and Null. Accordingly all sources being standardised should have their Gender field populated with one of these three values.

## Source Internal Duplicates

Sources may not provide a unique customer list, since the data is often sourced from transactional systems that re-enter patient data each time they present. This is the case for both the dental and optical sources used in the CME SSAA processing.

Where multiple records in a source refer to the same entity (customer) it is highly desirable that this data be matched internally to the source such that only one customer record from that source is matched against the master.

Sources that contain multiple records which match a single master record (as would be expected if there are multiple records in a source for a single entity) are problematic, since they introduce product joins to any attempt to create a consolidated master record, containing the master key and the keys from each source as separate columns.

For example, If dental has 3 records that match a master record (because a single customer attended dental 3 times) and optical has 4 records that match (single customer 4 presentations) then we would see 12 records in a consolidate master.

There are numerous instances of this circumstance captured in the [Duplicate Match Investigation](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/1915044030454767/command/1915044030454768) notebook. In the [CME Consolidated Master](https://adb-6334213679475485.5.azuredatabricks.net/?o=6334213679475485#notebook/3873140696955074/command/1915044030454779) notebook the consolidated master is rendered into a customer cohort that accounts for this, with a columns specifying if a member is a dental customer and/or an optical customer., but not the actual records.

# Rule Execution

Rules are executed sequentially, with the unmatched records for one rule used as input to the next rule.

## Clustering

Matching cannot be applied scalably between every record and every other record as this results in a quadratic explosion of the match candidates considered. This leads to performance and resource consumption issues with all but the smallest data volumes. As we are considering in excess of 4M records, we must first cluster the groups of entites that are matched.

In the table above cluster is used to pre-qualify records to match prior to obtaining any fuzzy matching score. Different rules have different clusters applied to them as required.

## Defined Rules

The rules and the sequence they are run are shown below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Rule ID** | **Description** | | **Cluster** | **Confidence Algo** | | **Source/Targets** | |
| **Rule ID** | | **Description** | | **Cluster** | **Confidence Algo** | | **Source/Targets** |
| Deterministic-Contact-Name | | Match of Client First, Last, Gender with matching Email or Mobile | | email or mobile identical | 100% Hardcoded (i.e. 1.0) | | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Deterministic-FundNumber-Name | | Match of Client First, Last, Gender with matching Health Fund | | Health Fund Number Identical | 100% Hardcoded (i.e. 1.0) | | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Fuzzy-Contact-Name | | Fuzzy Match on Concatenated First, Last, Gender, Email, Mobile with matching email or mobile | | email or mobile identical | Fuzzy Match on Concatenated First, Last, Gender, Email, Mobile | | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Fuzzy-DOB-Name | | Fuzzy match on Concatenated First/Last/Gender with matching DOB. This is used as the base calculator for Name. Address, Mobile, Fund Number etc. fuzzy match calculations used in compound rules. | | Date of Birth Identical | Fuzzy Match on Concatenated First, Last, Gender | | * Dental Patient - HI Membership * Optical Client - HI Membership |
| Combined-Weighted | | Final weighted combination of other rules | | N/A | Subject to tuning | | * All matches present |



BDP4 Rehosted CME Solution Design

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 21, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3286368348&selectedPageVersions=2&selectedPageVersions=3)

1 min read8 people viewed

This page contains the design for the rehosting of the CME to BDP4 infrastructure using strategic data sources available on that platform. This rehosted implementation takes the form of a coexistence hub MDM pattern.

The Coexistence hub pattern has the following features:

* The hub maintains the golden customer record including contact information and identifiers
* It is fed by the various sources of customer data present within the organisation
* Data sources maintain their own customer information, but may elect to consume golden record data from the CME via periodic feeds, however this is not mandatory.

A diagram depicting the relationships between the sources and the MDM system is shown below:

As this is a coexistence hub design, no changes should be required to the source systems.

Design Sections

Individual sections of the design have their own pages. Please refer to information via the below links:

* [CME Source Onboarding Procedure](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286171806/CME+Source+Onboarding+Procedure)
* [CME Data Models](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286433910/CME+Data+Models)
* [CME Processing Engine Requirements](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286630564/CME+Processing+Engine+Requirements)
* [CME Development Environment](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286663521/CME+Development+Environment)
* [CME Security](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286499612/CME+Security)
* [CME Testing Approach](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286630829/CME+Testing+Approach)
* [CME Data Sourcing](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3289514111/CME+Data+Sourcing)
* [CME Source Data Validation](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3289613610/CME+Source+Data+Validation)
* [CME Data Publication](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3289974702/CME+Data+Publication)
* [CME Match/Split Interface](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3310780893)

CME Source Onboarding Procedure

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 21, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3286171806&selectedPageVersions=11&selectedPageVersions=12)

1 min read6 people viewed

The procedure for onboarding a source is defined on this page.

Key steps in the onboarding process are:

1. [Source Data Identification](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247)
2. [Source Data Profiling](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3312648986)
3. [Source Data Registration & Standardisation](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3321595280)
4. [Data Quality Evaluation](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3289613610)
5. [Test Data Generation](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3321038625)
6. [Privacy/Preference Tagging](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3334406241)
7. [Rule configuration](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3183116473)
8. [Testing](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286630829)
9. [Publishing](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3289974702)

# CME Source Data Identification

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 25, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3313009247&selectedPageVersions=2&selectedPageVersions=3)

1 min read2 people viewed

Source data identification is the first step in the data onboarding process for the customer matching engine.

This step considering the data made available by a source system within the BDP4 platform (typically from the bronze layer) and identifying a data set that contains the required attributes.

Required and optional attributes are shown in the table below:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Required/Optional** |
| First Name | Customer Given Name | Required |
| Last Name | Customer Surname | Required |
| Full Name | Customer Full Name ( First, Last and potentially initial in a single field) | Required if First/Last name are not available separately |
| Date Of Birth | Customer date of birth | Optional (highly desirable) |
| Health Fund / Member Number | Customer member number as well as health fund identifier | Optional (highly desirable) |

In addition to the above attributes, two or more contact related attributes (or groups of attributes in the case of address) must also be available. These contact attributes include:

|  |  |  |
| --- | --- | --- |
| **Contact Attribute** | **Description** | **Comments** |
| Email Contact |  | Ideally should include usage (e.g. Personal/Work) |
| Mobile Number Contact |  | Typically used as primary phone contact, but may also be tagged with usage (e.g. Work) |
| Phone Number |  | Should include usage (e.g. Home/Work) |
| Address |  | Ideally should include usage (e.g. Residential/Postal/Business) |

If data is not available within the BDP4 environment it must first be ingested. Depending on business priority of merge/matching particular datasets the onboarding of the source to CME may also include this ingestion work, however this will not be the approach taken when executing BDP4 integration PI.



Add label

**Related pages**

[CME Data Models](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286433910/CME+Data+Models)

**CME Data Models**

[Bupa Data Program](https://bupateam.atlassian.net/wiki/spaces/HDP)

Often read together

[Golden Customer Record Model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3302686878/Golden+Customer+Record+Model)

**Golden Customer Record Model**

[Bupa Data Program](https://bupateam.atlassian.net/wiki/spaces/HDP)

Often read together

[C360 - Build MVP Customer Matching Process & Core Customer Profile](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3074555958)

**C360 - Build MVP Customer Matching Process & Core Customer Profile**

[Bupa Data Program](https://bupateam.atlassian.net/wiki/spaces/HDP)

Organised together

:thumbsup:

:clap:

:tada:

Be the first to add a reaction

## 2 page comments

### [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence)

### **Feb 12, 2024**

As per showcase discussion with Daniel Hall, if data present in development environment is not representative (in terms of completeness or quality) of the data set to be ingested, quickly move the standardisation to pre-prod to assess match feasibility is an important capability.

* [Reply](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247/CME+Source+Data+Identification)
* [Edit](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247/CME+Source+Data+Identification)
* [Delete](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247/CME+Source+Data+Identification)

### [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence)

### **Feb 12, 2024**

Key input to the initial selection of the sources is the ingestion master list which is accessible via the below link:

[Ingestion Master List V1.0.xlsx](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/Shared%20Documents/Data%20Ingestion%20Squad/Ingestion%20Master%20List%20V1.0.xlsx?d=wdd12b04c6ff54435bf1c39c919cdd188&csf=1&web=1&e=bJEdn1)

And HUGO source column information is available here: [Hugo Table Tracker v3.xlsx](https://bupaau.sharepoint.com/:x:/r/sites/HealthInsuranceBuildWorkingGroupBupaandEY-DataIngestionSquad-Private/Shared%20Documents/Data%20Ingestion%20Squad%20-%20Private/Hugo/Hugo%20Table%20Tracker%20v3.xlsx?d=w40e2aa3de68d47db932151fb64f6f85a&csf=1&web=1&e=HGBZTb)

* [Reply](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247/CME+Source+Data+Identification)
* [Edit](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247/CME+Source+Data+Identification)
* [Delete](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247/CME+Source+Data+Identification)

# CME Source Data Profiling

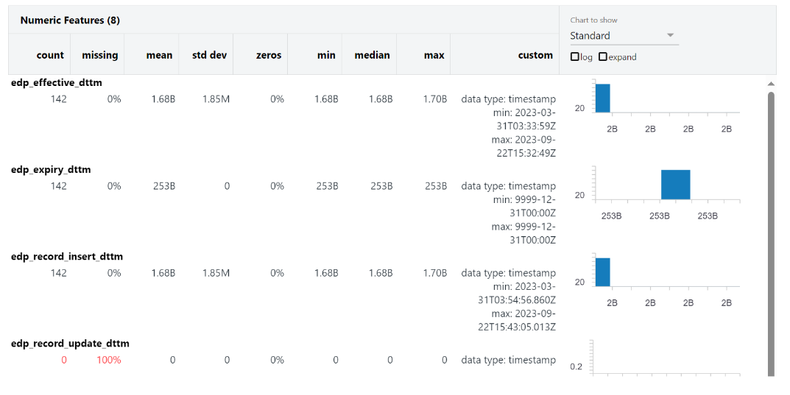
Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Mar 01, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3312648986&selectedPageVersions=5&selectedPageVersions=6)

4 min read3 people viewed

## Profiling Approach

Profiling is performed using Databricks to source the data into a notebook environment which is then displayed with a profile view. The profile view shows the columns present within the dataset and the number of nulls and frequency distribution (depending on data type).



If the data set is very large, then the retrieval operation within the notebook may include a limit to reduce the number of records that are profiled.

## Unmasked Production Data is what is profiled

Source data profiling is undertaken against umasked production data so that the feasibility of matching against that set is known prior to performing standardisation and matching configuration activities. This production data is sourced into the pre-production environment via a service request.

Assuming the objects to be profiled are present within the development environment this profiling notebook can be developed and unit tested such that it can be deployed to pre-production and executed by operations staff. In this way providing access to production data to the developer of the standardisation and matching code is not necessary.

## Performing Profiling

Profiling is performed by preparing a profiling notebook as per the below described profiling template. The notebook is supplied to production operations staff who then can run it and capture results for return to the developer.

### Profiling Notebook Template

A template that can be used to perform the profiling is shown below:

# Databricks notebook source

# MAGIC %md

# MAGIC

# MAGIC # Initial Profiling Notebook - Genesys

# MAGIC

# MAGIC This notebook is intended to be run in a production setting which has access to the unmasked data that will ultimately be onboarded to the customer matching engine (CME). It should be run under a user context that is able to see the data absent any masking.

# COMMAND ----------

# Paramters - these should be localised to the production environment

SOURCE\_CATALOG = "bdp4\_dev\_lh"

SOURCE\_SCHEMA = "brz\_genesys"

SOURCE\_OBJECT = "user\_data\_cust\_dim\_1"

print(f"Profile will be performed of {SOURCE\_CATALOG}.{SOURCE\_SCHEMA}.{SOURCE\_OBJECT}")

# COMMAND ----------

# Retrieve the data into a dataframe

source\_df = spark.sql(f"SELECT \* FROM {SOURCE\_CATALOG}.{SOURCE\_SCHEMA}.{SOURCE\_OBJECT};")

# COMMAND ----------

# Display the summarised profile

print(f"PROFILE FOR {SOURCE\_CATALOG}.{SOURCE\_SCHEMA}.{SOURCE\_OBJECT}:")

dbutils.data.summarize(source\_df)

This notebook when run will execute a profiling operation using the dbutils.data.summarize feature that is used when displaying a dataframe profile using the UI. The advantage here is that the raw data is not exposed, in the output, just the data profile.

To use this template you must customise it to the object you wish to profile by substituting in lines 11-13:

* The production catalog into SOURCE\_CATALOG
* Object schema in SOURCE\_SCHEMA
* Object name in SOURCE\_OBJECT

When multiple objects need to be profiled you can use the below template to indicate a list of objects that will be included in the SOURCE\_OBJECTS (rather than a single object name in SOURCE\_OBJECT). The included example lists 8 separate objects in the SOURCE\_OBJECTS and each will be separately profiled.

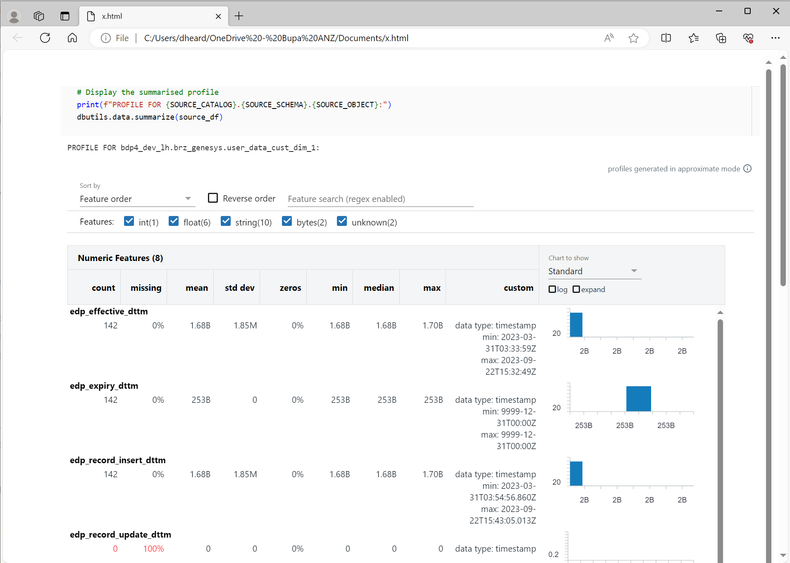
Multi-Object Profiling Notebook Template Example

### Capturing Profile Output

After the notebook is run the output is captured by:

1. Selecting “Export Cell” from the cell menu
2. Copying the HTML that is displayed in the pop-up box into a file with the extension .html.
3. At the top of the html file add the text “<html><body>”
4. At the bottom of the html file add the text </body></html>”

The HTML file may then be supplied to the developer. When the html file is opened the profile displayed should look like this:



The developer can then scroll through the results viewing the profiles for each attribute.

# CME Source Data Registration & Standardisation

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 28, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3321595280&selectedPageVersions=1&selectedPageVersions=2)

2 min read1 person viewed

## Metadata Registration

Data being onboarded to the CME needs to be registered within the metadata tables. This popualtes the following information:

* Source System name
* Rules to be applied to source

### Source Registration

Each source has a directory in under the matching folder in the CME repository. The folder name matches the “source\_name” attribute in the MATCH\_SRC table. Within the source directory is a notebook register\_metadata which performs the registration. The registration is done within %sql steps that contain insert statements.

To register the dental source the following sql statement would be executed:

INSERT INTO ${cme.bronze\_schema}.MATCH\_SRC (source\_id, source\_name, source\_description)

VALUES ('dental', 'dental\_transaction', 'Transaction file for dental presentations');

### Rule Registration

Rule registration is not done at this point, since data is not available for tuning. As each rule is enabled the insert statements for the SRC\_MATCH\_RULE table are included.

## Data Standardisation

Standardisation must be applied to data being matched by the CME so that the rules that are being applied may be reused across different sources. This minimises the amount of work required to integrate a new source into the engine.

The modelling approach used for the standard customer data can be found on the [CME Contact Information Modelling](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3294429606) page.

### Standardisation Notebook

Each source has its own standardisation notebook that effects the standardisations. This notebook takes the data from the source, standardises it and persists it to the BRZ\_CME\_<Source\_Name> schema. The name of the standardised source table is <source\_name>\_std.

### Structural Standardisation

Data that has been standardised is conformed to a model that mirrors the golden record. This model includes the following entities:

* <source\_name>\_CUST
* <source\_name>\_CUST\_ATTR
* <source\_name>\_CUST\_CONTACT
* <source\_name>\_CUST\_CONTACT\_ADDRESS

Please see the [CME Contact Information Model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3294429606) page for further details of this structure.

In the process of structural standardisation a key is generated for each incoming record. This is the primary key of the <source\_name>\_CUST table and a foreign key in the other contact and attribute tables.

The natural key (or keys if the source has a composite key) is also stored within the <source\_name>\_CUST table to allow linking CME standardised data (and therefore the golden record data) back to the source.

### Field Name Standardisation

Rules require a set of fields to have standardised names/types so they can be run across all data sets. When performing the standardisation the attributes present in the source object are mapped to equivalent attributes within the appropriate objects.

CME Test Data Generation

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Feb 14, 2024

1 min read1 person viewed

CME requires the ability to perform testing of the system, rules and publication logic without access to real data sets.

Please refer to the [CME Testing Approach](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286630829) page for details of the test data generation.

CME Privacy/Preference Tagging

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Feb 21, 2024

1 min read1 person viewed

As data is acquired into the CME standardised source area it is tagged for privacy and preferences. The rules used to source the privacy/preference classification of the data are source-dependent and are built into the standardisation notebook.

The tagging takes the form of a tag identifier that is put on each source record. Where specific attributes are tagged differently, the tags are applied to the attributes table, the address table or the contact tables as required. For details of these structures please refer to the [Golden Customer Record Model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3302686878)

# CME Data Models

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 22, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3286433910&selectedPageVersions=22&selectedPageVersions=23)

8 min read9 people viewed

This page summarises the data models in use by CME, and also provides a starting point for understanding CME data management approach. Other pages that contain additional information are:

* [Golden Customer Record Model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3302686878/Golden+Customer+Record+Model)
* [CME Contact Information Modelling](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3294429606/CME+Contact+Information+Modelling)
* [CME Data Classification](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3336372324/CME+Data+Classification)
* [CME Data Storage Approach](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3317826174/CME+Data+Storage+Approach)

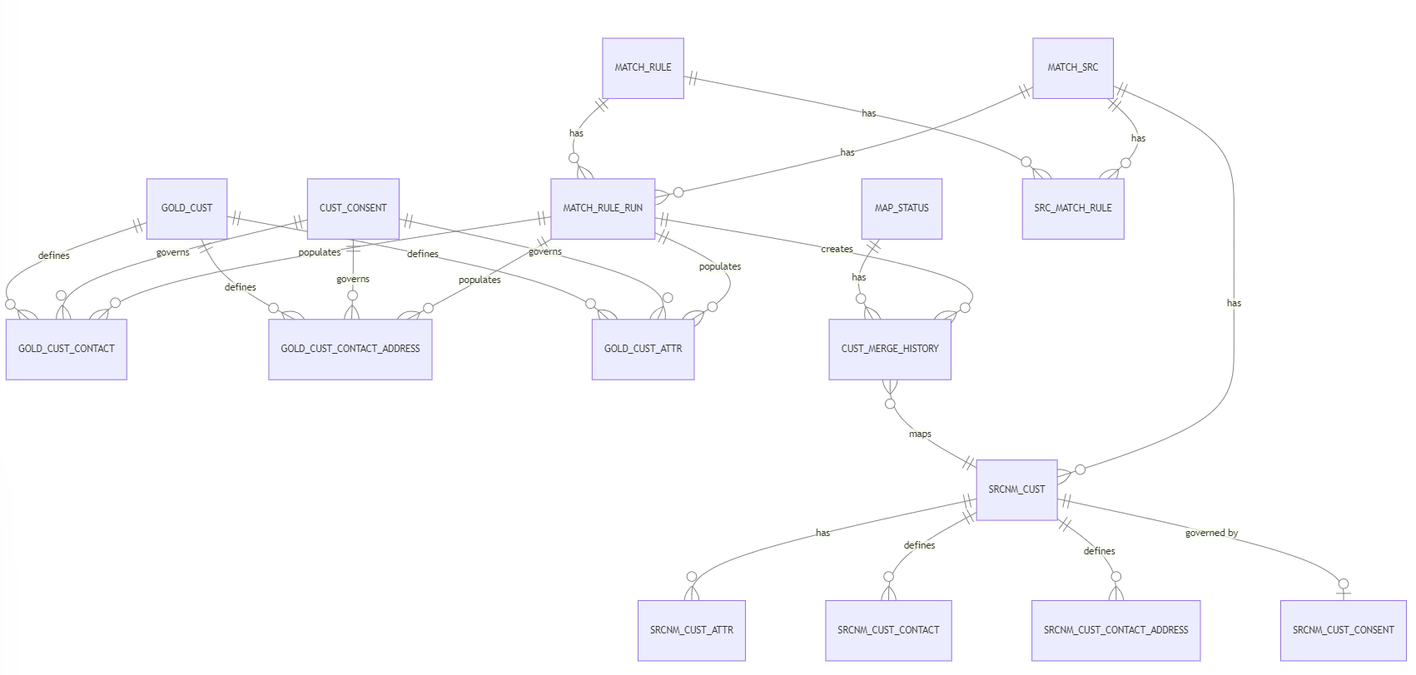
## Data used by CME

Data that is used by CME comprises the following:

* Source data - Data sources that are to be used to match against the golden record
* Golden record data - Data that contains the golden record and mappings of the golden records back to the sources from which they were drawn
* Intermediate data sets - Ephemeral and longer lived data that was used in the process of performing the matches
* Matching metadata - Data that describes the sources, rules and runs of the matching process

This page describes the models for all but the source data, as the source data structure varies according to the way the data is provided.

Here is an entity level diagram of the model as it stands:



For attribute level diagrams please refer to the individual discussions of each area

Code to generate entity level model

erDiagram

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

CUST\_CONSENT ||--o{ GOLD\_CUST\_CONTACT : governs

CUST\_CONSENT ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : governs

CUST\_CONSENT ||--o{ GOLD\_CUST\_ATTR : governs

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_CONTACT : populates

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : populates

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_ATTR : populates

CUST\_MERGE\_HISTORY }o--|| SRCNM\_CUST : maps

SRCNM\_CUST ||--o{ SRCNM\_CUST\_ATTR : has

SRCNM\_CUST ||--o{ SRCNM\_CUST\_CONTACT : defines

SRCNM\_CUST ||--o{ SRCNM\_CUST\_CONTACT\_ADDRESS : defines

MAP\_STATUS ||--o{ CUST\_MERGE\_HISTORY : has

SRCNM\_CUST ||--o| SRCNM\_CUST\_CONSENT : "governed by"

MATCH\_RULE\_RUN ||--o{ CUST\_MERGE\_HISTORY : creates

MATCH\_RULE ||--o{ MATCH\_RULE\_RUN : has

MATCH\_RULE ||--o{ SRC\_MATCH\_RULE : has

MATCH\_SRC ||--o{ MATCH\_RULE\_RUN : has

MATCH\_SRC ||--o{ SRCNM\_CUST : has

MATCH\_SRC ||--o{ SRC\_MATCH\_RULE : has

## Golden Record Data

The golden record data comprises the following major items:

* Golden record primary table
* Golden record attributes
* Golden record contacts
* Golden record addresses
* Golden record to matched source record mapping

Further detail is available on the [Golden Customer Record Model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3302686878) page.

## Intermediate Data Sets

Intermediate data sets include the following:

* Ephemeral working data sets - for example:
  + Match candidates
  + Candidate Inclusion / Exclusion scoring drivers
* Standardised version of source data that allows matching engine to execute rules without customisation

Ephemeral working data sets will be documented within the matching engine design page and will not be discussed further here.

### Standardised Source Data

Standardised source data includes several entities for each source. These are:

* Core source data
  + ID within source
  + Effectivity information (drawn directly from the standardised Bronze (TBC) layer source table)
  + Core attributes (Name, DOB)
* Source record attributes
  + Contains the attributes associated with the source records
  + Is stored in attribute name/value type of structure to facilitate maximum flexibility
  + Includes a reference to the source id and effectivity
* Source privacy/permission data
  + Defines privacy/permission characteristics pertaining to the source record
  + This is used to determine allowability of publishing and matching based on source or user preferences.

#### Physicalisation and the use of views

Given that the source data is expected to be drawn from the bronze layer, these standardised tables may actually not be persisted, but take the form of views against the bronze tables. Depending on performance experienced (and requirements relating to the ability to audit match sources) these objects may need to be physicalised to tables.

#### Naming Standard

The source data tables have a prefix that indicates the type of the data that is stored within them. The source core table has a suffix of SRC\_CORE, the attribute table has a suffix of SRC\_ATTR and the privacy/permissions table SRC\_PRV.

There is also a prefix that indicates the source. Each source is registered in the match metadata table along with a code for that source. All source specific tables are prefixed with that code.

For example, if the dental transaction file has a code of DENTALTXN then the core and attribute tables would be DENTALTXN\_SRC\_CORE and DENTALTXN\_SRC\_ATTR.

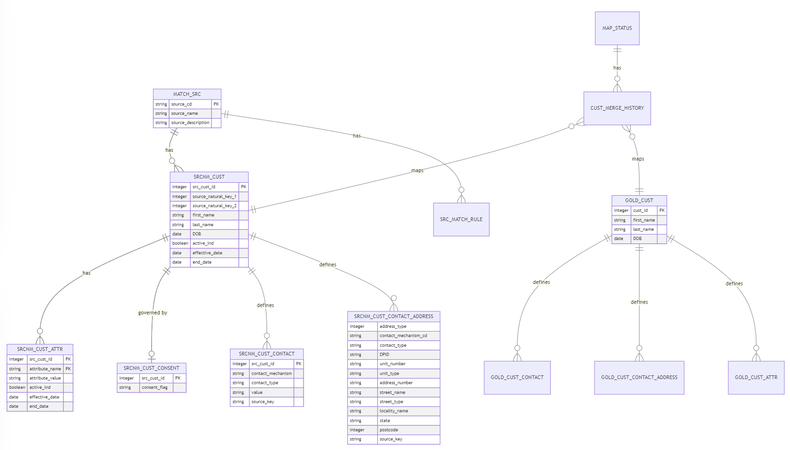
### Storage of Source Natural Keys

All data within the standardised version of a source must be able to be associated back to the original source record. This is done by providing the source’s natural key within the core standardised table.

In the below example SRCNM\_CUST contains the natural key

#### Data Model Diagram

The standardised source data model is shown below:



Code used to generate diagram

erDiagram

GOLD\_CUST {

integer cust\_id PK

string first\_name

string last\_name

date DOB

}

SRCNM\_CUST {

integer src\_cust\_id PK

integer source\_natural\_key\_1

integer source\_natural\_key\_2

string first\_name

string last\_name

date DOB

boolean active\_ind

date effective\_date

date end\_date

}

SRCNM\_CUST\_ATTR {

integer src\_cust\_id PK

string attribute\_name PK

string attribute\_value

boolean active\_ind

date effective\_date

date end\_date

}

SRCNM\_CUST\_CONSENT {

integer src\_cust\_id PK

string consent\_flag

}

SRCNM\_CUST\_CONTACT {

integer src\_cust\_id PK

string contact\_mechanism

string contact\_type

string value

string source\_key

}

SRCNM\_CUST\_CONTACT\_ADDRESS {

integer address\_type

string contact\_mechanism\_cd

string contact\_type

string DPID

string unit\_number

string unit\_type

string address\_number

string street\_name

string street\_type

string locality\_name

string state

integer postcode

string source\_key

}

SRCNM\_CUST ||--o{ SRCNM\_CUST\_ATTR : has

SRCNM\_CUST ||--o| SRCNM\_CUST\_CONSENT : "governed by"

MATCH\_SRC {

string source\_cd PK

string source\_name

string source\_description

}

MATCH\_SRC ||--o{ SRCNM\_CUST : has

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

CUST\_MERGE\_HISTORY }o--|| SRCNM\_CUST : maps

CUST\_MERGE\_HISTORY }o--|| GOLD\_CUST : maps

SRCNM\_CUST ||--o{ SRCNM\_CUST\_ATTR : has

SRCNM\_CUST ||--o{ SRCNM\_CUST\_CONTACT : defines

SRCNM\_CUST ||--o{ SRCNM\_CUST\_CONTACT\_ADDRESS : defines

MAP\_STATUS ||--o{ CUST\_MERGE\_HISTORY : has

SRCNM\_CUST ||--o| SRCNM\_CUST\_CONSENT : "governed by"

MATCH\_SRC ||--o{ SRCNM\_CUST : has

MATCH\_SRC ||--o{ SRC\_MATCH\_RULE : has

## CME Matching Metadata

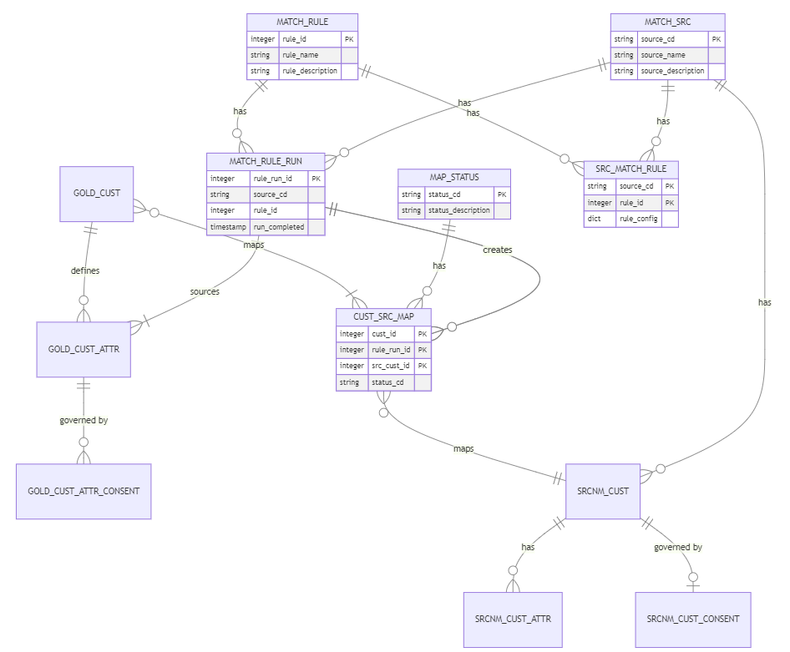
The CME matching metadata describes the sources, rules and runs of the matching process. This permits users of the golden record data to understand the sources for matches and how the matching process is performed. This also allows detailed audit of the matching process.

The main entities within the metamodel include:

* Source register
* Rule register
* Rule Run operational metadata
* Survivorship rules
* Audit information for match creation and breaking

#### Data Model Diagram

The metamodel data model is shown below:



Code used to generate this diagram

erDiagram

    GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

    GOLD\_CUST\_ATTR ||--o{ GOLD\_CUST\_ATTR\_CONSENT : "governed by"

    MATCH\_RULE\_RUN ||--|{ GOLD\_CUST\_ATTR : sources

    GOLD\_CUST }o--|{ CUST\_SRC\_MAP : maps

    CUST\_SRC\_MAP }o--|| SRCNM\_CUST : maps

    SRCNM\_CUST ||--o{ SRCNM\_CUST\_ATTR : has

    MAP\_STATUS ||--o{ CUST\_SRC\_MAP : has

    SRCNM\_CUST ||--o| SRCNM\_CUST\_CONSENT : "governed by"

    MATCH\_SRC {

        string source\_cd PK

        string source\_name

        string source\_description

    }

    MATCH\_RULE {

        integer rule\_id PK

        string rule\_name

        string rule\_description

    }

    SRC\_MATCH\_RULE {

        string source\_cd PK

        integer rule\_id PK

        dict rule\_config

    }

    MATCH\_RULE\_RUN {

        integer rule\_run\_id PK

        string source\_cd

        integer rule\_id

        timestamp run\_completed

    }

    MAP\_STATUS {

        string status\_cd PK

        string status\_description

    }

    CUST\_SRC\_MAP {

        integer cust\_id PK

        integer rule\_run\_id PK

        integer src\_cust\_id PK

        string status\_cd

    }

    MATCH\_RULE\_RUN ||--o{ CUST\_SRC\_MAP : creates

    MATCH\_RULE\_RUN ||--o{ CUST\_SRC\_MAP : creates

    MATCH\_RULE ||--o{ MATCH\_RULE\_RUN : has

    MATCH\_RULE ||--o{ SRC\_MATCH\_RULE : has

    MATCH\_SRC ||--o{ MATCH\_RULE\_RUN : has

    MATCH\_SRC ||--o{ SRCNM\_CUST : has

    MATCH\_SRC ||--o{ SRC\_MATCH\_RULE : has

## Data Classification

Source data being used by CME to perform match/merge activities is expected to already be classified during the process of ingestion. Data in the CME source bronze layer objects (standardis Data within the CME golden model objects, the metadata objects and the source

# Golden Customer Record Model

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 25, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3302686878&selectedPageVersions=17&selectedPageVersions=18)

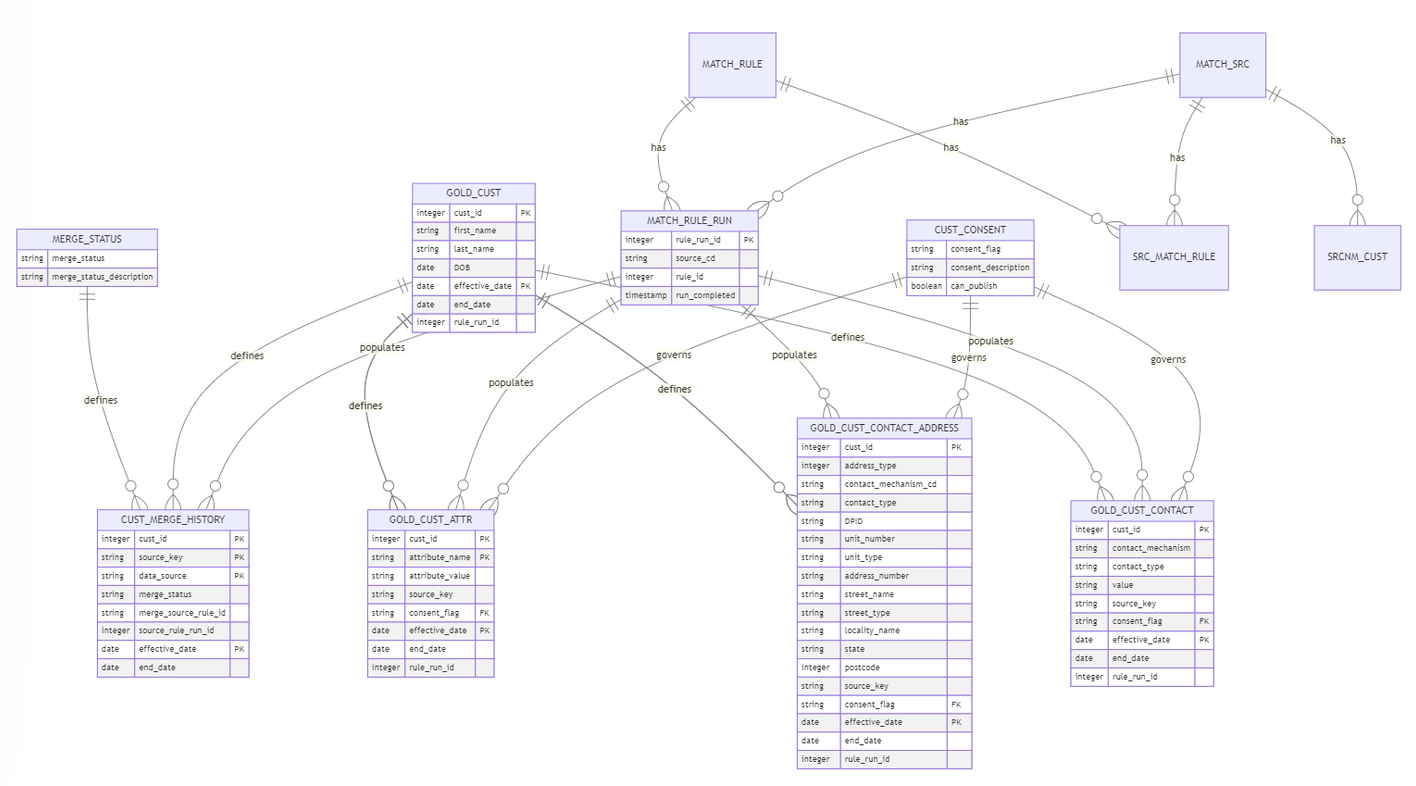
13 min read10 people viewed

The golden customer record model contains customer attributes that have been persisted from the sources into the central record via the matching/survivorship process.

The golden record data comprises the following major items:

* Golden record primary table
  + Contains the golden record id and core attributes such as name, DOB.
  + All other attributes are stored in the attributes or contact tables
* Golden record contact table
  + Contains the various contact information for a customer
  + Has a contact mechanism (e.g. mobile phone, email)
  + Has a contact type (e.g. home, work)
  + Stores the contact specifics within a “value” field
  + This is not used for address, since address is more usefully stored in multiple fields (street, city, postcode etc)
* Golden record contact address table
  + Contains the various addresses associated with a customer
  + Has an address type (e.g. home, work)
  + Addresses are associated with customers and are not standalone as one would when an address database of record is available/used.
  + Is decomposed into fields such as unit number, unit type, address number, street name, street type, locality name, state and postcode.
* Golden record attributes
  + Contains the attributes that have been associated with the golden record id via survivorship rules which are not contact data (since these will be in the Contact or Address entities)
  + Is stored in attribute name/value type of structure to facilitate maximum flexibility
  + Includes a reference to the source of the survived attribute
  + Some attributes only have use when considered together. An example of this would be address. These attribute clusters may have their own entities. See the [CME Contact Information Modelling](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3294429606) page for current design regarding this.
  + Attributes have a type - e.g. email or phone number.
* Golden record consent
  + Defines the consent tag that is applicable for an attribute or contact detail record
* Golden record to matched source record mapping
  + Defines the relationship between the golden record and the source record or records it has been associated with
  + Includes confidence of match and the rule used to qualify the match
  + Also can be used to explicitly indicate that a match is explicitly not valid so re-match of a candidate will not be permitted.
    - This is what is populated when users indicate a match should be split.
  + Contains a key that can be used to link to the operational metadata of the process that merged the source into it.

The overall golden record data model and how it relates to key match metadata tables is shown below. These will be discussed further in the sections that follow:



Code for this diagram

erDiagram

GOLD\_CUST {

integer cust\_id PK

string first\_name

string last\_name

date DOB

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST\_ATTR {

integer cust\_id PK

string attribute\_name PK

string attribute\_value

string source\_key

string consent\_flag FK

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST\_CONTACT {

integer cust\_id PK

string contact\_mechanism

string contact\_type

string value

string source\_key

string consent\_flag FK

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST\_CONTACT\_ADDRESS {

integer cust\_id PK

integer address\_type

string contact\_mechanism\_cd

string contact\_type

string DPID

string unit\_number

string unit\_type

string address\_number

string street\_name

string street\_type

string locality\_name

string state

integer postcode

string source\_key

string consent\_flag FK

date effective\_date PK

date end\_date

integer rule\_run\_id

}

CUST\_MERGE\_HISTORY {

integer cust\_id PK

string source\_key PK

string data\_source PK

string merge\_status

string merge\_source\_rule\_id

integer source\_rule\_run\_id

date effective\_date PK

date end\_date

}

MERGE\_STATUS {

string merge\_status

string merge\_status\_description

}

MERGE\_STATUS ||--o{ CUST\_MERGE\_HISTORY : defines

CUST\_CONSENT {

string consent\_flag

string consent\_description

boolean can\_publish

}

MATCH\_RULE\_RUN {

integer rule\_run\_id PK

string source\_cd

integer rule\_id

timestamp run\_completed

}

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

GOLD\_CUST ||--o{ CUST\_MERGE\_HISTORY : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_CONTACT : populates

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : populates

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_ATTR : populates

MATCH\_RULE\_RUN ||--o{ CUST\_MERGE\_HISTORY : populates

CUST\_CONSENT ||--o{ GOLD\_CUST\_CONTACT : governs

CUST\_CONSENT ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : governs

CUST\_CONSENT ||--o{ GOLD\_CUST\_ATTR : governs

MATCH\_RULE ||--o{ MATCH\_RULE\_RUN : has

MATCH\_RULE ||--o{ SRC\_MATCH\_RULE : has

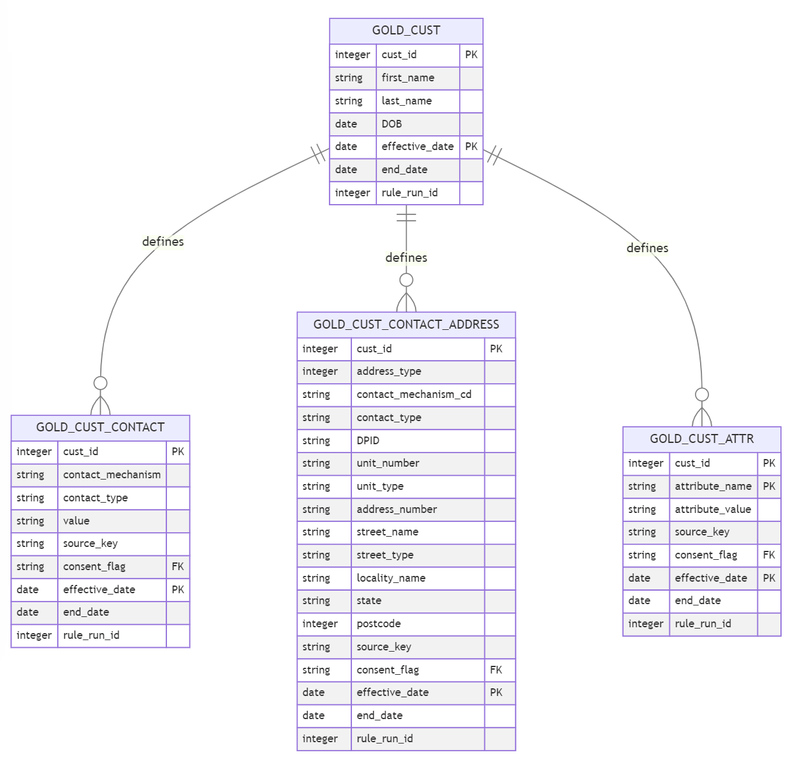
MATCH\_SRC ||--o{ MATCH\_RULE\_RUN : has

MATCH\_SRC ||--o{ SRCNM\_CUST : has

MATCH\_SRC ||--o{ SRC\_MATCH\_RULE : has

## Customer Contact and Address Detail Entities

See below for a diagram of the entities that store the core customer and contact details (note the source record mapping and consent relationship entities/relationships are not shown for clarity).



Code that generated above diagram

erDiagram

GOLD\_CUST {

integer cust\_id PK

string first\_name

string last\_name

date DOB

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST\_ATTR {

integer cust\_id PK

string attribute\_name PK

string attribute\_value

string source\_key

string consent\_flag FK

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST\_CONTACT {

integer cust\_id PK

string contact\_mechanism

string contact\_type

string value

string source\_key

string consent\_flag FK

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST\_CONTACT\_ADDRESS {

integer cust\_id PK

integer address\_type

string contact\_mechanism\_cd

string contact\_type

string DPID

string unit\_number

string unit\_type

string address\_number

string street\_name

string street\_type

string locality\_name

string state

integer postcode

string source\_key

string consent\_flag FK

date effective\_date PK

date end\_date

integer rule\_run\_id

}

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : defines

%% CUST\_CONSENT {

%% string consent\_flag PK

%% string consent\_descirption

%% }

GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

%% CUST\_CONSENT |o--o{ GOLD\_CUST\_ATTR : "governed by"

%% CUST\_CONSENT |o--o{ GOLD\_CUST\_CONTACT : "governed by"

%% CUST\_CONSENT |o--o{ GOLD\_CUST\_CONTACT\_ADDRESS : "governed by"

%% MATCH\_RULE\_RUN ||--|{ GOLD\_CUST\_ATTR : sources

### Mapping back to Source

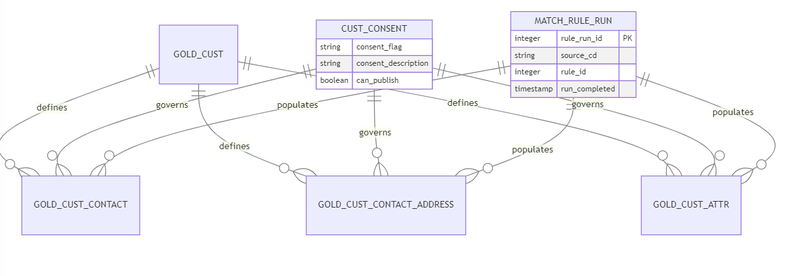
Each attribute/contact/address table contains the key of the originating source record to permit quick reference back to originating system.

### Historisation

Each entitiy is effective/end dated to permit accumulation of history.

### Consent Tagging

They are also linked to the consent and rule run via tagging columns. Here is the model that includes these relationships:



Code used to generate this diagram

erDiagram

GOLD\_CUST {

}

GOLD\_CUST\_ATTR {

}

GOLD\_CUST\_CONTACT {

}

GOLD\_CUST\_CONTACT\_ADDRESS {

}

CUST\_CONSENT {

string consent\_flag

string consent\_description

boolean can\_publish

}

MATCH\_RULE\_RUN {

integer rule\_run\_id PK

string source\_cd

integer rule\_id

timestamp run\_completed

}

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : defines

GOLD\_CUST ||--o{ GOLD\_CUST\_ATTR : defines

CUST\_CONSENT ||--o{ GOLD\_CUST\_CONTACT : governs

CUST\_CONSENT ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : governs

CUST\_CONSENT ||--o{ GOLD\_CUST\_ATTR : governs

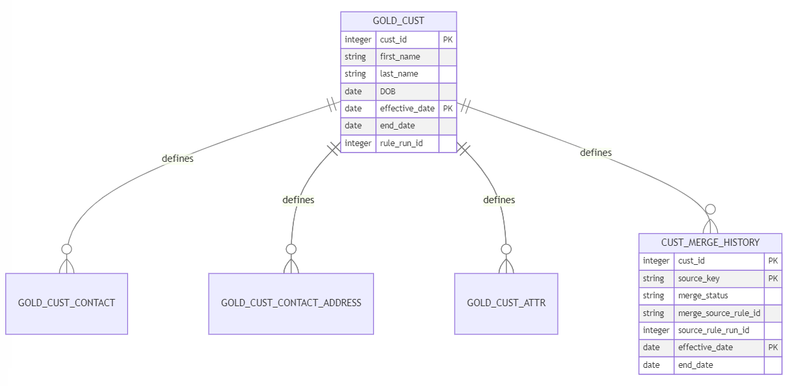
MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_CONTACT : populates

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_CONTACT\_ADDRESS : populates

MATCH\_RULE\_RUN ||--o{ GOLD\_CUST\_ATTR : populates

## Support for Merge/Split History

The golden record has a lineage, namely the source of the record or records that have contributed attributes to it. Over time as new customer records are merged or merges are split, metadata relating to these merge/splits is captured into this history.



This merge split history includes the following attributes:

|  |  |
| --- | --- |
| **Attribute** | **Definition** |
| Golden Record ID | Identifier assigned to golden record |
| Source Record ID | Identifier assigned to source record - note that this is the identifier within the CME, rather than the natural key of the record in the source. |
| Data Source Identifier | Identifier for the source system record source as registered in the matching engine metadata |
| Merge Status | Status of the record merge - True if the record is merged, or false if it has been split |
| Merge Source Rule | Match rule that led to the association of the source record to the golden record |
| Source Rule Run | Run ID of the match rule responsible for the merge (used to obtain operational metadata relating to the match) |
| Effective Date | The date that the record merg/split was effective |
| End Date | The end date for the merge/split |

### Merged Attribute Tagging

When a merge takes place attributes are “survived” from the source record into the golden record. The source for each attribute that is survived must be stored on that record so that:

* Users of the golden customer data know where it was sourced (and hence how trustworthy/usable that attribute may be)
* It is possible to disassociate the attribute from the golden record in the event that the record is subsequently split.

Merged attribute source is available in the source\_key attribute.

### Splitting Process Data Footprint

When a split is performed:

* The Merge/Split history table is updated to have a status of False (actually a new record is persisted with a False merge status)
* All Gollden Record attribute records that were survived based on the source record end dated.
* Depending on the complexity of survivorship rules simple end dating of attributes may not be sufficient and there may need to be a full recalculation of all attributes initiated

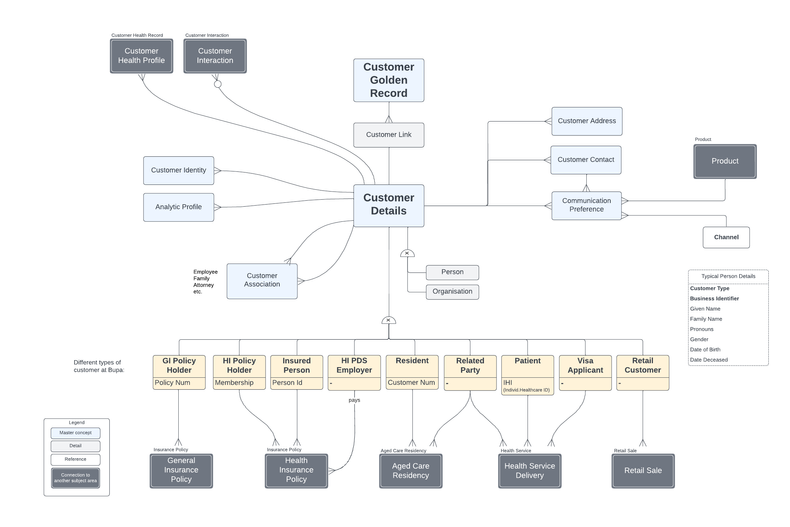
## Relationship / Hierarchy Support

Relationships between customers are often used to create customer hierarchies within the Golden Record environment. Additionally, relationships between other entities (e.g. Account or Service) are also used to establish hierarchies.

The Golden Customer record needs to be able to handle these relationships, however this aspect of the model is not covered in further detail here. As mentioned previously, the golden record should ideally interoperate with FIHR, and FHIR supports a number of relationship types and reference data relating to them.

## Publication Model

Data when published to the Gold Layer should align to the [CBDM V5 - Customer](https://bupateam.atlassian.net/wiki/spaces/DA/pages/2982805595) model as this is the advocated enterprise model for customer data within the organisation.



See the [CME Data Storage Approach](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3317826174) for further details of what is expected to be stored in each tier.

## Relationship to FHIR

FHIR is a platform specification that defines a set of capabilities for use across the healthcare process, in all jurisdictions, and in lots of different contexts. Over time more and more of the data exchanged by health related systems will be compliant with this specification.

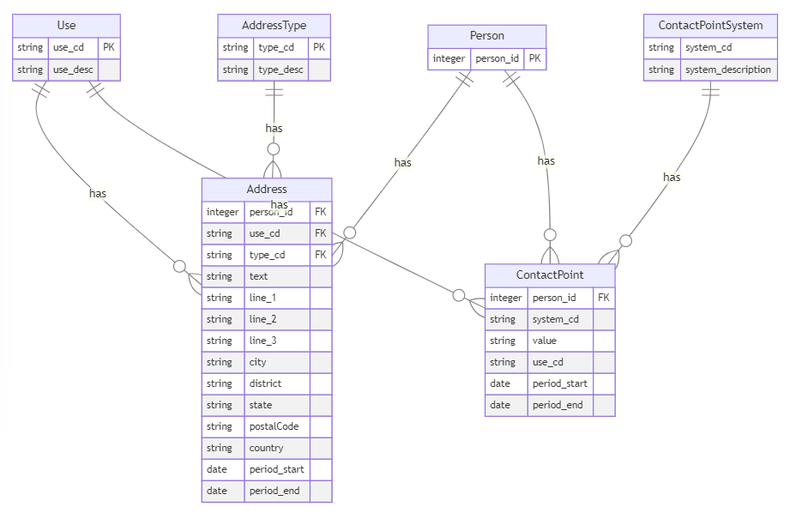
FHIR defines models for exchanging person-related data, including core person attributes (name, DOB etc), as well as contact information (e.g. Address, Phone, email etc.). It also specifies reference information models and code sets relating to these.

Given these points, it would be advantageous for the golden record model to be readily interoperable with FHIR’s message model relating to Person and ContactPoint and Address.

Documentation of these models can be found via the below links:

* [Person](https://build.fhir.org/datatypes.html#ContactPoint)
* [ContactPoint](https://build.fhir.org/datatypes.html#ContactPoint)
* [Address](https://build.fhir.org/datatypes.html#Address)

A diagram illustrating some of these is shown below:

Open image-20240208-000639.png

Code used to generate this diagram

erDiagram

Person {

integer person\_id PK

}

Address {

integer person\_id FK

string use\_cd FK

string type\_cd FK

string text

string line\_1

string line\_2

string line\_3

string city

string district

string state

string postalCode

string country

date period\_start

date period\_end

}

Use {

string use\_cd PK

string use\_desc

}

AddressType {

string type\_cd PK

string type\_desc

}

Person ||--o{ Address : has

Use ||--o{ Address : has

AddressType ||--o{ Address : has

ContactPointSystem {

string system\_cd

string system\_description

}

ContactPoint {

integer person\_id FK

string system\_cd

string value

string use\_cd

date period\_start

date period\_end

}

Use ||--o{ ContactPoint : has

ContactPointSystem ||--o{ ContactPoint : has

Person ||--o{ ContactPoint : has

As can be seen this largely mirrors the structures described in the contact/address details section above.

## Codesets

Where possible the golden model should leverage the codesets described in the [Standard Codesets](https://bupateam.atlassian.net/wiki/spaces/EDP/pages/3309961264) page.

Codesets to be leveraged include:

* Address Type
* Address Use
* Gender
* Status
  + Record effectivity status
  + There will be more statuses used than just Active, Inactive and Deleted for merge status however…

The specific codesets used over and above those listed above will be documented here as the model is realised. Contact Type is a likely future inclusion

CME Contact Information Modelling

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 21, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3294429606&selectedPageVersions=1&selectedPageVersions=2)

2 min read5 people viewed

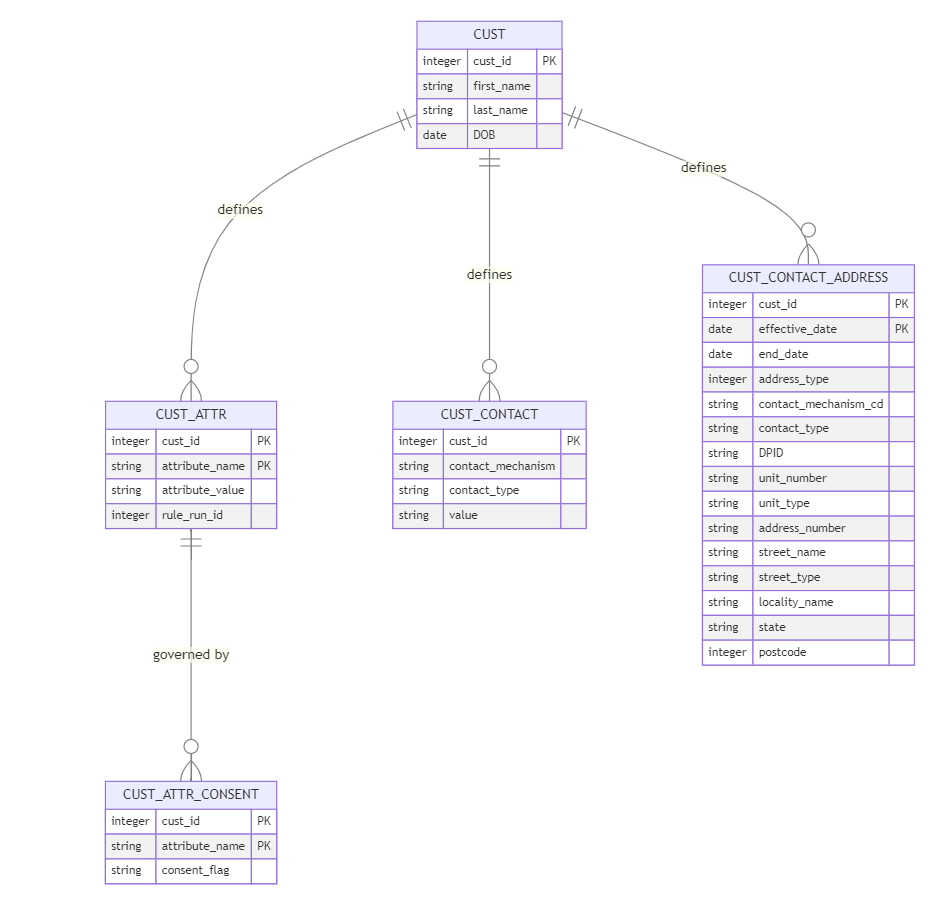
Contact information is a key input to the CME matching rules.

Different types of contact information include:

* Address
  + Postal
  + Home
  + Other (e.g. Business)
* Phone
  + Mobile
  + Home
  + Any given customer may have multiple of these
* Email
  + Personal
  + Work
  + Organisational

The way the contact information is modelled in both the golden and standardised sources must be compatible with metadata driven rules. This means it must have separate locations to store multiple versions of attributes (key-value type structure), customer contacts, and contact addresses.

The generic entities that store contact information are shown below:



Code that generated above diagram

For further details of this please refer to the [Golden Customer Record Model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3302686878)

# CME Data Classification

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Mar 04, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3336372324&selectedPageVersions=6&selectedPageVersions=7)

3 min read4 people viewed

As mentioned in the data model pages and the storage approach page CME has several different types of data. The classification appropriate to each piece of data needs to be defined, and this page outlines the classification approach which is to be used.

Data classifications that are to be used are documented on the [Data Classifications](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/2898133739) page.

The standard to which confidentiality classification is performed is the [Information Classification Standard](https://bupaau.sharepoint.com/:b:/r/sites/policiesprocedures/Shared%20Documents/Enterprise%20Information%20Security%20and%20Technology%20Standards/Information%20Security,%20Information%20Classification.pdf?csf=1&web=1&e=Y8thIK). According to this standard the confidentiality Classifications are:

* Level 0 - Can be made available to anyone without restriction
* Level 1 - Business Use Only
* Level 2 - Confidential
* Level 3 - Highly Confidential

Data is further classified as:

* Non-Sensitive
* PII
* PSI
* PHI
* PCI
* CIC
* HC

Data may be further tagged by integrity and availability. These will not be covered here at this time.

## CME Attribute Classifications

Attributes stored in CME and their associated fine-grained classifications are shown in the below table.

NOTE: This is a placeholder preliminary list sourced external to CME and is subject to localisation to CME attributes specifically, however it does list many of the attributes concerning CME

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Context** | **Category Type (Level 1)** | **Category Sub-type (Level 2)** | **Attribute Type (Level 3)** | **Sensitivity** | **CME**  **Scope** |
| Customer | Contact Details | Address Details | Accommodation Details | PII | Y |
| Customer | Contact Details | Address Details | Address Line 1 | PII | Y |
| Customer | Contact Details | Address Details | City or Region | Non-Sensitive | Y |
| Customer | Contact Details | Address Details | Country | Non-Sensitive | Y |
| Customer | Contact Details | Address Details | Delivery Point Identifier DPID | PII | Y |
| Customer | Contact Details | Address Details | Full Address | PII | Y |
| Customer | Contact Details | Address Details | Geocoded National Address File (G-NAF) Identifier | PII | TBC |
| Customer | Contact Details | Address Details | Geographic or geo-spatial coordinates | PII | TBC |
| Customer | Contact Details | Address Details | Postcode or ZIP | Non-Sensitive | Y |
| Customer | Contact Details | Address Details | State or Territory | Non-Sensitive | Y |
| Customer | Contact Details | Address Details | Suburb or Locality | Non-Sensitive | Y |
| Customer | Contact Details | Email Details | Email Address | PII | Y |
| Customer | Contact Details | Phone Details | Corporate Phone Number | Non-Sensitive | Y |
| Customer | Contact Details | Phone Details | Fax Number | PII | Y |
| Customer | Contact Details | Phone Details | Home Phone Number | PII | Y |
| Customer | Contact Details | Phone Details | Mobile Phone Number | PII | Y |
| Customer | Contact Details | Phone Details | No Phone Number Provided | NA | N |
| Customer | Contact Details | Phone Details | Phone Type Preference | Non-Sensitive | TBC |
| Customer | Contact Details | Phone Details | Preferred Phone Number | PII | Y |
| Customer | Contact Details | Phone Details | Work Phone Number | PII | Y |
| Customer | Personal Details | Name Details | Bank Account Name | PII | N |
| Customer | Personal Details | Name Details | Corporate Name | NA | N |
| Customer | Personal Details | Name Details | First or Given Name | PII | Y |
| Customer | Personal Details | Name Details | First or Given Name initial | Non-Sensitive | Y |
| Customer | Personal Details | Name Details | Full Name | PII | Y |
| Customer | Personal Details | Name Details | History or Name Changes | PII | TBC |
| Customer | Personal Details | Name Details | Last or Surname Initial | Non-Sensitive | Y |
| Customer | Personal Details | Name Details | Last or Surname or Name Suffix | PII | Y |
| Customer | Personal Details | Name Details | Middle Name | PII | TBD |
| Customer | Personal Details | Name Details | Middle or Given Name initial | Non-Sensitive | TBD |
| Customer | Personal Details | Name Details | Military Title or Rank | PII | TBD |
| Customer | Personal Details | Name Details | Name Prefix or Salutation | PII | TBD |
| Customer | Personal Details | Name Details | Pet First Name | Non-Sensitive | N |
| Customer | Personal Details | Name Details | Preferred Name or Nickname | PII | TBD |
| Customer | Personal Details | Age | Current Age | Non-Sensitive | N |
| Customer | Personal Details | Age Range or Band | Age Band | Non-Sensitive | N |
| Customer | Personal Details | Date of Birth | Date of Birth - Complete | PII | Y |
| Customer | Personal Details | Date of Birth | Date of Birth - Month and year | Non-Sensitive | N |
| Customer | Personal Details | Date of Birth | Date of Birth - Year | Non-Sensitive | N |
| Customer | Personal Details | Date of Birth | Pet Date of Birth | Non-Sensitive | N |
| Customer | Personal Details | Date of Death | Date of Death - Complete | PII | N |
| Customer | Personal Details | Date of Death | Date of Death - Month and year | Non-Sensitive | N |
| Customer | Personal Details | Date of Death | Date of Death - Year | Non-Sensitive | N |
| Customer | Personal Details | Date of Death | Deceased Indicator / Deceased Timestamp / Deceased Flag | PII | N |

## Use Case Assessment

The CME use case is subject to an assessment in terms of consent, ethics and security. A key input to this assessment is the attribute level classifications. At such time as the classification is complete

Details of information required by the assessment can be found via the below links:

* [Collection, Consent, Usage and Ethics Checklist](https://bupateam.atlassian.net/wiki/spaces/DGBDP/pages/3206316597)
* [Privacy Impact Assessment](https://bupaau.sharepoint.com/:x:/r/sites/policiesprocedures/_layouts/15/Doc.aspx?sourcedoc=%7B87B54EA2-AD27-402C-8D8E-E41B9472D358%7D&file=Privacy%20Impact%20Assessment%20Template.xlsm&action=default&mobileredirect=true&DefaultItemOpen=1)

## Classifications appropriate to Data Tiers

Data used by CME is spread across tiers and type. Each of these is described below.

### CME Metadata

CME Metadata comprises the tables that describe the sources, rules and runs of the matching process. This data is system configuration only, and should therefore be rated as **Non-Sensitive - BUSINESS USE ONLY**.

### CME Standardised Source

The standardised source data is a distillation of data that is being sourced into the CME into a standardised schema. Each source has their own schema with a standardised set of table names.

As this data is restructured only, the classification of data in this schema is simply the classification that was captured when the source was onboarded. Given that PII is used for the purposes of matching, the standardised sources should be classified as **PII - HIGHLY CONFIDENTIAL.**

Trackers that have been used to capture classifications for the sources anticipated for CME are linked to below:

* [Ingestion Tracker Optomate - Final 271123.xlsx](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/Shared%20Documents/Data%20Requirements/System%20Requirements%20-%20Source%20Systems/Optomate/Ingestion%20Tracker%20Optomate%20-%20Final%20271123.xlsx?d=w05d56f46399443e6999f89b7f290eaa3&csf=1&web=1&e=wd2Z6F)
* [Ingestion Attribute Tracker Dentally.xlsx](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/Shared%20Documents/Data%20Requirements/System%20Requirements%20-%20Source%20Systems/Dentally/Ingestion%20Attribute%20Tracker%20Dentally.xlsx?d=w22e05f673aff418eb6c4d3ed1e7188d7&csf=1&web=1&e=WJHGwl)
* [Hugo Data Dictionary object tracker](https://bupaau.sharepoint.com/:u:/r/sites/BupaDataProgramAll/Shared%20Documents/Data%20Requirements/System%20Requirements%20-%20Source%20Systems/Hugo/Hugo%20Data%20Dictionary%20object%20tracker.url?csf=1&web=1&e=tpmScw)
* [Data Classification Catalogue for BDP4 Sherlock.xlsx](https://bupaau.sharepoint.com/:x:/r/sites/TechnologyDataGovernance/DG%20References/Data%20Classification%20Catalogue%20for%20BDP4%20Sherlock.xlsx?d=w3d2a4cb1039546d280034768385aef1c&csf=1&web=1&e=SEGH9x)

See above for the fine-grained classifications appropriate to individual attributes.

### CME Silver/Gold

Output of the CME match/merge process includes mapping tables, merge candidate data and merged customer records.

The merged customer records should be considered **PII-HIGHLY CONFIDENTIAL**, with the mapping and merge candidate data considered **PII-CONFIDENTIAL.** See above for the fine-grained classifications appropriate to individual attributes.

NOTE: These classifications are preliminary and for further discussion.

# CME Data Storage Approach

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 23, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3317826174&selectedPageVersions=7&selectedPageVersions=8) by [Bing Chai](https://bupateam.atlassian.net/wiki/people/712020:36fd65a7-f8d2-4396-a9f5-420040dd8898?ref=confluence&src=profilecard)Version comment

3 min read4 people viewed

This page documents the different types of data and where they are stored, including the location within the BDP4 platform and the specific schema to be used.

## CME Data Types

CME has a number of different types of data that are maintained in the course of performing and publishing customer matches.

The data includes:

* Standardised Source Data
  + Restructured version of the source data set in the CME standard table structure with standardised table and attribute names
  + This is sourced from elsewhere in the BDP4 Bronze layer
* Ephemeral working objects
  + Work tables used during the matching process
  + Rebuilt each time a source matching run executes
  + Useful for tuning/troubleshooting matching processing
  + Purged at start of each match run
* Match Candidates (with Confidence)
  + Output of matching process
  + Used to determine records that are to be merged
* Matching Metadata
  + Inventory of the rules that are defined, the sources they apply to
  + Operational metadata pertaining to the actual match runs etc.
  + This has some overlap with control framework (when match job runs), but is hosted external to the control framework metadata.
  + **NOTE:** Venue of particular metadata items and handoffs between CME and control framework metamodel is to be discussed further
* Golden Record Data
  + Golden Customer Record and associated Address/Contact Point attributes
  + Mapping of customer record and attributes back to originating source system (source system id and natural keys)
  + Tagged with operational metadata as per Control Framework standards
* Synthesised data
  + Data Generation techniques are used in non-production environments to create realistic data for the purposes of development and unit testing.

## Matching Process Flow

The matching process and how it relates to these different data sets is shown in the below flow:

## CME Storage Strategy

CME is effectively an application, which takes data from BDP4 Bronze, processes it (creates candidate matches and merges them into a golden record), and publishes the golden record.

### CME Bronze Area

#### CME Source Standardised Objects

CME also distils data from sources into a standardised format that is used to match using standard routines. The standardised data is also persisted in the CME bronze area, with each source having its own schema to compartmentalise each source, permitting easy implementation of source aligned security configurations using Unity Catalog.

The schema naming standard for the standardised objects is BRZ\_CME\_<SOURCE\_IDENTIFIER> where the source identifier distinguishes the data sources and aligns to the source id in the CME metadata tables.

#### CME Metadata Objects

CME metadata tables are also conceptually owned by the application, therefore should be present in the bronze layer. These are also stored in the BRZ\_CME schema.

### CME Silver Area

The Silver layer is where disparate data from across Bupa comes together in an integrated model. This model aligns to the CBDM customer model (see [CBDM V5 - Customer](https://bupateam.atlassian.net/wiki/spaces/DA/pages/2982805595) ).

For CME the silver area contains the core golden record attribute and contact tables, and the mapping tables back to the originating sources. Tables related to the creation of the silver data objects are not persisted in silver, but remain in bronze as conceptually they are part of the CME when acting as a BDP4 data source application.

***NOTE: Use of silver or direct Bronze to Gold approach is something that is to be discussed.***

### CME Gold Area

The CME Gold Area is where the Golden record and mappings back to the source system data sets are exposed to end users.

These objects may be virtual (views against CME) or physical, depending on performance and usage characteristics.

### Managed Objects

Temporary objects that are used in the process of performing matching/merging operations are not stored in the external BDP4 Gold/Bronze storage area as they are ephemeral. These objects are stored within a managed schema named CME\_TMP (NOTE: Schema name to be finalised upon review and endorsement of this documentation)

Synthesized data used in development is also housed in a managed schema named CME\_SYNTHESIS.

# CME Processing Engine Requirements

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Jan 22, 2024

1 min read4 people viewed

The CME processing engine will continue to be DataBricks, but migrates from the current BDP3 SSAA environment to the BDP4 environment.

## Python Libraries

The processing engine is implemented as Databricks notebooks, containing pyspark and SQL steps. The Pyspark steps depend on a number of python packages which must be available to the platform.

BDP3 has fewer restrictions on the libraries available than the SSAA environment, so these required libraries must be explicitly defined such that they can be made available in the relevant Nexus repository.

The libraries include:

* TheFuzz
  + Fuzzy matching library providing the Levenshtein distance (LD) algorithm
* dbldatagen
  + Databricks data generation library
  + Used to generate representative data sets such that realistic data can be used when performing configuration of new sources/rules into the environment.
* nicknames
  + Used to provide the ability to match nicknames as synonyms for first names when performing matches
* faker
  + Python native data generation library
  + Used to generate representative data sets such that realistic data can be used when performing configuration of new sources/rules into the environment.
* jaro-winkler
  + Provides additional fuzzy matching algorithms that permit tuning of rules over and above LD

These libraries also require additional libraries as their dependencies. All dependencies must also be made available to BDP4 such that they can be employed in a notebook using %pip install <library\_name> without further intervention.

### Custom Packages

Custom libraries that are prepared for the CME or data generation may also be installed in the Nexus repository - this facilitates separation of logic from the match notebooks making it easier to support moving forward.

An example of this is the custgen library that provides faker generators for realistic customer data. Further details will be included

CME Development Environment

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 14, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3286663521&selectedPageVersions=9&selectedPageVersions=10)

2 min read4 people viewed

CME development is performed on the BDP4 environment. Further details can be found in the sections below whch include:

* Schema details for [CME data model](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286433910) entites
* Repository location for CME notebook and python library code

CME Schema

The storage location strategy is described on the [CME Data Storage Approach](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3317826174) page. This describes the rationale for object storage. Please refer to that page for further information.

The schema required are summarised in the below table:

|  |  |  |
| --- | --- | --- |
| **Schema** | **Description** | **Tier/Location** |
| BRZ\_CME | Results of match/merge as well as CME metadata objects | BDP4 Bronze (External) |
| BRZ\_CME\_VW\_ALL | Exposes metadata and matches but without PII | BDP4 Bronze (External) |
| BRZ\_CME\_<SOURCE\_ID> | Standardised objects for each source system.  VW\_ALL (All shareable no issues)  VW\_HI (business unit specific) | BDP4 Bronze (External) |
| GLD\_CUSTOMER (TBC) | Publication of golden record data as per gold layer modelling standards.  NOTE: This schema already exists for the purposes of the dashboard and is being built out with CME related output. | BDP4 Gold (External) |
| CME\_TMP | Ephemeral object storage for temporary objects used performing match/merge operations | Managed Objects (Internal) |
| CME\_SYNTHESIS | Synthetic data used in development environment for dev and unit testing | Managed Objects (Internal) |

Moving forward these schema should be present in the bdp4\_dev\_lh catalog, however at the current time all the external schema are present within the managed hive\_metastore - this will be remediated in the current sprint.

CME Code Repository

The CME code is present within the [dp-cme](https://bupaaunz.visualstudio.com/DefaultCollection/Data%20Program/_git/dp-cme) repository. This repository is structured in the standard way with the following directory hierarchy:

* databricks (top level directory for all databricks notebooks/libraries)
  + common (contains global imports and parameters used by all CME notebooks)
  + matching (contains the generic matching/merging notebooks that are used by each source to match/merge their data)
  + sources (parent of the per-source directories that contain the matching notebooks)
    - <source\_name> (parent folder of all notebooks specific to a single source system)
  + setup (directory containing initial environment setup code such as schema creation)
  + synthesis (parent of the code used to synthesize data used in matching activities)
    - <source\_name> (parent folder of all notebooks specific to synthesizing data for a single source system)
* metadata (directory containing sql statements that register CME processing within the control framework)
* pipeline (directory containing json representation of ADF pipelines that effect CME processing - prepared as per the [Azure DevOps Releases (CI/CD)](https://bupateam.atlassian.net/wiki/spaces/EDP/pages/2651848856) procedure)

Artifacts within the repository should follow the practices defined within the [Developer Guides](https://bupateam.atlassian.net/wiki/spaces/EDP/pages/2685534465) documentation.

# CME Security

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Jan 23, 2024

2 min read2 people viewed

Security for the CME on the BDP4 platform differs significantly from SSAA on BDP3 due to:

* A broader audience for the CME management, publishing etc. (vs single developer PoC)
* The ability to execute and support production CME workloads (ADF job submission and management)
* Additional options available for technically implementing security on the BDP4 platform

## CME Security Roles

A variety of user types are expected to engage with CME on the BDP4 platform:

* Match Developer
* Match consumers
* Data Stewards
* Privacy/Consent administrators
* Audit

It is expected that RBAC is used to define these roles and assign appropriate permissions to each of them.

### Match Developer Users

Match developers perform the following tasks:

* Standardise new data sources into the platform
* Perform data quality and match suitability profiling
* Register source and matching metadata into the CME Metamodel
* Create orchestration jobs to effect execution of the matching process

These users need:

* BDP4 Access
* Read/Write access to the repository that contains the Databricks code
* Read access to the bronze objects that will be used for source data acquisiton
* Read/Write access to the schema in which CME metamodel data is stored
* Read/Write access to the schema intermediate data is stored
* Read/Write access to the schema in which the matches are published

### Match Consumer Users

Match consumers perform the following tasks:

* Read data from the golden record publishing schema

These users need:

* BDP4 Access
  + Read access to the schema in which CME metamodel data is stored
  + Read access to the schema in which the matches are published  
      
    AND/OR
* Read access to PowerBI or whatever tool is used to provide end user access

### Data Steward Users

Data Steward users are empowered to create/confirm/break source record matches as required and review the published and prospective match candidates.

These users need:

* BDP4 Access
* Read access to the bronze objects that will be used for source data acquisiton
* Read access to the schema in which CME metamodel data is stored
* Read access to the schema intermediate data is stored
* Read/Write access to the schema in which the matches are published
* Ability to rerun match publishing jobs

### Privacy/Consent Administrator Users

Privacy/Consent admin users need the ability to review and update privacy and consent data associated with matches and sources for those matches, as well as view the merged/split themselves.

Further details of this persona will be included at a later date.

### Audit Users

Audit users can review the setup of the matching engine and the source, intermediate and published data in support of audit activities.

Further details of this persona will be included at a later date.

# CME Testing Approach

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 14, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3286630829&selectedPageVersions=1&selectedPageVersions=2)

3 min read2 people viewed

CME requires the ability to perform testing of the system, rules and publication logic without access to real data sets.

This page describes the testing approach that will be used.

## Generation of Test Data

Each source to be matched has a data definition that describes:

* The structure of the data
  + Attributes
  + Cardinality
* The data dictionary
  + Description of each attribute
  + Demographics/Domain of each attribute

This data definition is used to generate representative data. This data is generated using custom Faker providers to create values, and dbldatagen to create broadly representative data sets using them. This is the same approach that was used to generate the data used in the MDM tool vendor PoC toward the end of 2023.

### Test Data Generation Method

Test data is generated for each source, as well as for the golden record using the faker “aussie” provider. This is then transformed to ensure matchable records are present that can be merged to master.

### Generation of the Source RAW table

The “aussie” provider supplies a record with the following attributes:

* First Name
* Last Name
* Date of Birth
* Gender
* Landline Number
* Mobile Number
* Email Address
* Address, including attributes:
  + Full address (concatenation of granular address attributes into a single field)
  + Number
  + Street
  + Locality
  + State
  + Postcode

Each source and the golden customer record has at least 10k records created in a “raw” table that mirrors the above “aussie” provider structure. The raw table for each source is named <source\_name>\_raw.

The raw table is transformed to ensure that the data:

1. Looks like the source being simulated (same attributes, cardinality etc.)
2. Contains attribute values that align across sources to ensure the CME can actually match data.

### Scenario Allocation

When the raw tables are first generated there is likely no matches at all, since all data was generated randomly and in isolation.

CME matching scenarios are pre-defined match criteria that the data synthesis notebooks use to transform raw data to an output data set that contain the matches. The available scenarios are registered in the CME\_SYNTHESIS.MATCH\_SCENARIO table.

Each source is allocated a proportion of records that fit one or more match scenarios. These allocations are defined in the CME\_SYNTHESIS.SOURCE\_MATCH\_SCENARIO table. Records in this table include a percentage figure for each scenario to be generated for each source.

Finally, the scenario map table is populated for the source with the scenario id to apply for each raw record.

### Scenario Transformation

After the scenarios are allocated to a source, a notebook in that source transforms records in the raw table according to the rules for scenario into the <source\_name>\_out table.

During this operation attributes from the raw record are replaced with one or more attributes in the golden table. Each scenario has its own cell in the notebook with code that effects the transformation.

### Munging

So that fuzzy matching can be exercised, various munging routines are passed over the source data to introduce things such as typos, nicknames and nulls. The munging applied depends on the observed data within the source during profiling. For example, if during profiling it is observed a source specifies gender as ‘Male' and ‘Female’, then munging transforms the standard ‘M’ and ‘F’ domain to ‘Male’ and 'Female’ as required by the source.

### Finalisation

Once scenario transformation and munging have been performed, the data is finally restructured to match the source format. This puts data in the same table structure with the same columns and types as the source.

This is stored in the table <source\_name>\_out\_final

### Using the finalised object for development and testing

This object can then be used to develop and test the matching notebooks for the source rather than the actual source, since one can be assured matches will be present and one is not reliant on the source system preparing a representative data set in the development environment that is suitable for integration testing

CME Data Sourcing

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 28, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3289514111&selectedPageVersions=6&selectedPageVersions=7)

2 min read4 people viewed

CME Data Requirements

Requirements the data matched/merged to the golden record by CME are documented in the [CME Source Data Identification](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3313009247) page. This includes the attributes and structure that is needed.

Data Sourcing Current State

The current state of the data sourcing efforts is described in the [Data Sourcing WIP](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3330343652) page. This describes the candidate sources, findings from investigation and which sources are being progressed.

Data Provisioning

Data being matched into the CME environment is expected to be already provisioned into the Bronze or layer of the BDP4 Platform via one of the BDP4 supported ETL patterns.

This is summarised in the below diagram:

Sourcing from Bronze or Silver

As mentioned above, data may be sourced from either bronze or silver layers. This is to provide maximum flexibility and permit data to be matched even before it has been integrated into the silver environment.

If data is already present in silver, then it should be sourced from the silver layer. If it is not available in Silver, but only in Bronze, then Bronze is the location the matched data set is sourced from.

Where a data set is available only in Bronze when initially added to CME, but it is subsequently provisioned into Silver, then the data standardisation job that creates the standardised table should be updated to source from Silver.

Source Data Standardisation

Data when it is sourced is picked up by a CME data standardisation job which:

* Restructures the data to the CME source format
  + See [CME Data Models](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3286433910) page for details of the source standardised format
* May perform filtering of records that are of poor quality and unusable for matching purposes
* May map/patch data to address data quality issues such that standardised data is able to be matched using standardised match processing
  + If drawing from Silver, then it is likely this patching should already have been performed during Silver onboarding

Sourcing Data Consent / Privacy Details

Privacy and consent details should be sourced at the same time as the data is onboarded into CME. This involves acquiring consent flags during the standardisation process and loading them into the CME source schema.

# CME Sourcing WIP

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 27, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3330343652&selectedPageVersions=15&selectedPageVersions=16)

23 min read4 people viewed

This page details the work in progress around the selection of the initial sources for data ingested by CME.

For requirements these sources are attempting to satisfy, please refer to the [CME Data Sourcing](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3289514111) page.

The prospective sources described here are currently available on BDP4, however more sources/feeds are being actively onboarded to the BDP4 platform, so new opportunities may emerge during the PI, in which case the content on this page will be revisited.

Key artifacts used to establish the prospective sources are linked to below:

* [Master Ingestion Backlog](https://bupateam.atlassian.net/wiki/spaces/HDP/pages/3205890185)
* [Ingestion Master List](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/_layouts/15/Doc.aspx?sourcedoc=%7BDD12B04C-6FF5-4435-BF1C-39C919CDD188%7D&file=Ingestion%20Master%20Backlog%20V1.0.xlsx&action=default&mobileredirect=true)

## HUGO Data

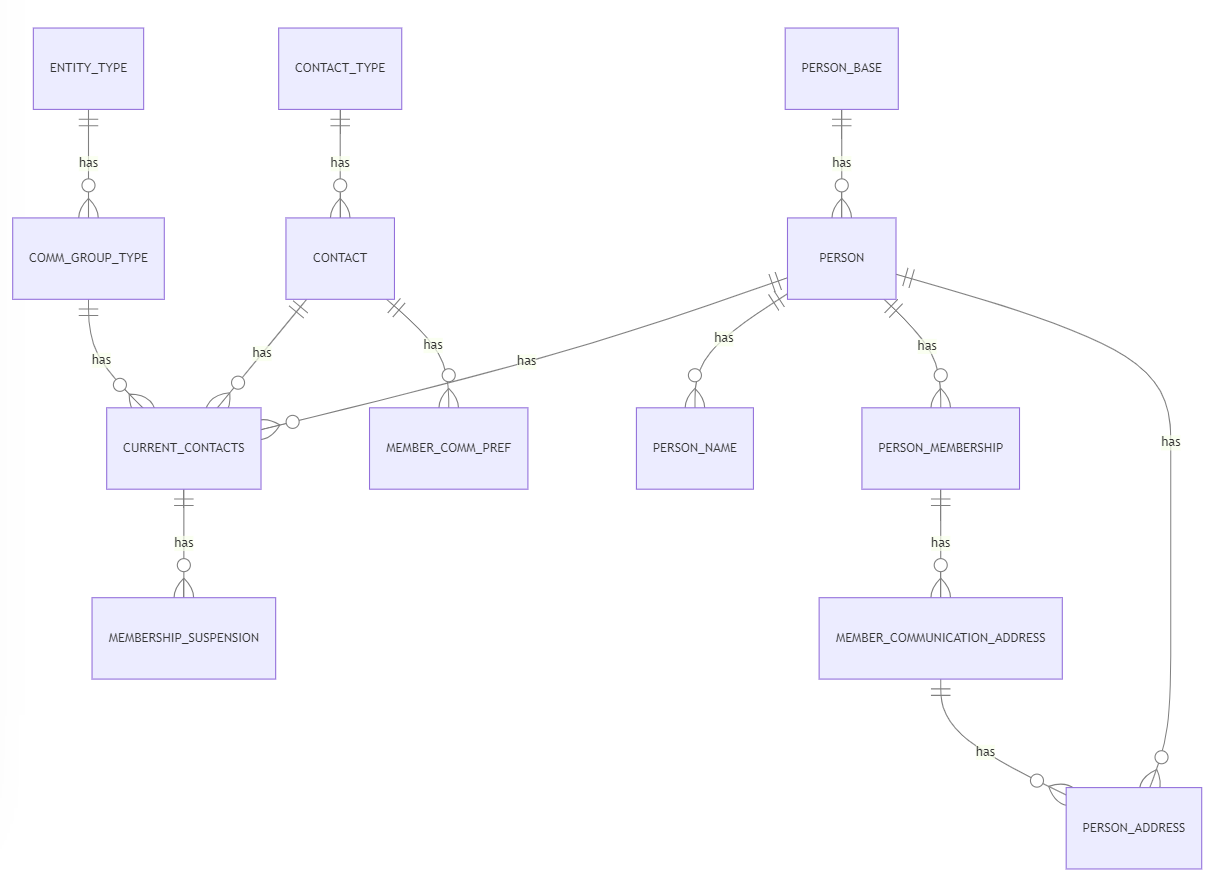
Hugo, BOSS and associated Domains are described in the [Hugo BOSS Domains](https://bupateam.atlassian.net/wiki/spaces/DSW/pages/58064961) page. The domain we wish to use for CME is the Person logical area.

Hugo stores customer data across a range of entities in a highly normalised structure, with contact information and name/DOB etc. stored in separate objects which must be joined to get a data set that is readily transformed into the CME standardised source format.

To review aspects of this please see the [Address and Comm Preference Changes](https://bupateam.atlassian.net/wiki/spaces/DSW/pages/1415414284) page which includes an ER diagram of the communication and address entities and their relationships.

In the Oracle BOSS schema the **ADVANCED\_SEARCH** entity allows Cyclops to perform sophisticated searches by person, address, and other such details. As such this entity pre-joins the atomic person tables and should be considered for the CME source. **PERSON\_MERGE** is also of interest, as it relates multiple person entries that are in actuality the same person. CURRENT\_CONTACT\_V is a view that also joins all of the required objects into a denormalised object. The base tables and join paths for these views are captured in the DDL found below. Note: The View Definitions found below were extracted from BOSSDEV1, mirrored from BOSSPRD True as of 02/02/2024. Its subject to change. Always Confirm the change in definition prior to re-creating the login & operationalizing it within BDP4.

Here is an ER diagram that shows at an entity level the relationships between the entities that underpin these views:



The status of the objects is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Ingestion Master List Status** | **In Pre-Prod** | **In Prod** |
| PERSON | Awaiting Ingestion | Y | Y |
| PERSON\_NAME | Awaiting Ingestion | N | Y |
| PERSON\_MEMBERSHIP | Awaiting Ingestion | Y | Y |
| MEMBER\_COMMUNICATION\_ADDRESS | Awaiting Ingestion | N | N |
| PERSON\_ADDRESS | Awaiting Ingestion | N | N |
| CONTACT\_TYPE | Complete | N | Y |
| CONTACT | Complete | N | Y |
| MEMBER\_COMM\_PREF | Awaiting Ingestion | N | Y |
| ENTITY\_TYPE | N/A | N | Y |
| COMM\_GROUP\_TYPE | In Progress | N | Y |
| CURRENT\_CONTACTS | In Progress | N | Y |
| MEMBERSHIP\_SUSPENSION | Awaiting Ingestion | N | Y |
| PERSON\_MERGE | Awaiting Ingestion | N | N |

#### UPDATE

Further review of the situation with regards to Address data is that the contact table does contain addresses, so by joining the following listed tables we should be able to constitute a view that sources the required Hugo data:

* Person
* Person\_Name
* Person\_Membership
* Contact
* Contact\_Type
* Entity\_Type
* Comm\_Group\_Type
* Membership\_Suspension

Person Merge is no longer used (populated) in source, and should not be factored into the design. Internal matching should be used rather than leveraging “same member” type mappings in person\_merge.

ER Diagram Source:

### View Definitions

CURRENT\_CONTACT\_V Definition

Person Merge Definition

Advanced Search Definition

## Optomate Data

The tracker for Optomate data ca be found in the [Optimate Tracker Workbook](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/_layouts/15/Doc.aspx?sourcedoc=%7BA9F2158F-FCD7-4F4C-BCFC-474D91F7FB09%7D&file=Ingestion%20Tracker%20Optomate.xlsx&action=default&mobileredirect=true)

Patient appears to be the entity that satisfies requirements for CME in a single object, [**h**](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/_layouts/15/Doc.aspx?sourcedoc=%7BA9F2158F-FCD7-4F4C-BCFC-474D91F7FB09%7D&file=Ingestion%20Tracker%20Optomate.xlsx&action=default&mobileredirect=true)**owever this object is in the “Awaiting Ingestion” status** according to the Ingestion Master List. The relative priority and timeline for the ingestion of Patient is a key item to unearth ASAP.

Other entities that would appear interesting are not usable, since PII used for the matching is all present in Patient and linked to these entities via a foreign key.

## Genesys Data

Genesys IVR data is of immediate interest, as ingesting it to CME delivers business benefit day 1. Considering the ingestion master list there are 24 feeds currently onboard. The one most immediately of interest is **USER\_DATA\_CUST\_DIM\_1** as it should contain customer attributes that can be matched. This table is already showning as “Complete” so should be present, however it is not currently available in pre-production. Use of this as a source is currently under further investigation (liaising with Rajshri Sathish).

The tracker for Genesys data onboarding (including attributes) can be found in the [Genesys Tracker Workbook](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/Shared%20Documents/Data%20Requirements/System%20Requirements%20-%20Source%20Systems/3.0%20Genesys%20(IVR%20%26%20GIA)/Genesys%20Tracker/Genesys%20Tracker_v1.3.xlsx?d=w828ea5756cc94f398fceafb652ba73c1&csf=1&web=1&e=aZxDX7)

**USER\_DATA\_CUST\_DIM\_1** is present within the development environment, but appears to be heavily masked, making that data unusable for development. As described in the onboarding approach data generation will need to be performed to onboard this source.

|  |  |  |
| --- | --- | --- |
| **Col No.** | **Col Name** | **Type** |
| 1 | edp\_pk\_hash | binary |
| 2 | edp\_row\_hash | binary |
| 3 | edp\_effective\_dttm | timestamp |
| 4 | edp\_expiry\_dttm | timestamp |
| 5 | edp\_source\_file\_name | string |
| 6 | edp\_source\_app\_name | string |
| 7 | edp\_record\_type | string |
| 8 | edp\_record\_insert\_dttm | timestamp |
| 9 | edp\_record\_update\_dttm | timestamp |
| 10 | edp\_process\_instance\_id | string |
| 11 | edp\_update\_process\_instance\_id | string |
| 12 | edp\_is\_current | boolean |
| 13 | ID | decimal(10,0) |
| 14 | TENANT\_KEY | decimal(10,0) |
| 15 | CREATE\_AUDIT\_KEY | decimal(19,0) |
| 16 | DIM\_ATTRIBUTE\_1 | string |
| 17 | DIM\_ATTRIBUTE\_2 | string |
| 18 | DIM\_ATTRIBUTE\_3 | string |
| 19 | DIM\_ATTRIBUTE\_4 | string |
| 20 | DIM\_ATTRIBUTE\_5 | string |
| 21 | edp\_partition\_year\_month | int |

## Dentally

Currently BDP4 has only ingested the Dentally treatment\_categories data set, so no data containing customer details is available. As such dental data should be considered out of scope for use as initial CME source.

The tracker for Dentally can be found in the [Dentally Ingestion Attribute Tracker](https://bupaau.sharepoint.com/:x:/r/sites/BupaDataProgramAll/_layouts/15/Doc.aspx?sourcedoc=%7B22E05F67-3AFF-418E-B6C4-D3ED1E7188D7%7D&file=Ingestion%20Attribute%20Tracker%20Dentally.xlsx&action=default&mobileredirect=true)

## Apollo

Apollo is another source that has been identified as something that should be integrated into CME. This data is not currently ingested, so should be considered for future PIs.

## ECR

ECR is another source that has been identified as something that should be integrated into CME. This data is not currently ingested, so should be considered for future PIs.

CME Source Data Validation

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Jan 24, 2024

1 min read3 people viewed

The CME requires a way to assess the data quality of source customer data so that match rules are appropriately prioritised.

Data Quality Scoring

Where an attribute or entire record is of poor quality (e.g. missing key attributes, or containing values that do not correspond with that expected for the attribute domain), the quality of the record should be scored such that this score can be used when calculating match confidence.

DQ Scoring is typically attribute based, so as data is standardised into the source customer attribute table the DQ score can be saved in that entity. Inclusion of the DQ score in the attribute table makes the quality score easily incorporated in the match confidence calculations that reference the evaluated attribute.

Contact data validation

Contact data is absolutely key to customer matching, so where possible validation of customer contact attributes should be performed.

A key set of attributes that can be readily validated is customer address. Customer address for Australian customers should be validated against a data set of Australian customer addresses, such as the GNAF data source. This involves determining if an address can be located in the GNAF data set.

CME Data Publication

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 21, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3289974702&selectedPageVersions=2&selectedPageVersions=3)

1 min read3 people viewed

CME outputs a golden record and a mapping back to customer data sourced from various source systems. This information must be published so that end users are able to leverage it.

BDP4 Published Data

The BDP4 platform has three main areas - Gold, Silver and Bronze. As Gold is the layer that is to be used for reporting, Gold is the logical location for data publication.

For further information on the Gold layer please refer to the [Gold Data Design](https://bupateam.atlassian.net/wiki/spaces/DA/pages/3252617287) page. For details on modelling guidelines to be employed refer to [BDP4 Gold Data Modelling Guidelines](https://bupateam.atlassian.net/wiki/spaces/DA/pages/2913828969)

The Gold layer prioritises user access and performance over load efficiency and is to be a star schema.

Key entities within the schema include:

* Source Customer Dimension
  + Contains the source of the customer record, and a standard set of attributes
  + Implemented as a view
* Golden Customer Dimension
* Golden Customer Fact
  + Contains the survived attribute values pertaining to each golden customer record

CME Match/Split Interface

Owned by [Duncan Heard](https://bupateam.atlassian.net/wiki/people/712020:b7fe98d2-0cb5-4ed5-9409-9f4cb40b4ca5?ref=confluence&src=profilecard)

Last updated: [Feb 08, 2024](https://bupateam.atlassian.net/wiki/pages/diffpagesbyversion.action?pageId=3310780893&selectedPageVersions=1&selectedPageVersions=2)

1 min read3 people viewed

Typically in a customer matching engine (or any master data management solution) a mechanism is provided which allows users to confirm candidate matches so that they can be merged, and to identify merged matches that need to be split.

When using MDM tooling this mechanism is provided by the MDM Tool User Interface, and may also be exposed using an API or similar to permit downstream systems to identify merges or matches.

UI/API Program Increment Scope

The BDP4 rehosted CME will not provide any UI or API mechanism to permit merges or splits to be identified during this Program Increment. It is expected that the strategic MDM tool will provide this capability.

Provision of Manual Merge/Split Capability

Absent the UI/API interface a geneic (non source specific) table will be prepared which can be used to identify records to merge or split, and code prepared that uses this table to perform the merge/split operations.

Management of data in this table is expected to be manually performed using insert/delete statements within DataBricks notebooks, and as such end users are not expected to be able to do this themselves - requiring the assistance of the CME operations staff.

In future this table may be exposed via an API (e.g. minimal FastAPI application), but once again, not in this PI.