

Version 1.0

December 4, 2018

Prepared by: M Solaimani

hcsc

Dental Claims GOLD

technical handbook

# Dental Claims GOLD

### Stake Holders

| Title | Name | Date 1 |  |
| --- | --- | --- | --- |
| Claims Data Architect | John Maenpaa |  |  |
| DDA Lead Architect | Balaji Gunniya |  |  |
| Sr. Datalake Ingestion Architect | Rakesh Billakanti |  |  |
| Sr. Datalake Ingestion Architect | M Solaimani |  |  |
| Application Architect | Arvind Misra |  |  |
| Product Manager | Sean Nicol |  |  |
| Data Modeler | Gosia Menyhart |  |  |
| Ingestion Scrum Master | Arivazhagan Marimuthu |  |  |
| Lead Developer | Deba kar |  |  |
| Developer |  |  |  |
| ITQA |  |  |  |
| DEV OPS |  |  |  |
| AMS |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Dental Claims GOLD 1](#_Toc531866272)

[Stake Holders 1](#_Toc531866273)

[1. Objective 3](#_Toc531866274)

[2. Project Summary 3](#_Toc531866275)

[3. Target Audience 3](#_Toc531866276)

[4. Target Product 3](#_Toc531866277)

[5. Scope and assumptions 3](#_Toc531866278)

[6. TRANSFORMATION high level process flow 4](#_Toc531866279)

[7. MEMBER MATCH Process 4](#_Toc531866280)

[8. TEMPLATE MAPPING DB2 DATABASE (CATDB) 5](#_Toc531866281)

[9. SPARK JOB REUSABILTY 5](#_Toc531866282)

[10. TRANSFORMATION LOW level process flow 5](#_Toc531866283)

[11. Environment and Standards 6](#_Toc531866284)

[11.1. TRIGGER FILE 6](#_Toc531866285)

[11.2. Incoming Files/Sources 7](#_Toc531866286)

[11.3. KAFKA 7](#_Toc531866287)

[11.4. Hadoop 8](#_Toc531866288)

[11.5. Yarn Queues 8](#_Toc531866289)

[11.6. Webroles/NPID 8](#_Toc531866290)

[11.7. HDFS Paths 8](#_Toc531866291)

[11.8. ZENA 8](#_Toc531866292)

[1.1. Application logs 9](#_Toc531866293)

[1.2. SCRIPT 9](#_Toc531866294)

[2. CLAIM PROC EXEC TABLE 9](#_Toc531866295)

[3. Audit and Balance 10](#_Toc531866296)

[4. Ancillary Processes Ancillary Processes 11](#_Toc531866297)

[4.1. GitHub 11](#_Toc531866298)

[4.2. Jenkins 11](#_Toc531866299)

[5. 11](#_Toc531866300)

[5.1. Appendix 11](#_Toc531866301)

## Objective

The objective of this document is to provide a low level technical design for the implementation of the Dental Claims Ingestion.

## Project Summary

Gold claims repository is the centralized location in the data lake to have all the claim categories stored in consistent common claims format. In current System Data lake receives a Dental Claim file daily in raw format. As part of this initiative Pharmacy claims in Dental file are to be parsed and converted to common claim format and converted claims are to be added to Kafka topic for Claim API to consume and stored in gold claims repository.

## Target Audience

Developers, Production Support Analysts, AMS, Dev ops, QA

## Target Product

**DDA Claims Gold**

## Scope and assumptions

Scope of the document is to provide detailed technical design for transforming Dental claims data present in incoming raw Dental files to Common Claim Format that will be stored in datalake Gold layer and sent to Kafka for Outside consumers.

* Dental
  + Dentalnet (DENTALNETTOCCF) – daily
  + Teledent (TELDENTTOCCF) - daily

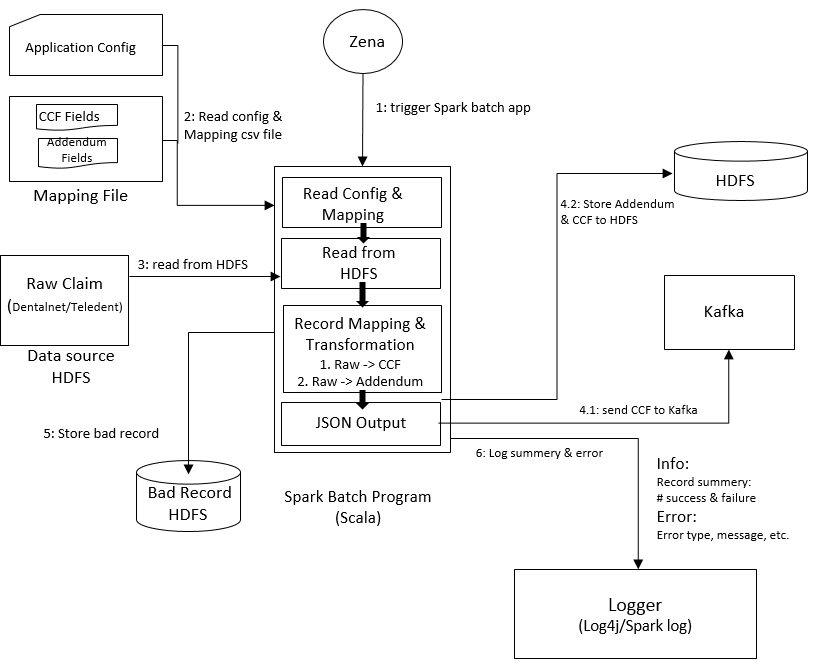
## TRANSFORMATION high level process flow

This section of the document describes detailed high level technical information about flow of how files ingested into datalake incoming layer will be transformed to CCF format and moved to Kafka and HDFS Gold layer.

* On successful completion of ingestion of file to incoming layer a Trigger will be fired

this will invoke new Dental to CCF transformation process.

* Dental Data (dentalnet and teledent) will be converted to CCF and Addendum JSON format.
* Data converted to CCF will be added to a Kafka topic for Consumption by Claim API.
* Data converted to CCF and Addendum will be stored to HDFS for consumption interfaces like search and Analytics.
* Count of records in each Dental file will be validated against successful and failed CCF records.



## MEMBER MATCH Process

Dental data does not have member information for all the claims.

A new process called Member Match process will be developed that will derive member information and append it to each claim in Dental file.

Mapping Details of field added by Member Match process will be added to Mapping Template so that same will be populated in output CCF json.

## TEMPLATE MAPPING DB2 DATABASE (CATDB)

A DB2 Database called CATDB will be created that will store mapping between various Claim source fields for various kinds of claims, corresponding CCF fields name, CCF json path, data types etc.

## SPARK JOB REUSABILTY

Spark job that will be converting Dental fixed length file to CCF json will be developed such that it can be reused for other sources as well.

Spark job will extract mapping between source fields and target json fields for CATDB database,

Until CATDB is created and populated Spark hob will use Template mapping file that represent mapping data in CATDB in CSV format.

Path to Mapping template that provides mapping between source fields and destination CCF fields will be passed as parameter to Spark job.

Spark Job will create Json structure based on json path provided in mapping template and populate them with values in source template.

This job can be used with any Fixed width file source passing Mapping template created for the given source.

## TRANSFORMATION LOW level process flow

This section of the document provides details of low level components that will be developed to ingest and transform data.

Below process flow is categorized into seven steps.

Quick summary of Ten logical steps:

**Step1:** Member match process will drop a trigger on successful completion.

**Step2:** A new Zena directory monitor will be setup to monitor for trigger and will invoke Spark job on receiving trigger.

**Step3:** Spark Job will read Template mapping csv file for mapping between Dental (a separate job for dentalnet and teledent) and CCF fields and load it to a Map.

**Step4:** Spark Job will read Daily raw file and convert each field to corresponding CCF field and create a CCF Json and Addendum Json.

**Step5:** Spark Job will validate data type of fields based on data type specified in Template Mapping.

**Step6:** If data validation is passed Job will write record to Kafka and Good data HDFS location.

**Step7:** For every good record two Json claim json and Addendum json will be generated and written to separate HDFS locations as per HDFS path specified in configuration file.

**Step8:** Failed records will be written to Bad Data HDFS location as per HDFS path specified in configuration file.

**Step9:** Spark Job will validate count of claims in each dental file against total claims written to Good data and Bad data HDFS location.

**Step10:** In case of count mismatch error will be recorded in logs and Application Exception will be Thrown.

## Environment and Standards

### TRIGGER FILE

**Dental to CCF Trigger:**

Trigger for process will be a trigger file

Dental\_TRG\_DENTALNET\_<MMDDYY\_MM\_SS>.trg

dropped to location “/datalakedata/prod/gold/integration/work/claims/dental/processed/”

Dental\_TRG\_ TELDENT\_<MMDDYY\_MM\_SS>.trg

dropped to location “/datalakedata/prod/gold/integration/work/claims/teledent/processed/”

Trigger file will have Information on Dental sources file to be processed following format

hdfsFile="< DENTALNET/TELDENT HDFS file fully qualified path and name>"

**History Trigger:**

claims\_dental\_dentalnet\_hist\_raw\_memmatch\_<DateTimeFromFileName>.trg

**On Going Trigger:**

claims\_dental\_dentalnet\_raw\_memmatch\_YYYYMMDDHH24MISS.trg

claims\_ dental \_teledent\_raw\_memmatch\_YYYYMMDDHH24MISS.trg

claims\_dental\_dentalnet\_memmatch\_gold\_YYYYMMDDHH24MISS.trg

claims\_dental\_teledent\_memmatch\_gold\_YYYYMMDDHH24MISS.trg

Note : YYYYMMDDHH24MISS --> System Date and Time

**Sample Trigger Files:**

claims\_dental\_dentalnet\_raw\_memmatch\_20181201030432.trg

HDFSFilePath=/prod/incoming/raw/claims/dental/dnoa/ascii/dentalnet/2018-11-30

LakeFileName=<SOURCE FILE NAME>

claims\_pharmacy\_teledent\_raw\_memmatch\_20181201030432.trg

HDFSFilePath=/prod/incoming/raw/claims/dental/dnoa/ascii/teledent/2018-11-30

LakeFileName=<SOURCE FILE NAME>

claims\_dental\_dentalnet\_memmatch\_gold\_20181201030432.trg

HDFSFilePath=/prod/incoming/raw/claims/dental/dnoa/member\_match/dentalnet/2017/10/28/023400

LakeFileName=<RAW\_FILE\_NAME>\_MEMMATCH

claims\_dental\_teledent\_memmatch\_gold\_20181201030432.trg

HDFSFilePath=/prod/incoming/raw/claims/dental/dnoa/member\_match/teledent/2017/10/28/023400

LakeFileName=<RAW\_FILE\_NAME>\_MEMMATCH

### Incoming Files/Sources

/prod/incoming/raw/claims/dental/dnoa/ascii/dentalnet

/prod/incoming/raw/claims/dental/dnoa/ascii/teledent

/prod/incoming/raw/claims/dental/dnoa/member\_match/dentalnet

/prod/incoming/raw/claims/dental/dnoa/member\_match/teledent

### KAFKA

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Producer | Consumer | Topic | Message Key | Message Body | PartitionsYN | Producer | Consumer | Message Retention |
| MCEF to CCF Spark process | ClaimAPI | topic\_claim\_ccf\_complete | CorporateEntityCode—AdjustmentNumber—ClaimStatusCode-DCN | CCF Json | Yes | MCEF to CCF Spark process | ClaimAPI | 7 days |

### Hadoop

|  |  |  |  |
| --- | --- | --- | --- |
| Environment | Hostname | IP Address |  |
| Dev | **dazrslenap0001.app.dev.hcscint.net** | **10.233.37.144** |  |
| SIT | **dwauslaenapp02.app.dev.hcscint.net** | **10.130.1.87** |  |
| Pre Prod | twauslaenapp04.app.test.hcscint.net | 10.134.136.75 |  |
| Prod | **pwauslaenapp04.app.hcscint.net** | **10.136.136.37** |  |

### Yarn Queues

ETL Queue.

### Webroles/NPID

|  |  |  |  |
| --- | --- | --- | --- |
| Source | ETL Web role | NPID | Environments |
| MCEF | HDOS Hdp ETL Enterprise Claim | hdhlmn01 | Dev/SIT/Pre Prod/Prod |

### HDFS Paths

GOLD Paths:

/prod/gold/claims/certified/full/ccf/dental/teledent/YYYY/MM/DD/

/prod/gold/claims/certified/full/ccf/dental/dentalnet/YYYY/MM/DD/

/prod/gold/claims/certified/full/addendum/dental/teledent/YYYY/MM/DD/

/prod/gold/claims/certified/full/addendum/dental/dentalnet/YYYY/MM/DD/

Bad Folders:

/prod/gold/claims/baddata/dental/teledent/YYYY/MM/DD/

/prod/gold/claims/baddata/dental/dentalnet/YYYY/MM/DD/

Status code

**DENTALNETTOCCF**

**TELDENTTOCCF**

### ZENA

All the Zena process should be Directory Monitor event-based triggers and process events should be developed in one of below location of Zena Dev environment as per Claim Type.

**TIS Datalake\GOLD\Claims\01-Pharmacy**

**TIS Datalake\GOLD\Claims\02-Medical**

**TIS Datalake\GOLD\Claims\03-Dental**

**TIS Datalake\GOLD\Claims\04-Vison**

**Launch Process:**

TIS Datalake CLAIMS Dental History Launch Process

                with two events

                                one for dentalnet

                                one for teledent

**Raw to CCF Conversion:**

TIS Datalake CLAIMS Dental Dentalnet Member Match Process

TIS Datalake CLAIMS Dental Teledent Member Match Process

**Member Match to CCF conversion:**

TIS Datalake CLAIMS Dental Dentalnet to CCF Producer

TIS Datalake CLAIMS Dental Teledent to CCF Producer

### Application logs

Application logs will be generated at following location.

/datalakedata/prod/gold/integration/logs/claims/<claimType>/

With log file name as follows.

Dentalnet\_claims\_MMDDYY\_HHMM.log

Teledent\_claims\_MMDDYY\_HHMM.log

### SCRIPT

Above Zena process on being triggered will call following shell script

“/datalakebin/prod/gold/integration/src/scripts/claims/dentalnet/claim\_dental\_dentalnet\_to\_ccf.sh”

“/datalakebin/prod/gold/integration/src/scripts/claims/teledent/claim\_dental\_teledent\_to\_ccf.sh”

Above shell script will submit Spark job for execution.

## CLAIM PROC EXEC TABLE

A hive table “PROC\_EXEC\_CLAIM\_GOLD” under hive database “GOLD\_CLAIM\_WORK” will be created.

This table will be used to record execution details of every job run and will be partitioned by job execution date.

This table will also record count of claims in source system and number of claims transformed successfully to CCF and number of Claims failing CCF transformation.

Following is Hive DDL for table.

DROP TABLE IF EXISTS GOLD\_CLAIM\_WORK.PROC\_EXEC\_CLAIM\_GOLD;

CREATE EXTERNAL TABLE GOLD\_CLAIM\_WORK.PROC\_EXEC\_CLAIM\_GOLD (

CLAIM\_CATG STRING

,SRC\_SYS STRING

,SRC\_FILE STRING

,OPERATION STRING

,EXEC\_STS STRING

,ZENAJOB\_NM STRING

,START\_TS STRING

,END\_TS STRING

,SRC\_REC\_COUNT STRING

,CCF\_GOOD\_REC\_COUNT STRING

,CCF\_BAD\_REC\_COUNT STRING

,DESCRIPTION STRING

)

PARTITIONED BY (EXEC\_DT STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '|'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

LOCATION '/prod/work/gold/claims/proc\_exec\_claim\_gold'

TBLPROPERTIES('serialization.null.format'='?');

## Audit and Balance

At end of Processing Spark job will match count of records in Dental files against total failed and successful records converted to CCF.

Total counts and job completion status will be logged to “PROC\_EXEC\_CLAIM\_GOLD” Hive table and in case of mismatch error will be logged and System exit will be invoked with non-zero return code.

Non-zero return code will be trapped by shell script and will result in failure of Zena Job.

## Ancillary Processes Ancillary Processes

### GitHub



Datalake Gold repository will be used for project.

URL: https://ghe.fyiblue.com/HCSC-Pilot/datalake\_gold

Repositories: HCSC-Pilot/ datalake\_gold

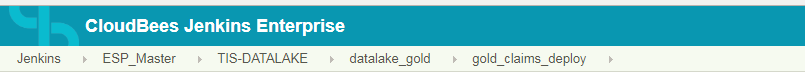
### Jenkins

Jenkins will be used to automate migration of code components and objects from GIT repository to Datalake Test and Prod environments.

URL: <https://cjp.fyiblue.com>

Full project name: TIS-DATALAKE/datalake\_gold/gold\_claims\_deploy

Locations of Jenkins Jobs which are used for configure and migrate.



## 

### Appendix