Low Level Design Document

For

Data Ingestion Process

**Version History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Updated by | Description of changes made | Intended Audience |
| 1.0 | 09/08/2021 | Rajaraman  Nandakumar | First draft created | IEEE Core Development Team |
| 2.0 | 24/08/2021 | Rajaraman  Naveen  Shanthapriya | Sprint 3 Activities | IEEE Core Development Team |
|  |  |  |  |  |
|  |  |  |  |  |

[**1.** Introduction 3](#_Toc82500878)

[**1.1.** Purpose 3](#_Toc82500879)

[**1.2.** Tools Involved: 3](#_Toc82500880)

[**2.** Data Standardization Strategy 3](#_Toc82500881)

[**2.1.** Data Standardization Strategy for Address 3](#_Toc82500882)

[**2.2.** Data Standardization Strategy for First Name, Middle Name, Last Name 4](#_Toc82500883)

[**2.3.** Data Standardization Strategy for Organization Name 5](#_Toc82500884)

[**2.4.** Data Standardization Strategy for DOB 5](#_Toc82500885)

[**2.5.** Data Standardization Strategy for Phone 6](#_Toc82500886)

[**2.6.** Data Standardization Strategy for Email 6](#_Toc82500887)

[**3.** Validation Logic 7](#_Toc82500888)

[**4.** Data Grouping Logic 7](#_Toc82500889)

[**4.1.** Level 1 Grouping based on last name 8](#_Toc82500890)

[**4.2.** Level 2 Grouping based on first letter of last name , first letter of first name and for each group from 2nd Bucket in L1 grouping. 8](#_Toc82500891)

[**4.3.** Level 2 Grouping based on first letter of last name and first letter of first name for bucket 3 8](#_Toc82500892)

[**5**. Needs to be changed (Logical Merging of Source records for across group matching) 9](#_Toc82500893)

[6. Creating Configurable Code for Address Standardization and Domain Specific Grouping 11](#_Toc82500894)

[7. Map Source File Attribute to Domain Attribute 11](#_Toc82500895)

# Introduction

## Purpose

The Purpose of the document is to define the Data ingestion development design which includes Data Profiling, Data Cleansing, Standardization of Data rules, Grouping strategy and GRAD Rules. This will be a recurring document which will be updated as on when the sprint development is completed.

## Tools Involved:

|  |  |
| --- | --- |
| **Storage** | AWS S3 |
| **NoSQL Data Base** | Mongo/Document DB |
| **Technology** | Python 3.7 |
| **Modules** | Pandas 1.3.1, Regex, boto3 |
| **Repository** | GitHub (https://git.ieee.org/projects/AA) |
| **Tracking** | JIRA (https://jira.ieee.org/projects/AA) |

# Data Standardization Strategy

The below section describes various standardization strategies being used in the project.

## Data Standardization Strategy for Address

**~~Note:~~** ~~This is the temporary functionality we have developed for workaround and to move ahead with the development. Once any Address Validation service is confirmed and implemented this function will be removed.~~

**Functionality –** Reading the file using pandas and apply the developer defined regex function for Address Standardization

Regex Functionalities used in the code are:

* Removing the multiple space with Single space,
* Removing Special characters
* Removing Starting and ending spaces(trimming)

|  |
| --- |
| **reg1 = ' +'**  **reg2 = '[^\w\s]'**  **reg3 = '[ ]+$'**  **reg4 = '^[ ]+'** |

**Using dict for replacing address abbreviations**

|  |
| --- |
| **' (rd);' : " road;",' (rd)\|' : " road|",' rd ' : ' road ','^(pl )+' : 'place ',' (pl);' : " place;",' (pl)\|' : " place|",' pl ' : ' place ','^(ext )+' : 'extension ',' (ext);' : " extension;",' (ext)\|' : " extension|",' ext ' : ' extension ','^(ln )+' : 'lane ',' (ln);' : " lane;",' (ln)\|' : " lane|",' ln ' : ' lane ','^(ave )+' : 'avenue ',**  **' (ave);' : " avenue;"' (ave)\|' : " avenue|"' ave ' : ' avenue ','^(hwy )+' : 'highway ',' (hwy);' : " highway;"}** |

Note: The above dict file will be configured and saved in ~~MongoDb~~ DocumentDB so that in future it can be edited and used directly

* Replaced all the "unspecified" string value from the address to empty string
* Replaced all the multi space with single space
* Replaced all the punctuation marks and special characters with single space
* Removed non-English characters from the address fields of all types
* Removed empty space from starting and ending of the address fields of all types
* Categorized Personal & Professional Address
  + Home address to Personal\_Address
  + Primary, work, school & other Address to Professional Address
* Differentiate columns with ';' delimiter and differentiate the category with '|' delimiter
* removing extra pipe delimiter from end and start after concatenation.
* The abbreviations are standardized

**End output columns for ML Processing**  – Personal\_address & Professional\_address

Below shown is one example of the input record and its corresponding output record after passing through the address Standardisation code

**Input Record**s : 1620\* NE northwood $dr apt j203,pullman,wa,usa#,|99163 5146 3131 'walnut st apt 223,philadelphia,pa, usa,19104 3417'

**Output Records:**

Professional \_address:1620 north east northwood dr apt j203;pullman;wa;usa;99163 5146

Personal\_ adddress:3131 walnut street apartment 223;philadelphia;pa;usa;19104 3417

## Data Standardization Strategy for First Name, Middle Name, Last Name

**Functionality –**

Reading the file using pandas and apply the developer defined regex function for Name Standardization

Analyzing the below naming convention and excluding it from the columns using regexp\_replace

|  |
| --- |
| pat= r'(\||\.|Mr|Mrs|Miss|Ms|Mx|Sir|Dr|Lady|Lord|Madam|Maam|Mr|Mrs|Miss|Ms|Dr|Admiral|Air Comm|Ambassador|Baron|Baroness|Brigadier|Brother|Canon|Capt|Chief|Cllr|Col|Commander| |

Regex expressions used for removing the Special characters and the honorific from first name columns

|  |
| --- |
| **r'^[^@\s]**[**+@[^@\s]+\.[^@\s]+$**](mailto:+@[%5e@\s%5d+\.%5b%5e@\s%5d+$)  **r'[^a-zA-Z\s]+''** |

* Remove Accent Characters
* Remove Non-English Characters
* Remove Email Strings
* Remove Honorifics
* Remove Special Characters
* Remove Unwanted spaces

**End output columns for ML Processing** – first\_name, middle\_name,last\_name

Below is a sample record input and output to go through the name standardization

~~Lets have a sample record to pass through this address Standardisation and check the output~~

**Input Records**: “paxson,vern”, “pax,$vern”, “&paxs vern1998”

**Output Records**: “paxson,vern”

## Data Standardization Strategy for Organization Name

**Functionality –** Reading the file using pandas and apply the developer defined regex function for org\_name Standardization

Analyzing the below naming convention and replacing it using regexp\_replace

|  |
| --- |
| **abbr = { "ltd": "Limited", "Ltd": "Limited", "corp": "Corporation", "Corp": "Corporation", "inc": "Incorporated", "Inc": "Incorporated", "org": "Organization", "Org": "Organization"}**  **pat = r'(ltd|Ltd|org|Org|uni|Uni|inc|Inc|inst|Inst|corp|Corp|Univ|univ)$** |

* Remove Accent Characters
* Remove the Values if it Contains Only Numbers
* Remove Special Characters
* Remove extra spaces
* Standardize the Abbreviations e.g., Ltd – Limited
* Multiple values separated by pipe delimiter.

**End output column**: org\_name

Let’s have a sample record to pass through this org\_name Standardisation and check the output

Input Records: Rutgers Univ,

Output Records: Rutgers University

## Data Standardization Strategy for DOB

**Functionality –** Reading the file using pandas and apply the developer defined regex function for DOB Column Standardization

|  |
| --- |
| **reg = '[^+0-9]+'** |

**End output Columns** – DOB

* Replaced the delimiter as -, to maintain uniformity
* Converted date format as dd-mm-yyyy

Let’s have a sample record to pass through this DOB Standardisation and check the output

Input Records: 1927-01-05

Output Records: 05-01-1927

## Data Standardization Strategy for Phone

**Functionality –** Reading the file using pandas and apply the developer defined regex function for Phone\_Number Standardization. Please see the details below

Using Reg expressions and applying Regex functions for replacing Special Character and space

|  |
| --- |
| **reg = '[^|+0-9]+'**  **eg1 = '\|+'**  **reg2 = r'[+]\*$'**  **reg\_lead = r'^[|]\*'**  **reg\_lag = r'[|]\*$'** |

Note: If the length of the phone number is greater than 15 and less than 7 then record is considered as invalid

* Removed spaces
* Removed special characters like -, () and XXXX
* Removed suffix + sign
* Mark Invalid phone number for length greater than 15 or less than 7
* Multiple values separated by pipe delimiter.

**End Output Columns** – P**hone\_Number**

Let’s have a sample record to pass through this Phone\_number Standardisation and check the output

**Input records:** +92 21 111 128 128 and +1-917-4355028

**Output records:** +9221111128128 and +19194355028

## Data Standardization Strategy for Email

**Functionality –** Reading the file using pandas and apply the developer defined regex function for email column Standardization. Please see the details

Using Reg expressions and applying Regex functions for replacing Special Character and space

|  |
| --- |
| regexp\_replace(lower(df\_email["con\_email\_addr\_txt"]), "[[^a-zA-Z0-9@\.\_\-|](mailto:%5ea-zA-Z0-9@\._\-|) ]", "")) \  .withColumn("con\_alt\_email\_addr\_txt", functions.regexp\_replace(lower(df\_email["con\_alt\_email\_addr\_txt"]), "[[^a-zA-Z0-9@\.\_\-|](mailto:%5ea-zA-Z0-9@\._\-|) ]", "")) |

**C**all regexp\_extract\_all to extract the array of valid emails.

|  |
| --- |
| **extract\_expr = expr( "regexp\_extract\_all(concat, '(\\\w+([\\\.-]?\\\w+)\*@\\[A-Za-z\-\.]+([\\\.-]?\\\w+)\*(\\\.\\\w{2,3})+)', 0)")** |

Call regexp\_replace for replacing Special characters

|  |
| --- |
| **regexp\_replace ("Email1", '[|,]\*$', ""))** |

* Checked for @ symbol present in mail.
* Checked for domain present in mail
* Remove all invalid emails that does not contain @ or domain
* Multiple values separated by pipe delimiter
* check for.(period) symbol present in mail

**End Output Colums** – **email**

Let’s have a sample record to pass through this Email Standardisation and check the output

**Input Record** : "razansadeq@yahoo.com","razansadeq@yahoocom"

**Output Record** : "[razansadeq@yahoo.com](mailto:razansadeq@yahoo.com)

The **JSON**File with all the Applied Standardization logic will be processed further to Data Validation logic

# Validation Logic

Null check in Last\_name Column and Primary key Column(Source\_id), Those record will be removed and written to the error File.

After the Data Standardization and Validation the below are the list of columns which we are used to send the details to the ML team

1. **con\_id**
2. **First\_Name**
3. **Middle\_Name,**
4. **Last\_Name**
5. **Email**
6. **Phone\_Number**
7. **Date\_of\_Birth**
8. **org\_name**
9. **Personal\_Address**
10. **Professional\_Address**
11. **Source**

# Data Grouping Logic

Once Data Standardization and validation is completed, Data set file is created based on the Alphabetical order (First Letter of the Last Name) Total 26 File for each Source Siebel and Pubs (Editor, Author and Reviewer), Merge all Files in to one JSON file and we are converting to CSV and that file is used for Data Grouping logic

## Level 1 Grouping based on last name

1. Bucketing: Records between 2-999: Files created and shared with ML team
2. Bucketing: Records >= 1000:
3. Bucketing: Records <2:

The 2 and 3 Bucketing Recorder will be processed for Level 2 Grouping.

## Level 2 Grouping based on first letter of last name , first letter of first name and for each group from 2nd Bucket in L1 grouping.

1. Bucketing for Records between 2-999 : Files created and shared with ML team
2. Bucketing for  Records >= 1000 : Files created and shared with ML team
3. Bucketing for  Records < 2 : Files created and shared with ML team

## Level 2 Grouping based on first letter of last name and first letter of first name for bucket 3

1. Bucketing for Records between 2-999 : Files created and shared with ML team
2. Bucketing for  Records >= 1000 : Files created and shared with ML team
3. Bucketing for  Records < 2 : Files created and shared with ML team

|  |  |
| --- | --- |
| Input Folder Structure | |
| Parent folder (3) | Subfolder |
| Level 1 (Grouping based on last names) | Groups with 2-999 records |
| Level 2 (Grouping based on first letter of first name and first letter of last name for groups containing >=1000 records after Level 1) | Groups with 2-999 records |
| Groups with >=1000 records |
| Groups with unique record |
| Level 2 (Grouping based on first letter of first name and first letter of last name for groups containing unique record after Level 1) | Groups with 2-999 records |
| Groups with >=1000 records |
| Groups with unique record |

The Final set file Buckets will be sent to ML for disambiguation

# Needs to be changed (Logical Merging of Source records for across group matching)

* Python code is build for comparing the GUID Repository with a new record and it is performed with a lookup on GUID Repository using Source Primary Identifier, If the same value in the primary identifier matches in the GUID the entire document in the GUID will be replaced with a new source records and that record with primary identifier which replaced GUID record will be removed from the disambiguation pipeline ( it wont go the ML)

for eg:

{

Source id : PERSBL123

Source primary Identifier: 1357111

}

* Created configuration parameter in Mongo DB domain Master collection where we are setting parameters required for bucketing size and criteria.

"grouping": {

"level1": {

"criteria": "last\_name",

"upper\_limit": 1000,

"lower\_limit": 2

} ,

"level2": {

"criteria": "first\_name",

"criteria2": "substring('first\_name', 1, 1)",

"criteria1": "substring('last\_name', 1, 1)",

"lower\_limit": 2,

"upper\_limit": 1000

} ,

"path": {

"root\_path": "D:/grouping/",

"l2\_buckt\_path": "temp/G2\_2\_999/",

"l2\_buckt\_path1": "temp/G2\_1000+/",

"l2\_buckt\_path2": "temp/G2\_Unique/"

} ,

"prefix": {

"l1\_file\_prefix": "L1\_2-999\_",

"l2\_file\_prefix": "L2\_2-999\_",

"l2\_file\_prefix1": "L2\_1K+\_",

"l2\_file\_prefix2": "L2\_Unique\_",

"l2\_u\_prefix": "L2\_U\_2-999\_",

"l2\_u\_prefix1": "L2\_U\_1K+\_",

"l2\_u\_prefix2": "L2\_U\_Unique\_"

}

}

* Created a Python code for creating a bucketed records in groups based on the configuration parameter created in the MongoDB above and creating one file for each bucket , Hence 7 files will be created for each bucket

**Module** - PyMongo

* Created a mongo DB document where all source folder structure will be configured. While reading the files from the source, value will be read from MongoDB collection(Source Registration)
* Python Code is created based on the folder structure as mentioned below. Please find the folder structure which is expected to be in S3

|  |
| --- |
| Siebel /Input  Siebel/output  Siebel/Error  Siebel/Archive |

**Module** - OS(Local – Windows), Threading

* In Addition, the above python code is to check for files in the corresponding Source Folders and if any of the source does not have the file, the corresponding thread alone will stop and the flow continues with the remaining source thread
* All the sources which land to s3 bucket under respective source folders and below are the pointers
* Developed a python code to do a File level validation for all the sources which lands to s3 bucket under respective source folders and below are the pointers
  + - 1. Creating Multiple thread to run all sources in Parallel
      2. **File Size Validation** - If the size of the file is zero, we will stop processing the thread function for the particular source and we will notify the DSO by mail
      3. **Record Count Check** - Created a python function which will check the record in the respected input file
      4. **File Extension Validation** - Created a python function which will check the record in the particular input file ( CSV or JSON)
      5. **Validation on File Naming Convention** - Created a python function to check if the file name is following proper naming convention

**Naming Convention needs to be mentioned**

* If the above validation fails the corresponding error function will be called and will notify the DSO by mail (Design in Progress)

* Python function is created which will check for the null values for the mandatory columns which are configured in source registration collection of MongoDB.          This function will take a file either csv/ json and will read the attributes from source registration for which **IsNULL flag** is set False and then it will filter out those records from the files to error record file.    (Need to confirm on the Notification which is in discussion)

      Eg : Put a document

# Creating Configurable Code for Address Standardization and Domain Specific Grouping

* Create Mongo DB **Domain master collection** to hold all the configuration details for each and every attribute.

  **Sample collection**

[{"domain\_id":"PER123","domain\_name":"Person","attributes":[{"a\_name":"first\_name","a\_datatype":"string","isNull":"true","std\_function":"name\_std","singlevalued":"true","validation":"!=('John','spiderman')","columnorder":2,"active":"true","tags":["Name","F\_Name","First\_N"]},{"a\_name":"last\_name","a\_datatype":"string","isNull":"false","std\_function":"name\_std","singlevalued":"true","validation":"!=('John','spiderman')","columnorder":3,"active":"true"},{"a\_name":"phone","a\_datatype":"string","isNull":"true","std\_function":"phone\_std","singlevalued":"false","columnorder":4,"active":"true"},{"a\_name":"source\_primaryid","a\_datatype":"string","isNull":"false","singlevalued":"true","columnorder":1,"active":"true"},{"a\_name":"email","a\_datatype":"string","isNull":"true","singlevalued":"false","columnorder":5,"active":"true","std\_function":"email\_std"},{"a\_name":"parsed\_address","lvl2\_att":[{"l2\_name":"line1","l2\_datatype":"string","l2\_isNull":"true","l2\_order":2,"active":"true"},{"l2\_name":"line2","l2\_datatype":"string","l2\_isNull":"true","l2\_order":3,"active":"true"},{"l2\_name":"line3","l2\_datatype":"string","l2\_isNull":"true","l2\_order":4,"active":"true"},{"l2\_name":"city","l2\_datatype":"string","l2\_isNull":"true","l2\_order":5,"active":"true"},{"l2\_name":"state","l2\_datatype":"string","l2\_isNull":"true","l2\_order":6,"active":"true"},{"l2\_name":"postalcode","l2\_datatype":"string","l2\_isNull":"true","l2\_order":7,"active":"true"},{"l2\_name":"country","l2\_datatype":"string","l2\_isNull":"true","l2\_order":8,"active":"true"},{"l2\_name":"type","l2\_datatype":"string","l2\_isNull":"true","l2\_order":1,"active":"true"}],"std\_function":"parseadd\_std","columnorder":6}],"no\_match\_threshold":35,"identical\_match\_threshold":75,"MLfileattributes":{"attributes":["first\_name","last\_name","phone","parsed\_address"],"delimiter":"|"}}

]

# Map Source File Attribute to Domain Attribute

* Flat file complex data into json format with configurable code using python.

**Configurable details for columns from – Can g**

[columns]    
    
multivalued\_fields = con\_email\_addr\_txt,con\_emplymnt\_emplyr\_nm    
  

reg\_fields =  record\_id,con\_cust\_num,con\_first\_nm,con\_midl\_nm,con\_last\_nm,con\_home\_phn\_num\_txt,con\_birth\_dt,con\_email\_addr\_txt,con\_emplymnt\_emplyr\_nm    
  

address\_field = address    
  

address\_subset = con\_pri\_addr\_ln\_1\_txt,con\_pri\_addr\_ln\_2\_txt,con\_pri\_addr\_ln\_3\_txt,con\_pri\_addr\_city\_nm,con\_pri\_addr\_state\_nm,con\_pri\_addr\_cntry\_nm,con\_pri\_addr\_pstl\_cd

**Sample Input data**

 record\_id,

[4609027,41423998,Richard,P,Barnes,wg3a@arrl.net,+1](mailto:4609027,41423998,Richard,P,Barnes,wg3a@arrl.net,+1) 410 437 3686,2/23/1951,,3641 Dorshire Ct, Edificio Los Robles, Chapinero,Pasadena,MD,USA,21122-6469

**Sample output data**

{

   "record\_id":4609027,

   "cust\_num":41423998,

   "fname":"Richard",

   "mname":"P",

   "lname":"Barnes",

   "phone":"+1 410 437 3686",

   "date\_of\_birth":"2\/23\/1951",

   "address":[

      {

         "line1":"3641 Dorshire Ct",

         "line2":"Edificio Los Robles",

         "line3":"Chapinero",

         "city":"Pasadena",

         "state":"MD"

         "country":"USA",

         "zip":"21122-6469",

      }

   ],

   "email":[

      "[wg3a@arrl.net](mailto:wg3a@arrl.net)"

   ],

   "org\_name":[

      "Honeywell International, Inc."

   ]

}

* Created the mapping .json file for source to domain mapping to make standardized column names across sources.

**Mapping .json file**

{    
        "record\_id": "record\_id",    
        "con\_cust\_num" : "cust\_num",    
        "con\_first\_nm": "fname",    
        "con\_midl\_nm": "mname",    
        "con\_last\_nm": "lname",    
        "con\_email\_addr\_txt": "email",    
        "con\_home\_phn\_num\_txt": "phone",    
        "con\_birth\_dt": "date\_of\_birth",    
        "con\_pri\_addr\_ln\_1\_txt": "line1",    
        "con\_pri\_addr\_ln\_2\_txt": "line2",    
        "con\_pri\_addr\_ln\_3\_txt": "line3",    
        "con\_pri\_addr\_city\_nm": "city",    
        "con\_pri\_addr\_state\_nm": "state",    
        "con\_pri\_addr\_cntry\_nm": "country",    
        "con\_emplymnt\_emplyr\_nm" : "org\_name",    
        "con\_pri\_addr\_pstl\_cd":"zip",    
        "address": "address"    
}

**Input Sample Data**

record\_id,con\_cust\_num,con\_first\_nm,con\_midl\_nm,con\_last\_nm,con\_email\_addr\_txt,con\_home\_phn\_num\_txt,con\_birth\_dt,con\_emplymnt\_emplyr\_nm,con\_pri\_addr\_ln\_1\_txt,con\_pri\_addr\_ln\_2\_txt,con\_pri\_addr\_ln\_3\_txt,con\_pri\_addr\_city\_nm,con\_pri\_addr\_state\_nm,con\_pri\_addr\_cntry\_nm,con\_pri\_addr\_pstl\_cd

[4609027,41423998,Richard,P,Barnes,wg3a@arrl.net,+1](mailto:4609027,41423998,Richard,P,Barnes,wg3a@arrl.net,+1) 410 437 3686,2/23/1951,,3641 Dorshire Ct, Edificio Los Robles, Chapinero,Pasadena,MD,USA,21122-6469

**Sample output data**

{

   "record\_id":4609027,

   "cust\_num":41423998,

   "fname":"Richard",

   "mname":"P",

   "lname":"Barnes",

   "phone":"+1 410 437 3686",

   "date\_of\_birth":"2\/23\/1951",

   "address":[

      {

         "line1":"3641 Dorshire Ct",

         "line2":"Edificio Los Robles",

         "line3":"Chapinero",

         "city":"Pasadena",

         "state":"MD"

         "country":"USA",

         "zip":"21122-6469",

      }

   ],

   "email":[

      "[wg3a@arrl.net](mailto:wg3a@arrl.net)"

   ],

   "org\_name":[

      "Honeywell International, Inc."

   ]

}