



MDG Data Load Design

Enterprise Master Data Technical Specifications Document (MD.070)

Last Saved: 08-06-2020 14:30:00

Last Saved by: Kumar, Rohit

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Author | Affected Sections | Description of Change | Date of Revision |
| 1.0 | Matthew Wright |  | Initial Version | 3/30/2020 |
| 1.1 | Karen Adair | 4, 5, 6, 7 | Updated high level and detail architecture diagrams; added logic for section 6; added open item | 4/20/2020 |
| 1.2 | Rohit Kumar | 7,8 | Added section for Unit Testing and load testing | 4/22/2020 |
|  |  |  |  |  |
|  |  |  |  |  |

Approvals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Author | Affected Sections | Description of Change | Date of Revision |
| 1.0 |  |  | Initial Version | 03/27/2016 |

Contents

[1. Background 5](#_Toc38448602)

[2. Scope 5](#_Toc38448603)

[3. Assumptions 5](#_Toc38448604)

[4. Technical Requirements - Emailer 6](#_Toc38448605)

[4.1.1. High level approach. 6](#_Toc38448606)

[4.1.2. Detail Process 7](#_Toc38448607)

[4.1.3. Code details 9](#_Toc38448608)

[4.1.4. Exception and Error Handling 9](#_Toc38448609)

[4.1.5. Program Files: 9](#_Toc38448610)

[4.1.6. Jobs and Schedule 9](#_Toc38448611)

[4.1.7. Restart Strategy 9](#_Toc38448612)

[4.1.8. Table Load Order: 10](#_Toc38448613)

[4.1.9. Stage Data: 10](#_Toc38448614)

[4.1.10. Program Files: 10](#_Toc38448615)

[4.1.11. Informatica Workflows: 10](#_Toc38448616)

[4.1.12. Backup 10](#_Toc38448617)

[4.1.13. Jobs and Schedule 10](#_Toc38448618)

[4.1.14. Restart Strategy 11](#_Toc38448619)

[5. Technical Requirements – Data Collection Form 12](#_Toc38448620)

[5.1.1. High level approach. 12](#_Toc38448621)

[5.1.2. Detail Process 12](#_Toc38448622)

[5.1.3. Code details 15](#_Toc38448623)

[5.1.4. Exception and Error Handling 15](#_Toc38448624)

[5.1.5. Table Load Order: 15](#_Toc38448625)

[5.1.6. Stage Data Transform and Master: 15](#_Toc38448626)

[5.1.7. Program Files: 15](#_Toc38448627)

[5.1.8. Restart Strategy 15](#_Toc38448628)

[6. Technical Requirements – Data Transformation and Mastering 17](#_Toc38448629)

[6.1.1. High level approach. 17](#_Toc38448630)

[6.1.2. Detail Process 17](#_Toc38448631)

[6.1.3. Code details 18](#_Toc38448632)

[6.1.4. Exception and Error Handling 18](#_Toc38448633)

[6.1.5. Table Load Order: 19](#_Toc38448634)

[6.1.6. Stage Data Transform and Master: 19](#_Toc38448635)

[6.1.7. Program Files: 19](#_Toc38448636)

[6.1.8. Informatica Workflows: 19](#_Toc38448637)

[6.1.9. Backup 19](#_Toc38448638)

[6.1.10. Jobs and Schedule 19](#_Toc38448639)

[6.1.11. Restart Strategy 19](#_Toc38448640)

**[7.](#_Toc38448641)****[Unit Testing](#_Toc38448641)** [19](#_Toc38448641)

[7.1.1. Unit Testing 19](#_Toc38448642)

**[8.](#_Toc38448643)****[Load testing](#_Toc38448643)** [20](#_Toc38448643)

[8.1.1. Load Testing 20](#_Toc38448644)

[9.](#_Toc38448645) **[Open and Closed Issues](#_Toc38448645)** [20](#_Toc38448645)

[9.1. Open and Closed Issues 20](#_Toc38448646)

[9.1.1. Open Issues 20](#_Toc38448647)

[9.1.2. Closed Issues 20](#_Toc38448648)

# Background

The Data Entry strategy described in this document is to address the below high level requirements

* Notification Process
* Data Collection
* Stage Data Transform and Master

In addition, the document describes the data entry form details, API Calls, technical process flow, development components & frequency

# Scope

* Contact applicable users for Data Entry via Emailer
* Collect Data via web application and store data in staging table
  + Metadata Data Space
  + External Application Mapping Dataset
* Transform and Master Data
  + Geography and Organization Data Space

# Assumptions

* Out of the box EBX API calls leveraged to GET and POST data stage table
* Manual uploads of data to be done to initiate stage table.
* Google API to be leveraged to geocode addresses.

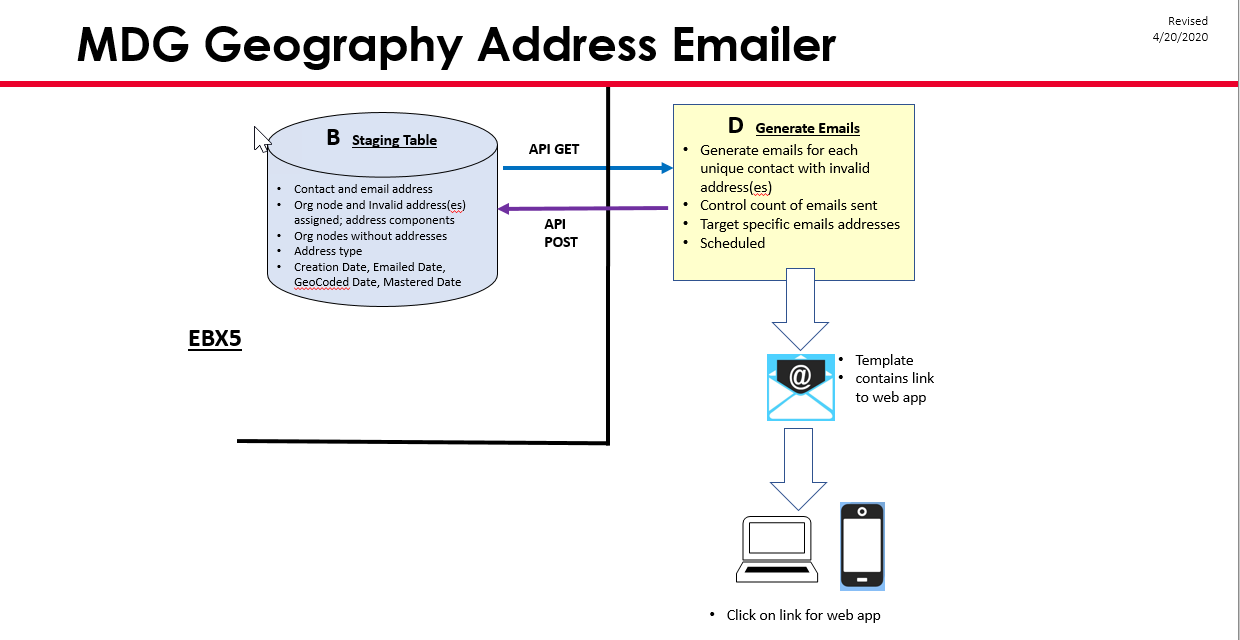
# Technical Requirements - Emailer

### High level approach.

#### 

High level architecture and flow of data between EBX5, the client, and Google Address.

### Detail Process





**Generic process detail**

**Emailer:**

1. The Geo Coder App will initiate a process daily that will check the count of emails to be sent out for users to do address validation from the MDG Emailer Config Table.
2. Based on the Config Table Count and Frequency, the Geo Coder App will query the MDG Address Staging Table to gather a set of unique email addresses that equals the Config Table Count and will only look at emails that have a Update Date greater than the Config’s Frequency
3. The Geo Coder App will leverage SMTP to email the users an email containing a link to the Address Validation Forms/Screens
4. The Geo Coder App will leverage an MDG API to POST Emailed Dates and Emailed Flags to all rows for the emailed users on the MDG Address Stage Table
5. The Geo Coder App will send out to admins the count of success and failed emails

### Code details

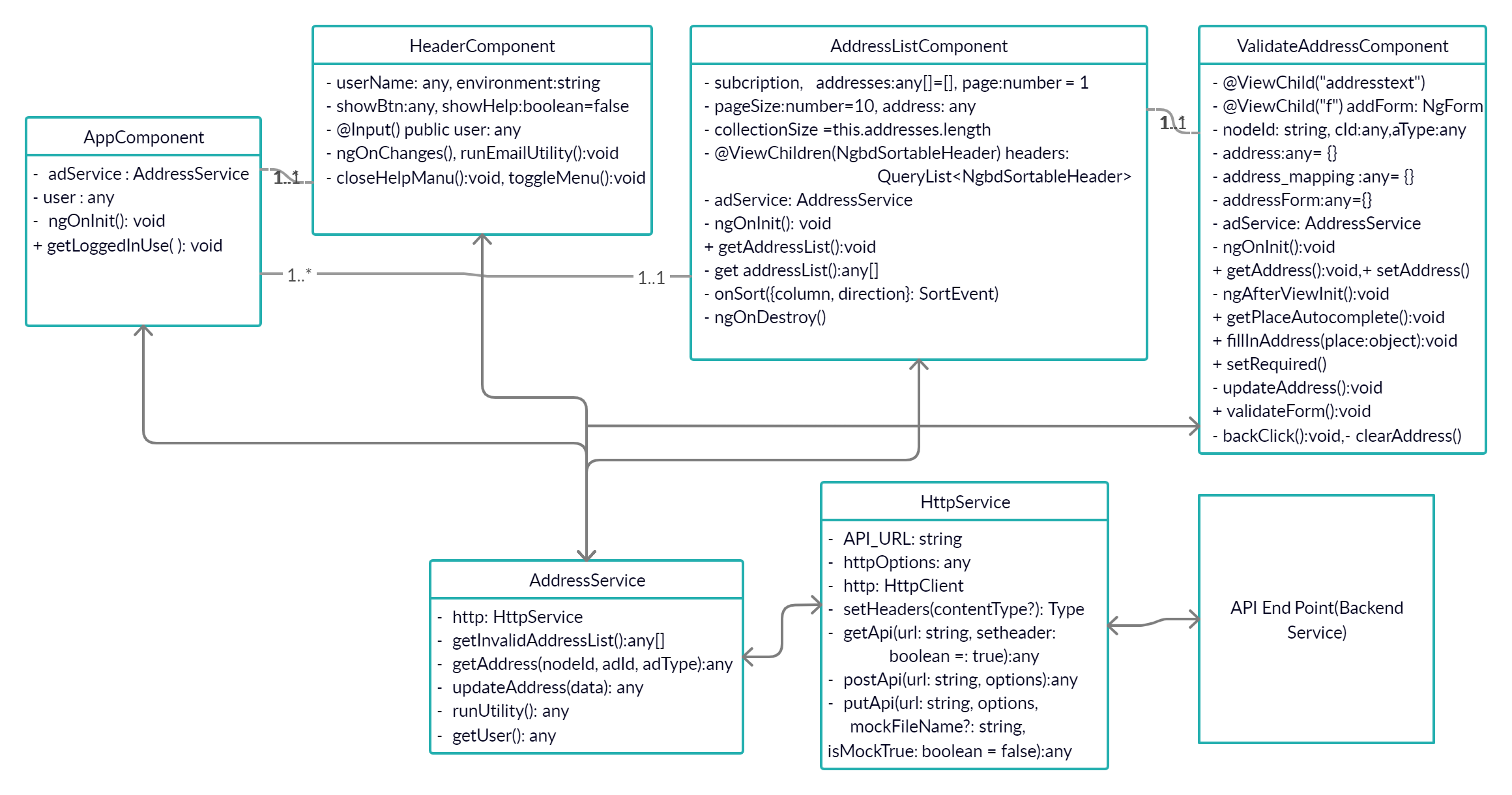
**Angular:**

To execute the **Emailer,** admin user will click on the “Email Utility” link on the header section of the application.

Flow of execution:

**“Email Utility” (HeaderComponent) => AddressService => runUtility(method) =>HttpService => postAPI(http://usphl0qmdmapv1.fss.aramark.com:7001/validator/run-email-utility)**

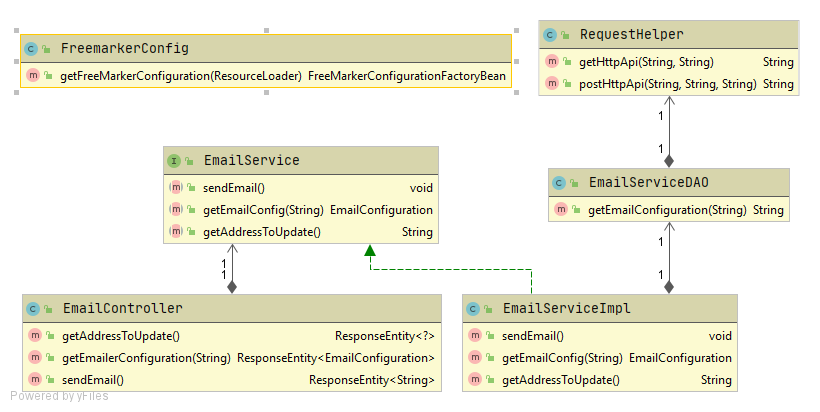
**Class Diagram:**

****

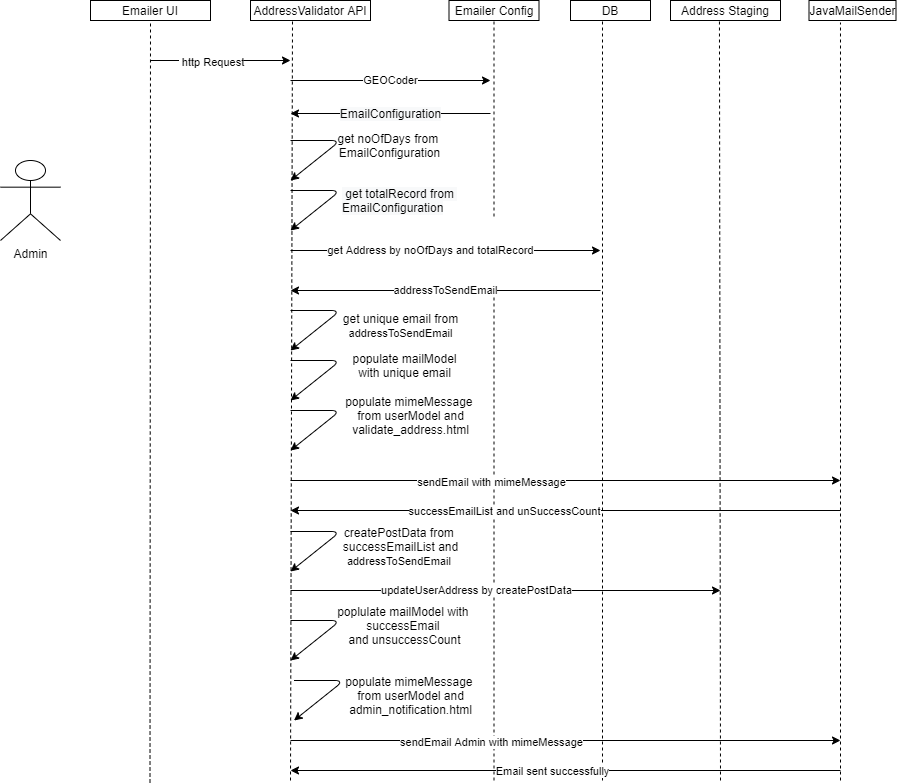
**Java:**

Spring-boot, lombok, Restful Services, Spring JPA(For DB Connectivity), JavaMailSender.

**UML Class Diagram:**

****

**Sequence Diagram:**



Flow of execution:

* **EmailController:sendEmail()=> EmailService:sendEmail() => EmailServiceImpl:sendEmail() => getEmailConfig() => DbService:getAddressToSendEmail(noOFDays, totalRecord) => mimeMessage()=> JavaMailSender: sendUserMail() => DataServiceDAO:updateUserAddress() => EmailServiceImpl: sendAdminMail().**
* **EmailController:getEmailerConfiguration()=>EmailService:getEmailConfig()=> EmailServiceDAO:getEmailConfiguration() => RequestHelper.getHttpApi() => JsonUtils.write()=> EmailConfiguration.class.**
* **EmailController:getAddressToUpdate()=>EmailService:getAddressToUpdate() =>getEmailConfig(systemName)=> DbService.getAddressToSendEmail(noOFDays, totalRecord)=> JsonUtils.read()=> String.class.**

1. **Email Process APIs:** Here we are exposing three rest end points for completing mailer functionality:
2. A rest api call which will return email configuration.

**URL:**http://usphl0qmdmapv1.fss.aramark.com:7001/validator/emailerConfig?systemName=GEOCoder

**Method:** GET

**Request:** String (systemName)

**Response:**

{

"rows": [

{

"label": "GEOCoder",

"details": “String”,

"content": {

"system": {

"content": "GEOCoder"

},

"recipientCount": {

"content": "2"

},

"frequency": {

"content": "2"

}

}

}

],

"pagination": {}

}

1. A Get rest api call for getting list of addresses .

**URL:** <http://usphl0qmdmapv1.fss.aramark.com:7001/validator/getAddressToUpdate>

**Method:** GET

**Request: NA**

**Response:**

[

{

"contactId":"10624",

"primaryEmail":"fazio-stacy@aramark.com",

"origOrgNodeId":"323416",

"addrTypeId":"2"

},

{

"contactId":"10624",

"primaryEmail":"fazio-stacy@aramark.com",

"origOrgNodeId":"323416",

"addrTypeId":"2"

}

]

1. A rest call which will run email utility for sending emails to user and admin.

**URL:** http://usphl0qmdmapv1.fss.aramark.com:7001/validator/run-email-utility

**Method:** POST

**Request:** NA

**Response:**  “Email sent successfully!!”.

### Exception and Error Handling

#### Warning/Error Conditions:

| **Condition** | **Error Message** | **Error / Warning** | **Program Action** |
| --- | --- | --- | --- |
| Only Admin user is Authorized to initiate Email endpoint | Admins email not found | NotAuthorizedException | Email endpoint cannot be triggered by any-other user accept Admin. Status code 401 |
| If Email template not given in expected format. | ERROR OCCURRED in Template OR Message Creation OR | TemplateException | Logs will be printed with details.  Status-code 500 |
| If Connection failed while connecting with Mail host. | Unable to send email | MailException | Logs will be printed with details.  Status code = 500 |
| If Geo Coder App is not able to connect with ebxdataservice | ERROR OCCURRED in Template OR Message Creation OR | IOException | Logs will be printed with details.  Status code = 500 |

### Program Files:

|  |  |  |  |
| --- | --- | --- | --- |
| Server | File name and Path | TFS location | Description |
| Bitbucket | <https://bitbucket.org/kumar-rohit1/aramark-addressmap>.git | NA | Complete Code is committed in to the bitbucket. |

### Jobs and Schedule

N/A

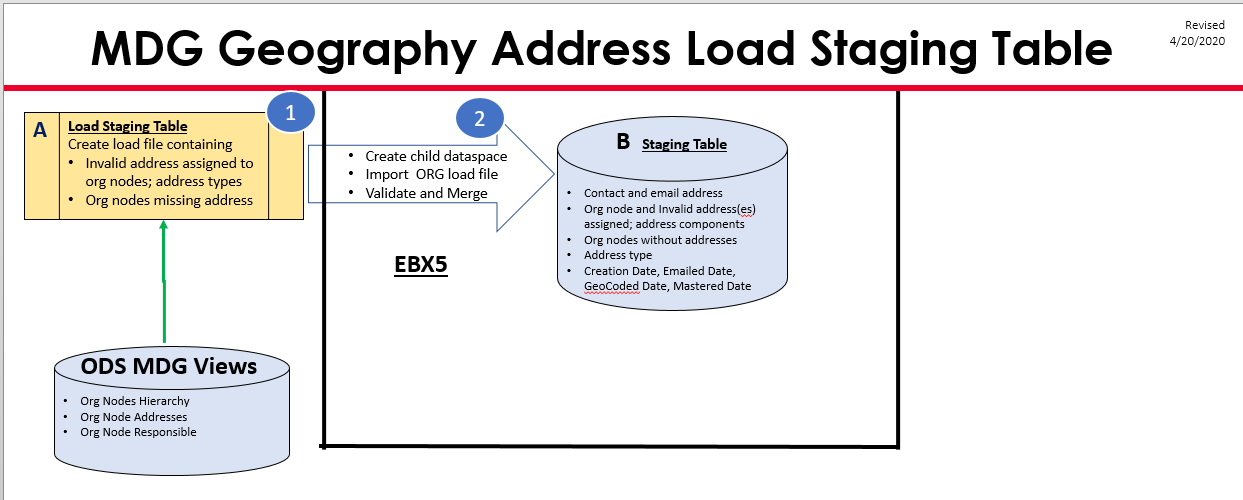
### Restart Strategy

To restart/rerun:

|  |  |  |
| --- | --- | --- |
| **Condition** | **Resolution** | **Comments** |
| Manual Rerun available to Admins within WebApp | Clicking the Emailer button will manually rerun this process |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Table Load Order:

### Stage Data:



### Program Files:

|  |  |  |  |
| --- | --- | --- | --- |
| Server | File name and Path | TFS location | Description |
|  |  |  |  |

### Informatica Workflows:

### Backup

### Jobs and Schedule

N/A

### Restart Strategy

To restart/rerun:

|  |  |  |
| --- | --- | --- |
| **Condition** | **Resolution** | **Comments** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

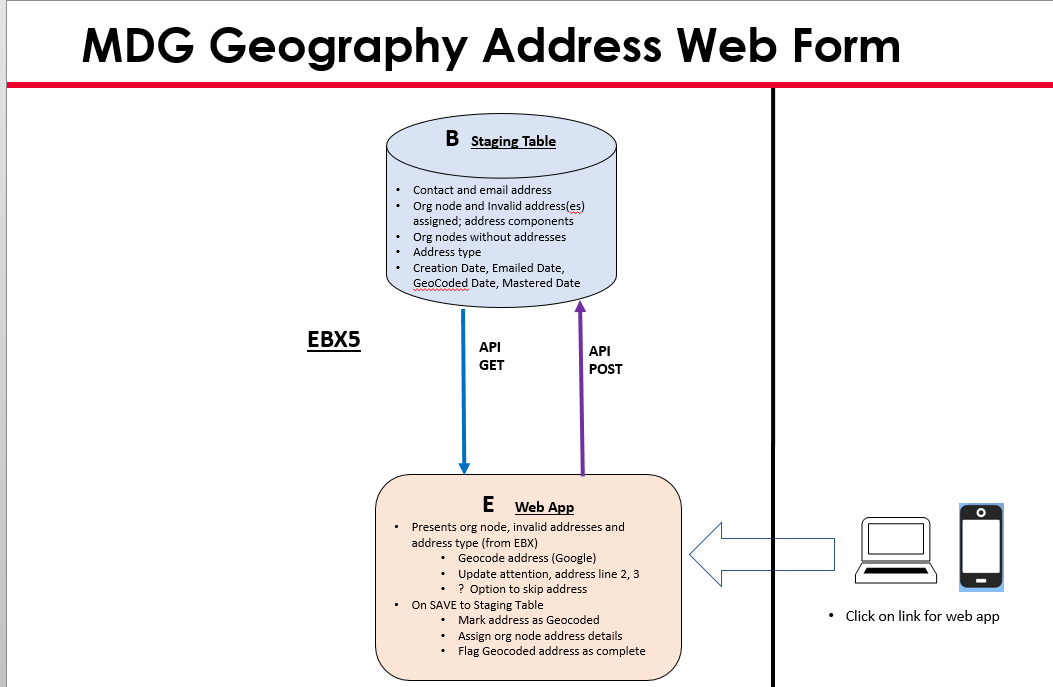
# Technical Requirements – Data Collection Form

### High level approach.



High level architecture and flow of data between EBX5, the client, and Google Address.

### Detail Process





**Generic process detail**

**Address Collection Forms:**

1. Upon clicking on the link within the email the user will be brought a screen showing all of the addresses they are required to validate. The list of Org Node Addresses to be validated is populated from the MDG Address Staging Table based on the logged in users email
2. The User clicks the link to be brought to the validation form for each Org Node Address. The Invalid address is displayed on the screen by leveraging the MDG Address Staging Table for that invalid Org Node Address.
3. The user fills out the autocomplete address field, attention to, address line 2 and address line 3. The autocomplete address field leverages Google API to geocode the address. The other fields are optional.
4. Upon selecting a Google Autocompleted Address, the address components are displayed on the screen in the Place ID, Street Number, Street Name, City, State, Postal Town, Postal Code fields. (If no street number, premise is put as street number. If not City, Postal Town is put in City and Postal Town fields)
5. Upon a user submitting the form, all address components are Posted via MDG API to the MDG Stage Table. The Geocoded Date and Geocoded Flag are also updated during this call.

### Code details

The following frameworks were used to develop ui screens/forms for users to input address data.

* **Angular-**

**Show logged-in user info=>**

To show the logged-in user info, following component has been used:

**app.component.ts => ngOnInit() => getLoggedInUser() => address.service.ts => getUser =>**

**http.service.ts => getAPI (http://usphl0qmdmapv1.fss.aramark.com:7001/validator)**

Above API will return the logged-in user name & other related info.

**Addresses to validate:** Upon clicking on the link within the email the user will be brought a screen showing all of the addresses they are required to validate. To get the list of address, following component & services is in use.

**Component -** address-list.component

**Service-** address.service.ts, http.service.ts

**Service Method-** getInvalidAddressList

**Code execution flow-**

**address-list.component.ts => ngOnInit=> getAddressList => address.service.ts =>**

**getInvalidAddressList =>http.service.ts => getAPI**

**(http://usphl0qmdmapv1.fss.aramark.com:7001/validator/addresses)**

**Validate address form:** The User clicks the link to be brought to the validation form for each Org Node Address. The Invalid address is displayed on the screen by leveraging the MDG Address Staging Table for that invalid Org Node Address.

On the validate address form following component & services is in use,

**Component -** validate-address.component

**Service-** address.service.ts, http.service.ts

**Service Method-** getAddress(), updateAddress()

**Code execution flow- Get Address Detail:**

**validate-address.component.ts => ngOnInit=> getAddress()=>**

**address.service.ts=> getAddress(nodeId, cId, aType)=>http.service.ts => getAPI**

**(http://usphl0qmdmapv1.fss.aramark.com:7001/validator/address)**

Above API will return the selected invalid address details.

**Code execution flow- Update Address:**

Once user is selected the valid address with the help of GOOGLE Auto-complete place API,

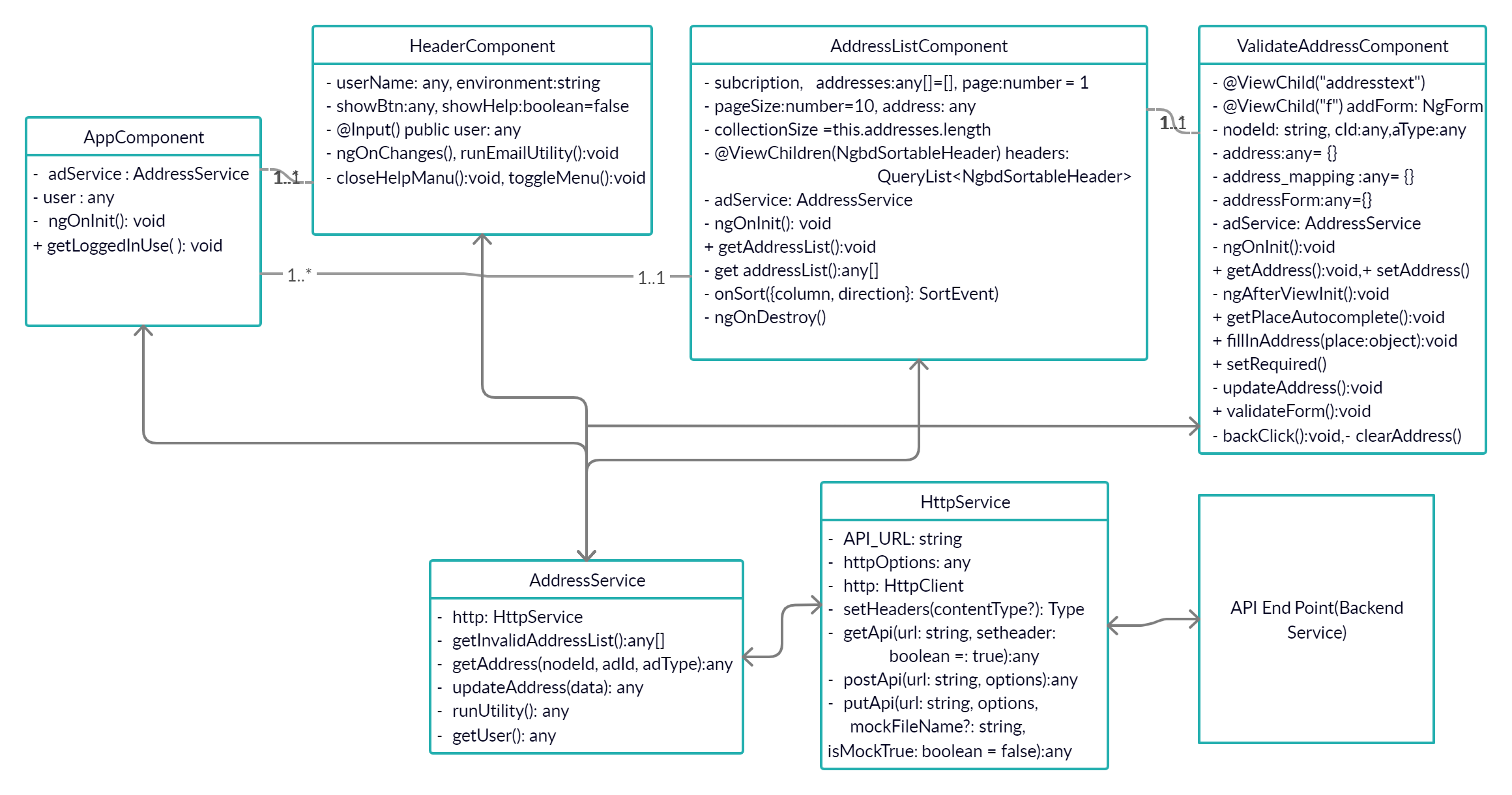
update the address with the help of following flow:

**validate-address.component.ts =>updateAddress()=> address.service.ts=> updateAddress**

**=> http.service.ts => postAPI**

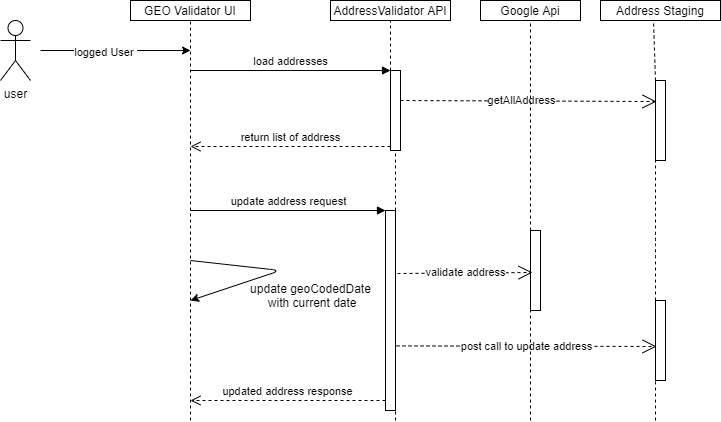
**(http://usphl0qmdmapv1.fss.aramark.com:7001/validator/address)**

**Class Diagram:**

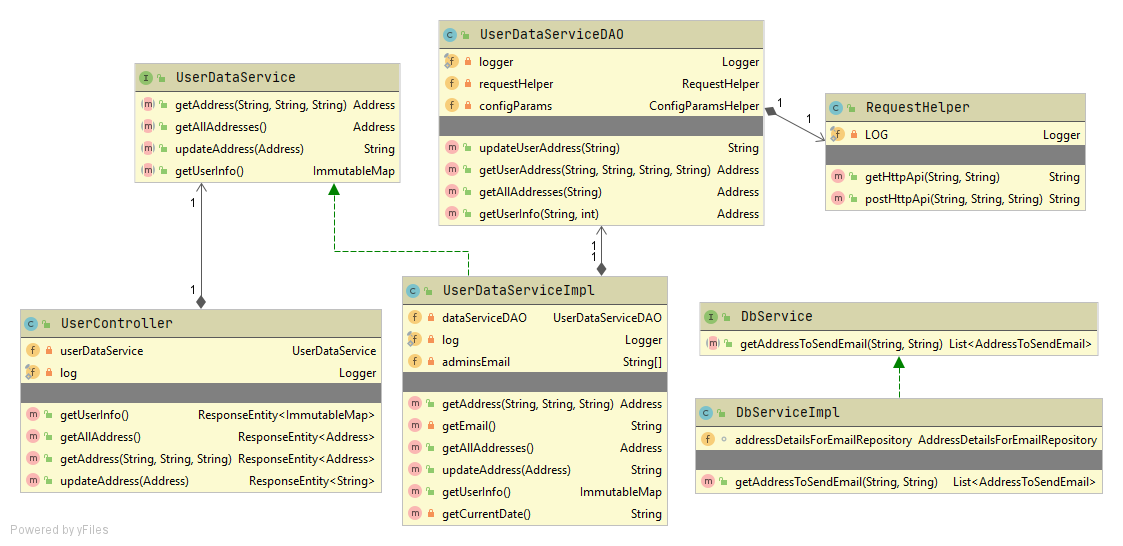


* **Java/EBX5**
* **Java:** **Spring-boot, Restful Services.**

**Sequence Diagram:-**

****

**UML Diagram: -**

****

**Get user info=>**

To get user info, following component has been used:

**Controller-** UserController

**Service-** UserDataService,UserDataServiceDAO

**Service Method-** getUserInfo

**Helper-** RequestHelper

**Helper Method-**getHttpApi

**Utils-**JsonUtils

**Utils Method-** write

**Code execution flow-**

* **UserController:getUserInfo() => UserDataService:getUserInfo() => UserDataService:getEmail() =>SecurityContextHolder:getContext():getAuthentication() => UserDataServiceDAO:getUserInfo() => RequestHelper:getHttpApi => JsonUtils.write => Address.class**

Below API will return the user name & other related info.

**URL:**http://usphl0qmdmapv1.fss.aramark.com:7001/validator/

**Method:** GET

**Request:** NA

**Response:**

{

"username": "Sara Bintrim",

"isA": true

}

**Get all Address =>**

To get all address, following component has been used:

**Controller-** UserController

**Service-** UserDataService,UserDataServiceDAO

**Service Method-** getAllAddresses

**Helper-** RequestHelper

**Helper Method-**getHttpApi

**Utils-**JsonUtils

**Utils Method-** write

**Code execution flow-**

* + **UserController:getAllAddress() => UserDataService:getAllAddresses() => UserDataService:getEmail() =>SecurityContextHolder:getContext():getAuthentication() => UserDataServiceDAO:getAllAddresses() => RequestHelper:getHttpApi => JsonUtils.write => list< Address.class>**

**URL:**http://usphl0qmdmapv1.fss.aramark.com:7001/validator/addresses

**Method:** GET

**Request:** NA

**Response:**

{

"rows": [

{

"label": "10051 - 326028 - 2",

"details": "http://usphl0qmdmapv1.fss.aramark.com:7001/ebx-dataservices/rest/data/v1/BMetaData-Hub/External\_App\_Mapping/root/address/addressStaging/10051\_7C326028\_7C2",

"content": {

"contactId": {

"content": "10051"

},

"origOrgNodeId": {

"content": "326028"

},

"addrTypeId": {

"content": "2"

},

"eMailedDate": {},

"placeID": {},

"streetNumber": {},

"streetName": {},

"cityLocality": {},

"postalTown": {},

"stateProvince": {},

"county": {},

"neighborhood": {},

"establishment": {},

"adminArea3": {},

"postalCD": {},

"country": {},

"formattedAddress": {},

"latitude": {},

"longitude": {},

"source": {

"selector": "http://usphl0qmdmapv1.fss.aramark.com:7001/ebx-dataservices/rest/data/v1/BMetaData-Hub/External\_App\_Mapping/root/address/addressStaging/10051\_7C326028\_7C2/source?selector=true"

},

"geoCodedDate": {},

"nodeMasterDate": {},

"createdDateTime": {

"content": "2020-03-17T14:52:23.887"

},

"addressLine2": {},

"addressLine3": {},

"attnLine": {},

"geoAddressId": {},

"contactEmail": {

"content": "adair-karen@aramark.com"

},

"f\_name": {

"content": "Robert"

},

"l\_name": {

"content": "Johnson"

},

"origNodeAddrId": {

"content": "0"

},

"orig\_org\_node\_type": {

"content": "Location"

},

"orgValue": {

"content": "L00010943"

},

"orig\_org\_node\_name": {

"content": "Corpus Christi Conc Cabaniss Baseball"

},

"parentOrgNode": {

"content": "62716"

},

"parent\_org\_node\_type": {

"content": "Profit Center"

},

"parentOrgValue": {

"content": "400187500"

},

"parent\_org\_node\_name": {

"content": "Corpus Christi ISD Concessions"

},

"addrTypeDesc": {

"content": "Shipping"

},

"origAddrId": {

"content": "0"

},

"origFrmtAddress": {},

"origCounty": {}

}

}

],

"pagination": {}

}

**Get Address =>**

To get address, following component has been used:

**Controller-** UserController

**Service-** UserDataService,UserDataServiceDAO

**Service Method-** getAddress

**Helper-** RequestHelper

**Helper Method-**getHttpApi

**Utils-**JsonUtils

**Utils Method-** write

**Code execution flow-**

* + **UserController:getAddress() => UserDataService:getAddress() => UserDataService:getEmail() =>SecurityContextHolder:getContext():getAuthentication() => UserDataServiceDAO:getAddress() => RequestHelper:getHttpApi => JsonUtils.write => Address.class**

**URL:**http://usphl0qmdmapv1.fss.aramark.com:7001/validator/address

**Method:** GET

**Request:** NA

**Response:**

{

"rows":[

{

"label":"10051 - 326028 - 2",

"details":"http://usphl0qmdmapv1.fss.aramark.com:7001/ebx-dataservices/rest/data/v1/BMetaData-Hub/External\_App\_Mapping/root/address/addressStaging/10051\_7C326028\_7C2",

"content":{

"contactId":{

"content":"10051"

},

"origOrgNodeId":{

"content":"326028"

},

"addrTypeId":{

"content":"2"

},

"eMailedDate":{

},

"placeID":{

},

"streetNumber":{

},

"streetName":{

},

"cityLocality":{

},

"postalTown":{

},

"stateProvince":{

},

"county":{

},

"neighborhood":{

},

"establishment":{

},

"adminArea3":{

},

"postalCD":{

},

"country":{

},

"formattedAddress":{

},

"latitude":{

},

"longitude":{

},

"source":{

"selector":"http://usphl0qmdmapv1.fss.aramark.com:7001/ebx-dataservices/rest/data/v1/BMetaData-Hub/External\_App\_Mapping/root/address/addressStaging/10051\_7C326028\_7C2/source?selector=true"

},

"geoCodedDate":{

},

"nodeMasterDate":{

},

"createdDateTime":{

"content":"2020-03-17T14:52:23.887"

},

"addressLine2":{

},

"addressLine3":{

},

"attnLine":{

},

"geoAddressId":{

},

"contactEmail":{

"content":"adair-karen@aramark.com"

},

"f\_name":{

"content":"Robert"

},

"l\_name":{

"content":"Johnson"

},

"origNodeAddrId":{

"content":"0"

},

"orig\_org\_node\_type":{

"content":"Location"

},

"orgValue":{

"content":"L00010943"

},

"orig\_org\_node\_name":{

"content":"Corpus Christi Conc Cabaniss Baseball"

},

"parentOrgNode":{

"content":"62716"

},

"parent\_org\_node\_type":{

"content":"Profit Center"

},

"parentOrgValue":{

"content":"400187500"

},

"parent\_org\_node\_name":{

"content":"Corpus Christi ISD Concessions"

},

"addrTypeDesc":{

"content":"Shipping"

},

"origAddrId":{

"content":"0"

},

"origFrmtAddress":{

},

"origCounty":{

}

}

}

],

"pagination":{

}

}

**Update Address =>**

To update address, following component has been used:

**Controller-** UserController

**Service-** UserDataService,UserDataServiceDAO

**Service Method-** updateAddress

**Helper-** RequestHelper

**Helper Method-**postHttpApi

**Utils-**JsonUtils

**Utils Method-** read

**Code execution flow-**

* + **UserController:updateAddress() => UserDataService:updateAddress() => UserDataServiceDAO:getAddress() => RequestHelper:postHttpApi()=> JsonUtils.read()=> String.class**

**URL:**http://usphl0qmdmapv1.fss.aramark.com:7001/validator/address

**Method:** POST

**Request:**

{

"rows":[

{

"content":{

"addressLine2":{

"content":""

},

"addressLine3":{

"content":""

},

"addrTypeDesc":{

"content":"Physical"

},

"addrTypeId":{

"content":"1"

},

"adminArea3":{

"content":null

},

"attnLine":{

"content":""

},

"cityLocality":{

"content":"Sanger"

},

"contactEmail":{

"content":"kumar-rohit@aramark.ca"

},

"contactId":{

"content":"10714"

},

"country":{

"content":"US"

},

"county":{

"content":"Fresno County"

},

"createdDateTime":{

"content":"2020-03-17T14:52:24.067"

},

"eMailedDate":{

"content":"2020-04-23"

},

"establishment":{

"content":null

},

"f\_name":{

"content":"Yanet"

},

"formattedAddress":{

"content":"1239 Faller Ave, Sanger, CA 93657, USA"

},

"geoAddressId":{

},

"geoCodedDate":{

"content":"2020-05-21T00:00:00.000"

},

"l\_name":{

"content":"Vergara"

},

"latitude":{

"content":"36.698819"

},

"longitude":{

"content":"-119.5476983"

},

"neighborhood":{

"content":null

},

"nodeMasterDate":{

},

"orgValue":{

"content":"L00036526"

},

"orig\_org\_node\_name":{

"content":"Sapient Miami Cafeteria"

},

"orig\_org\_node\_type":{

"content":"Location"

},

"origAddrId":{

"content":"0"

},

"origCounty":{

},

"origFrmtAddress":{

"content":""

},

"origNodeAddrId":{

"content":"0"

},

"origOrgNodeId":{

"content":"338210"

},

"parent\_org\_node\_name":{

"content":"Sapient Miami"

},

"parent\_org\_node\_type":{

"content":"Profit Center"

},

"parentOrgNode":{

"content":"151894"

},

"parentOrgValue":{

"content":"000008707"

},

"placeID":{

"content":"ChIJ09S5YgP5lIARDbnixSUbxrc"

},

"postalCD":{

"content":"93657"

},

"postalTown":{

"content":null

},

"source":{

"content":"Google",

"selector":"http://usphl0qmdmapv1.fss.aramark.com:7001/ebx-dataservices/rest/data/v1/BMetaData-Hub/External\_App\_Mapping/root/address/addressStaging/10714\_7C338210\_7C1/source?selector=true"

},

"stateProvince":{

"content":"CA"

},

"streetName":{

"content":"Faller Avenue"

},

"streetNumber":{

"content":"1239"

}

},

"details":"http://usphl0qmdmapv1.fss.aramark.com:7001/ebx-dataservices/rest/data/v1/BMetaData-Hub/External\_App\_Mapping/root/address/addressStaging/10714\_7C338210\_7C1",

"label":"10714 - 338210 - 1"

}

]

}

**Response:** 200 .

### Exception and Error Handling

#### Warning/Error Conditions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Condition** | **Error Message** | **Error / Warning** | **Program Action** |
| When the user is not authorised | Unauthorised | 401 | User should authorise with a valid mail address |
| When the service is not found | NotFoundException | 404 | Service not found |
| When the required service is not available | IOException | 503 | Wait for end server to respond |
|  |  |  |  |

### Table Load Order:

### Stage Data Transform and Master:

### Program Files:

|  |  |  |  |
| --- | --- | --- | --- |
| Server | File name and Path | TFS location | Description |
| Bitbucket | <https://bitbucket.org/kumar-rohit1/aramark-addressmap>.git | NA | Complete code is available in bitbucket. |
|  |  |  |  |

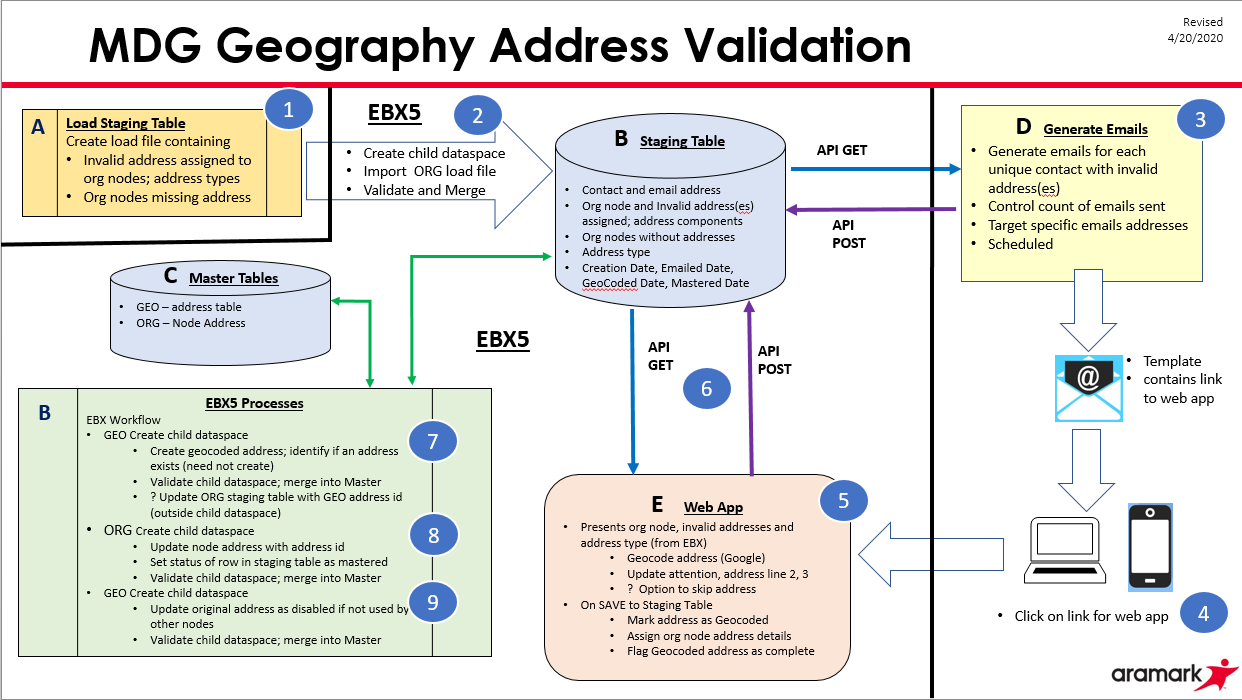
### Restart Strategy

To restart/rerun:

|  |  |  |
| --- | --- | --- |
| **Condition** | **Resolution** | **Comments** |
| Manual Rerun available to Admins within WebApp | Clicking the Emailer button will manually rerun this process |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Technical Requirements – Data Transformation and Mastering

### High level approach.



High level architecture and flow of data from source to target datasets in EBX5. Refer to tables and processes within the ***green*** Boxes.

### Detail Process

**Generic process detail**

1. Master address -> GEO Address table
   1. Identify which addresses exist or do not exist in GEO Address table
      1. Select
         1. from staging table, original Org Node ID, original address ID, address type id, place ID, street number, street name, city/locality, postal town, county, state/province, postal code, country
            1. All rows in staging tbl where place ID, GEO ID, and GEO coded date is null
         2. From Address table, address ID, source, for existing addresses
            1. for matching row in GEO Address table based on place ID (if exists) and is active
      2. If Original address id is pulled, address exists
   2. Create GEO address load file and master addresses
      1. For distinct set of new addresses that do not exist in Address table (no match on place ID)
      2. Source = 'GEO CODER'
   3. Select
      1. Address ID from address table for all newly mastered addresses, based on source of GEO CODER and?? (anything else?), to identify the newly created addresses
      2. Address id from above 1.a.i , for matching row in GEO Address table based on place ID match and is active
2. Create a staging table load file and master
   1. To update the GEO address id and GEO Coded date columns using results from 1.c.
3. Master node address -> node address table
   1. Select from staging table
      1. Original org node id, address type id, geo address id, where GEO address id is present and Node Address Mastered Date is null.
   2. \*\* ,may need to create a file to inactivate the invalid addresses already assigned to an org node
      1. Select Original org node id, address type id, geo address id, original node address id, where GEO address id is present.
   3. \*\* Create Node Address load file into child dataspace
      1. \*\* For 2.b first, if needed to inactivate invalid addresses on the node resp table
   4. Create Node Address load file into same child dataspace
      1. For 2.a
   5. Merge child dataspace
4. Create a staging table load file and master
   1. To update the Node Address mastered date for all rows updated in #3.

### Code details

The following frameworks were used…...

**Java/EBX5:**

### Exception and Error Handling

#### Warning/Error Conditions:

| **Condition** | **Error Message** | **Error / Warning** | **Program Action** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Table Load Order:

### Stage Data Transform and Master:

### Program Files:

|  |  |  |  |
| --- | --- | --- | --- |
| Server | File name and Path | TFS location | Description |
|  |  |  |  |

### Informatica Workflows:

### Backup

### Jobs and Schedule

### Restart Strategy

To restart/rerun:

|  |  |  |
| --- | --- | --- |
| **Condition** | **Resolution** | **Comments** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# **Unit Testing**

### Unit Testing

## We performed manual unit testing:

**Email Process:**

1. In valid case when we have valid email config, email list and address data to update , then successfully sent emails then we got response:

201: "Email sent successfully!!".

1. When we not login with admin account, then we will get error response:

401: "admins email not found"

1. When we are not able to send email with following conditions:- address data from DB not found, error connecting to mail server then we will get error response:

500: "Unable to send email".

**Address Process:**

1. In valid case when we have valid email the apis get address, get all address and update address give 200 response .
2. When we not login with account, then we will get error response: 401: "email not found".
3. When EBX connection failure then we will get error response: 422: ”Unprocessable entity”.

# **Load testing**

### Load Testing

# **Open and Closed Issues**

## Open and Closed Issues

1. Add open issues that you identify while writing or reviewing this document to the open issues section. As you resolve issues, move them to the closed issues section and keep the issue ID the same. Include an explanation of the resolution.  
     
   When this deliverable is complete, any open issues should be transferred to the project- or process-level Risk and Issue Log (PJM.CR.040) and managed using a project level Risk and Issue Form (PJM.CR.040). In addition, the open items should remain in the open issues section of this deliverable, but flagged in the resolution column as being transferred.

### Open Issues

| ID | Issue | | Resolution | | Responsibility | Target Date | | Impact Date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  |  | |  |
| 1 | International addresses: three additional address attributes added | | **Modify code to populate three new column in staging table, if returned from Google: establishment, administrative level 3, neighborhood** | | New requirement | 4/28/2020 | | Available for DEV deployment |
| 2. | emailerConfig: query param as systemName added | | Add a new query param systemName based on that value it will fetch records default value is GeoCoder if systemName not pass | | New requirement | 4/28/2020 | | Available for DEV deployment |
| 3 | customize swagger UI: added aramark logo and api-catalog | | Added aramark logo image and append api-catalog.html in the swagger url | | New requirement | 4/28/2020 | | Available for DEV deployment |
|  |  | |  | |  |  | |  |
|  |  | |  | |  | |  |  |
|  |  |  | |  | |  | |  |

### Closed Issues

| ID | Issue | | Resolution | | Responsibility | Target Date | | Impact Date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  |  | |  |
|  |  | |  | |  |  | |  |
|  |  | |  | |  |  | |  |
|  |  | |  | |  |  | |  |
|  |  | |  | |  |  | |  |
|  |  | |  | |  | |  |  |
|  |  |  | |  | |  | |  |
|  |  |  | |  | |  | |  |