1 Document Control

1.1 Change Policy Document Management

Table 1. Revision History

Version	Status	Date	Author	Changes/Justifications
0.1	Draft		Govind Shukla	Baseline version

Table 2. Document Properties

Item	Details
Document Title	File Upload Processing
Template Author	
Published Date	
Document ID	

Table 3. Glossary and Definitions

Acronym		Description

File Upload and processing,

Triggers

- The batch job can be initiated on demand or through external service call
- The batch can be scheduled at a defined frequency (interval or Daytime of the day)

Step - 1

1. The job would poll a location to detect the file with a specified mask in the filename
2. Alternatively, the file would be available in Blob column of a file upload table with processing status as RFP and record status as BAA – ready for Processing
3. For processing based on file in location, a record is inserted into compensationsummary table with status as RFP and BAA and entire file is loaded into a blob column
4. The job would read the file from the compensationsummary table - Blob column and mark the processing status as FUV – File under processing
5. Application performs data validation on the records,
    a. In case the header is not correct,
        i. file is rejected immediately with reject reason and reject code in the compensationsummary table
        ii. processing status is marked is FPF – File processing Failed
    b. In case the no. of error in content of file(child record) is beyond a configurable threshold, entire file is rejected, with reject code and reject reason
    c. Individual records are validated for data type inline with Jhipster generated entities based on the validation defined in the entities
    d. For each record in the file
        i. In case successfully validated, the record is inserted into child table – CompensationDetails with
            1. Recordstatus as BAU
            2. Processing status as SFU
        ii. In case record fails due to data validation issue, it is inserted into

compensationdetail with

1. Recordstatus as BAU
2. Processing status as SFF

6. For successful load, the application updates the table - CompensationSummary
    a. Total amount
    b. Total no of records
    c. Record successfully loaded
    d. Records failed
    e. Processing status is changed to FUS – File Uploaded successfully

Technical considerations:

- The application to support upload of multiple files either through multiple threads or instances of the job or any other suggested mechanism as required, this should be a configurable parameter
- The application to be resilient in terms of processing, in case of failure, the job should be able to pick up for last point and complete the process
- Admin dashboard to be able to review and monitor the batch job (Spring Cloud Data flow)
- Duplicate check is based on reference field in the upload file

Section interface for payments

For each record in payment table with processingstatus as QFI or PFR

- Fire an interface
    - if response success,
        - update the payment table status = PFS
        - update employee table, allocateamount = allocateamount-payamount
    - else

- update payment table status = PFR

Technical Considerations

1. Reuse Jhipster code as much as possible
2. Preference for JPA based operations
3. MaxTotalThread = 10
4. MaxthreadPerEmployee=2
5. -MaxRetries = 3 (with exponential backoff)
6. Request response pattern to be inserted into paymwentintf table for each call
7. the call to API would be a rest call or SOAP call

Sample Rest Request and response

Sample Request	Sample Response

{
"requestId":"XYZD",
"requestDateTime":"20201225155634",
"languageCode":"EN",
"sourceChannel":"5",
"sourceUserName":"GCEMP",
"sourceEnv":"DEV",
"serviceName":"initiateEmpPayment",
"entityCode":"456353",
"message":{"<Information from the table>"}
}
	{
	"requestId":"XYZD",
	"responseDateTime":"20201225155634",
	"lang":"EN",
	"serviceName":" initiatePayment ",
	"customerCode":"456353",
	"serviceStatus":"S",
	"serviceStatusCode":"2003",
	"serviceStatusDesc":"Successfully received",
	"message":{
	"payReference":"3243200020000087",
	"Status":"R",
	"StatusCode":"0012",
	"StatusDesc":"Payment value exceeds the balance for the employee record"
	}
	}

1. Requestid is generated using ${=import java.text.SimpleDateFormat ; UUID.randomUUID().toString()}
2. For Soap Based transaction, the message header should be inline with rest header mentioned above