***EQUIFAX***

CMS Fusion

**INPUT Technical Flow**

**DOCUMENTATION**

Table of Contents

[1. UI 3](#_Toc508010486)

[1.1 Input Menu 3](#_Toc508010487)

[1.2 Import New File 3](#_Toc508010488)

[1.3 Save/Continue 3](#_Toc508010489)

[1.4 Import Layout 4](#_Toc508010490)

[1.5 Submit 4](#_Toc508010491)

[1.6 Input Zip Codes 4](#_Toc508010492)

[1.6.1 SUBMIT to Oracle DB 4](#_Toc508010493)

[1.6.2 SUBMIT to Submit to REST 4](#_Toc508010494)

[1.7 UI Validation 5](#_Toc508010495)

[2. REST 6](#_Toc508010496)

[3. CORE Application 8](#_Toc508010497)

[3.1 FilePrepareWorkItemProcessor 8](#_Toc508010498)

[3.2 GPWorkItemProcessor 8](#_Toc508010499)

[3.3 ZipWorkItemProcessor 9](#_Toc508010500)

|  |  |
| --- | --- |
| **CREATED BY** | **UST Global** |
| **CREATED ON** | **11/20/2017** |
| **VERIFIED BY** |  |

**Revision History**

| **Revision** | **Author** | **Date** | **Status and Description** |
| --- | --- | --- | --- |
| 1.0 | UST Global | 11/20/2017 | Initial Draft |
| 1.1 | UST Global | 12/22/2017 | Modified based on review comments |
|  |  |  |  |

**Updates**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Story** | **Requirement** | **Section** | **Modified by** |
| 27/02/2018 | Sprint58(CF2-2621) | Error w.r.t Length validations for the DOB,DOB delimited | [2.7 UI Validation](#UIValidation) | Varnika Priydarshini |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

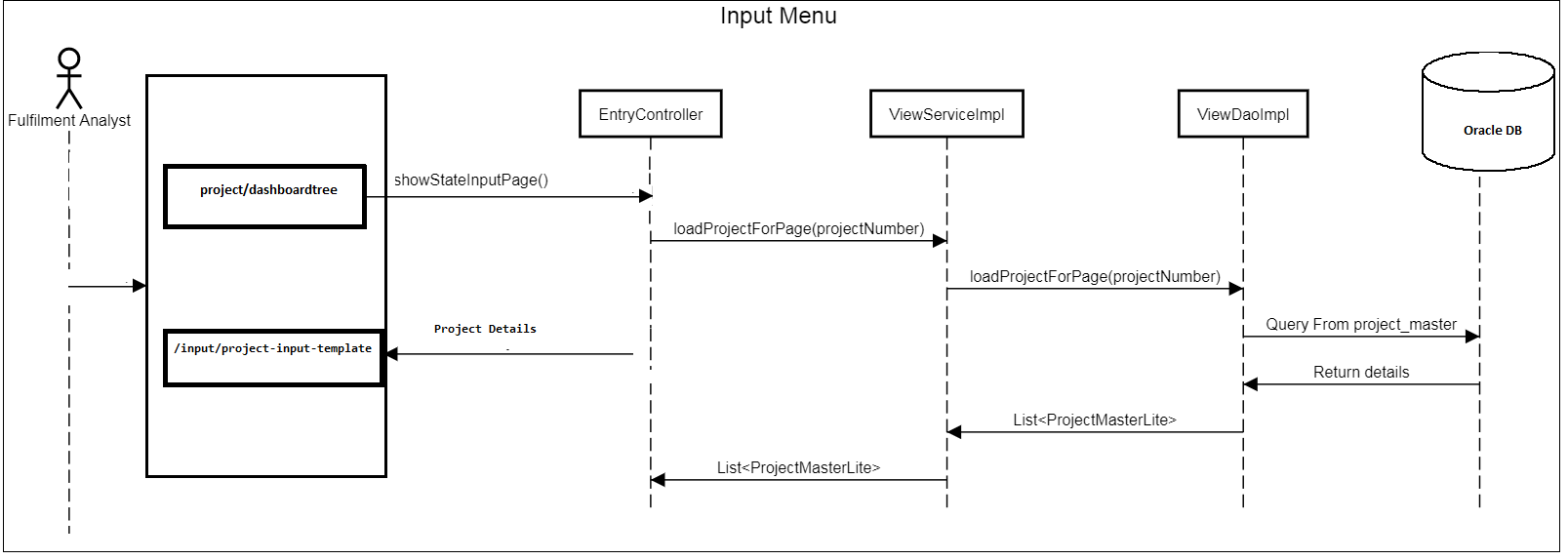
# UI

## Input Menu

* Request ‘/input/Entry’ will be mapped to showStateInputPage()

/cms-fusion-controller/src/main/java/com/equifax/cms/fusion/controller/input/EntryController.java

* Load the details of the Project from table PROJECT\_MASTER table - viewService.loadProjectForPage
* Return to view /input/project-input-template, which will be resolved by ViewResolver org.springframework.web.servlet.view.InternalResourceViewResolver which is mentioned in *cmsFusion-servlet.xml*
* This will redirect to jsp page project-input-template and project-input-grid-content.jsp.

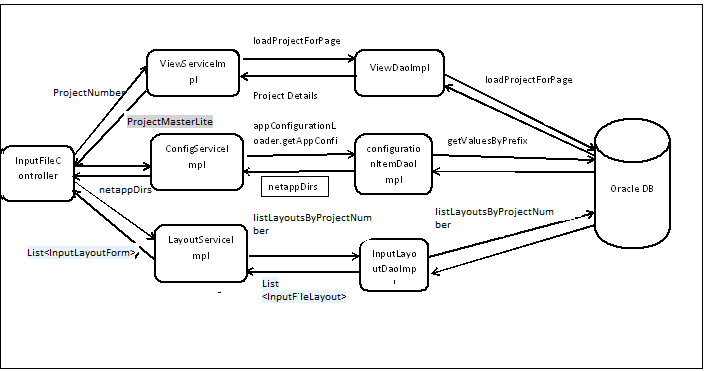


## Import New File

* Request /input/import-file will be mapped to showImportFilePage()

/cms-fusion-controller/src/main/java/com/equifax/cms/fusion/controller/input/InputFileController.java

* Load the details of the Project from table PROJECT\_MASTER table - viewService.loadProjectForPage
* Load the netappDirs where the input files are loaded in the ‘nas’ location, from the CONFIGURATION\_ITEM table.
* Get the layout details of the project from INPUT\_FILE\_LAYOUT table and the import layout details from INPUT\_IMPORT\_LAYOUT
* The details for the page will be stored in InputLayoutSearchForm.



## Save/Continue

* Request /input/save-input-process will be mapped to saveInputProcess()

/cms-fusion-controller/src/main/java/com/equifax/cms/fusion/controller/input/InputFileController.java

* Validate the input file for the format
* Enter the details selected as configurations in the INPUT\_FILE\_PROCESS table.
* Insert the input file details in INPUT\_FILE table
* Add the imported layout details if any for the process in INPUT\_IMPORT\_LAYOUT table
* Save the input file process details in inputProcessService.saveInputProcess().
* The Input file will be parsed based on the Input file format and will be displayed in the Grid.
* The user can select the corresponding fields that need to be saved in the Oracle DB

## Import Layout

* Request /input/import-layout-page-fixed for Fixed format, /input/import-layout-page-delimited for Delimited format
* The request is mapped with the methods showImportFixedLayoutPage () for Fixed Layout and showImportDelimitedLayoutPage() for Delimited layout.
* The methods will load the Layouts from INPUT\_IMPORT\_LAYOUT
* These .csv files will be parsed in the parseCsvFile() in LayoutServiceImpl.java
* Populate the grid details from getImportLayoutGridDetailFromCSV()
* Add the details to Model attribute to be displayed in jsp page model.addAttribute("gridJson").
* For Fixed format, the import layout form is ImportFixedLayoutForm
* For Delimited, it is ImportDelimitedLayoutForm

## Submit

* Request input/summary/submit will be mapped to submitInputFileSummary()

/cms-fusion-controller/src/main/java/com/equifax/cms/fusion/controller/input/InputFileController.java

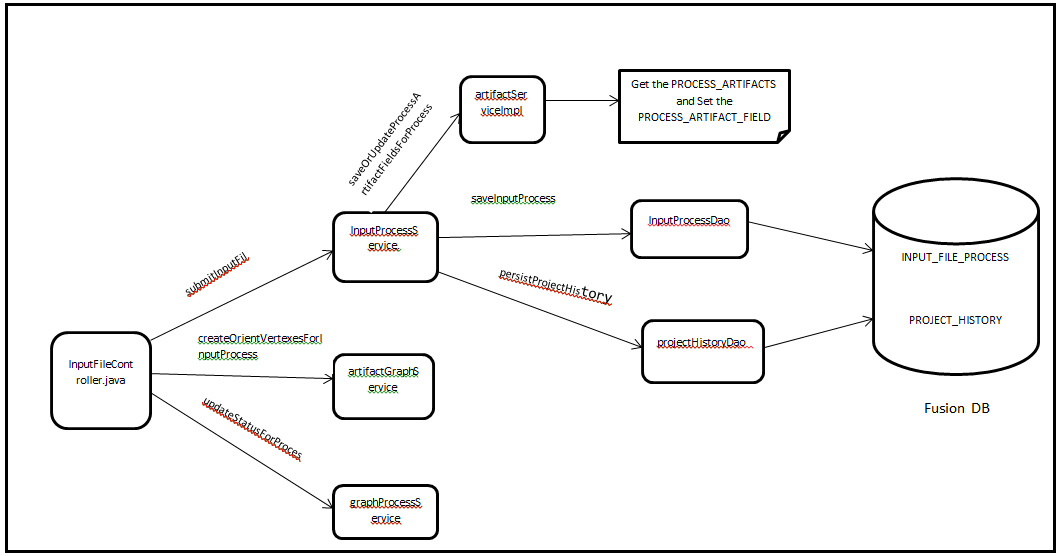
* Submit process happens through inputProcessService.submitInputFile()
* artifactGraphService. createOrientVertexesForInputProcess() will create the PROCESS\_RELATIONSHIP and Artifacts
* graphProcessService.updateStatusForProcess() will updates the status for the process.
* inputProcessService.submitInputFile()has two methods
* Saving the details in Oracle DB
* Sending the job Configurations to REST/ WorkFlow Manager

## Input Zip Codes

* Request /input/newZipEntry will be mapped to showZipCodesInputPage() in ZipCodeController.java
* This method will load the project details to be shown on the New Zip entry page
* Request /input/save-zipCodeEntry will save the details in the corresponding tables .
* Update the details in ZIPINPUT\_FILE and zipcode\_process tables.
* ZipCodeService.java will parse the zip file in the parseFile().
* Update and validate the Zip codes in the zipcode\_input\_entry table.
* zDao.saveZipCodeProcess will save the Zip code details in zipcode\_process table

### SUBMIT to Oracle DB

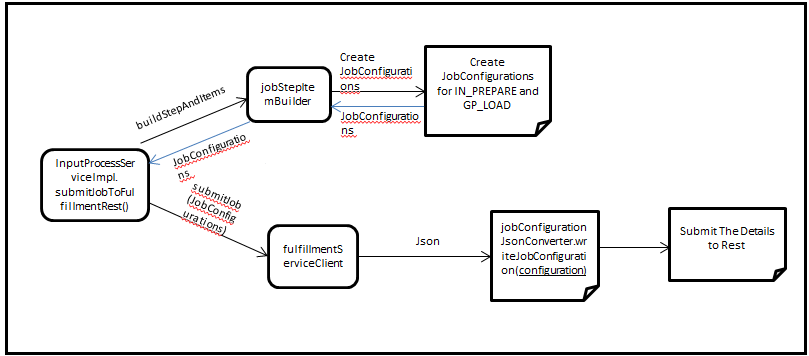
* Enter the details in PROJECT\_HISTORY table.
* Set the process status as SUBMITTED.
* Enter details in PROCESS\_ARTIFACT, PROCESS\_ARTIFACT\_FIELD tables
* inputProcessDao.saveInputProcess() will save the input details in INPUT\_FILE\_PROCESS table.



### SUBMIT to Submit to REST

* /cms-fusion-service/src/main/java/com/equifax/cms/fusion/service/AbstractBasicProcessService.java.submitJobToFulfillmentRest() will have two steps
* Create the JobConfigurations
* jobStepItemBuilder.buildStepAndItems() will create the Steps and WorkItems for the Process
  + Create itemName as IN\_PREPARE for File Prepare Item
  + Create itemName as IN\_GPLOAD for GP\_LOAD.
  + Create the Parameters and Mapping Strategies
* Create ScriptStepDefinition [Activity] which is [@JsonTypeName("script-step")] added as workflowDefinition.
* Create ParallelStepsDefinition [@JsonTypeName("parallel-step")] which will be the block of steps that all must be executed in parallel.
* Submitting the JobConfigurations to Rest
* fulfillmentServiceClient.submitJob() will submit the JobConfigurations created in the previous step.
* /cms-fusion-service/src/main/java/com/equifax/cms/fusion/service/fulfillment/FulfillmentServiceClient.java.submitJob()

Will convert the JobConfigurations to a JSON String and pass the details to REST



## UI Validation

UI validations for fields in EBCDIC format and ASCII format must be universal. Back end validation for both ASCII format and EBCDIC format is to be done for the fields.

* DOB
* DOB\_DDMMYYYY
* DOB\_YYYYMMDD
* DOB\_YYYYDDMM
* DOB\_DELIMITED\_MDY
* DOB\_DELIMITED\_DMY
* DOB\_DELIMITED\_YMD
* DOB\_DELIMITED\_YDM
* BIRTH\_MONTH
* BIRTH\_DAY\_MONTH
* BIRTH\_YEAR
* BIRTH\_CENTURY
* DOB\_SIGNED

So that when User provides incorrect Length values for Date and related Fields i.e. DOB, DOB Delimited, Birth Date month, Day, year Century, system should display an error message to the User when user selects the Preview / Generate PDF / Save / Continue option. The error message displayed should be as…….."Birth Century(CC) field type should have length 2."

# REST

* /fulfillment-common/src/main/resources/com/equifax/cms/fusion/fulfillment/common/applicationContext-rest.xml will have the configuration to redirect to the REST application
* From the configuration in applicationContext-rest.xml, it will redirect to submitJobConfiguration() in

/fulfillment-common/src/main/java/com/equifax/cms/fusion/fulfillment/common/resource/FulfillmentJobResource.java

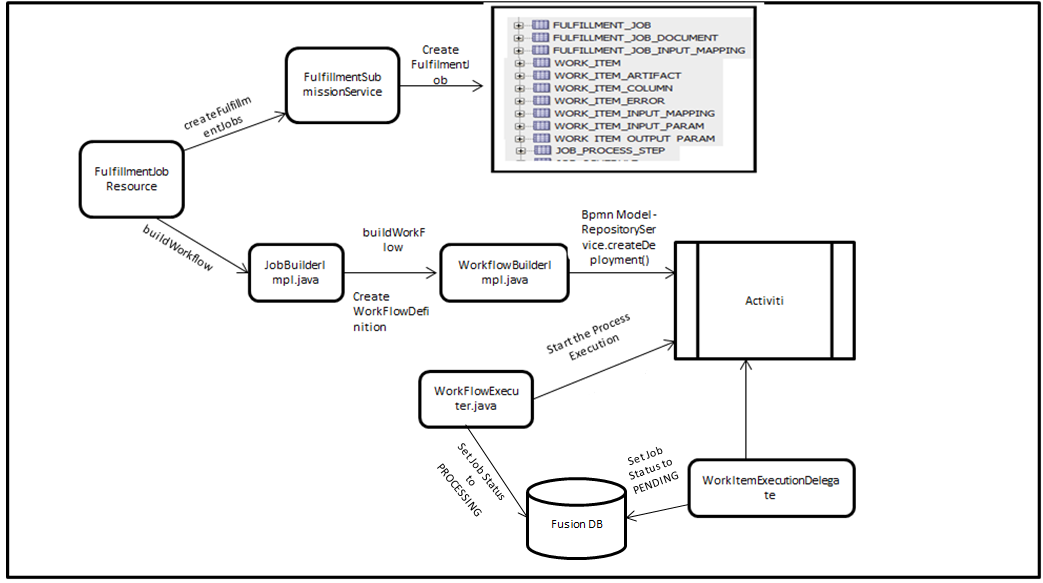
* Create Fulfillment job details in FULFILLMENT\_JOB table, in FulfillmentSubmissionService.createFulfillmentJobs()
* Insert into FULFILLMENT\_JOB\_INPUT\_MAPPING if it is a job stacking
* Enter the Json Details in FULFILLMENT\_JOB\_DOCUMENT table.
* It will create the Step details in JOB\_PROCESS\_STEP table.
* It will create the work item details in WORK\_ITEM and WORK\_ITEM\_INPUT\_MAPPING tables
* /fulfillment-common/src/main/java/com/equifax/cms/fusion/fulfillment/common/service/JobBuilderImpl.java - buildWorkflow() will create the Work Flow from the jobConfigurations
* /fulfillment-common/src/main/java/com/equifax/cms/fusion/fulfillment/common/service/WorkflowBuilderImpl.java .buildWorkflow() will build the work flow
* WorkflowDefinitionConversion will create a Bpmn process with the WorkflowDefinition
* Create a workflow with Bpmn process – [Activiti].
* Deploy the Bpmn model with the RepositoryService.createDeployment()
* WorkflowExecutor will start the process instance using RuntimeService [service task]
* processInstance = runtimeService.startProcessInstanceById(job.getFlowDefinitionId(), processVariables);
* set the job status as processing

job.setStatus(WorkItemStatusEnum.***PROCESSING***.getItemStatus());

* WorkItemExecutionDelegate will set the workitem status to ***PENDING*** in startExecution () method.

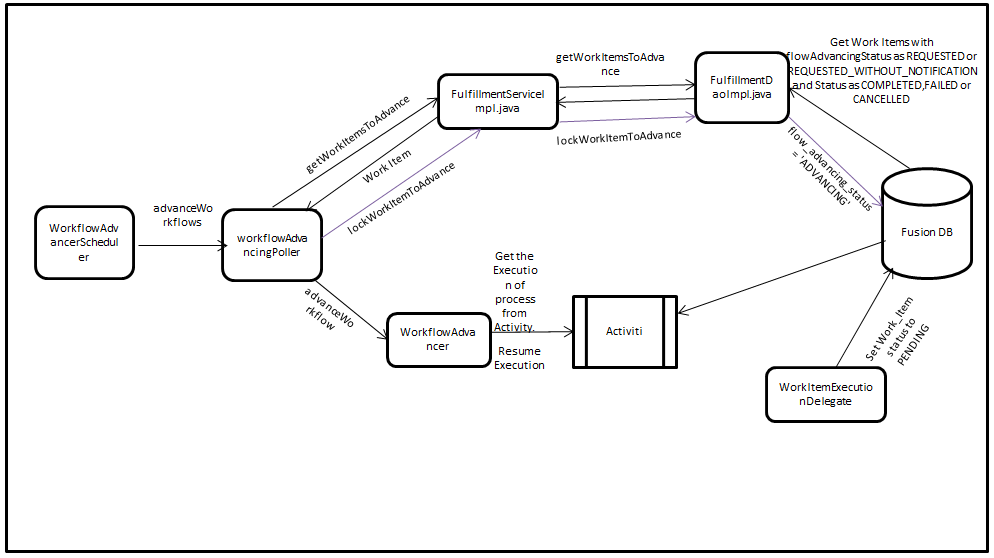
[Works based on Activiti -> Reference: https://www.activiti.org/javadocs/org/activiti/engine/delegate/class-use/DelegateExecution.html]

* Send notification for success or failure



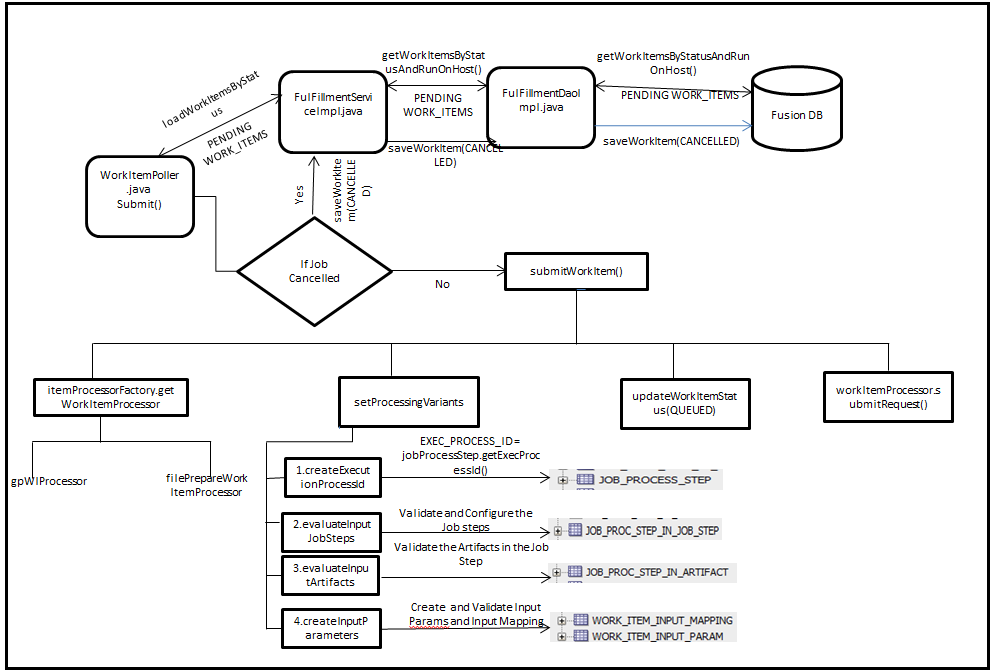
**Work Flow Advancing Flow**

* WorkflowAdvancerScheduler.java is the scheduler which will call the WorkflowAdvancingPoller.java
* WorkflowAdvancingPoller.advanceWorkFlow() will pick up the Work Item with the flow Advancing status as ***REQUESTED*** or ***REQUESTED\_WITHOUT\_NOTIFICATION***
* fulfillmentService.lockWorkItemToAdvance() will set the status to ***ADVANCING***
* Get the WorkItem in wait task and trigger it.
* Change the status to ***ADVANCED***
* WorkItemExecutionDelegate will set the workitem status to ***PENDING*** in startExecution () method.



# CORE Application

* /fulfillment-core-web/src/main/java/com/equifax/cms/fusion/fulfillment/service/WorkItemPoller.java will poll the WORK\_ITEM table and will get the Work\_Items which are in Pending status
* If Job is cancelled, update the workitem status to ***CANCELLED***
* submitWorkItem will Submit other workitems which are not cancelled
* Update the Work\_item status to ***Queued***
* getWorkItemProcessor() in /fulfillment-core-web/src/main/java/com/equifax/cms/fusion/fulfillment/service/WorkItemProcessorFactory.java will provide the WorkItemProcessor based on the WorkItemSubTypeEnum and WorkItemMainTypeEnum. For INPUT we have two work\_item
* IN\_PREPARE -> workItemProcessor = filePrepareWorkItemProcessor;
* GPLOAD -> workItemProcessor = gpWIProcessor;



## FilePrepareWorkItemProcessor

* Update the WorkItem Status to PROCESSING
* Prepare FulfillmentCoreRequest
* FileWorkItemHelper.process()will create the IN\_PREPARE file and update in nas location

o Get File Processor based on fileType

o Load input file

o Create intermediate file

o Calculate layout

o generate seq & check for cleanse flag - fileProcessor.generateRevisedInputFile(). This will create file in csv format.

The file processors are EbcdicFileProcessor, AsciiFixedFileProcessor, AsciiDelimitedFileProcessor

o Add output parameter Relative Revised File Path and gp\_seq\_start\_num - appendRelativeRevisedFilePath()

Save stats for the workitem

* Update the WorkItem Status to COMPLETED

## GPWorkItemProcessor

* Update the Work\_Item\_status to ***PROCESSING***
* /fulfillment-core-web/src/main/resources/com/equifax/cms/fusion/fulfillment/core/applicationContext-fulfillment-core.xml will provide the WorkItemCommandFinder

<entry key="IN.GPLOAD">

<bean

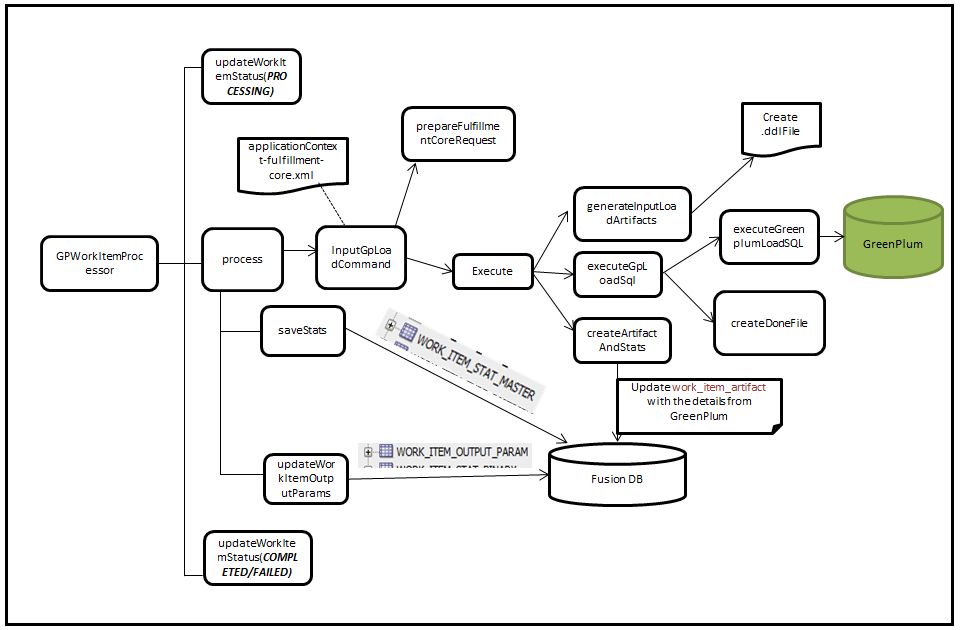
class="com.equifax.cms.fusion.fulfillment.command.WorkItemCommandFinder">

<lookup-method name="createWorkItemCommand" bean="inputGpLoadCommand" />

</bean>

</entry>

* Copy the file from the nas location
* Update the artifact details in work\_item\_artifact table.
* Generate GpTableDdlTempate for the file format
* executeGpLoadSql will execute the GreenPlum upload plsql
* Create the .done file.
* Update the artifacts in work\_item\_artifact table.



## ZipWorkItemProcessor

* Update the Work\_Item\_status to ***PROCESSING***
* Process the Work Item
* Parse the Zip file and update the details in zipcode\_runtime\_input\_entry
* Execute the procedure ZIPCODE\_EXPAND\_PROCEDURE and insert the values in ZIPCODE\_EXPANDED\_ENTRY.
* Create the Zip details to a file
* Update the Zip and State details in the State Master
* Save the Stats

*Stat stat = createStat("ZIPCODE\_EXPANDED\_STATS");*

*stat.add(new StatItem("UNIQUE\_ZIPCODE\_COUNT", String.valueOf(uniqueZipCodeList.size()), false, "Total count of unique ZIP codes"));*

*stat.add(new StatItem("ZIP\_FILE\_LOCATION", zipFile.getAbsolutePath(), false, "ZIP file location"));*

*stat.add(prepareStateWiseZipCountStats(item.getId()));*