|  |  |  |  |
| --- | --- | --- | --- |
| **GPO HEALTH** | | | |
| **Document Title** | **Design Specification Document for ETL Flow** | | |
| **Document ID** | **Ver** | **System Identifier** | **Page no** |
| GPO-001 | 1.0 | Talend ETL Framework |  |

**Design Specification Document for ETL Flow**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | | Job Title and Role | Signature | | | Date |
| The Author (a Developer) is signing to confirm that this document has been prepared in accordance with an approved Document management process,  That content is consistent with the deliverable definition found in specify system,  that relevant input from any contributory authors has been included and that an appropriate review /editing process has been conducted | | | | | | |
| Author by:  Deva Kishore Katta | Developer | | |  |  | |
| Approved by: |  | | |  |  | |

1. **Introduction**

This Document Architecture, Design and operational aspects for Talend ETL Framework. It provides details required for an implementation to satisfy the requirements to provide fit for provide system.

This Platform Established for loading Invoice,PO,AP data from source to target through various stages and provides meaningful information as per business requirement.

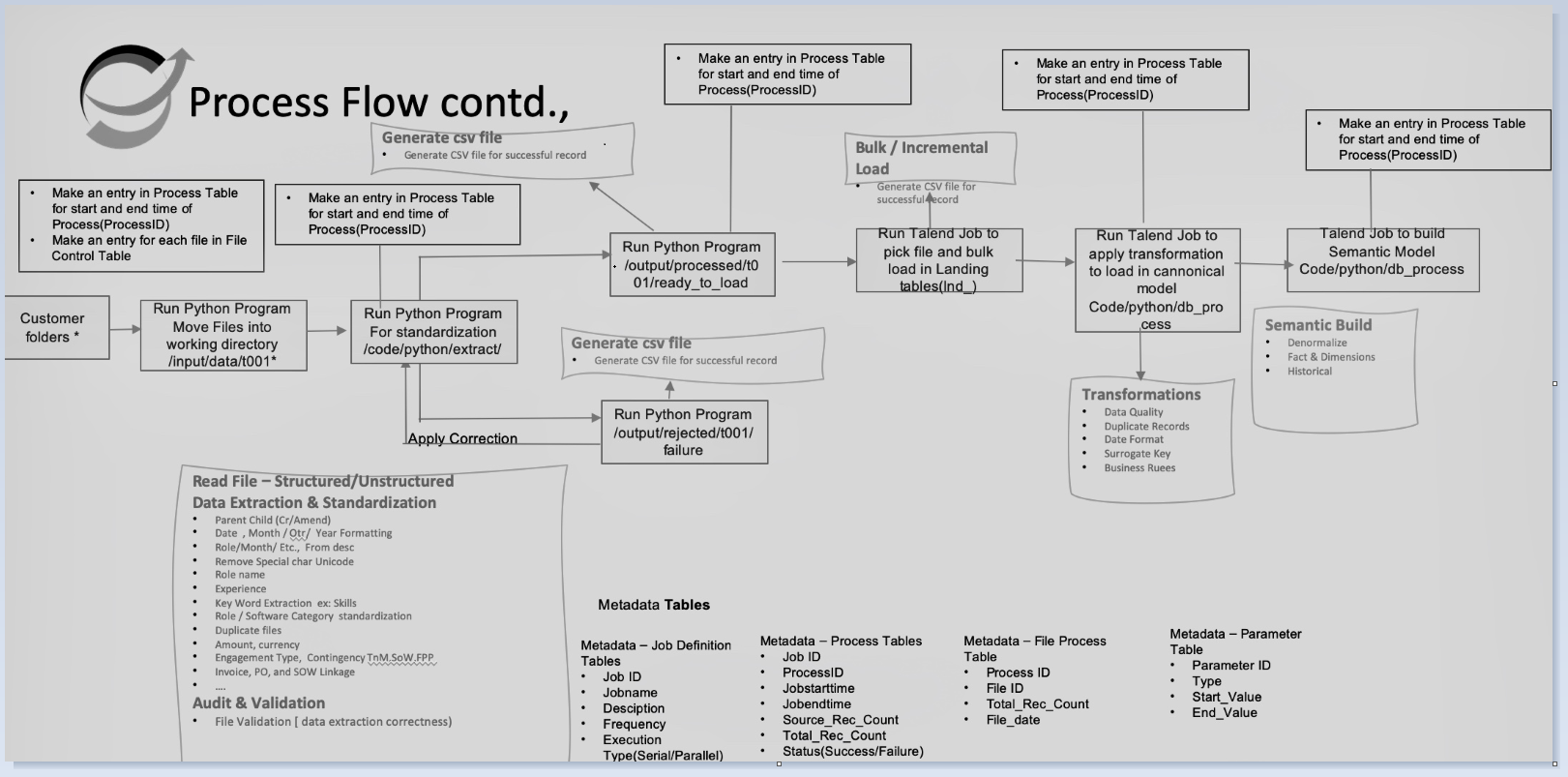
* 1. **Definitions:**

|  |  |
| --- | --- |
| TERM | DEFINITION |
| ETL | Extract Transformation Loading |
| DB | Data base |
| TDI | Talend Data Integration |

1. **Design Specification**

This design specification document describes Talend framework design end to end along with metadata, audit and logging mechanisms.

* **High level Architectural Diagram**



* 1. **Software Architecture**

Entire framework design managed with configuration/metadata tables available in MYSQL database. Source is L0 layer where source files will be placed. Using Talend framework the data driven workflows were created to orchestrate the processing of data. Talend pipelines triggering in Ad Hoc nature manually.

* + 1. **Component overview**

**2.1.1.1 Data base usage**

* SQL data base services are used in this ETL framework.
* It stores metadata which is used configuring ETL framework.
  + - 1. **ETL Tool**

Talend ETL tool used in this framework for designing data flows/ pipelines and transformations.

* + - 1. **Data Store**

The configuration, metadata, auditing, output tables are maintained in MYSQL database.

**2.1.2 Configuration Specification**

All database connection details, source paths, target paths are stored in context tables from MySQL database. These tables were used in Talend framework for executing talend jobs for processing source files to target tables etc., Below are the context tables which are used.

|  |  |  |
| --- | --- | --- |
| **Database** | **Table name** | **Purpose** |
| MYSQL | metadata. prod\_l1\_context\_db | This table contains all the connection details of target tables, Audit tables. |
| MYSQL | metadata. prod\_l1\_context\_dir | This table contains source paths, reject path,landing paths etc., |
| MYSQL | metadata. L1\_Audit\_data | It stores audit info of L1 jobs |
| MYSQL | metadata. L1\_job\_log | It stores the Log info of L1 jobs |
| MYSQL | metadata.prod\_context\_l2\_l3 | This table contains all the connection details and target tables, Audit tables. |
| MYSQL | metadata.prod\_context\_l4 | This table contains all the connection details and target tables, Audit tables. |

Table: Configuration details

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.no** | **Source** | **Dev** | **Preprod** |
| 1 | MYSQL DB | dev-mysql.c969yoyq9cyy.us-east-1.rds.amazonaws.com | gpo-db.c969yoyq9cyy.us-east-1.rds.amazonaws.com |

|  |  |
| --- | --- |
| **metadata.prod\_l1\_context\_dir** | |
| Delimiter\_File | C:\ETL\Prod\_Data\_Pipeline\_Execution\Header\_Validation\outputDelimiter.csv |
| dynamic\_PyCode | findDelimiterInFile\_v3.py |
| dynamic\_Pycode\_HomeDir | C:\ETL\Prod\_Data\_Pipeline\_Execution\Header\_Validation |
| file\_archive | C:\ETL\Prod\_Data\_Pipeline\_Execution\etl\_target\Archive |
| file\_processed | C:\ETL\Prod\_Data\_Pipeline\_Execution\etl\_target\Processed |
| file\_processed\_stg | C:\ETL\Prod\_Data\_Pipeline\_Execution\etl\_target\Processed\_stg |
| file\_rejected | C:\ETL\Prod\_Data\_Pipeline\_Execution\etl\_target\Reject |
| file\_splitpath | C:\ETL\Prod\_Data\_Pipeline\_Execution\etl\_target\Split\_File |
| py\_dedup | C:\ETL\Prod\_Data\_Pipeline\_Execution\Dedup |
| py\_sourcefiles\_download | C:\ETL\Prod\_Data\_Pipeline\_Execution\processLAtoL0 |
| py\_split\_file\_AP | C:\ETL\Prod\_Data\_Pipeline\_Execution\split\_updated\_ap |
| py\_split\_file\_inv | C:\ETL\Prod\_Data\_Pipeline\_Execution\split\_updated |
| py\_split\_file\_PO | C:\ETL\Prod\_Data\_Pipeline\_Execution\split\_updated\_po |
| rej\_schema\_out | C:\ETL\Prod\_Data\_Pipeline\_Execution\Header\_Validation\rej\_schema\_new.txt |
| sch\_validation\_pycode\_dir | C:\ETL\Prod\_Data\_Pipeline\_Execution\Header\_Validation |
| Sch\_validation\_src | C:\ETL\Prod\_Data\_Pipeline\_Execution\Header\_Validation\schema\_out.txt |
| source\_file\_path | C:\ETL\Prod\_Data\_Pipeline\_Execution\etl\_source |
| Srcfile\_copy\_dynamic | C:\ETL\Prod\_Data\_Pipeline\_Execution\Header\_Validation\Schema\_validation |

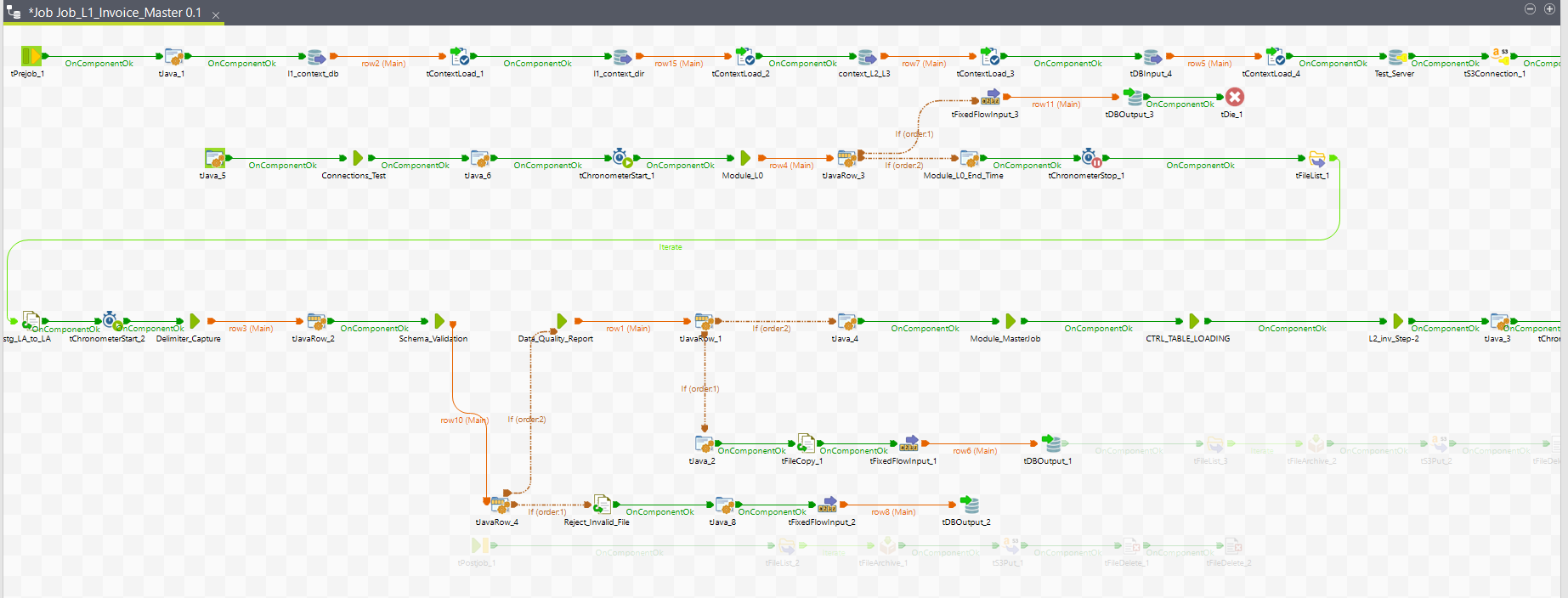
Table: Context path details

* 1. **Talend Data Integration Process(L1\_stage)**

For loading source files to target tables, all details with respect to paths need to configure in context tables.

* + 1. **Job\_L1\_Invoice\_Master**
* As Part of Talend ETL Data loading it holds all subjobs for various purposes.
* It has connections test job,Data quality jobs along with all module jobs.

**Job\_L1\_Invoice\_Master:**



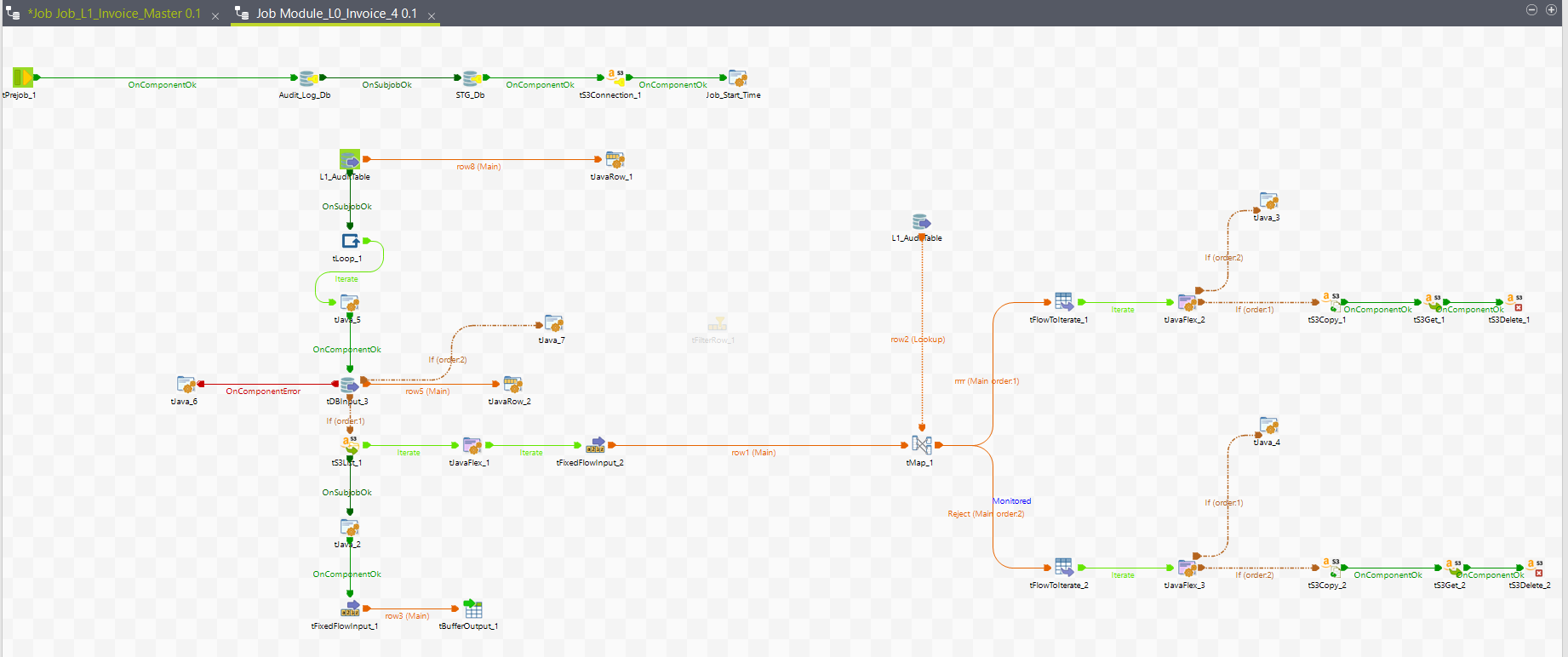
1. **Connection\_test:**

* It test the source and target connection details.
* Reads details from metadata table and validate the connection details before processing file.

1. **Data Quality Job:**

* This job validate the quality of data i.e., it finds for unwanted data.
* If bad data percentage is more than threshold value which is specified in metadata.
  + - 1. **Module\_L0**
* This Job read source files from S3 location and place the files in Landing Area
* It checks the duplicate file names and reject and place in reject folder if any duplicate file shared by customer.

**Module\_L0:**



* + - 1. **Master\_Job\_Invoice**
* As part of Talend ETL data loading, created various module talend jobs to get Various check points to validate data.
* All modules were configured in master job.

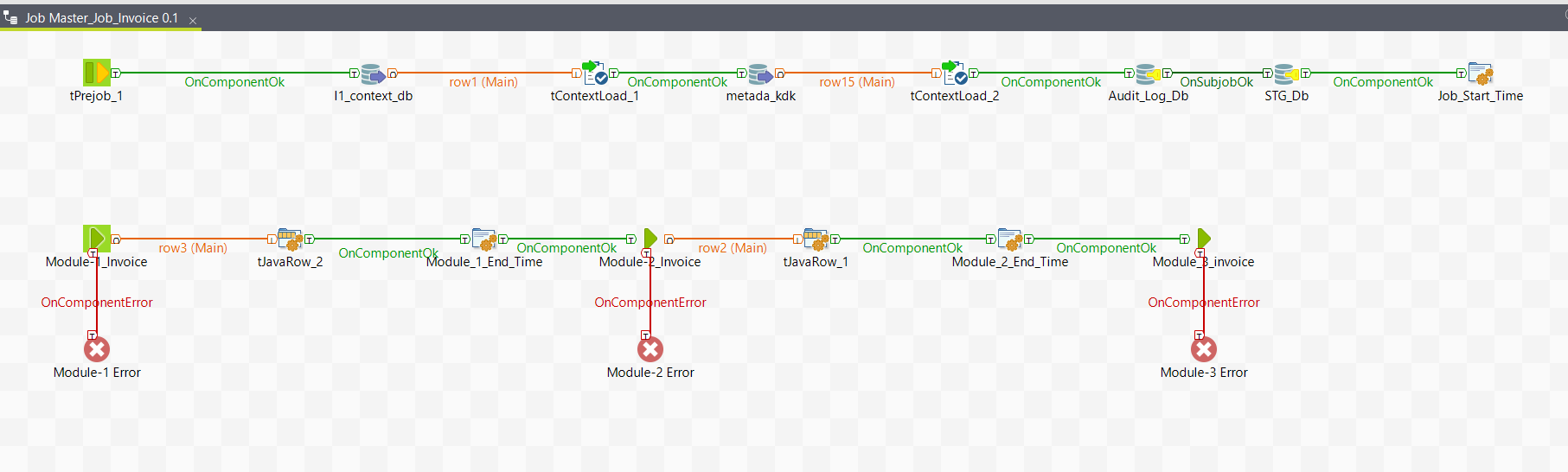


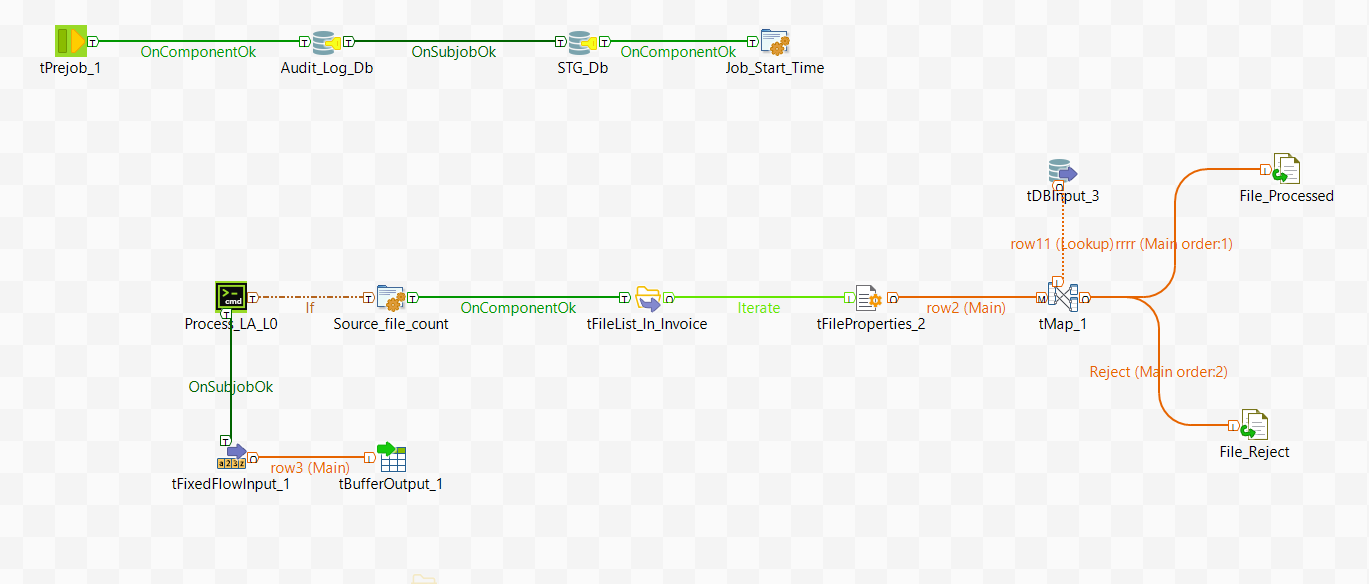
Table-2 Talend Pipeline details

|  |  |
| --- | --- |
| **Pipeline Type** | **Pipeline Name** |
| Master\_Job\_Invoice | Master\_Job\_Invoice |
| Module-1 |
| Module-2 |
| Module-3 |
| Module-4 |
| Schema\_Validation\_Inv |

1. **Child Job(Module-1)**

* This job will read source files from source path.
* Has python code will place Source File path, once we trigger master job python code will run in module-1 and place files.
* This job checks duplicate check, size check(file>500kb) to avoid empty files and name validation i.e., orphan file check.
* Once file passed all check points in module-1 source file will be placed into process\_stg path for further checks.
* If file failed in any check point file will be placed in reject path.

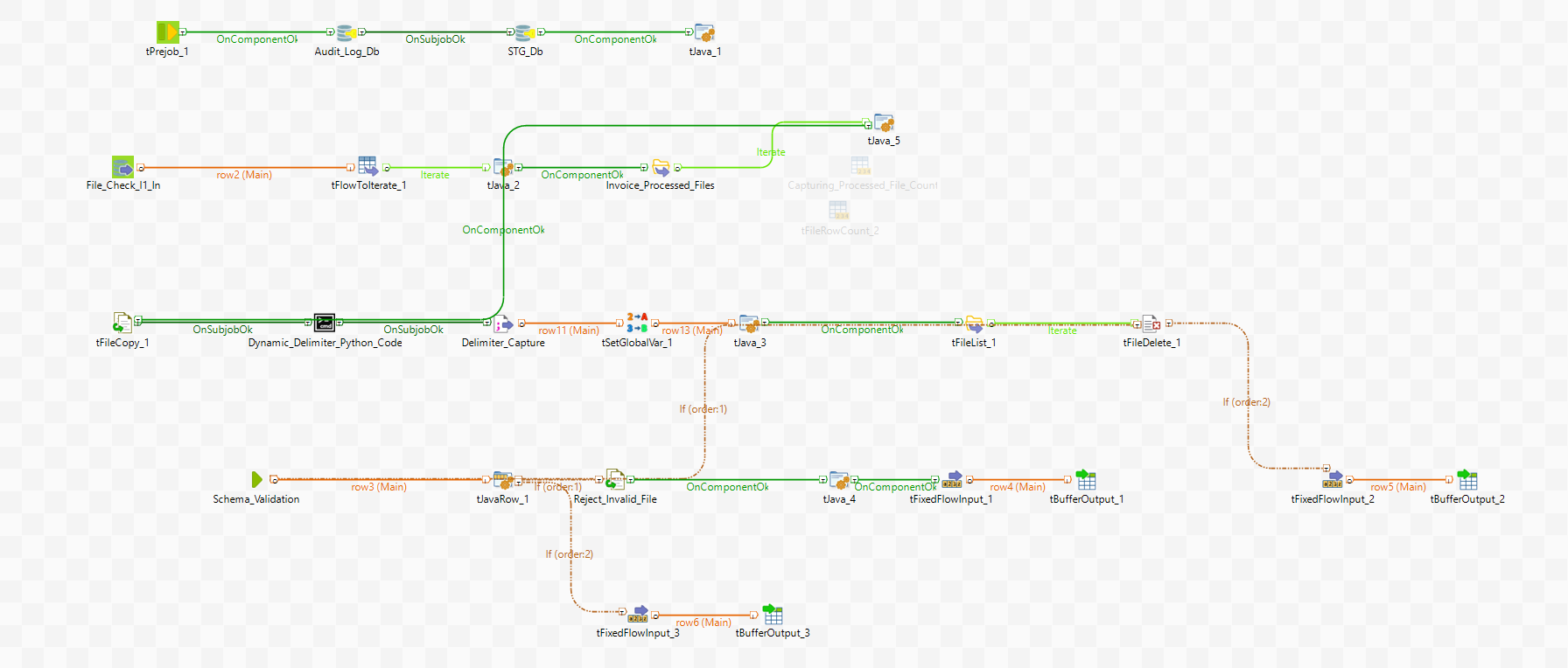
**Module-1**



1. **Child Job(Module-2)(Deprecated in Dec23 release)**

* This job will check and capture header, footer checks from file\_check\_l1table.
* Python code will trigger and capture the delimiter from the source file.
* This job will check the schema from source file and validate.
* If it failed at schema validation file will be placed at reject\_path.

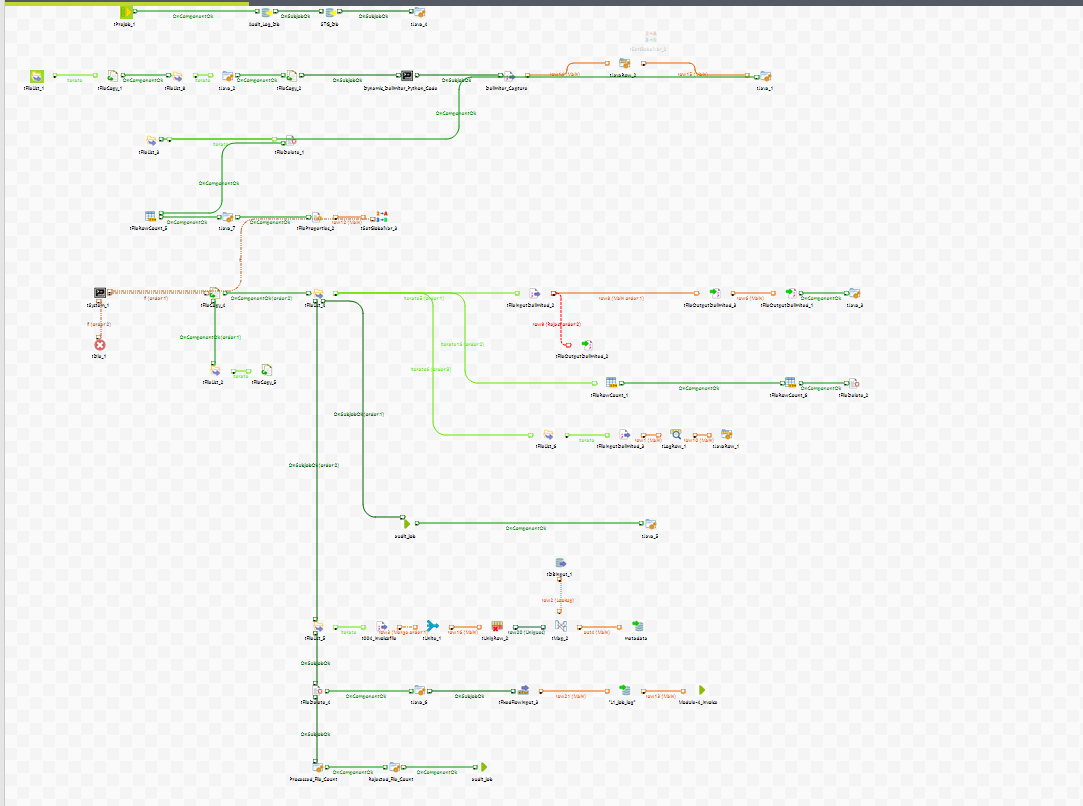
**Module-2**



1. **Child job(Module-3)**

* This job will split the source file into small files with the help of python code.
* This job capture the corrupted records into file in reject path.
* Once file created inn processed path audit information updated in audit table.
* Vendor names will be updated in ctrl\_vendor table.
* Loging table also will be updated in log tables.

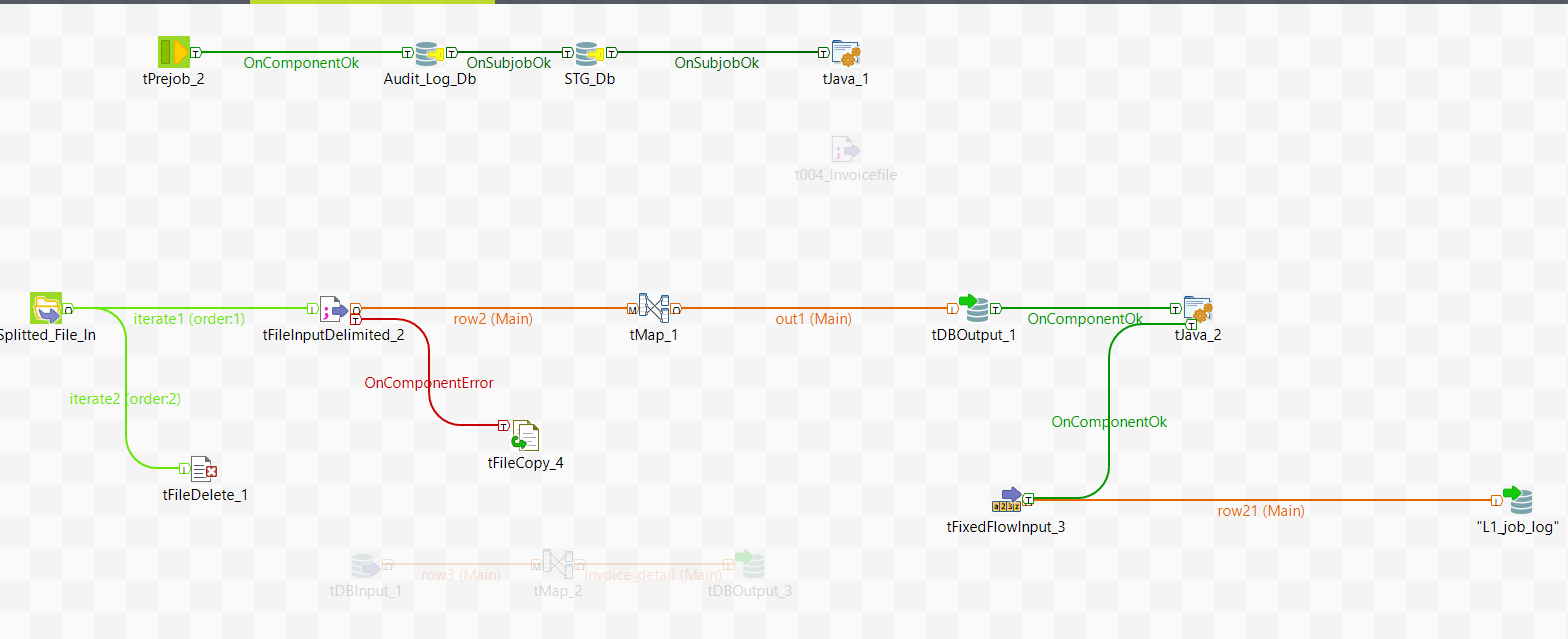
**Module-3**



1. **Child job(Module-4)**

* This job will load all valid files into target table.
* Once files loaded successfully logging information is updating in log table.

**Module-4**



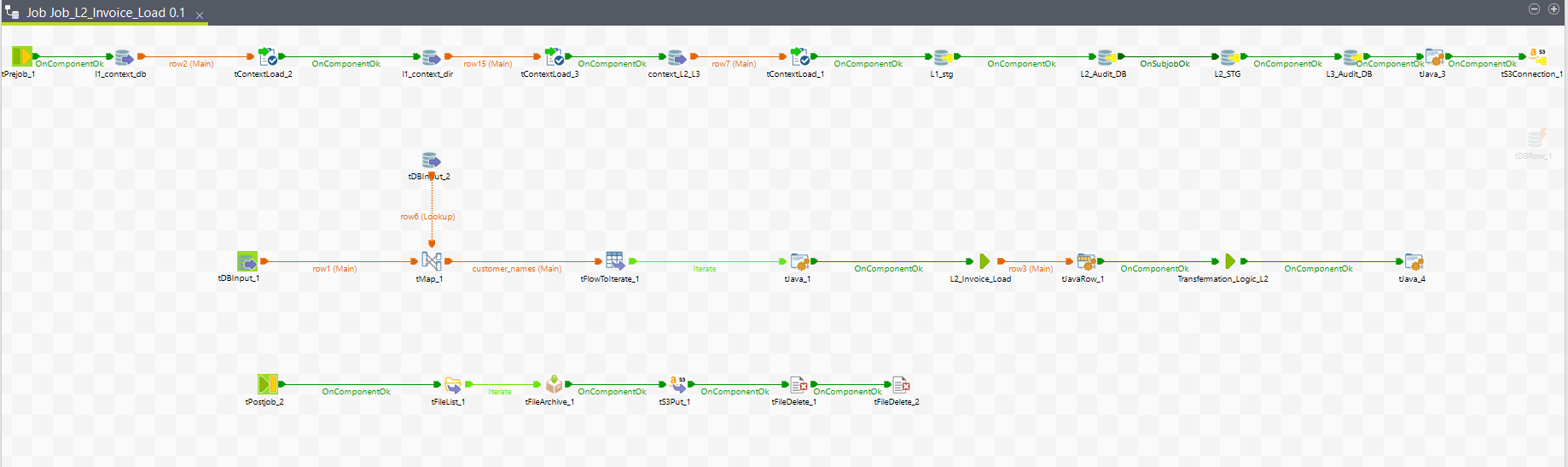
* 1. **Talend Data Integration Process(L2\_stage)**

This Job Loads Data from L1 stage to L2 stage.

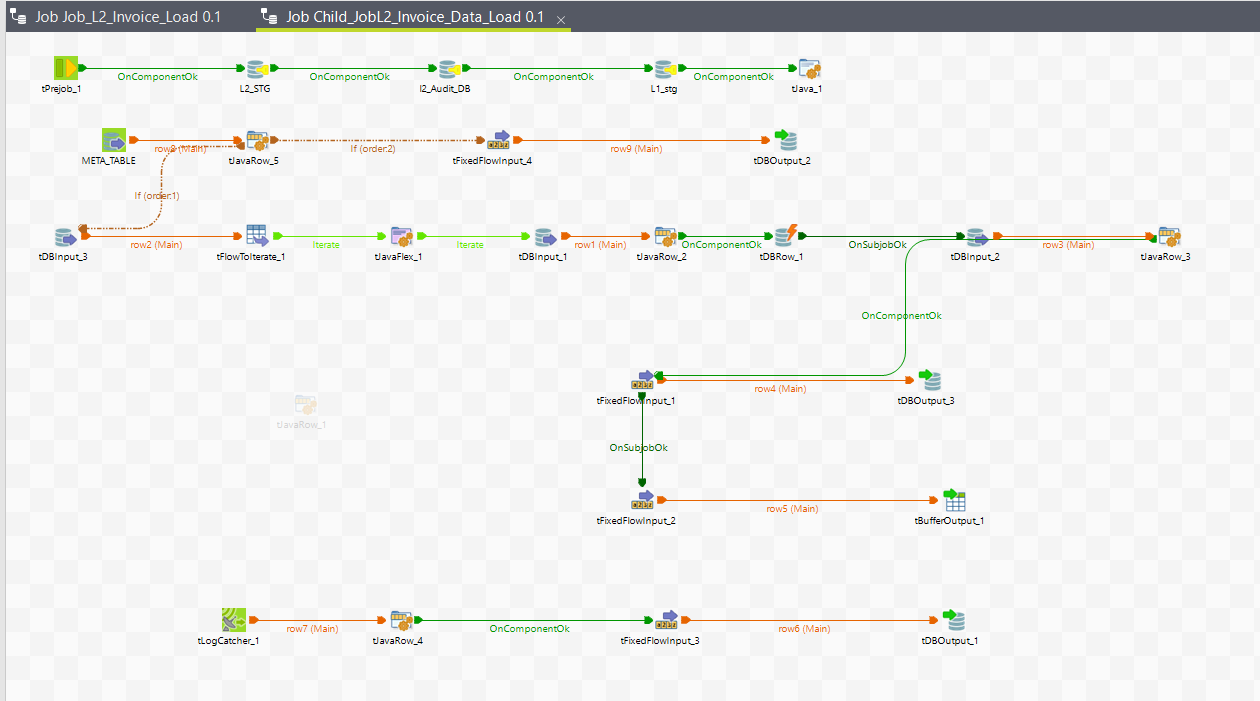
1. **Job\_L2\_Invoice\_data\_Load**

* We are passing customer name and Last\_Load\_date from L1\_invoice\_detail table and compare with L2\_Audit data.
* In Child job L2 invoice Load From metadata table we are passing queries and applying business logic on top of L1\_data and loading.
* Here we are maintaining all queries in metadata\_query table will pass dynamically.
* Maintaining audit and logging data in L2\_audit\_data.

**L2\_invoice\_data\_load**



**Child\_job\_L2\_invoice\_Load:**



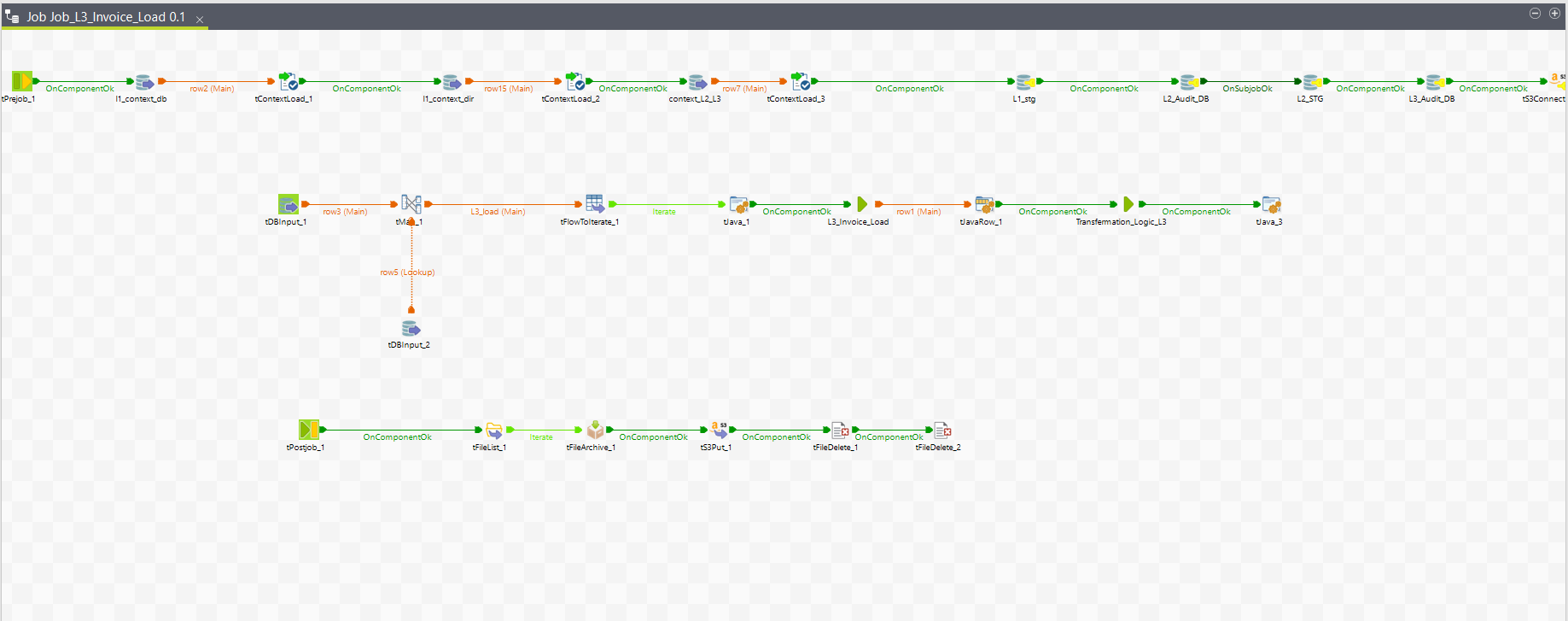
* 1. **Talend Data Integration Process(L3\_stage)**

The main aim of this job is to Load data to L3 satge tables from L2 Stage.

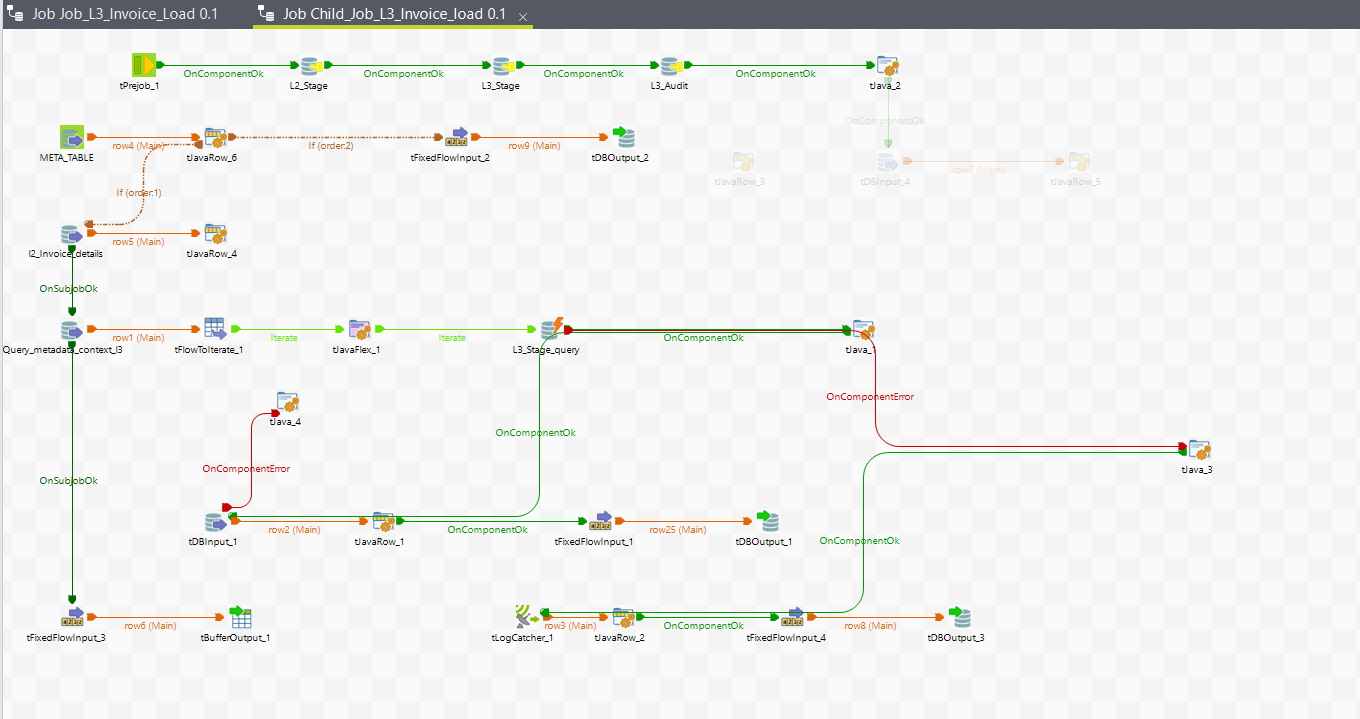
1. **Job\_L3\_Invoice\_Load**

* We are Passing customer name and Last\_Load\_date from L2\_invoice\_detail table and do lookup with L3 Audit table .
* Child\_job\_L3\_Invoice\_Load loads the data from mysql L2\_stage table to L3 Stage fact table by reading queries from metadata tables..
* Audit data is updating L3\_Audit\_data table once job completed.

**Job\_L3\_Invoice\_Load:**



**Child\_Job\_L3\_Invoice\_Load:**



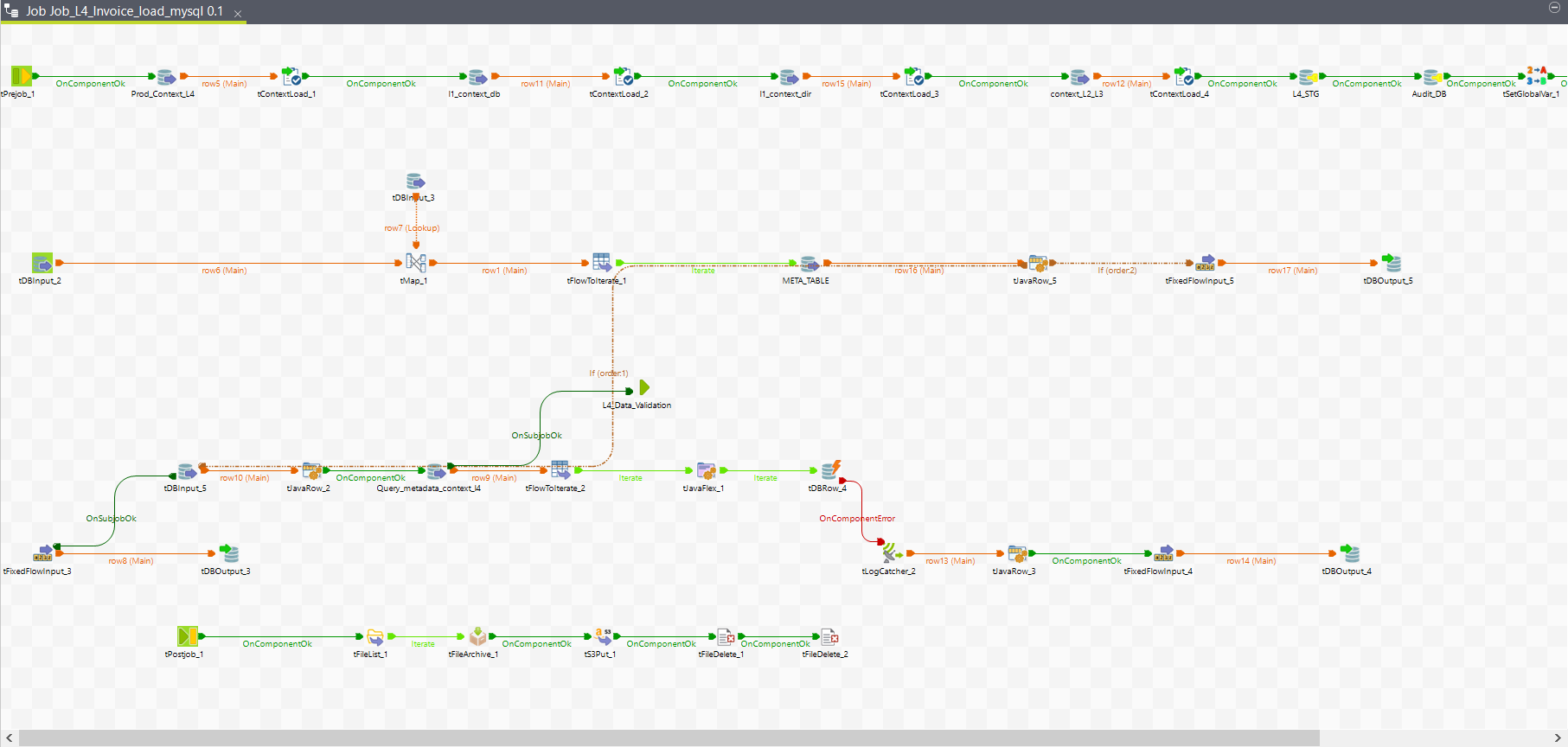
* 1. **Talend Data Integration Process(L4\_stage)**

In this stage we are loading aggregated data into L4 aggregated tables from L3 fact table.

1. **Job\_L4\_Invoice\_load**

* We Are passing party group key and last load date from l3 fact table and comparing with L4 Audit table.
* All aggregated queries were configured in metadata tables For loading to aggregate tables.
* Audit information is updating in L4 Audit data.

**Job\_L4\_Invoice\_load**

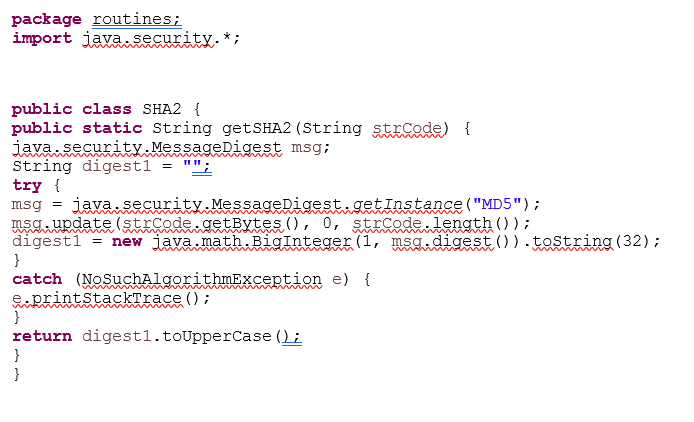


* 1. **Java Code(ROUTINE)**

There is customized java code for creating routine to generate unique Hash key. This Hash key will generated on SHA2 algorithm. The main aim of this routine is to eliminate the duplicate records while inserting to target table.

**DESIGN:**

* We came up with new implementation of generating Hash value record wise to compare the target values to source values.
* With this hash value we can compare source and target hash values and only unmatched hash value records will be processed
* For generating hash value, we need to generate customised routine in Talend. The same thing we can use in our jobs.
* Please find the below java code implemented for Hash value with MD5 algorithm.

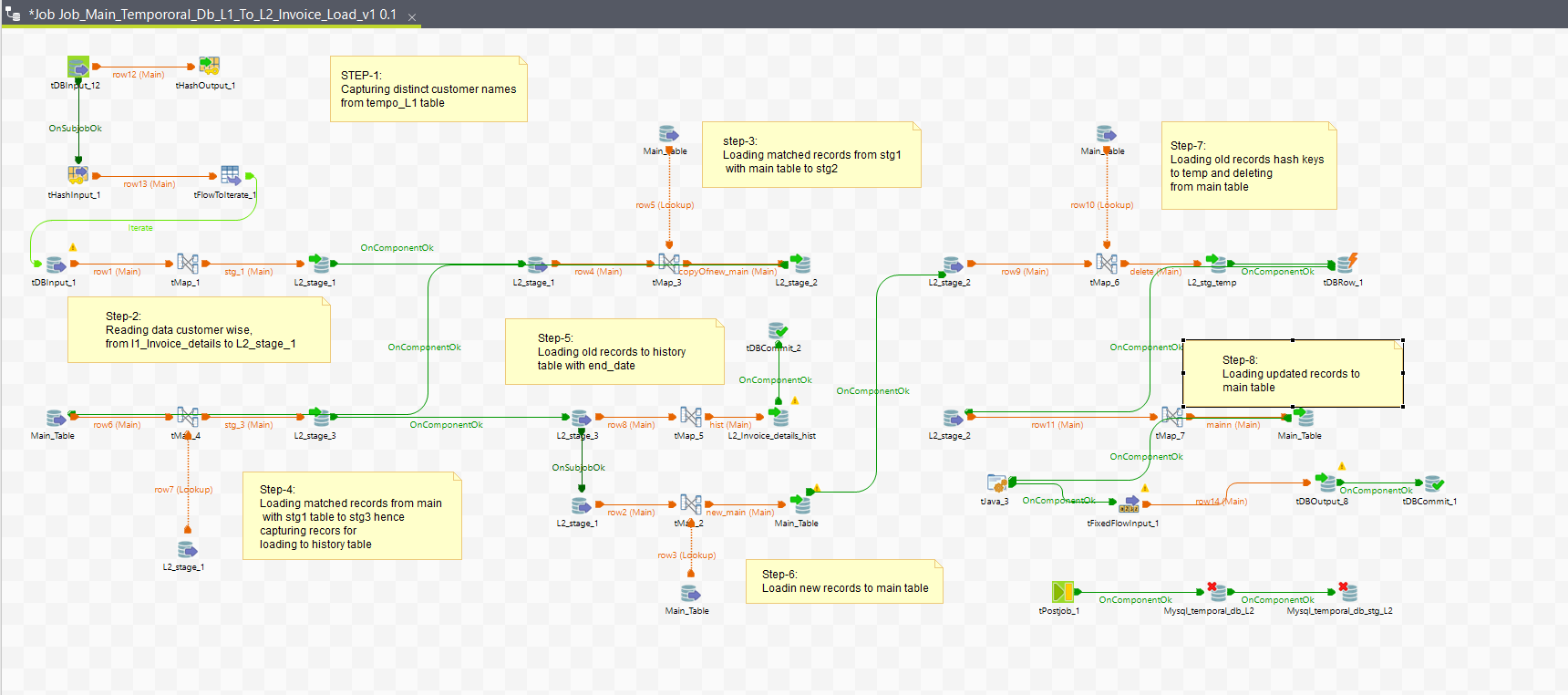


* 1. **Temporal DB:**
* While data is loading from L1 to L2 and L2 to L3 need to handle updated records in L2 stage and L3 stage tables.
* If old record comes with any updated data, our code will identify and update the target table record with updated value and will load old record into history table.
* This helps us to analyses the record level updated.
* To achieve this we are using different intermediate staging tables.
* All staging tables are truncate and load only.

1. **Process:**

* **Step-1:** Capturing Customer names from L1 stage table to process the from L1 layer. It helps to resolve the Out of memory issues while loading data.
* **Step-2:** Read the data from L1 table and load to stage table L2\_stage\_1.
* **Step-3:** It reads L2\_stage\_1 table and compare with L2 invoice detail table and load to l2\_stage\_2. Hence it has old records from L2 invoice detail table.
* **Step-4:** Loading matched records from main with stg1 table to stg3 hence capturing records for loading to history table.
* **Step**-**5:** Loading old records to history table with end date.
* **Step-6:** Loadin new records to main table from L2\_stg\_1 table.
* **Step-7:** Loading old records hash keys to temp and deleting from main table.
* **Step-8:** Loading updated records to main table.
* Once main table updated with records audit information will be load into Temporal Audit table

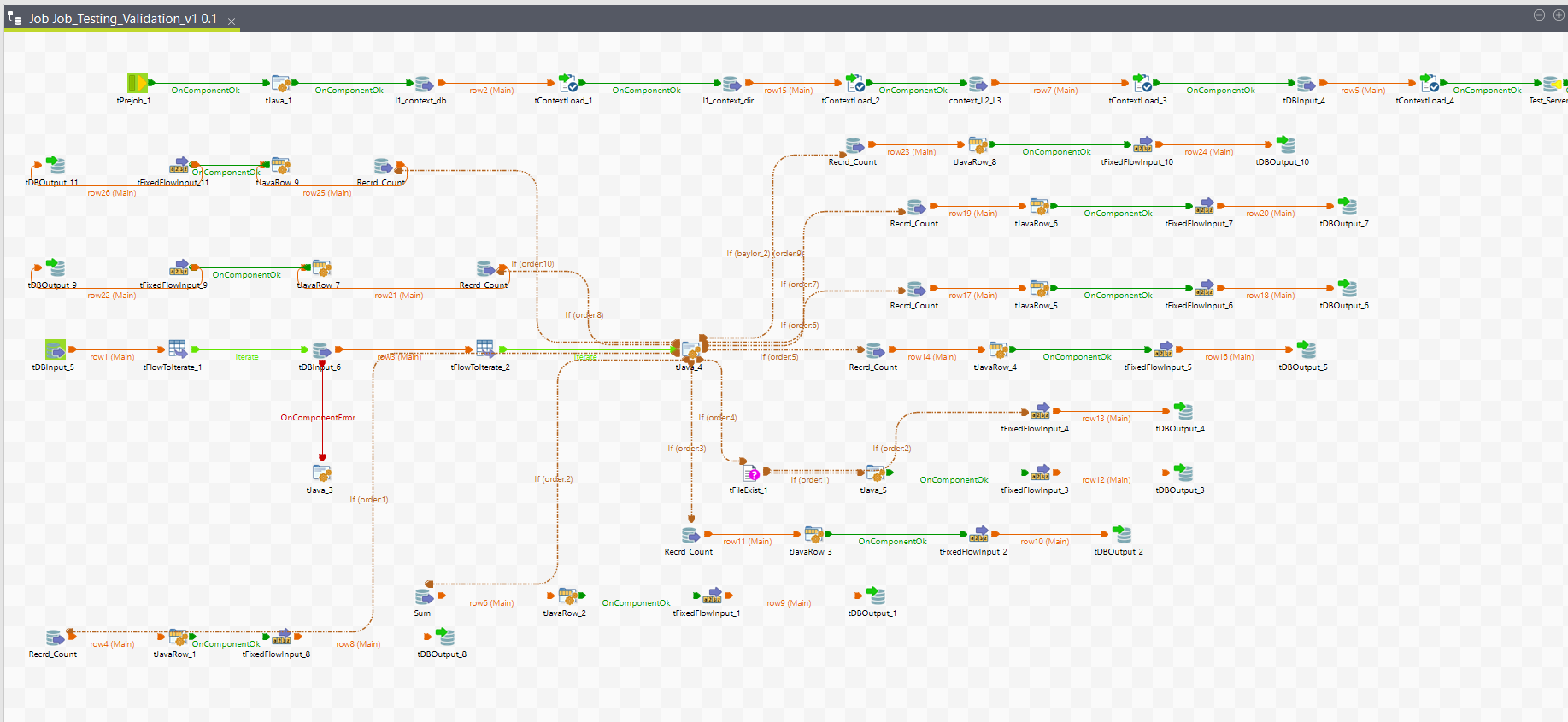
**Job\_Main\_temporalDB\_L1\_To\_L2\_Invoice\_Load:**



**2.8: Encryption of PPI Column:**

* 1. **Testing Automation:**
* Testing Automation is a talend code to test the ETL code.
* There are 90 test cases designed to validate the code including Functional level also.
* We have created a testing validation job in talend and configured all test cases in metadata table.
* Testing job will perform testing by reading test cases from metadata table as per active flag and update the test cases in table.
* Below list of tables used in testing.
* **Test\_metadata:** it has all test cases information.
* **Test\_validation:** it has all test cases results.

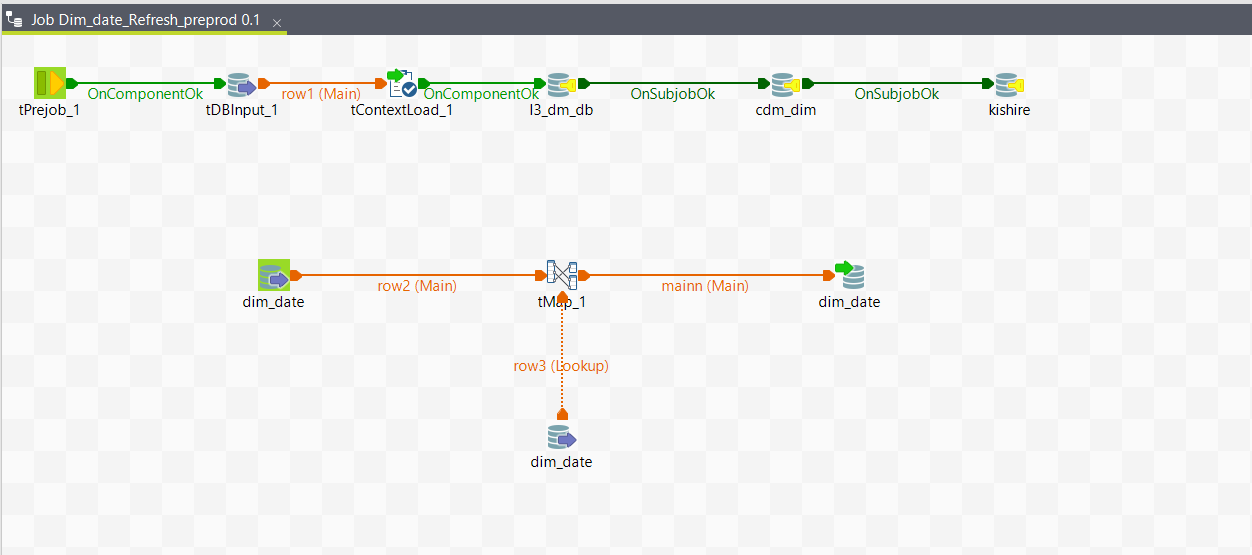
**Job\_Testing\_Validation:**



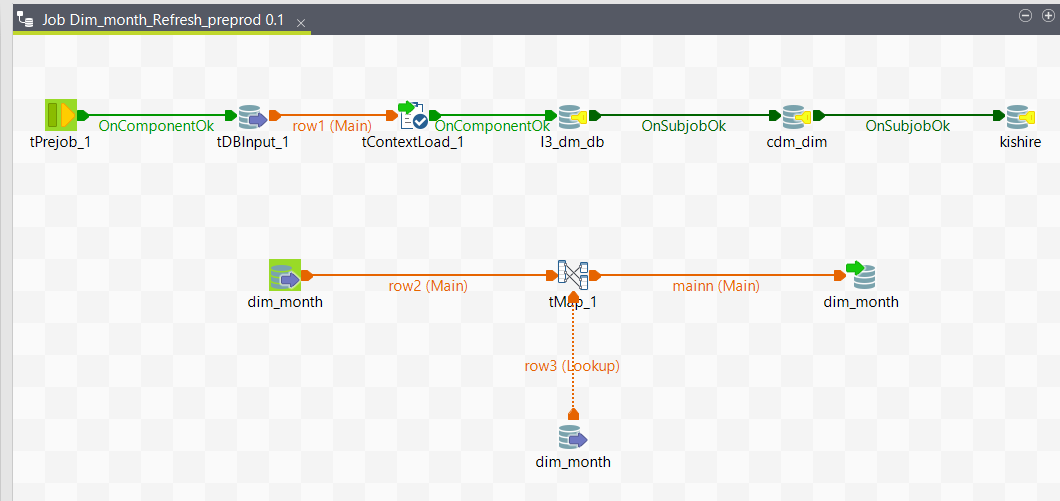
* 1. **Dimension Cloning:**
* In the process of Centralized database, we are making all dimension jobs data in Sync.
* Here we have refresh jobs and Sync Jobs.
* All the connection details are passing from dim\_metadata table.
* **Refresh Jobs:** it will refresh Data from preprod to dev.
* Dim\_date\_Refresh\_preprod.
* Dim\_month\_Refresh\_preprod.
* Dim\_Party\_group\_refresh\_preprod.
* Dim\_party\_org\_refresh\_preprod.
* Dim\_Party\_Refresh\_preprod.
* Dim\_Prd\_Refresh\_preprod.
* Dim\_slu\_Refresh\_preprod.
* Dim\_Vendor\_Group\_Refresh\_preprod.
* Dim\_Vendor\_Refresh\_preprod
* **Sync Jobs:** it will Sync the data in demo server.
* Dim\_date\_Sync2\_preprod.
* Dim\_month\_Sync2\_preprod.
* Dim\_Party\_group\_Sync2\_preprod.
* Dim\_party\_org\_Sync2\_preprod.
* Dim\_Party\_Sync2\_preprod.
* Dim\_Prd\_Sync2\_preprod.
* Dim\_slu\_Sync2\_preprod.
* Dim\_Vendor\_Group\_Sync2\_preprod.
* Dim\_Vendor\_Sync2\_preprod

**Screenshots:**

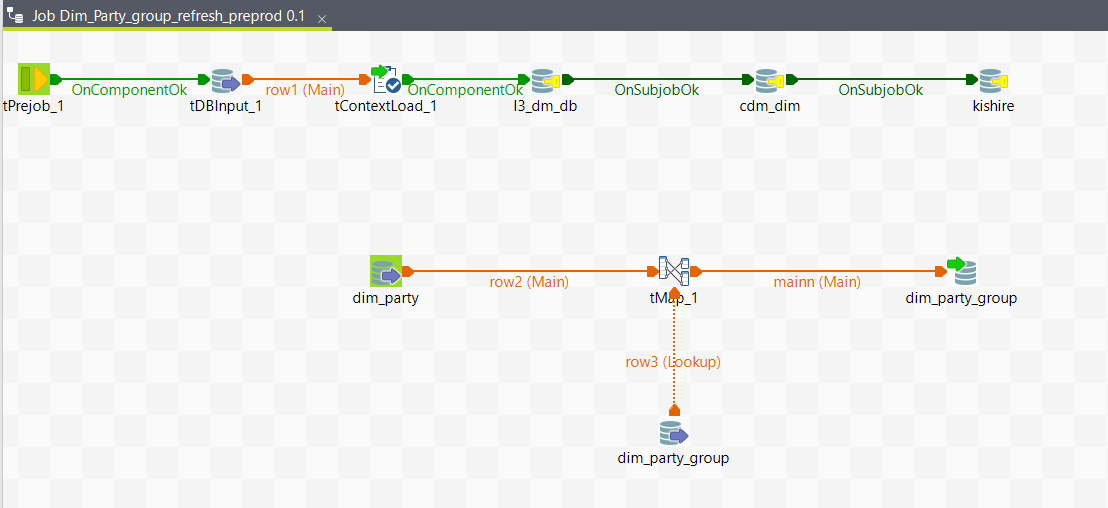
**Dim\_date\_Refresh\_preprod:**



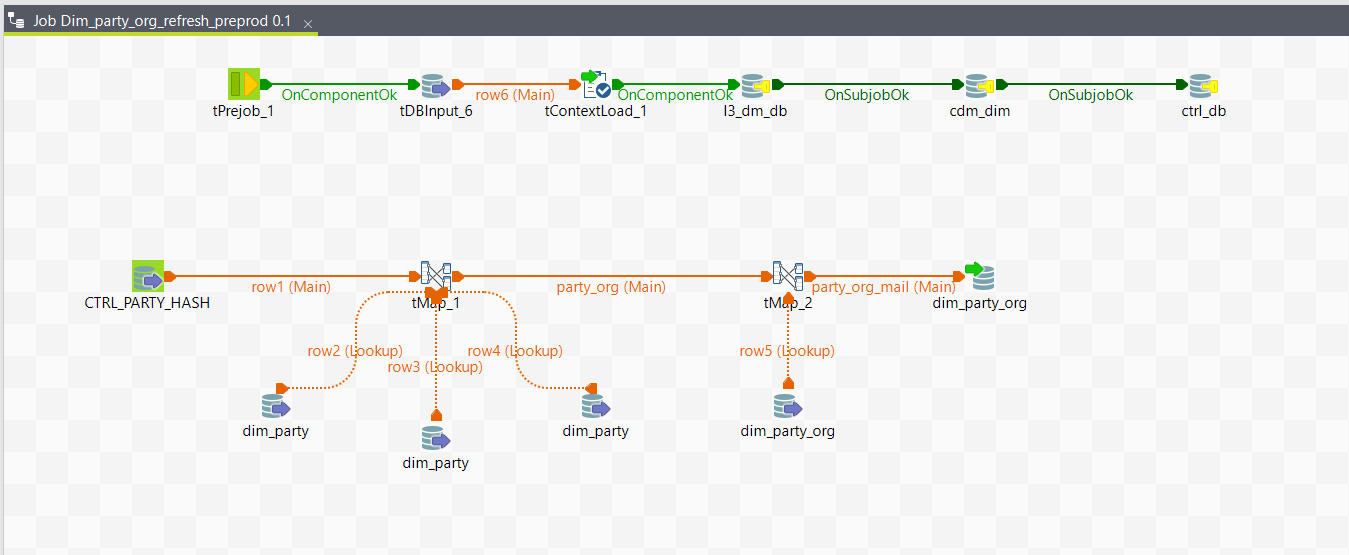
**Dim\_month\_Refresh\_preprod:**



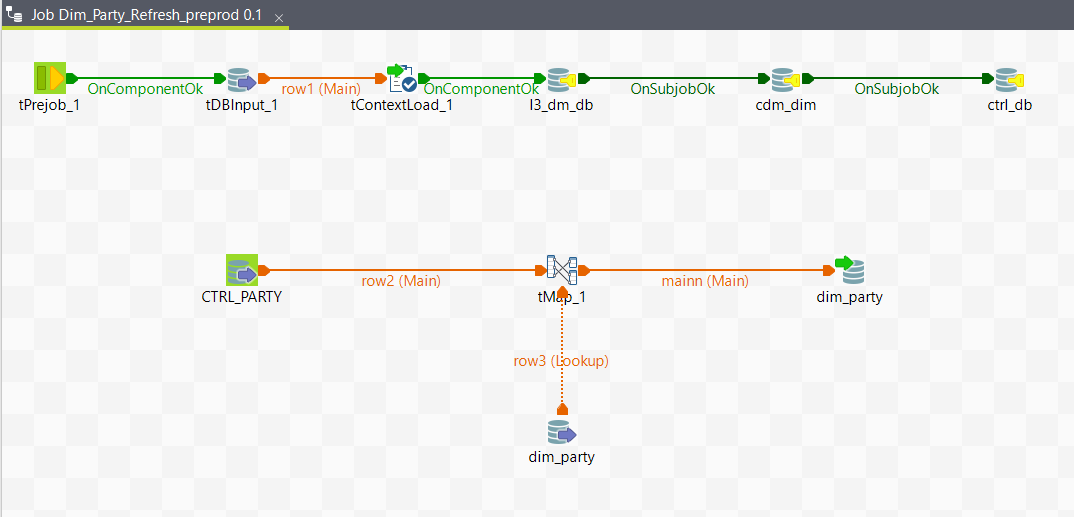
**Dim\_Party\_group\_refresh\_preprod:**



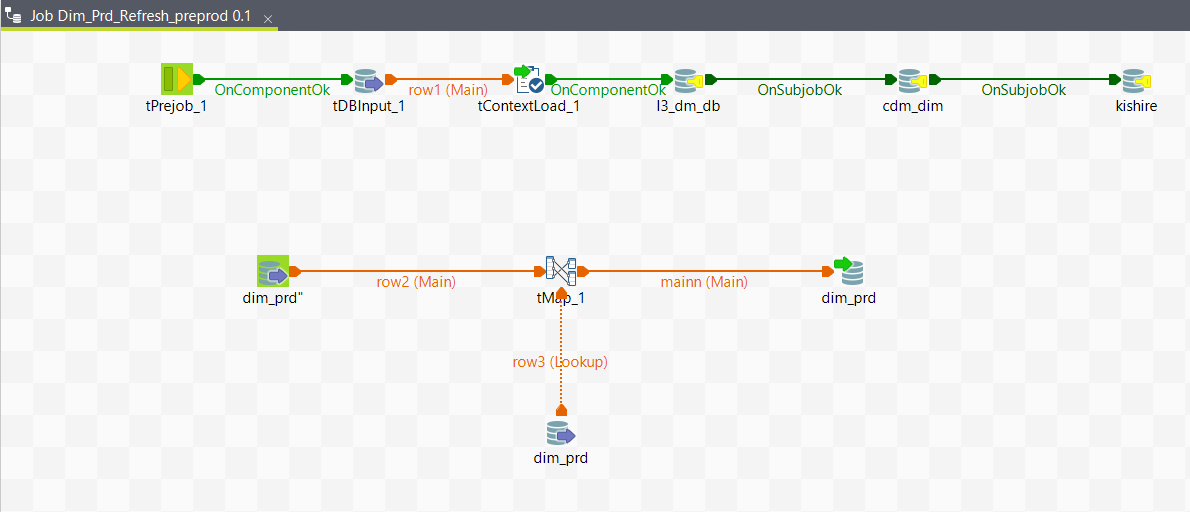
**Dim\_party\_org\_refresh\_preprod:**



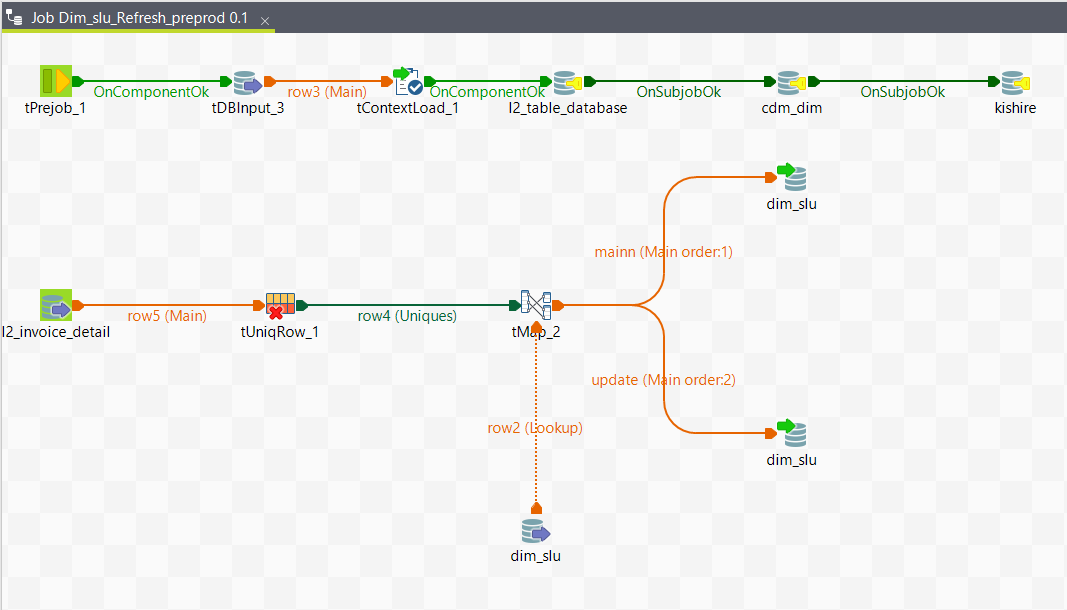
**Dim\_Party\_Refresh\_preprod:**



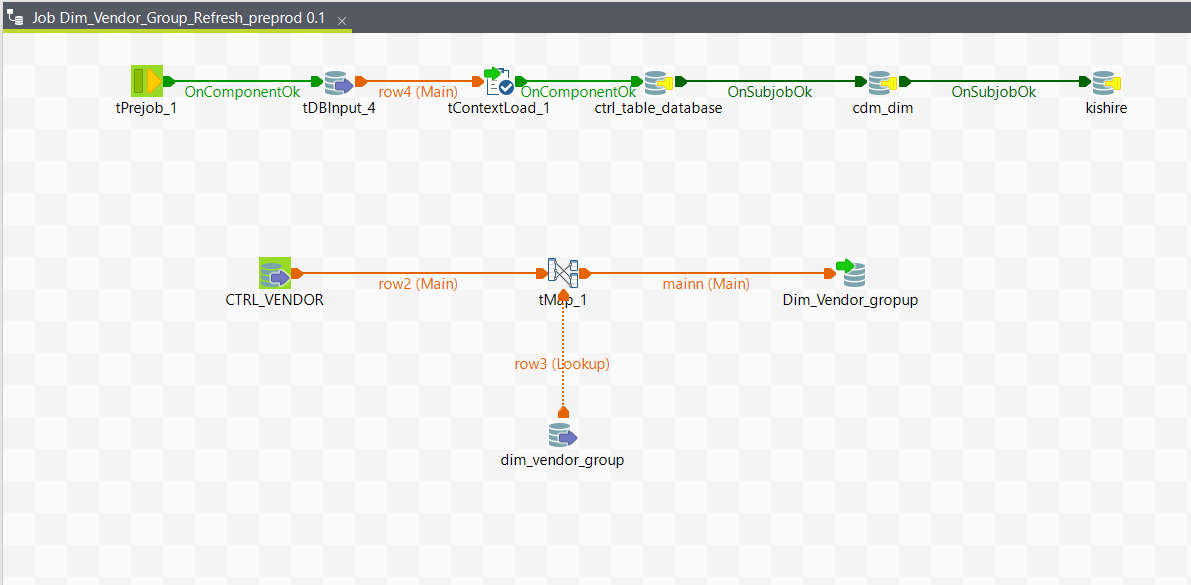
**Dim\_Prd\_Refresh\_preprod:**



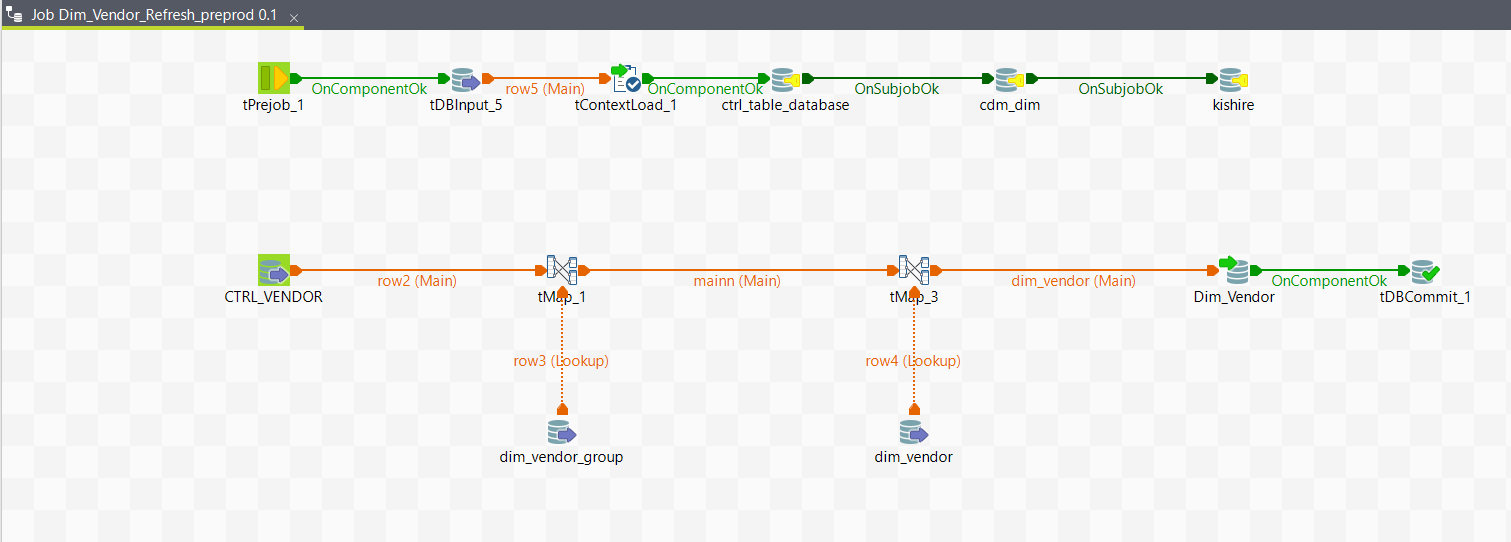
**Dim\_slu\_Refresh\_preprod:**



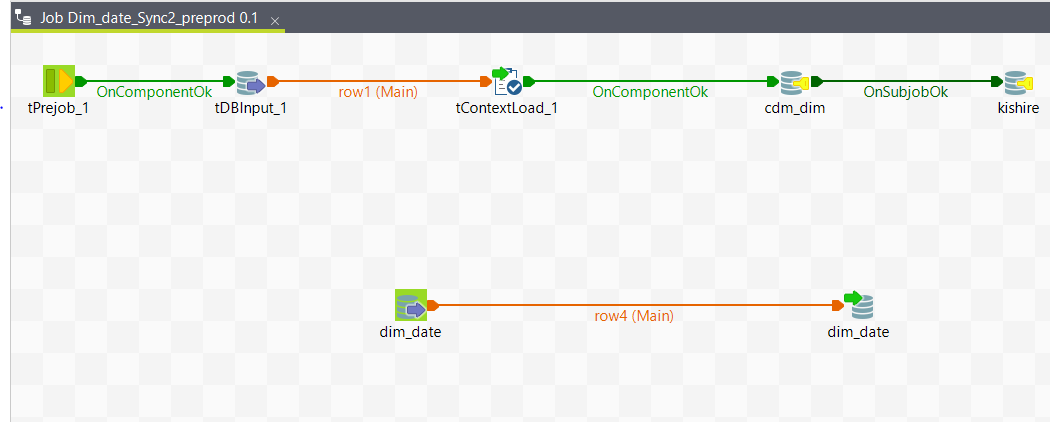
**Dim\_Vendor\_Group\_Refresh\_preprod:**



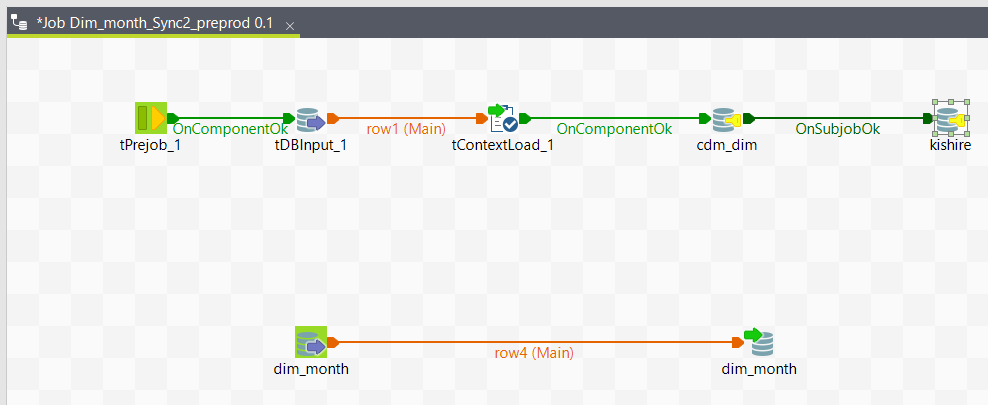
**Dim\_Vendor\_Refresh\_preprod:**



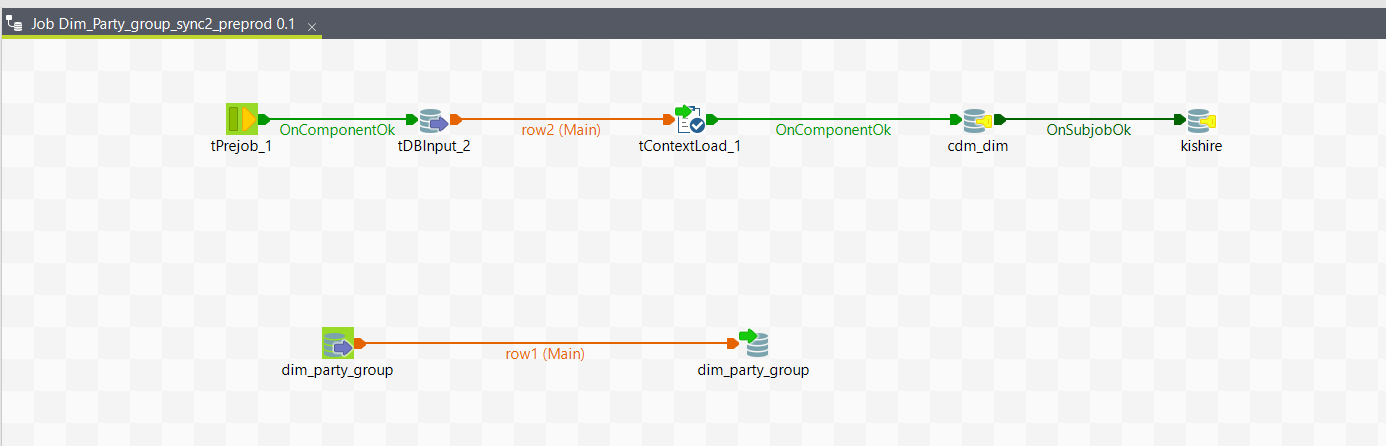
**Dim\_date\_Sync2\_preprod:**



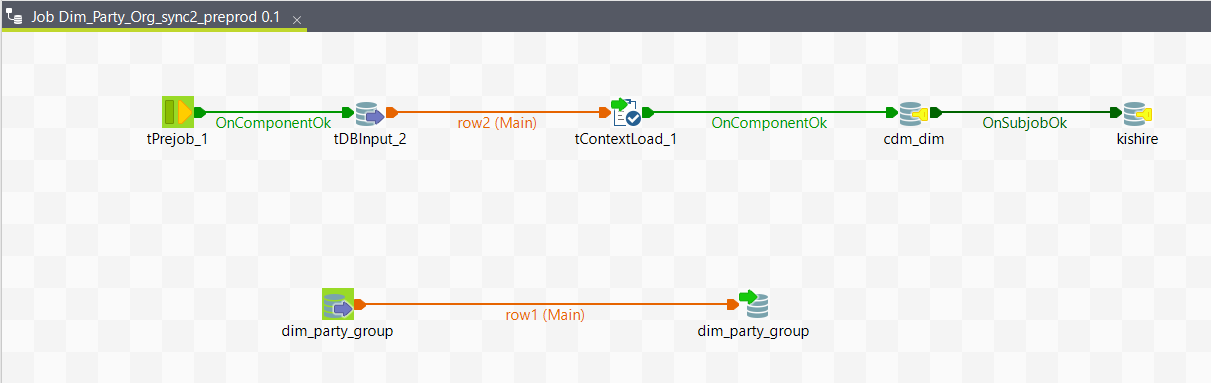
**Dim\_month\_Sync2\_preprod:**



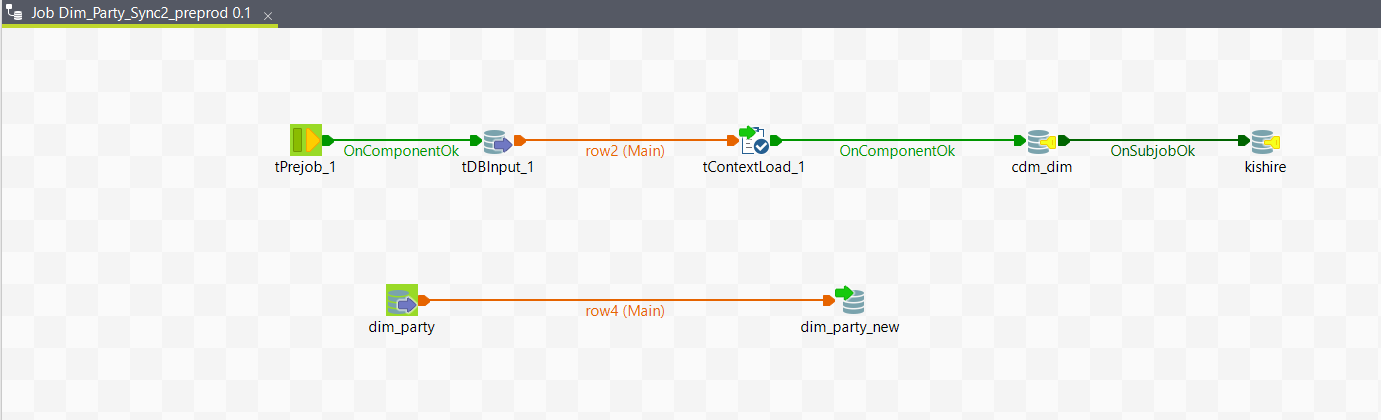
**Dim\_Party\_group\_Sync2\_preprod:**



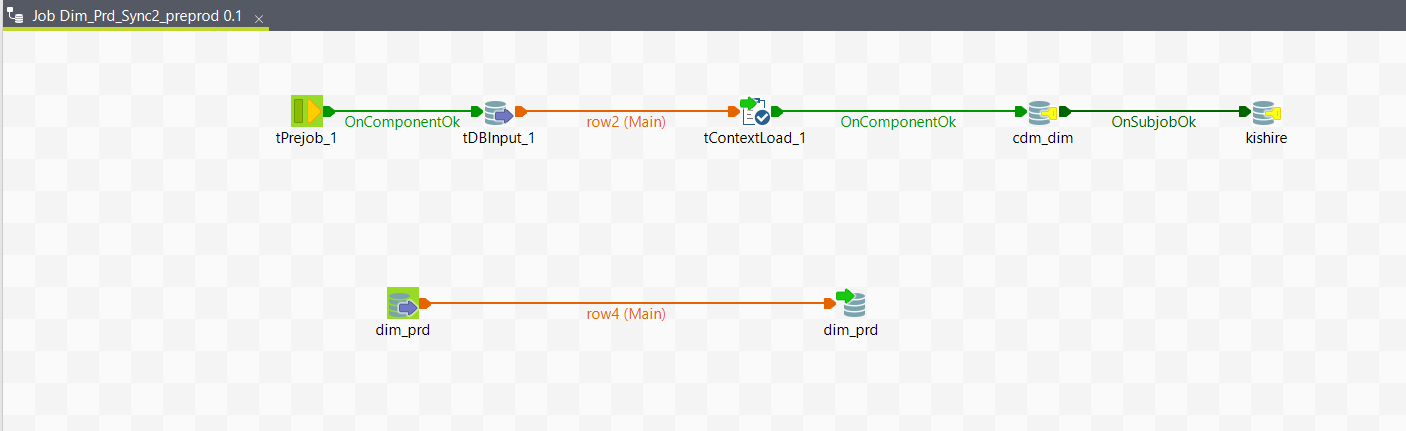
**Dim\_party\_org\_Sync2\_preprod:**



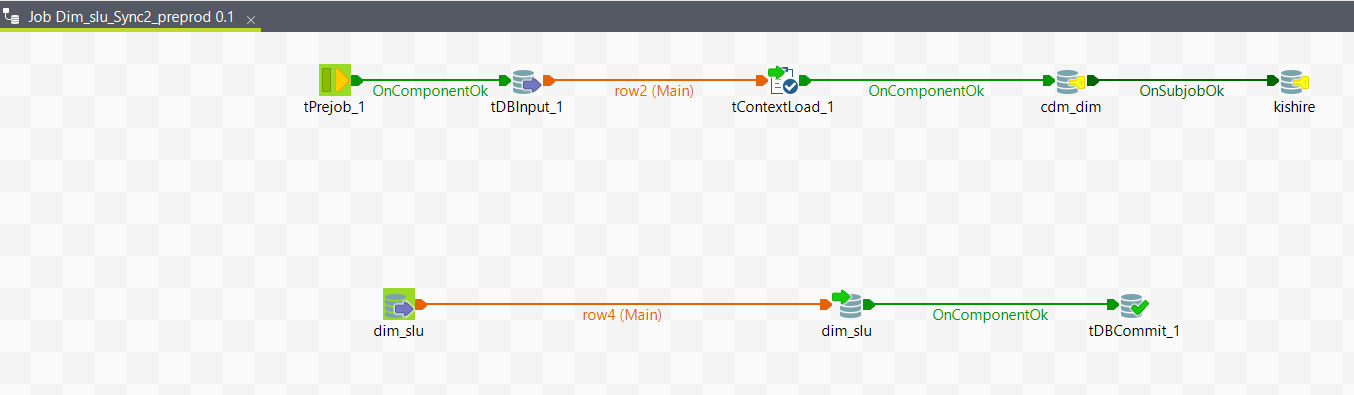
**Dim\_Party\_Sync2\_preprod:**



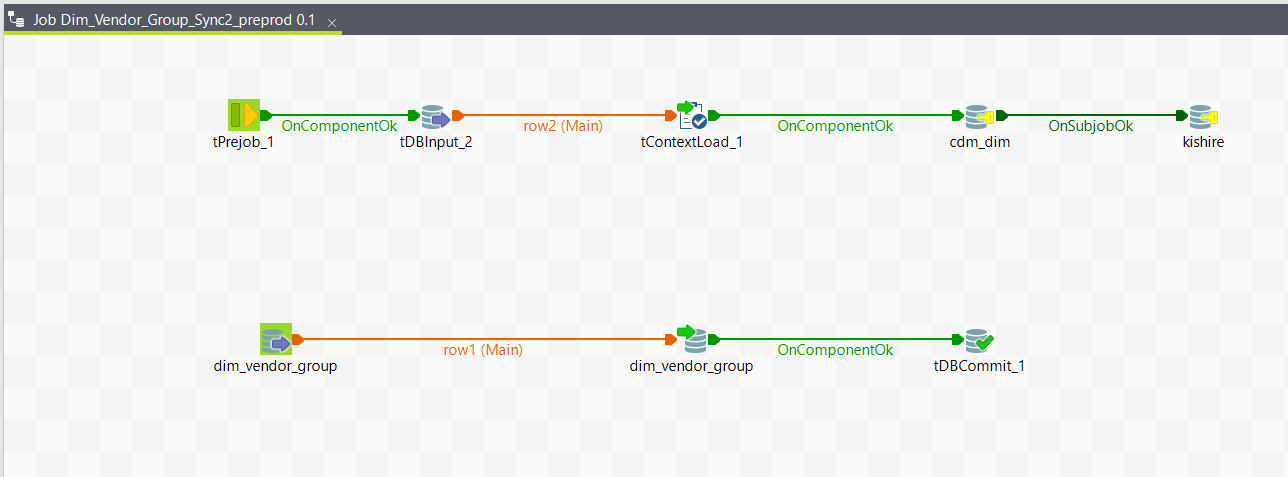
**Dim\_Prd\_Sync2\_preprod:**



**Dim\_slu\_Sync2\_preprod:**



**Dim\_Vendor\_Group\_Sync2\_preprod:**



**Dim\_Vendor\_Sync2\_preprod:**

