Talend Metadata Driven Process

Last updated by | Archana Balachandran | Jun 4, 2020 at 2:11 PM EDT

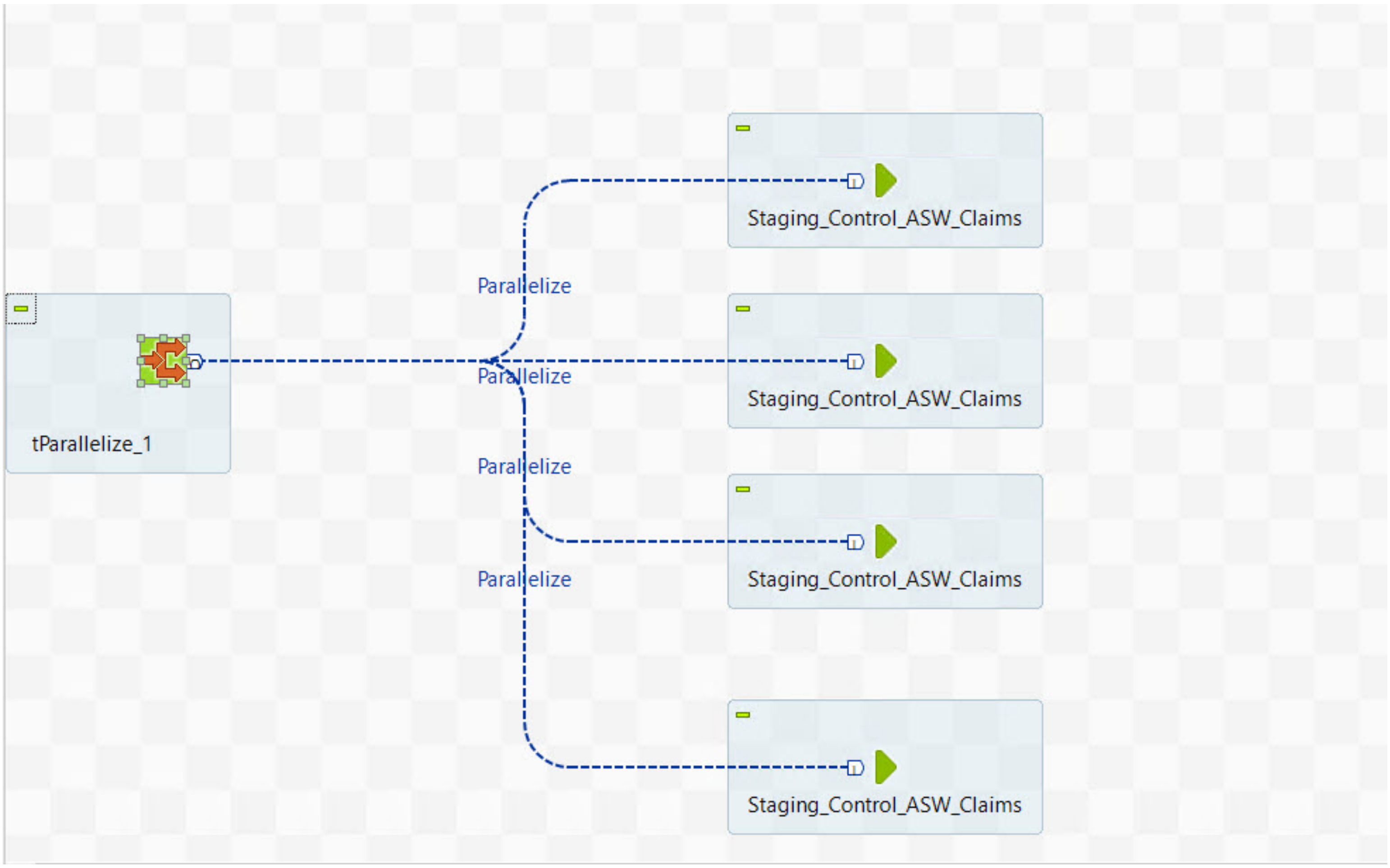
Contents

Problem Overview

Problem: Child jobs not running in correct context

The current orchestration Job design with the following set up (design attached):

- Multiple tRunJob components configured to run multiple sub-jobs. I've selected the 'QA' context in the context dropdown.
- tParallelize component that runs all the tRunJob components in parallel.


tParallelize_1
Parallelize
Parallelize
Parallelize
Parallelize
Staging_Control_ASW_Claims
Staging_Control_ASW_Claims
Staging_Control_ASW_Claims
Staging_Control_ASW_Claims

Now, when you publish this job to cloud and promote through various higher environments - the context selected in tRunJob remains the same and does not dynamically change based on the cloud environment in exists in.

What you need is: when you publish the job design to a cloud environment, the tRunJob component should dynamically pick up the current execution environment. How can you achieve this in Talend Studio, without manually changing the context in tRunJob component?

A case was opened with Talend Support to solve this issue (Case 00171909) and the support representative informed us that there is a Feature Request for this change on Talend's end, which is a few months away.

Solution: Metadata driven jobs using tContextLoad

Overview:

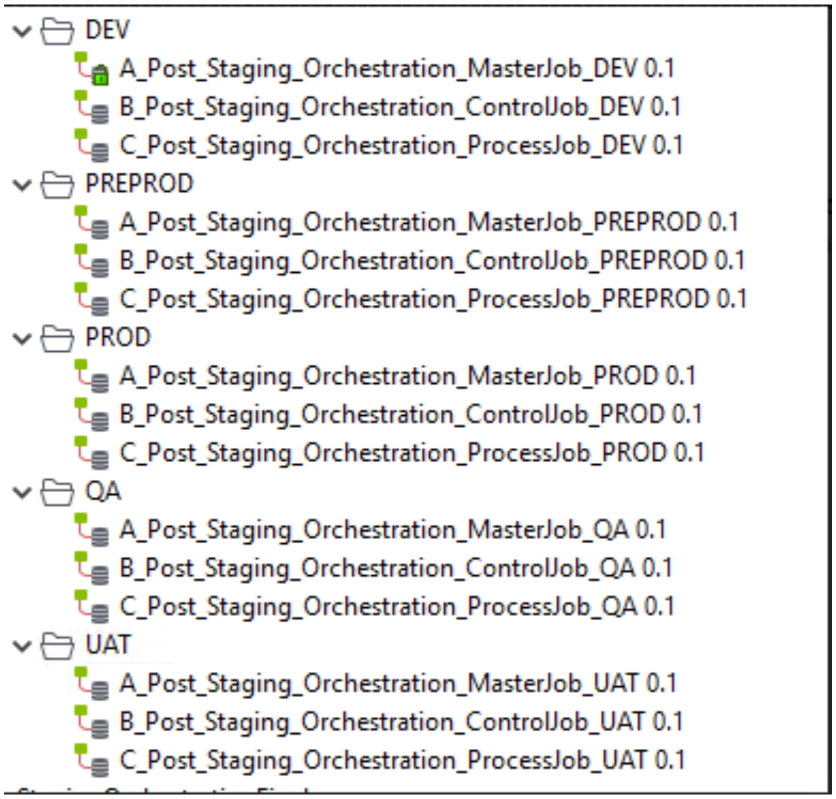
We have introduced two components in the PreJob step to fetch the required context variables for the job being executed at run-time and applying it to the job. Snowflake table Metadata.<environment>.talend_driver table stores all the values, and is queried by tSnowflakeRow component. The result of this query, which is a <key,value> pair of context variables are passed on to tContextLoad component which sets the variables.

Design Optimization: Metadata Driven Orchestration

Existing Setup:

Currently for orchestration, each environment has its own design (as shown in the image below). This is difficult to maintain because each design change had to be manually applied to all designs manually, and then published to all environments separately.

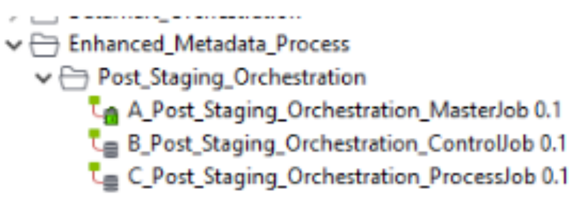
It was created in this manner because of a known Talend limitation where the context values(or environment variables) were not dynamically getting passed from parent to its nested child jobs. As per Talend Support team, a feature request for this fix is a few months away.



```
DEV
    A_Post_Staging_Orchestration_MasterJob_DEV 0.1
    B_Post_Staging_Orchestration_ControlJob_DEV 0.1
    C_Post_Staging_Orchestration_ProcessJob_DEV 0.1
PREPROD
    A_Post_Staging_Orchestration_MasterJob_PREPROD 0.1
    B_Post_Staging_Orchestration_ControlJob_PREPROD 0.1
    C_Post_Staging_Orchestration_ProcessJob_PREPROD 0.1
PROD
    A_Post_Staging_Orchestration_MasterJob_PROD 0.1
    B_Post_Staging_Orchestration_ControlJob_PROD 0.1
    C_Post_Staging_Orchestration_ProcessJob_PROD 0.1
QA
    A_Post_Staging_Orchestration_MasterJob_QA 0.1
    B_Post_Staging_Orchestration_ControlJob_QA 0.1
    C_Post_Staging_Orchestration_ProcessJob_QA 0.1
UAT
    A_Post_Staging_Orchestration_MasterJob_UAT 0.1
    B_Post_Staging_Orchestration_ControlJob_UAT 0.1
    C_Post_Staging_Orchestration_ProcessJob_UAT 0.1
```

Enhanced Setup:

This enhancement was applied In order to create an environment-agnostic Orchestration design, which allows: Simple and efficient design - one design which applies to all environments Easy maintenance (a change applied in design applies to all environments) Easy Publishing (one design published to DEV cloud, and can then be seamlessly promoted to higher environments within TMC) Less design errors due to change application in each design for every environment

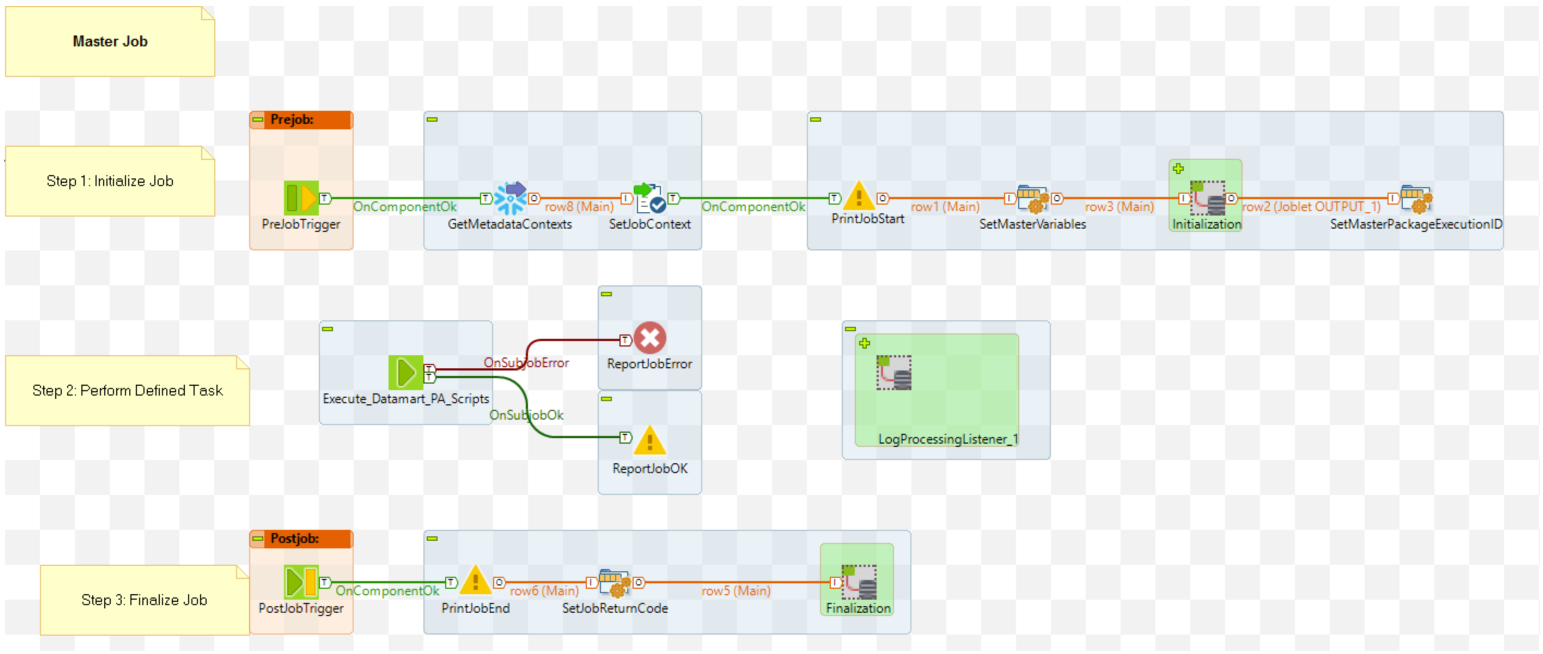


```
Enhanced_Metadata_Process
    Post_Staging_Orchestration
        A_Post_Staging_Orchestration_MasterJob 0.1
        B_Post_Staging_Orchestration_ControlJob 0.1
        C_Post_Staging_Orchestration_ProcessJob 0.1
```

We're leveraging the Metadata database in Snowflake, which stores all the required environment variables in a table. So when a Talend job is kicked off, the first step in the process is to load the context variables from the Snowflake table based on the execution environment. Therefore, only one job design is required to handle execution in all environments due to this metadata-driven capability.

Once this design is published from Talend Studio to Talend Cloud Dev Environment, we can use the Promotions tab to seamlessly promote the design to all other environments.

This enhanced design is applied to all Master, Control and Process job designs.



In snowflake talend_driver table, each schema represents the environment.

Select query from Talend's tSnowflakeRow for a job named 'DataMart_RM_Claim' will look like:

```
 SELECT parameter_name,parameter_value FROM talend_driver where job_name in ('Job Execution
Control', 'DataMart_RM_Claim') and environment=context.environment;
```

Note: 'Job Execution Control' is a default job that contains context values for ETLconfig table parameters which are necessary in all job designs to initialize and finalize the job.

Creating entries for a new Environment

Step 1: Create the talend_driver table in Snowflake

```
CREATE OR REPLACE TABLE METADATA.UAT.TALEND_DRIVER
(parameter_id varchar,
environment varchar,
parameter_name varchar,
parameter_value varchar,
parameter_desc varchar,
parameter_type varchar,
job_name varchar);
```

Step 2: Provide necessary Grants for Talend_metadata_role

```
GRANT USAGE ON DATABASE METADATA TO ROLE TALEND_METADATA_ROLE;
GRANT USAGE on schema METADATA.<environment> TO ROLE TALEND_METADATA_ROLE;
GRANT insert,update,DELETE,truncate,select on all tables in schema metadata.< environment> to role
GRANT CREATE TABLE ON SCHEMA metadata.< environment>  TO ROLE talend_metadata_role;
GRANT ROLE talend_metadata_role to USER talend_metadata_user;
```

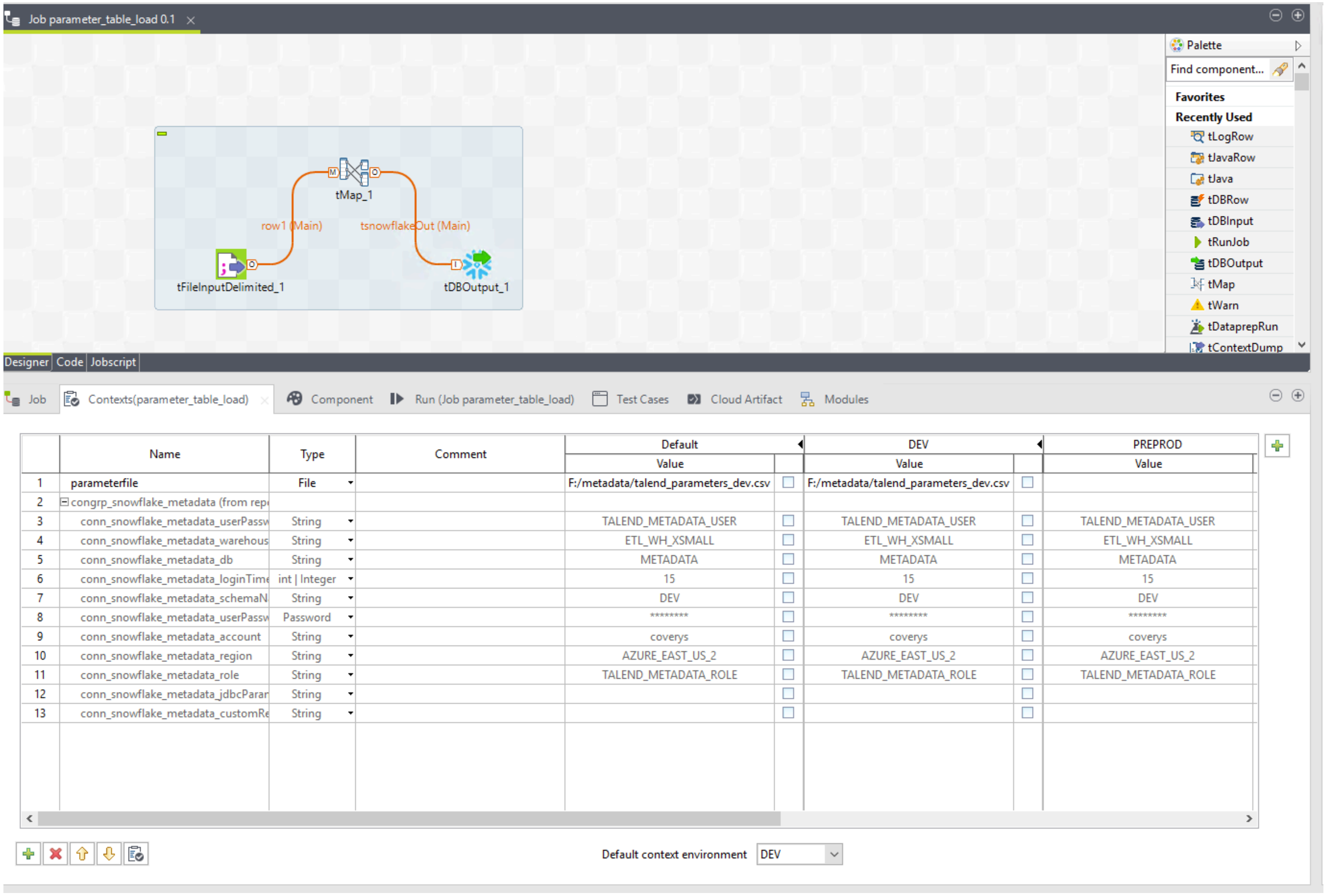
Step 3: Create a file (pipe-delimited, .csv) containing all the required parameters and place the file in F:\metadata folder in the remote VM. Note: In order to run the parameter_table_load job in cloud, you need to place this file in the same server where the remote engine for that environment is installed in.

For example, this is what the content of the file will look like:

```
PARAMETER_ID|ENVIRONMENT|PARAMETER_NAME|PARAMETER_VALUE|PARAMETER_DESC|PARAMETER_TYPE|JOB_NAME
1|UAT|conn_sqlserver_asw_party_host|172.17.45.47|SQL Server connection details for ASW_Party jobs|
2|UAT|conn_sqlserver_asw_party_port|21433|SQL Server connection details for ASW_Party jobs|Int|ASW
3|UAT|conn_sqlserver_asw_party_schema|dbo|SQL Server connection details for ASW_Party jobs|String|
4|UAT|conn_sqlserver_asw_party_database|PROD_PARTY00_RUN|SQL Server connection details for ASW_Par
5|UAT|conn_sqlserver_asw_party_username|CoverysETL_Snowflake|SQL Server connection details for ASW
6|UAT|conn_sqlserver_asw_party_table_name|PARTY_MBRSHP|SQL Server connection details for ASW_Party
```

Step 4: Load the data into talend_driver table from Talend

In Talend Studio, navigate to the job design named 'paramater_table_load' in Templates /MetadataDrivenOrchstration folder.



	Name	Type	Comment	Default		DEV		PREPROD
				Value		Value		Value
1	parameterfile	File		F:/metadata/talend_parameters_dev.csv	☐	F:/metadata/talend_parameters_dev.csv	☐	
2	⊟ congrp_snowflake_metadata (from rep							
3	conn_snowflake_metadata_userPassw	String		TALEND_METADATA_USER	☐	TALEND_METADATA_USER	☐	TALEND_METADATA_USER
4	conn_snowflake_metadata_warehous	String		ETL_WH_XSMALL	☐	ETL_WH_XSMALL	☐	ETL_WH_XSMALL
5	conn_snowflake_metadata_db	String		METADATA	☐	METADATA	☐	METADATA
6	conn_snowflake_metadata_loginTime	int | Integer		15	☐	15	☐	15
7	conn_snowflake_metadata_schemaN	String		DEV	☐	DEV	☐	DEV
8	conn_snowflake_metadata_userPassw	Password		********	☐	********	☐	********
9	conn_snowflake_metadata_account	String		coverys	☐	coverys	☐	coverys
10	conn_snowflake_metadata_region	String		AZURE_EAST_US_2	☐	AZURE_EAST_US_2	☐	AZURE_EAST_US_2
11	conn_snowflake_metadata_role	String		TALEND_METADATA_ROLE	☐	TALEND_METADATA_ROLE	☐	TALEND_METADATA_ROLE
12	conn_snowflake_metadata_jdbcParar	String			☐		☐	
13	conn_snowflake_metadata_customRe	String			☐		☐	

Default context environment: DEV

Switch to the Context tab, and create a new context/environment by clicking the '+' icon on the right corner. Enter the environment's variables, mainly the parameterfilename pointing to the file you created in Step 3.

Once you're ready to load the data, switch 'Run' tab and select the new Context/Environment you just created and Run the Job. Verify the data in Snowflake after the job has successfully completed.

Orchestration Processes following the Enhanced Design

The enhanced process was applied to the following Orchestration processes:

Script-driven processes:

1. Post Deployment Orchestration
2. Post Staging Orchestration
3. Transform Orchestration
4. Datamart PA Orchestration
5. Datamart RM Orchestration

6. CLH Orchestration
7. GL Extracts Orchestration
8. Downstream Orchestration

Talend Job -driven processes:

1. Datamart SQLServerLoad Orchestration

Staging Orchestration

Staging Orchestration is yet to be converted into the new process. By converting the staging process to the enhanced process, key points to note:

1. It will bring down the number of processes per environment to 'one process for all environments'
2. We will still have 1 Staging Master job, and 5 Staging Control Jobs due to tRunJob component's limitation.

Elaborating on the 2nd point mentioned above: If a tRunJob has too many Child Jobs selected, the Talend Job may fail to compile, and produce the following exception:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The code of method tFileInputDelimited_1Process(Map) is exceeding the 65535 bytes limit
```

This is because Talend Studio is a code generator and creates Java code for each Talend Job. Each subjob is a method in the Job class. If a subjob is too big, the size of the final generated code will exceed 65536 bytes.

Datamart_SQLSvrLoad_Orchestration job designs can be used to replicate the enhanced process for Staging jobs.

Steps to convert Staging Orchestration to follow the Enhanced Metadata driven process

Part A: Create Talend Job Designs for Master, Control processes.

Start with creating a new folder named 'Staging_Orchestration in Enhanced_Metadata_Process folder. Create 5 blank job designs with the following names:

- A_Staging_Orchestration_MasterJob
- B_Staging_Orchestration_ControlJob_1
- B_Staging_Orchestration_ControlJob_2
- B_Staging_Orchestration_ControlJob_3
- B_Staging_Orchestration_ControlJob_4
- B_Staging_Orchestration_ControlJob_5

Part B: Configure the Master Job A_Staging_Orchestration_MasterJob

Step 1: Add the required Context Variables and Groups

In the contexts tab, add the following context variables:

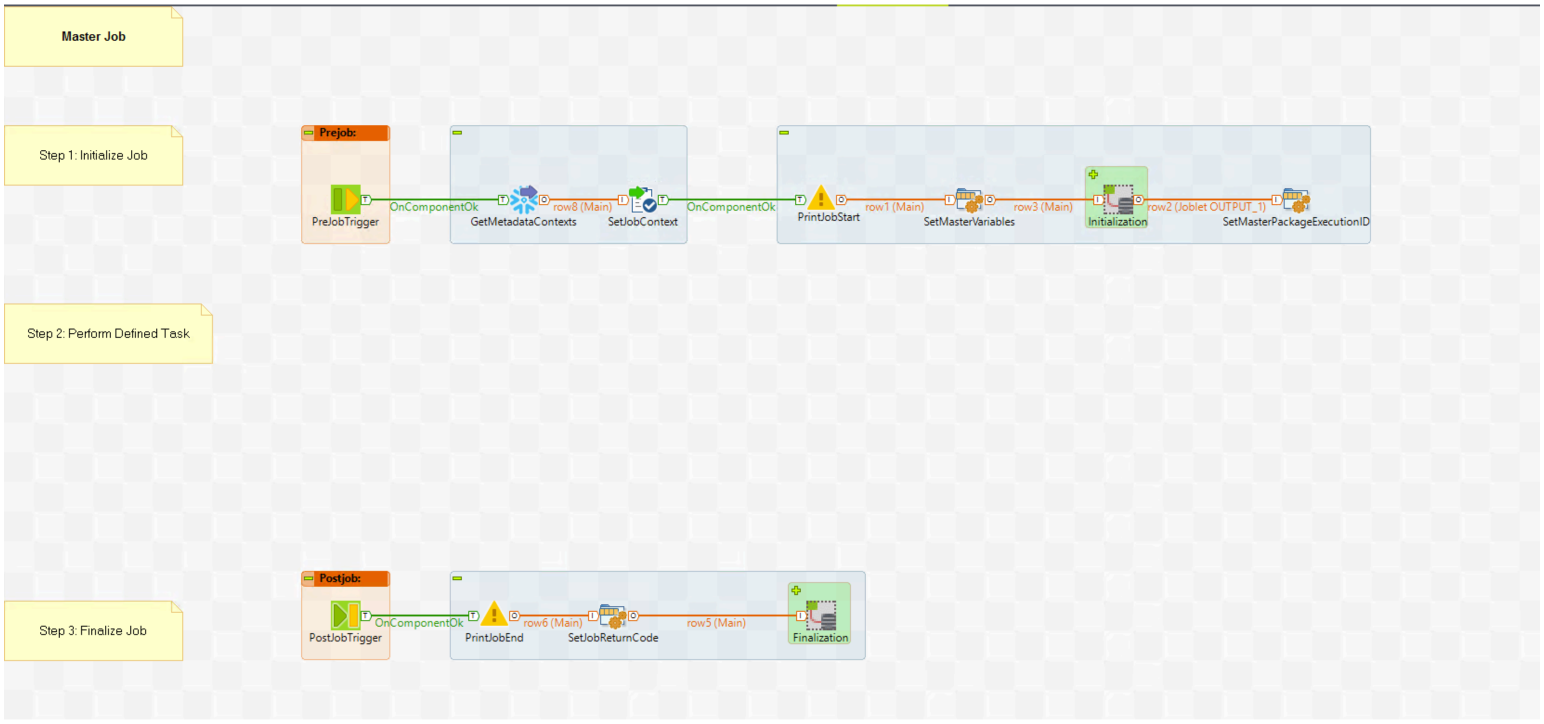
master_job_name : provide value as CoverysEDW_Staging_Master for all environments environment : provide value based on each environment (DEV,QA,UAT,PREPROD,PROD)

Then add the context groups:

ConGrp_Snowflake_ETLConfig_CONTROL congrp_snowflake_metadata Job_Configuration

Step 2: Add PreJob and PostJob components, along with LogProcessingListener Joblet

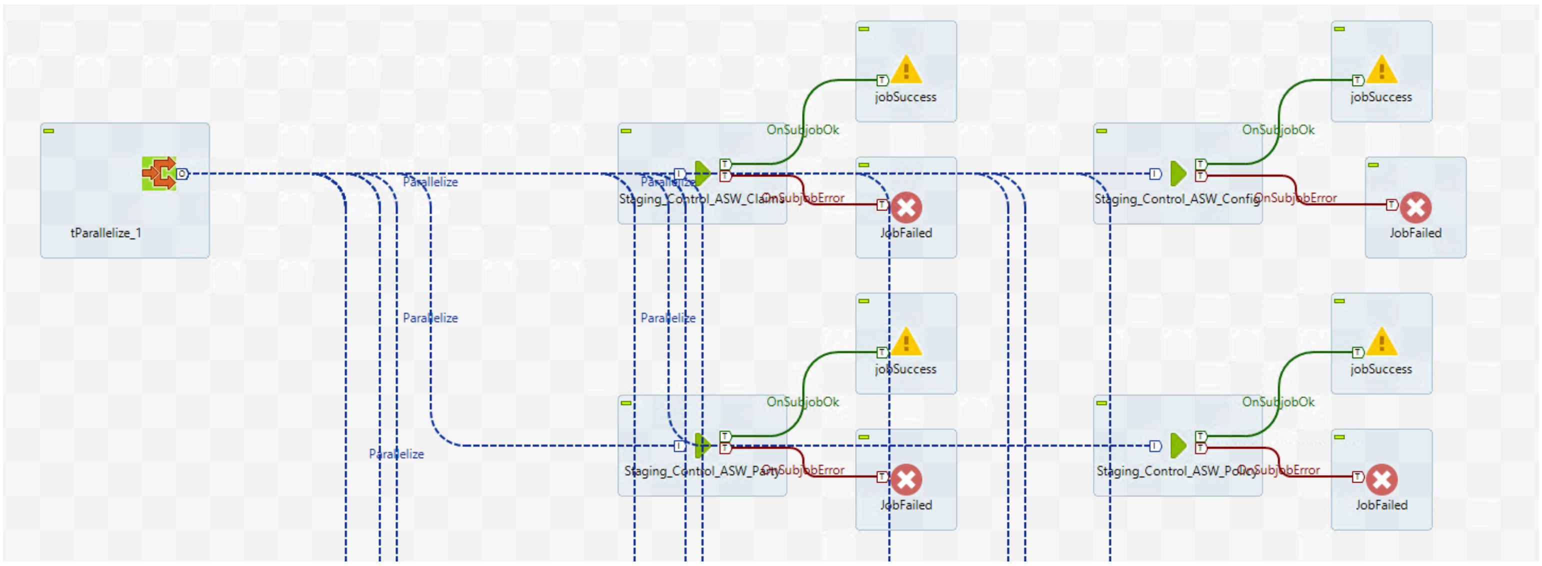
Paste the PreJob and PostJob pieces (along with all the components attached to these) from the Template job named 'TEMPLATE_A_Orchestration_Template_Master' in Templates/MetadatadrivenOrchestration folder. Also include the LogProcessingListener Joblet.



Step 3: Add tRunJob components that calls all Control Jobs

From the original Staging Orchestration design, copy the tParallize components, along with all the tRunJob components for controlJobs and the tWarn and tDie components attached.

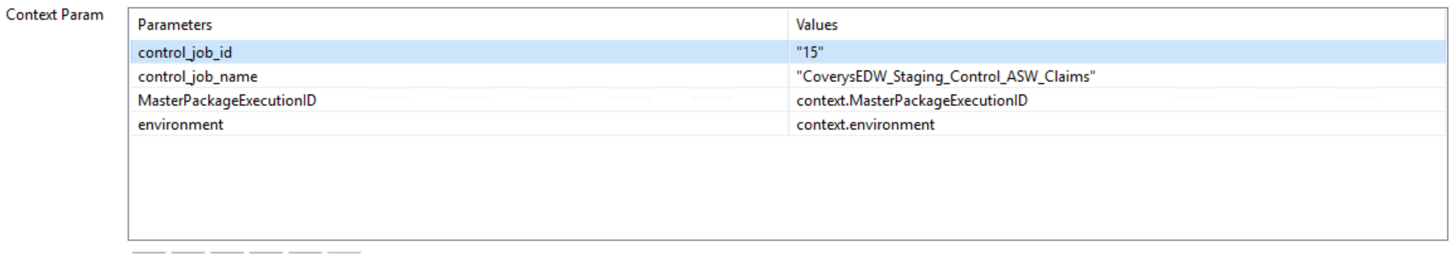
Paste the selection into the new Master Job design.


tParallelize_1
Parallelize
Staging_Control_ASW_Claims
Staging_Control_ASW_Config
Staging_Control_ASW_Party
Staging_Control_ASW_Policy
OnSubjobOk
OnSubjobError
jobSuccess
JobFailed

Step 4: Configure each tRunJob component

For each of the tRunJob component for each control schemas:

1. Add a new context parameter called 'environment' and set the value as context.environment.
2. Also ensure that the control_job_id value is specified in double quotes (shown below).



Context Param

Parameters	Values
control_job_id	"15"
control_job_name	"CoverysEDW_Staging_Control_ASW_Claims"
MasterPackageExecutionID	context.MasterPackageExecutionID
environment	context.environment

3. Select the correct ControlJob to be called (refer the table below):

Control Schema	Control Job Name
ASW_Claims	B_Staging_Orchestration_ControlJob_1
ASW_Config	B_Staging_Orchestration_ControlJob_3
ASW_Party	B_Staging_Orchestration_ControlJob_4
ASW_Policy	B_Staging_Orchestration_ControlJob_4
ASW_Shred	B_Staging_Orchestration_ControlJob_2
Clinical_Coding	B_Staging_Orchestration_ControlJob_1

Conformance	B_Staging_Orchestration_ControlJob_3
CSRP_ASW_Reporting	B_Staging_Orchestration_ControlJob_1
GoalWorkbook	B_Staging_Orchestration_ControlJob_1
Integrations	B_Staging_Orchestration_ControlJob_1
IS4Claims	B_Staging_Orchestration_ControlJob_1
MHAIC	B_Staging_Orchestration_ControlJob_3
OA_RM	B_Staging_Orchestration_ControlJob_4
PPIC	B_Staging_Orchestration_ControlJob_3
PPIC	B_Staging_Orchestration_ControlJob_3
PPIC	B_Staging_Orchestration_ControlJob_3
PPIC	B_Staging_Orchestration_ControlJob_3
SharePoint	B_Staging_Orchestration_ControlJob_1
Utility	B_Staging_Orchestration_ControlJob_1
XrefMapping	B_Staging_Orchestration_ControlJob_1
IS4West	B_Staging_Orchestration_ControlJob_5
IS4East	B_Staging_Orchestration_ControlJob_5

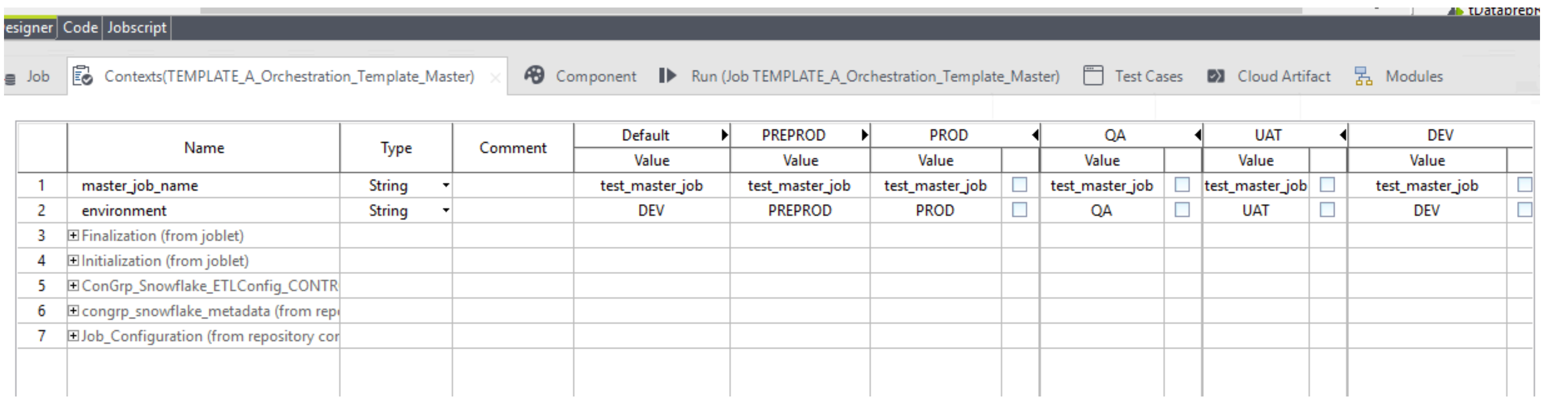
Step 5: Modify the code in PreJob section tJavaRow component

Select the tJavaRow component attached to PostJob component, and replace the code with the code below:

```
output_row.PackageExecutionID = String.valueOf(context.MasterPackageExecutionID);
output_row.packagename=context.JobName;

//Instantiate an Iterator to iterate through the globalMap keys
java.util.Iterator<String> it = globalMap.keySet().iterator();

//Instantiate an int variable for errors
int error = 0;
int error_code_int=0;
//Iterate through the keys until they are all done or error is something
//other than 0
while(it.hasNext()&&error==0){
String key = it.next();

//Check to see if the key contains "_DIE_CODE"
if(key.indexOf("_DIE_CODE")>-1)

{
//Retrieve the DIE_CODE and set it to error
error = ((Integer)globalMap.get(key)).intValue();
//Retrieve the Error Message
String component = key.replaceAll("_DIE_CODE", "");
String message = ((String)globalMap.get(component+"_DIE_MESSAGE"));
System.out.println("Printing from PostJob: "+(String)globalMap.get(component+"_DIE_MESSAGE"));
}

}

//If an error of value other than 0 is found, set the errorCode value
if(error!=0)

{ //status = "failure";
error_code_int = new Integer(error);
System.out.println("Errors Encountered!!");

}
else
{
System.out.println("No errors!!!");
error_code_int=2;
}

System.out.println("Final Error code is:"+error_code_int);
output_row.error_code=Integer.toString(error_code_int);
```

This will allow the ReturnCode values from all the tDie components in the job design.

Step 6: Save job design. Save and re-open the job design and confirm that the Context Groups from Initialize and Finalize Joblets are included in the Contexts tab.


esigner
Code
Jobscript
Job
Contexts(TEMPLATE_A_Orchestration_Template_Master)
Component
Run (Job TEMPLATE_A_Orchestration_Template_Master)
Test Cases
Cloud Artifact
Modules

	Name	Type	Comment	Default	PREPROD	PROD		QA		UAT		DEV	
				Value	Value	Value		Value		Value		Value	
1	master_job_name	String		test_master_job	test_master_job	test_master_job	☐	test_master_job	☐	test_master_job	☐	test_master_job	☐
2	environment	String		DEV	PREPROD	PROD	☐	QA	☐	UAT	☐	DEV	☐
3	⊞Finalization (from joblet)												
4	⊞Initialization (from joblet)												
5	⊞ConGrp_Snowflake_ETLConfig_CONTR												
6	⊞congrp_snowflake_metadata (from rep												
7	⊞Job_Configuration (from repository cor												

Part C: Configure Control Job

As mentioned in the previous section, there are 5 Control Jobs required for the Staging process in order to accommodate all 400+ staging jobs. We need to open each of the following control jobs and configure them.

- B_Staging_Orchestration_ControlJob_1
- B_Staging_Orchestration_ControlJob_2
- B_Staging_Orchestration_ControlJob_3
- B_Staging_Orchestration_ControlJob_4
- B_Staging_Orchestration_ControlJob_5

The difference in each of the designs lie in the Staging jobs selected in their tRunJob components.

Open each design and perform the following steps:

Step 1: Add the required Context Variables and Groups

Add the following context variable: environment. Provide values based on each environment (DEV,QA,UAT,PREPROD,PROD) Add the following context groups:

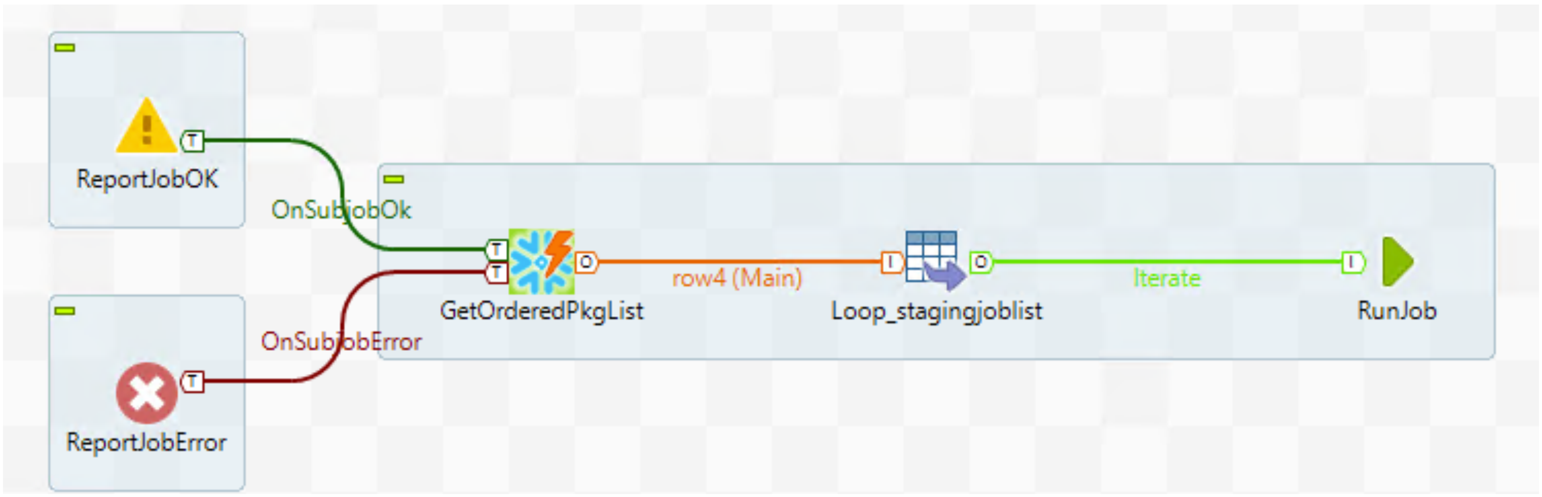
1. ConGrp_Snowflake_ETLConfig_CONTROL (from repository context)
2. congrp_snowflake_metadata (from repository context)
3. Job_Configuration (from repository context)

Step 2: Add PreJob, PostJob and LogProcessingListener components

Paste the PreJob and PostJob pieces (along with all the components attached to these) from the Template job named 'TEMPLATE_B_Orchestration_Template_Control' in Templates/MetadatadrivenOrchestration folder. Also include the LogProcessingListener joblet.

Step 3: Add Control Orchestration components

From the Datamart job 'B_Datamart_SQLSvrLoad_Orchestration_ControlJob', copy the control job driver components and paste it into the new job design.



Step 4: Configure tRunJob component

In the Context Param section, add a new context parameter called 'environment' and set the value as context.environment. You can leave the Context field as 'Default'. In the Job list, include the process/staging Talend Job names. For each of the 5 control Job designs, the tRunJob component has enabled different sets of jobs as indicated in the table specified in Part B Step 4.

Step 5: Save job design

Save and reopen the design to ensure the context groups from the Initialize and Finalize jobless are present in the Context tab.

Part D: Configure the Individual Staging Jobs

Open each job in Staging_Load folder, and apply the following changes. For reference, open any Talend Job in Datamarts folder.

Step 1: Add required context groups and variables

1. congrp_snowflake_metadata
2. Job_Configuration NOTE: ensure the 'environment' context variable is selected and included in the Job.

Step 2: Add PreJob, PostJob and LogProcessingListener components

Paste the PreJob and PostJob pieces (along with all the components attached to these) from the Template job named 'TEMPLATE_C_Orchestration_Template_Process' in Templates/MetadatadrivenOrchestration folder. Also include the LogProcessingListener joblet.

Modify the SELECT query in tSnowflakeRow component in PreJob section (labeled GetMetadataContexts), paste the following:

```
"SELECT parameter_name,parameter_value
FROM talend_driver
WHERE job_name in ('Job Execution Control','"+jobName+"')
AND environment='"+context.environment+ "';"
```

Step 3: Add context variable entries for each job in talend_driver table for all environments.

For example, taking job 'DataMart_PA_Dim_RiskParty' as an example, the following entries were made in the talend_parameters_<environment>.csv for all environments.

Here, the input component tDBInput(snowflake) was using context group 'Conn_Snowflake_EDW_DataMart_PA' and the output component tDBOutput (Microsoft SQL Server) was using context group 'Conn_SQLServer_COV_SQL_DataMart_PA'. Therefore, in the parameter file, we need to include all the context variables for both these context groups.

```
394|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_AdditionalParams||SQL Server connection details for Dat
395|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_Database|DataMart_PA_SF|SQL Server connection details f
396|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_Login|TalendUser|SQL Server connection details for Data
397|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_Password|T@lend45n0w|SQL Server connection details for
398|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_Port|1433|SQL Server connection details for Datamart_PA
399|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_Schema|dbo|SQL Server connection details for Datamart_P
400|UAT|Conn_SQLServer_COV_SQL_DataMart_PA_Server|cov-sql-test.pmg.local|SQL Server connection det
401|UAT|Conn_Snowflake_EDW_DataMart_PA_account|coverys|Snowflake connection details for Datamart_P
402|UAT|Conn_Snowflake_EDW_DataMart_PA_customRegionID||Snowflake connection details for Datamart_P
403|UAT|Conn_Snowflake_EDW_DataMart_PA_db|UAT_EDW|Snowflake connection details for Datamart_PA job
404|UAT|Conn_Snowflake_EDW_DataMart_PA_jdbcParameters||Snowflake connection details for Datamart_P
405|UAT|Conn_Snowflake_EDW_DataMart_PA_loginTimeout|15|Snowflake connection details for Datamart_P
406|UAT|Conn_Snowflake_EDW_DataMart_PA_region|AZURE_EAST_US_2|Snowflake connection details for Dat
407|UAT|Conn_Snowflake_EDW_DataMart_PA_role|UAT_ETL_ROLE|Snowflake connection details for Datamart
408|UAT|Conn_Snowflake_EDW_DataMart_PA_schemaName|DATAMART_PA|Snowflake connection details for Dat
409|UAT|Conn_Snowflake_EDW_DataMart_PA_userPassword_password|M1gr4t3D4t4|Snowflake connection deta
410|UAT|Conn_Snowflake_EDW_DataMart_PA_userPassword_userId|UAT_TALEND|Snowflake connection details
411|UAT|Conn_Snowflake_EDW_DataMart_PA_warehouse|UAT_ETL_XSMALL|Snowflake connection details for D
```

Once the parameter file is complete, load the file into snowflake using parameter_table_load job.