**Custom Reporting Activities & Dashboard**

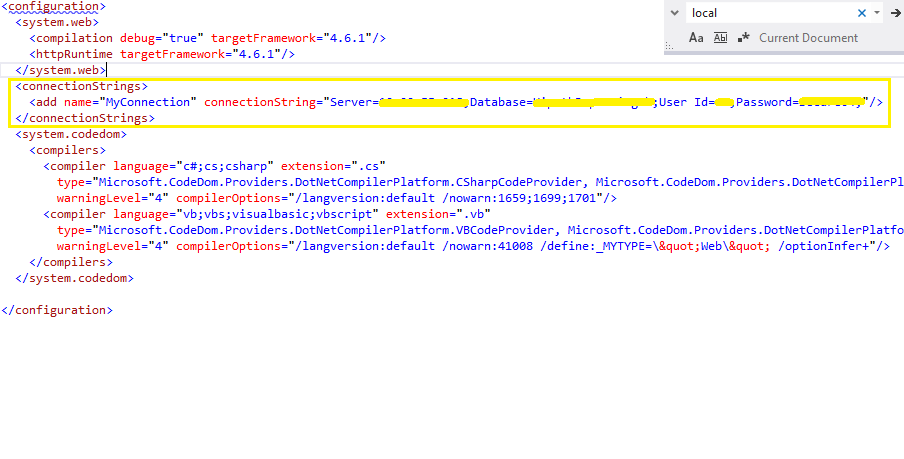
**Pre-Requisites:**

* UiPath Custom reporting activity pack
* MS SQL Server
* MS Excel

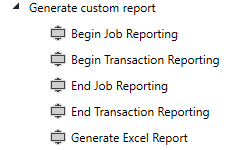
**Steps for setting up UiPath Custom reporting:**

1. Create new database in SQL server
2. Execute SQL server script on created database, provided in below Path: SqlServerExecutionScript\SQLReportGenerationStoredProcedure.sql
3. Update connection string object in Web.config file in Dashboard code

*File Path- RPACustomReportingDashboard\RPACustomReporting\Web.Config*



1. Add Custom report generation Activity pack into .Nuget folder in UiPath Studio installation directory. Inside UiPath studio add this package into your solution. Install the package with package explorer into UiPath. Once done you will be able to see it in Activities Pane as below:



**Details of activities in the Custom Reporting Activities Pack**

* **Begin Job Reporting:**

We are using SQL server to store Transaction and Job details, this activity Creates a Log for our Jobs & Transactions in defined SQL table. It returns a Unique job ID which is allocated to our job. Store this ID into a global variable as this ID will be required into End job reporting activity.

**Input/Output Details:**

**Input:**

ProcessName: Process Name for which log has to be created

SQLConnectionString: SQL Server connection String

**Output:**

JobID: Store this variable in global variable, to provide it to End Job Reporting activity

* **Begin Transaction Reporting:**

This activity creates Log for our Transactions in SQL table. It returns a Unique Transaction ID which is allocated to current transaction. Store this ID into a global variable as this ID is required into End Transaction Reporting activity

**Input/Output Details:**

**Input:**

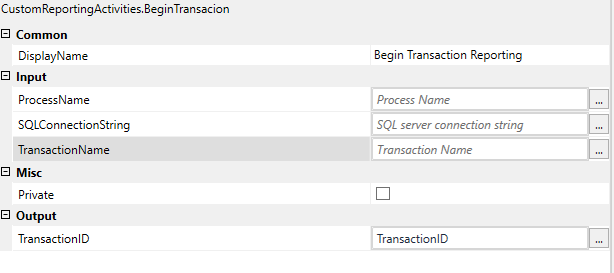
ProcessName: Process Name for which log has to be created

SQLConnectionString: SQL Server connection String

TransactionName: Name by which transaction is stored

**Output:**

TransactionID: Store this in global variable so that this can be provided as input to End Transaction Reporting activity.



* **End Transaction Reporting:**

This activity takes TransactionID as input parameter, it will update End time of the current job in SQL server. User has to provide the status for the transaction as Success, Faulted, Rule Exception.

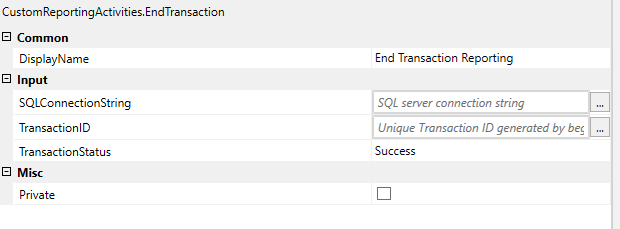
**Input/Output Details:**

**Input:**

TransactionID: Use value received from Begin Job Reporting activity

SQLConnectionString: Sql Server connection String

TransactionStatus: Update transaction status as Success/Faulted/RuleException



* **End job Reporting:**

This activity takes input of Report folder, JobID and creates Excel file with last Job Report.

**Input/Output Details:**

**Input:**

JobID: Use value received from Begin Transaction Reporting activity

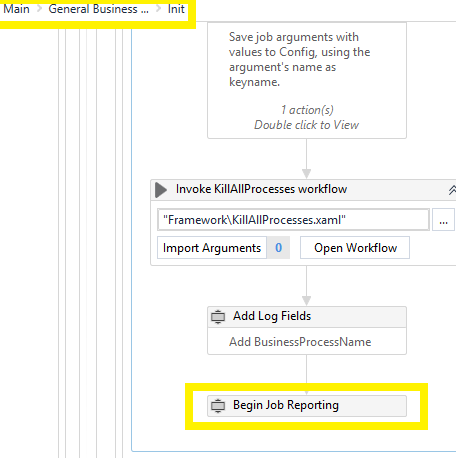
SQLConnectionString: Sql Server connection String

JobStatus: Update transaction status as Success/Faulted

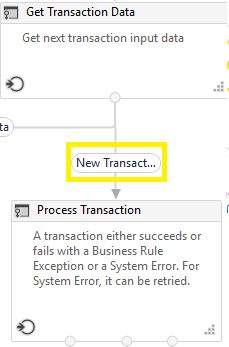


1. Place above activities into following locations inside RE Framework.

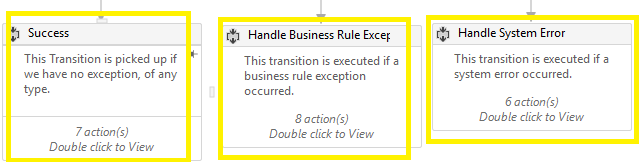
* Begin Job Reporting: This activity should be placed in Init State.



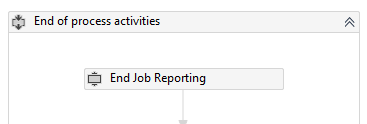
* Begin Transaction Reporting: This activity should be placed into Transition between GetTransactionData and Process State.



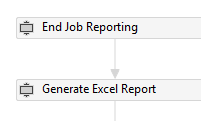
* End Transaction Reporting: This activity should be placed in Set Transaction Status Activity under this three Flow decisions with TransactionStatus input as [Success, RuleException, Faulted]



* End Job Reporting: This activity has to be added in End Process State in RE as follows:



* If you want to generate Excel report for each processed job for that particular run, add this activity below End Job Reporting Activity.



Once the above steps are completed you will successfully be able to log jobs and transactions.