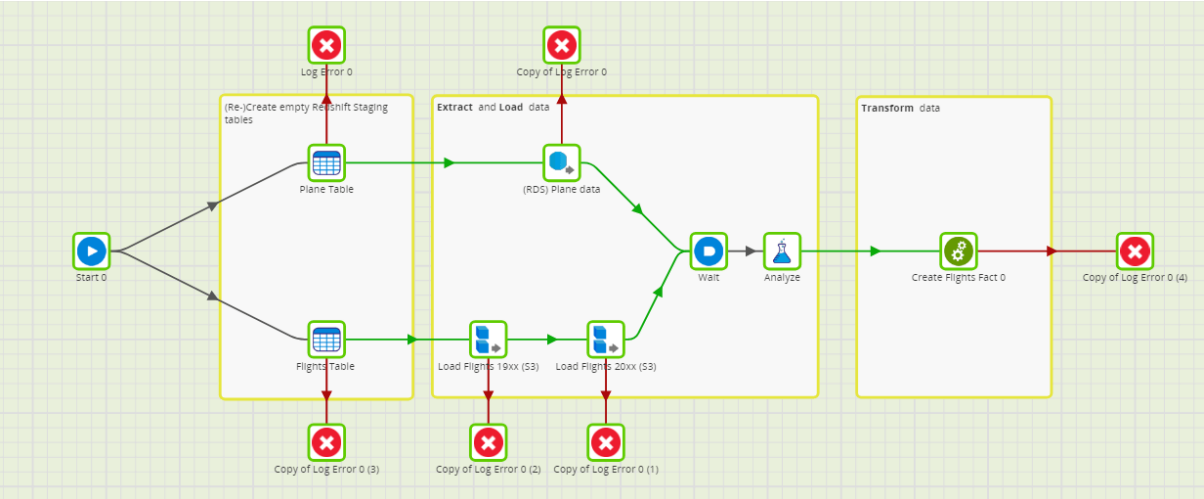
Error Handling Options in Matillion ETL – Alerting & Audit Tables

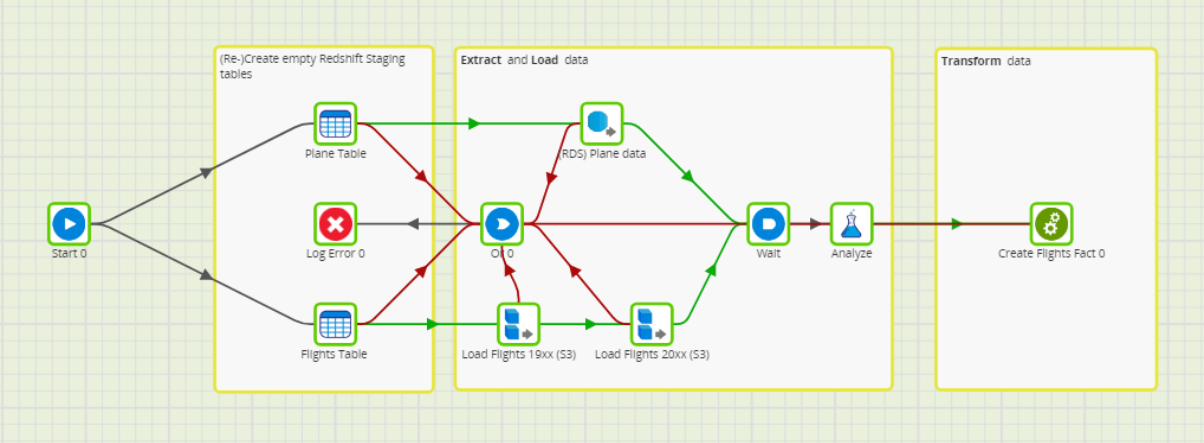
This is the second blog in a three-part series on Matillion ETL error handling options. This blog describes how to access component metadata used in alerting and logging.

In the **previous blog** in this series, I discussed the concept of a re-usable or shared job which would encapsulate the logic of our error handling to be used in multiple instances.

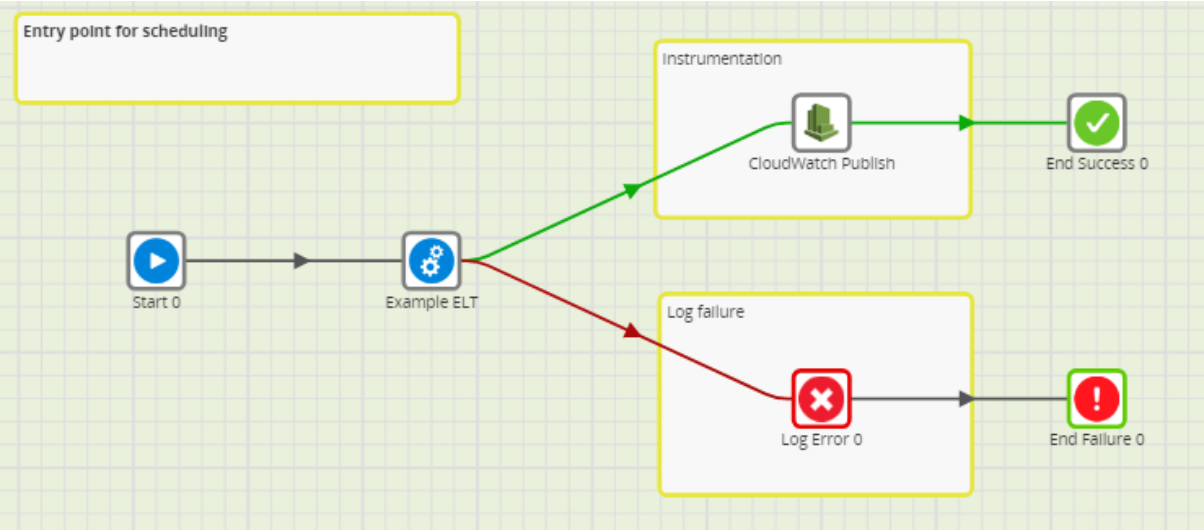
There are two patterns customers generally use when employing this strategy. The first is to use their shared job connected via the *On Failure* nub of selected components. This approach works well in terms of having no duplicated logic, however, it can make the canvas somewhat complicated by having lots of components on the screen.



This pattern is sometimes used with an **Or component** to connect a single error handling **Shared Job** to multiple components' *On Failure* nub. This reduces the number of components but can make the canvas more difficult to read as there are more intersecting lines.



The second pattern is to have a parent orchestration job which acts as the entry point for the scheduler and contains the main orchestration job and your error handling shared job connected via the *On Failure* Nub.



This is a much more readable and maintainable design, however, there are instances when the first pattern is more appropriate, such as when you are using the unconditional nub. This means a job won't terminate on an error and

so only the in the first occurrence an error will be detected by using a parent job.

To actually pass through the metadata that's going to be useful into these shared jobs you need to export it from a component that generates an error.

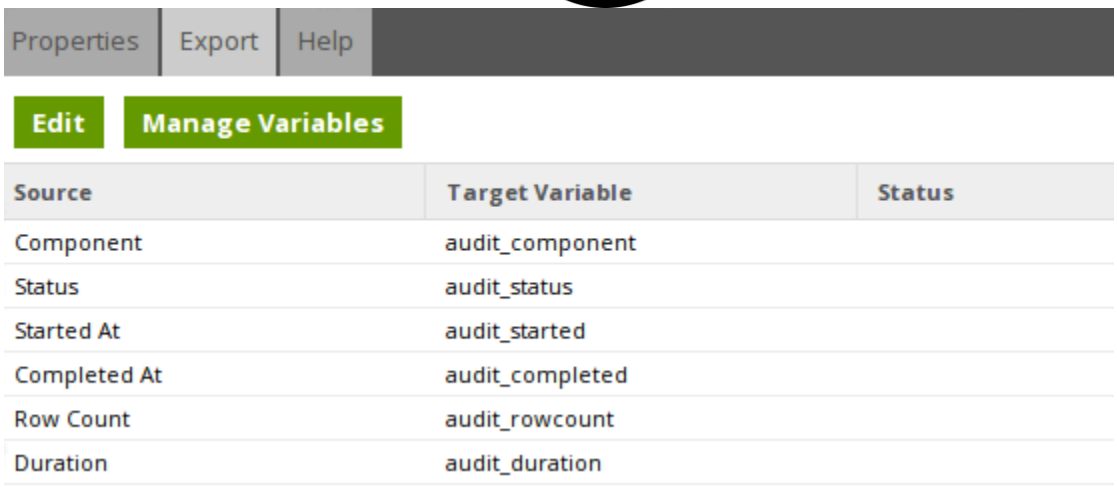
Component Export

Every orchestration component has an export tab which allows you to store some component metadata into variables. There are some standard attributes which are common across all components, they are:

- Component
- Started At
- Completed At
- Duration
- Message
- Status
- Row Count

And some components have metadata that's specific. For example, the **Create Table Component** has an attribute called "Table Recreated". Similarly, the **Data Stager Components** have attributes called "Time Taken To Load" and "Time Taken To Stage".

You will need to store those values in some variables for use later on. So that you do not need to create a set of variables for every job, you can use **environment variables** which will be accessible across all jobs, and because you need to access the values in a different branch (line originating at source component) you select the **shared** behavior.



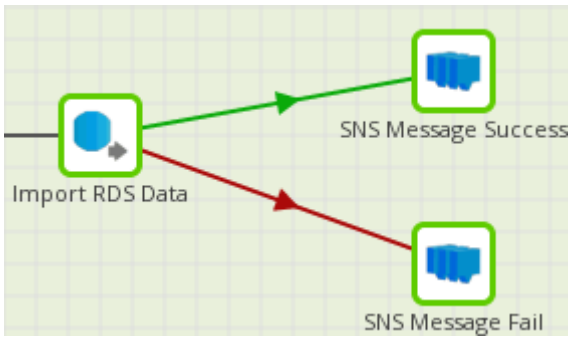
Alerting

Now that you have important metadata such as the message, start time and status of a component, you can compose a message to notify an administrator. The cloud platform you are using will determine the options available to you.

Amazon SNS (Amazon Redshift or Snowflake on AWS)

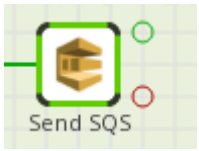
I recommend using SNS for alerting, as it easily allows subscription to topics by email, SMS as well as allowing you to push messages straight into SQS.

A detailed example of setting up an SNS alert can be found on the **SNS Component** documentation page.



Amazon SQS (Amazon Redshift or Snowflake on AWS)

You may have an external application that needs to be notified of the status of a job run, an alternative approach to the push architecture of SNS is to use a polling architecture of SQS.



This is useful for tools such as external scheduling systems or even Matillion ETL itself which may decide to run the job again if it receives a failure notification or triggers a downstream process if it receives a success notification.

An example of how to set up such a job can found on the **SQS Component** documentation.

Name	Value		Status
Name	Send SQS	...	OK
Region	eu-west-1	...	OK
Queue Name	[redacted]	...	OK
Message	{"method": "update_reports", "arg1": 5}	...	OK
Message Format	Plain	...	OK

Google Pub/Sub (BigQuery)

For BigQuery customers, you can use Cloud Pub/Sub to send messages and alerts in much the same way as Amazon SNS/SQS.

For a detailed guide on how to implement this, you can refer to the **Cloud Pub/Sub Component** documentation.

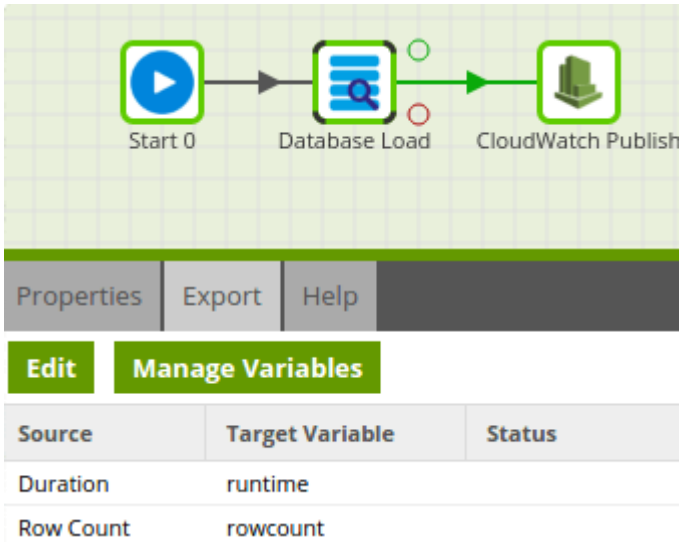
Properties | Export | SQL | Help

Cloud Pub/Sub — OK

Name	Value		Status
Name	Cloud Pub/Sub	...	OK
Project	emerald-test	...	OK
Topics	docspub	...	OK
Message	Test!	...	OK
Attributes	1, a, 2, b	...	OK

Cloudwatch alerts (Amazon Redshift or Snowflake on AWS)

AWS customers can also make use of CloudWatch for logging metrics such as duration & rowcount, and then set alerts when a job falls outside of the minimum and maximum threshold values.



This is explained in the **CloudWatch Publish Component** documentation.

Populating an Audit Table

I have previously written on the topic of **exporting your Matillion ETL instance history** for reporting purposes, however, this approach needs to be run on a different schedule from your existing jobs and can be too verbose for simple logging.

Therefore, a common practice in ETL processing is to have audit tables or control tables which contain the history of job runs and their status.

This **technical article** details how to use the environment variables we have set from the component export tab to populate an audit table.

References :

https://www.matillion.com/blog/error-handling-options-in-matillion-etl-alerting-audit-tables