User Defined Error Handling in Informatica PowerCenter

Error Handling is one of the must have components in any Data Warehouse or Data Integration project. When we start with any Data Warehouse or Data Integration projects, business users come up with set of exceptions to be handled in the ETL process. In this article, let’s talk about how do we easily handle these user defined error.

Informatica Functions Used

We are going to use two functions provided by Informatica PowerCenter to define our user defined error capture logic. Before we get into the coding lets understand the functions, which we are going to use.

1. ERROR()
2. ABORT()

**ERROR()**: This function Causes the PowerCenter Integration Service to skip a row and issue an error message, which you define. The error message displays in the session log or written to the error log tables based on the error logging type configuration in the session.

**ABORT()**: Stops the session, and issues a specified error message to the session log file or written to the error log tables based on the error logging type configuration in the session. When the PowerCenter Integration Service encounters an ABORT function, it stops transforming data at that row. It processes any rows read before the session aborts.

**Note**: Use the ERROR, ABORT function for both input and output port default values. You might use these functions for input ports to keep null values from passing into a transformation and use for output ports to handle any kind of transformation error.

Informatica Implementation

For the demonstration lets consider a workflow which loads daily credit card transactions and below two user defined data validation checks

1. Should not load any transaction with 0 (zero) amount, but capture such transactions into error tables
2. Should not process any transactions without credit card number and Stop the workflow.

**Mapping Level Changes**

To handle both the exceptions, let's create an expression transformation and add two variable ports.

- **TEST_TRANS_AMOUNT** as Variable Port
- **TEST_CREDIT_CARD_NB** as Variable Port

Add below expression for both ports. First expression will take care of the user defined data validation check No 1 and second expression will take care of user defined data validation check No 2.

- **TEST_TRANS_AMOUNT** :- IIF(TRANS_AMOUNT = 0, ERROR('0 (Zero) Transaction Amount'))
- **TEST_CREDIT_CARD_NB** :- IIF(ISNULL(LTRIM(RTRIM(CREDIT_CARD_NB))), ABORT('Empty Credit Card Number'))

The complete expression transformation is shown in below image.

Now insert this transformation in the mapping where you need the data validation and complete the mapping.
**Hint**: This Expression can be converted into a Reusable transformation, So that any mapping needs this data validation can reuse this transformation.

**Session Level Changes**

Once the mapping is complete, configure the session and provide the settings for row error logging as shown in below image. Please read the article Error handling made easy using Informatica Row Error Logging for more details on row error logging.

With this configuration we specified, Informatica PowerCenter will create four different tables for error logging and the table details as below.

- **ETL_PMERR_DATA**: Stores data about a transformation row error and its corresponding source row.
- **ETL_PMERR_MSG**: Stores metadata about an error and the error message.
ETL_PMERR_SESS: Stores metadata about the session.
ETL_PMERR_TRANS: Stores metadata about the source and transformation ports, when error occurs.

With this, we are done with the setting required to capture user-defined errors. Any data records which violates our data validation check will be captured into PMERR tables mentioned above.

Report the Error Data.

Now we have the error data stored in the error table, we can pull the error report using an SQL query. Below is a basic query to get the error report. We can be more fancy with the SQL and get more information from the error tables.

```sql
select
    sess.FOLDER_NAME as 'Folder Name',
    sess.WORKFLOW_NAME as 'WorkFlow Name',
    sess.TASK_INST_PATH as 'Session Name',
    data.SOURCE_ROW_DATA as 'Source Data',
    msg.ERROR_MSG as 'Error MSG'
from
    ETL_PMERR_SESS sess
left outer join ETL_PMERR_DATA data
    on data.WORKFLOW_RUN_ID = sess.WORKFLOW_RUN_ID and data.SESS_INST_ID = sess.SESS_INST_ID
left outer join ETL_PMERR_MSG msg
    on msg.WORKFLOW_RUN_ID = sess.WORKFLOW_RUN_ID and msg.SESS_INST_ID = sess.SESS_INST_ID
where
    sess.FOLDER_NAME = <Project Folder Name> and
    sess.WORKFLOW_NAME = <Workflow Name> and
    sess.TASK_INST_PATH = <Session Name> and
    sess.SESS_START_TIME = <Session Run Time>
```
Pros and Cons of this Approach.

We should know the Pros and Cons of this approach before applying this to your project.

Pros.

1. Out of the box Solution Provided by Informatica.
2. Less Coding and Testing efforts required by the development team.

Cons.

1. Added overhead to the Session performance, which is expected and acceptable.