**Parameter file - parameters and variables:**

It is always a best practice in coding that you should never hard code the values. The same applies to Informatica as well. It is always better to pass the values using parameters or variables in place of hard coding them. When you define parameters or variables in the code, you need to pass the values to those parameters and variables. A parameter file serves that purpose. Any value that you hard code can be passed through the Parameter file. You can define the Parameter file at the session level and workflow level.

You must have noticed that the system defined the variable *$PMSourceFileDir\* or *$PMTargetFileDir\*. Similar to that, we can define User Define variables. You can define the variable at both the mapping level and workflow level.

*If the value passed remains constant across the session run, it is called***parameter***, and if the value changes across the session run, it is called***variable***.*

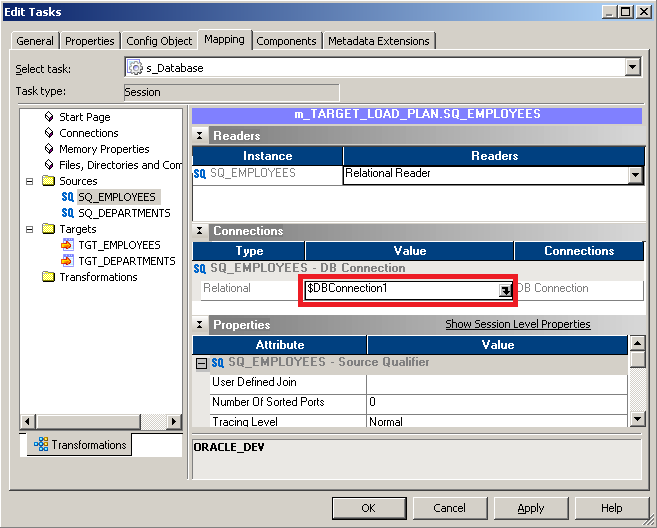
Let's take an example to understand the Parameter file. You are aware that Informatica will have three different repositories to cater to the need of three regions. Let's say we have three repositories REPO\_DEV, REPO\_TEST, REPO\_PROD serving the development, testing, and production regions respectively. Also, corresponding to three regions, we have three oracle databases--ORACLE\_DEV, ORACLE\_TEST, and ORACLE\_PROD respectively. When you start the coding in the development region under REPO\_DEV, you will hardcode the database connection value to ORACLE\_DEV. Your code is working successfully, and when you want to deploy the code to test the region, you will need to replace the database connection value to ORACLE\_TEST manually. Changing the code after testing is not allowed.

The same case applies when you wish to deploy the code from test to production. The solution comes as a Parameter file. Parameter files serve the purpose of passing the value based on the region in which you are running the code. We are defining the Parameter file for passing the value for the source database connection (*$DBCONNECTION1*), target file name (*$TGTFILENAME*), e-mail recipient (*$EMAILUSER*), and a mapping-level variable for the location ($$LOCATION). We are using the example of session-level variable, workflow-level variable, and mapping-level variable so you understand clearly how it works.

**Defining session-level variables**

The variables that are defined under session task are called session-level variables. There are various values that can be passed via variables, such as source, target data base connection value, source/target file name, source/target file path, session log file name/path, and so on. session task does most of the work in Workflow Manager screen and, hence, has been assigned special privilege. To pass the value through a variable, simply replace the hard-coded value with a variable of your choice.

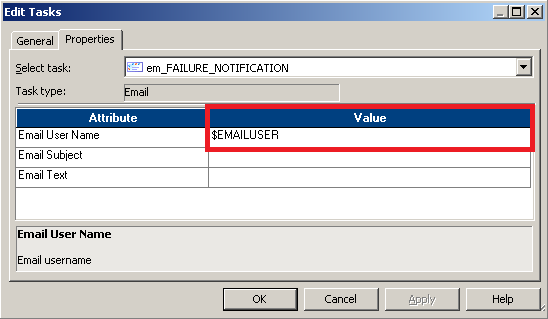
As shown in the following screenshot, we are using $DBCONNECTION1 as the database connection variable:



Similarly, assign the value to the target file name (*$TGTFILENAME*). Since we define the target file name under the session task, it will be referred to as session-level variable.

**Defining workflow-level variables**

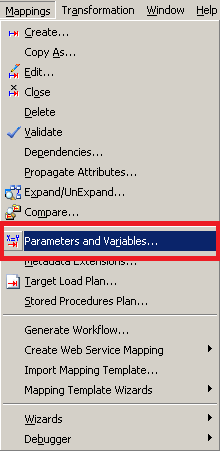
The variables that are defined under various tasks are called workflow-level variables. Note that you can define the session-level variables under workflow also in the Parameter file. In our case, we are passing an e-mail user value as the variable. To assign the value, simply replace the hardcoded value by the variable in the e-mail task as shown in the following screenshot:



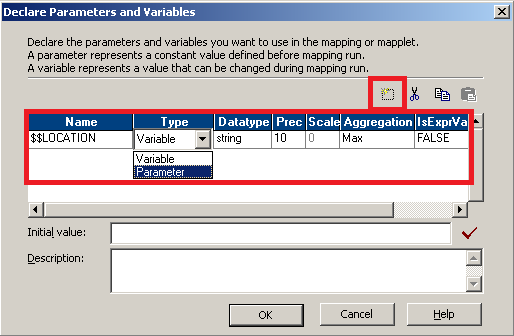
**Defining mapping-level variables**

As mentioned earlier, you can define the variable for the hardcoded values in mapping inside transformations also. You need to define parameters or variables under mapping before you can use them in transformations, else the mapping will become invalid. To add the values, perform the following steps:

1. Open the mapping for which you wish to add variables in the Mapping Designer, go to Mappings | Parameters and Variables:

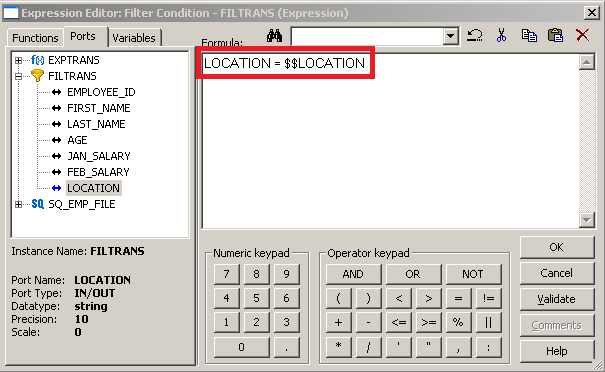


1. In the next screen, click on Add a new variable to this table. Define the variable, and select the type as Parameter and Variable based on your requirement. Click on OK:



As you can notice, mapping-level variables are always defined as $$.

Open the transformation to which you wish to assign the variable or parameter. We are using filter transformation to pass the value:

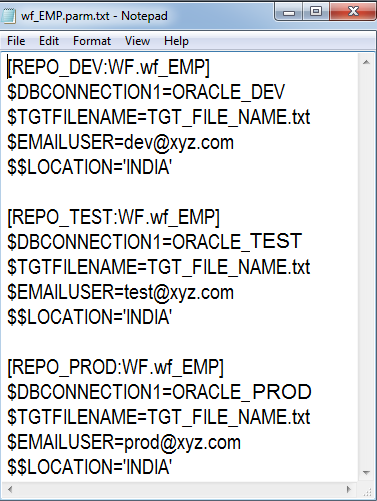


As you can notice, we have assigned $$LOCATION to the filter condition.

With this, we are done with defining the variables or parameters. Next, we will see how to pass the values to these using a parameter file.

**Creating a parameter file**

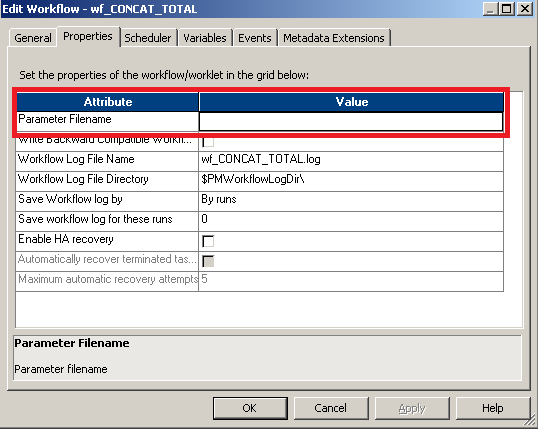
A Parameter file is nothing but a simple .txt file that contains the values of the variable to be passed A sample Parameter file for the variable defined in the previous steps is shown as follows:



As you can see, the Parameter file contains the values of the variables to be replaced in the three regions. Informatica matches the Repository name against the name of Repository defined in the Parameter file and replaces the values of the variables internally before it runs the workflow. wf\_EMP is the name of workflow for which you defined the Parameter file. So, suppose you are running in the production region, Informatica will match the Repository name REPO\_PROD against the same name in the Parameter file, replace the variables with the value internally, and execute the workflow with the replaced values.

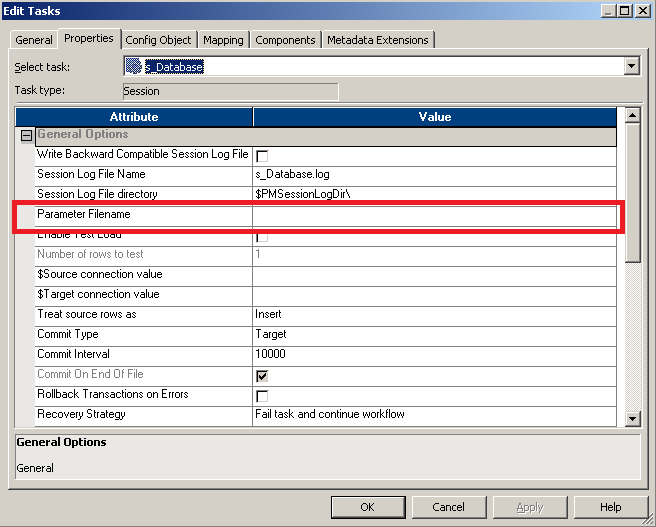
**Mentioning the Parameter file at the workflow level**

To define the Parameter file at the workflow level, open Workflow Manager, go to Workflow | Edit | Properties. Specify the path and name of the parameter file for the attribute as indicated in the following screenshot:



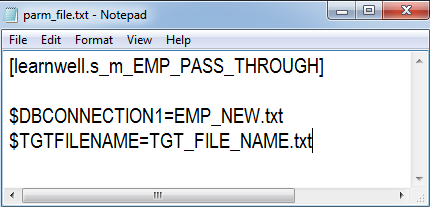
**Mentioning the Parameter file at the session level**

To define the Parameter file at the session level, open the session in the Workflow Manager, double-click on the session task, and click on Properties. Specify the path and name of the Parameter file for the attribute as indicated in the following screenshot:



You have learned how to use parameters and variables, create a parameter file, and define the Parameter file. Another importance of parameter file is that if the value of any variable is changing, you need not modify the code; simply change the value in the Parameter file, and the changed value will take effect the next time you run the code.

Also, note that Parameter files can be used to replace the values for the individual session run. They can also be defined at the folder level as against at repository level as seen earlier. It is not mandatory to define the workflow for all three regions. A sample Parameter file for an individual session run (s\_m\_EMP\_PASS\_THROUGH) defined at the folder level (learnwell) is shown in the following screenshot:



Parameters are different from variables in the fact that:

 The value of a **parameter is fixed** during the run of the mapping

 **Variables can change** in value during run-time   
  
Both parameters and variables can be accessed from any component in the mapping which supports it.

**Changing values of Variables**

To change the value of a variable, one of the following functions can be used within an expression: SETMAXVARIABLE($$Variable, value) , SETMINVARIABLE($$Variable, value), SETVARIABLE($$Variable, value) , SETCOUNTVARIABLE($$Variable), where:

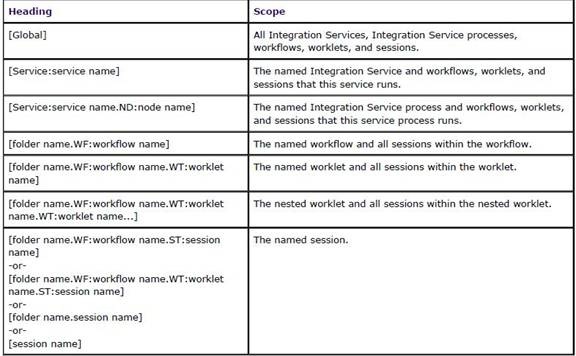
SETVARIABLE sets the variable to a value that you specify (executes only if a row is marked as insert or update). At the end of a successful session, the Integration Service saves either the MAX or MIN of (start value.final value) to the repository, depending on the aggregate type of the variable. Unless overridden, it uses the saved value as the start value of the variable for the next session run.

SETCOUNTVARIABLE - increments a counter variable. If the Row Type is Insert increment +1, if Row Type is Delete increment -1. A value = 0 is used for Update and Reject.

SETMAXVARIABLE - compare current value to value passed into the function. Returns the higher value and sets the current value to the higher value.

SETMINVARIABLE - compare current value to the value passed into the function. Returns the lower value and sets the current value to the lower value.

At the end of a successful session, the values of variables are saved to the repository. The SetVariable function writes the final value of a variable to the repository based on the Aggregation Type selected when the variable was defined.



Sample parameter file:

[Service:IntegrationSvc\_01]

$$SuccessEmail=dwhadmin@etl-tools.info

$$FailureEmail=helpdesk@etl-tools.info

[DWH\_PROJECT.WF:wkf\_daily\_loading]

$$platform=hpux

$$DBC\_ORA=oracle\_dwh

[DWH\_PROJECT.WF:wkf\_daily\_loading.ST:s\_src\_sa\_sapbw]

$$DBC\_SAP=sapbw.etl-tools.info

$$DBC\_ORA=oracle\_sap\_staging

**The Order of Execution of Paramater Values in Informatica:**

1. Paramaeter File

2. Value Assigned in Pre-Session Variable Assignment

3. Value saved in the Repository

4. Initial Value defined for the variable at declaration level

5. The default value defined by Informatica for the Datatype of the defined variable

Informatica Checks for the variable values in the above order and stops and assigns the value to the parameter/variable when it finds a value.

**Note:*In case there are values defined in two parameter files and the Parameter files are assignd at both Session Level and Workflow level, The Paramater file defined at the Workflow level always overrides the Paramater file defined at the Session Level.***

How these variable assignment actually, works :

1. Paramaeter File  
Assign a value to the variable in the parameter file itself

2. Value Assigned in Pre-Session Variable Assignment  
You can assign values for a variable explicitly using the Pre/Post Session variable assignment in the Session Properties.

3. Value saved in the Repository When you declare a variable, you also declare the aggregation type of that variable. It could be Min, mix or count. Based upon this declaration after the session is ended , the Integration Service writes and saves the final value of the variable into the Repository. SO, from the next load onwards, the variable displays the value stored in the Repository.  
Note: You can check the value stored in the Repository by righclicking on the session using that variable and clicking “View persistent values.”

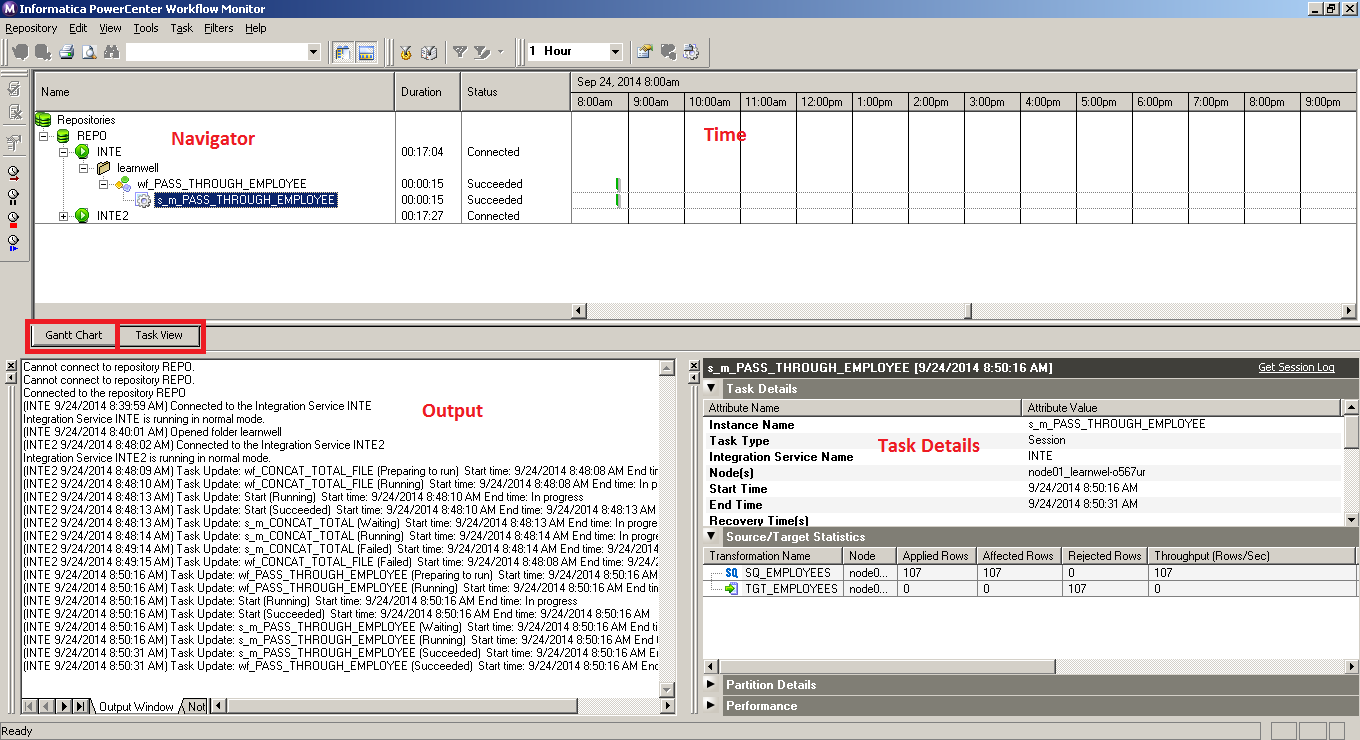
4. Initial Value/Default Value defined for the variable at declaration level You can provide an initial value to the variable while declaration itself.

5. The default value defined by Informatica for the Datatype of the defined variable In case, the Inetgration service doesn’t find any value assigned against the declared Variable, it assigns the Default value for the datatype of which the variable is declared.

"Hi experts, I read through various posts which stated that session parameter file is ignored when workflow parameter file is declared. However in help file there is section where it states that there is a solution via using the $PMMergeSessParamFile ( HELP FILE Session parameter file is ignored when a workflow parameter file is specified while running a PowerCenter session Problem Description A PowerCenter workflow with a session is configured with a parameter file in the session properties and the workflow properties. When running the workflow, the session parameter file is ignored and the workflow parameter file is used. Cause By default, when you define a workflow parameter file and a session parameter file for a session within the workflow, the Integration Service uses the workflow parameter file, and ignores the session parameter file. Solution You can force the workflow to use both the workflow parameter file and session parameter file by adding the following to the workflow section of the workflow parameter file: $PMMergeSessParamFile=TRUE This must be set for the section defined for the workflow in the workflow parameter file. Example [DEV\_FOLDER.wf\_workflow] $PMMergeSessParamFile=TRUE Note If you define the same parameter or variable in both the session and workflow parameter files, the Integration Service sets parameter and variable values according to the following rules: • When a parameter or variable is defined in the same section of the workflow and session parameter files, the Integration Service uses the value in the workflow parameter file. • When a parameter or variable is defined in both the session section of the session parameter file and the workflow section of the workflow parameter file, the Integration Service uses the value in the session parameter file. ) I tried this solution but it didn't work. Do you have any idea of what might be the good solution since i desparately looking for someway to use both Workflow parameter file and session parameter file in a workflow. For example ,Here is my Workflow parameter definition: $$Environment=DevEnvironment email@removed email@removed $PMMergeSessParamFile=TRUE Any help is appreciated,"

**Using the Workflow Monitor**

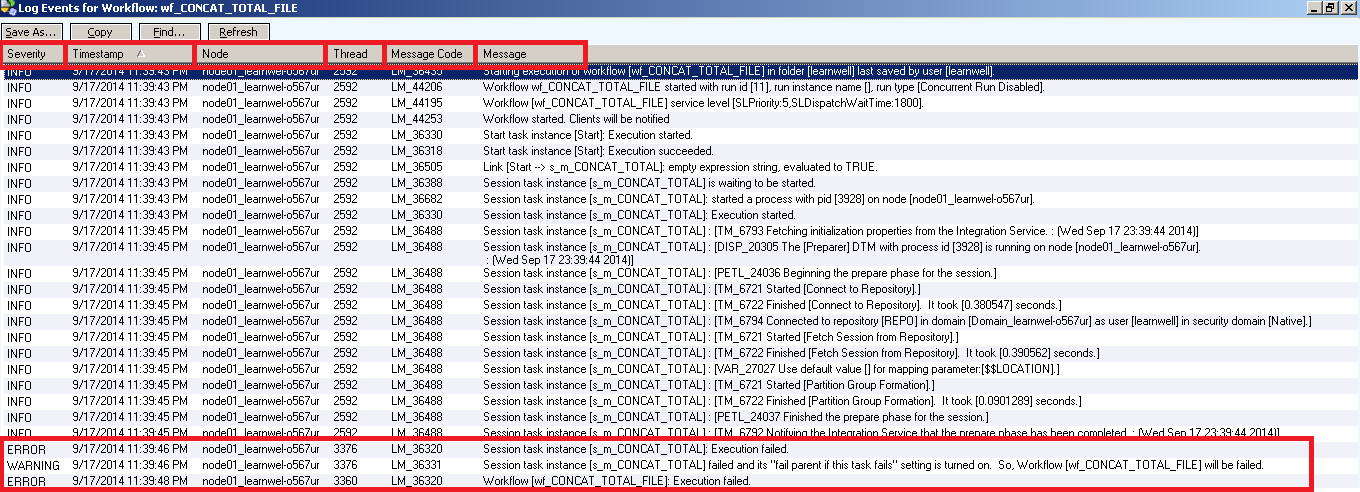
The Workflow Monitor Screen, as mentioned earlier, displays the status of the running workflow and tasks. It has two views to show the status -- the Gantt Chart view and the Task view. You can select the view you wish to see. The Workflow Monitor screen can be seen in the following screenshot:



The screen is divided into the following sections:

* **Navigator:** This section of the screen displays various repositories, integration service, and the workflow names running at the instant. This section shows all the objects that have been executed at least once.
* **Output:** This section displays various system-level information details received from integration eervice and repository.
* **Time:** This section displays the timings of the execution of various workflows.
* **Gantt Chart View:** This view shows the information of various workflow runs in chronological order.
* **Task View:** This view shows the information of various workflow runs in a report form.
* **Task Details:** This section shows the details of the task. Also, it shows the source-to-target statistics and Performance and Partition details if selected.

**Working with the workflow log**

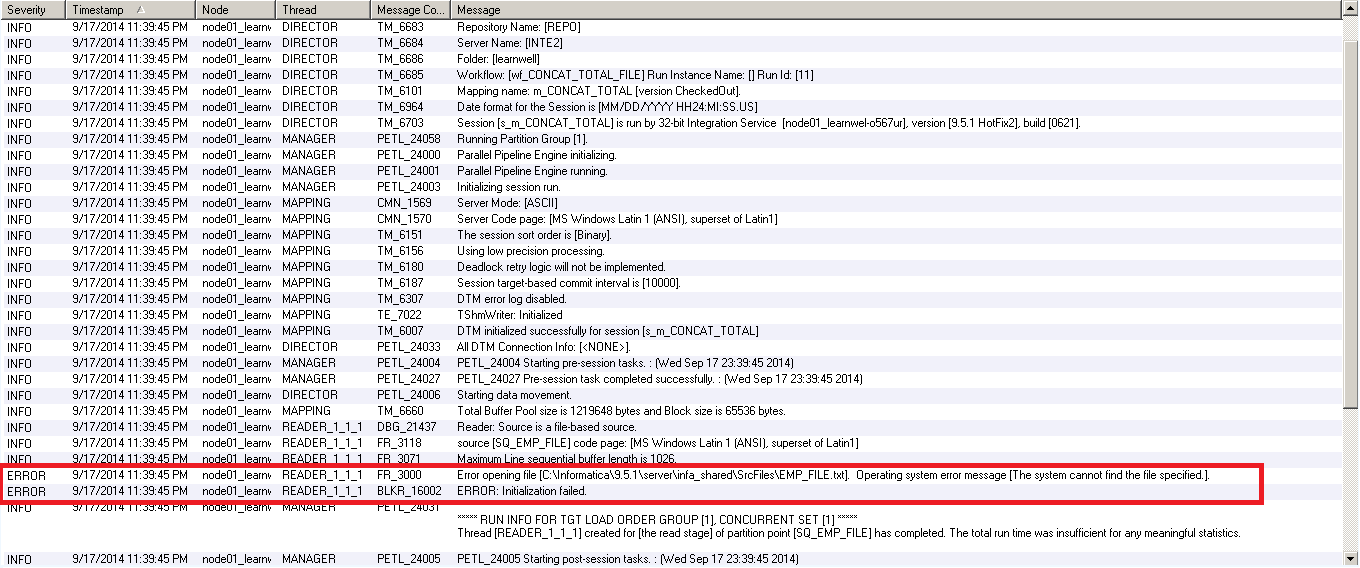


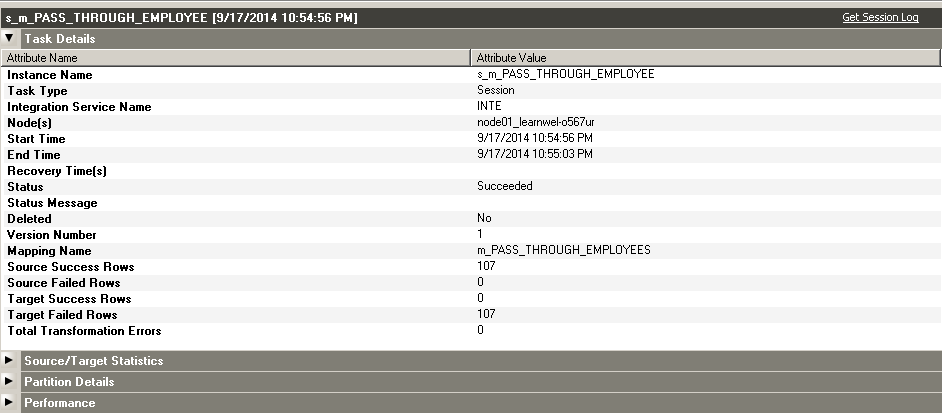
The following table describes the various options of the workflow log:

|  |  |
| --- | --- |
| Option | Description |
| Severity | It shows the status of the particular event that took place while executing the workflow.  INFO indicates general system-level information.  ERROR indicates the error that has occurred due to which the Workflow failed.  WARNING indicates the process that was not executed as per expectation. The workflow may not fail because of the warning. |
| Timestamp | This indicates the exact timing of the particular step that the workflow was running. |
| Node | This indicates the name of the node under which the workflow is executed. |
| Thread | This indicates the thread each step is using to execute. You can see the different threads in the session log. |
| Message Code | This indicates the system-defined message code. Usually, in Informatica PowerCenter, we do not refer to message codes. |
| Message | This shows the detailed message indicating the steps that occurred during the workflow execution. |

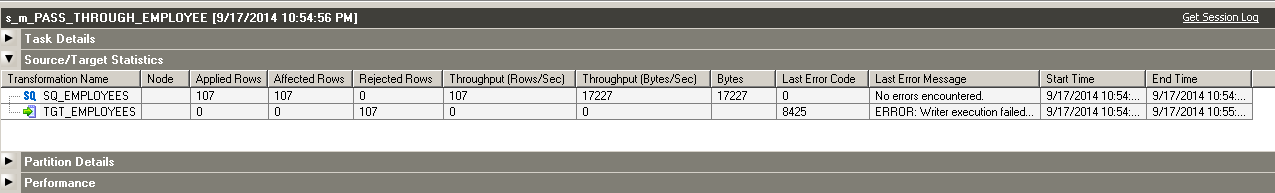
**orking with the session Log**

Similar to the workflow log, the session log also indicates the detailed level of information that gives you a complete understanding of the process that occurred while the workflow was running. The session log in Informatica gives the exact reason of the error that has occurred, using which you can debug the issue and correct the code. For example, in the following screenshot, the session log is indicating the error as The system cannot find the file specified. This directly indicates that the file you wish to access does not exist. Check the details in the session task, correct the path. and your workflow will succeed.





# Source/target statistics properties :Source/target-level task details under session run properties are shown here:



# Common Errors

When you execute the session and workflow, there are certain common error that you will face. Some of the common errors are as follows:

* **Source File Not found:** This indicates that the source file is not available at the specified location. Make sure you have placed the file in the correct folder.
* **Unable to generate Session log:** This may be due to some invalid session task or may show if you don't specify the lookup file name and path in the session task.
* **Table or view not found:** This indicates that the database table in which you are willing to load the data is not available. Make sure you have created the table before you load the data in the table.
* **Communication link failure:** This indicates that there are network issues affecting the integration service or repository.
* **Failed to allocate memory:** This indicates that the memory required for executing the process is not available.
* **Duplicate Primary/Foreign key:** This indicates that you are trying to load the duplicate data in the primary key in the table.

**Performace tuning:**

**Finding the target bottleneck**

Always consider checking the bottlenecks at the target side first. There can be various reasons for bottlenecks at the target. First, we need to know how to find the target bottleneck.

**Using thread statistics**

Thread statistics are a part of the session log. When you run the workflow, the session log generates the thread statistics, which can provide you with information about the bottlenecks present in Source, Target, or Transformation. Thread statistics give information about the total runtime, idle time, and busy percentage of Source, Target, and Transformation.

Thread statistics consist of Reader thread, Writer thread, and Transformation thread. Reader thread gives information related to the total runtime, idle time, and busy percentage of the sources in the mapping. Writer thread gives information about the total runtime, idle time, and busy percentage of targets in the mapping. Similarly, Transformation thread gives information related to transformations in the mapping, as indicated in the following sample thread statistics:

\*\*\*\*\* RUN INFO FOR TGT LOAD ORDER GROUP [1] \*\*\*\*\*

Thread [ READER\_1\_1\_1 ] created for [ the read stage ] of partition point [ SQ\_EMPLOYEES ] has completed.

Total Run Time = [ 100.11 ] Secs

Total Idle Time = [ 90.101 ] Secs

Busy Percentage = [ 10.888141628 ]

Thread [ TRANSFORMATION\_1\_1\_1 ] created for [ the transformation stage ] of partition point [ SQ\_EMPLOYEES ] has completed.

Total Run Time = [ 123.11 ] Secs

Total Idle Time = [ 100.23 ] Secs

Busy Percentage = [ 18.585005278 ]

Thread [ WRITER\_1\_1\_1 ] created for [ the target stage ] of partition point [ TGT\_EMPLOYEES ] has completed.

Total Run Time = [ 130.11 ] Secs

Total Idle Time = [ 1.23 ] Secs

Busy Percentage = [ 99.054646069 ]

Writer thread is busy for 99 percent as compared to Reader and Transformation. We can say that, in this case, the Target is the bottleneck. Similarly, you can identify if the source or transformation has bottlenecks.

**Configuring the sample target load**

It is a simple understanding that loading data in the target table will take more time as compared to loading data in the target file. Consider that you are loading data in the target table in your mapping, configuring a sample run, and trying to load the same data in a test target file. Check the difference in the runtime of both the processes. If there is a significant difference, you can easily say that the database target table has the bottleneck.

**Eliminating the target bottleneck**

There are various ways in which you can optimize the target loading.

**Minimizing target table deadlocks**

There can be a scenario when Informatica is trying to load data in a table that is already being used by another system. When Informatica encounters the deadlock, it hampers the processing by slowing the loading process. To avoid this, make sure that the target table is not being used by other processes at the same time.

**Dropping indexes and constraints**

Loading data in the table takes more time because there are multiple indexes and constraints created on the table. Each time a new record is loaded in the table, it is first checked for indexes and constraints. This hampers the performance. To avoid this, you can use pre-SQL and post-SQL commands in the Session task. Using pre-SQL commands, you can remove the indexes, and using post-SQL, you can apply the indexes. When you define pre-SQL and post-SQL, Informatica applies those commands before and after the data is loaded in the table.

Removing the indexes and constraints is not always recommended, but it definitely improves the performance. You can opt for this option if the data that you are loading is not very critical.

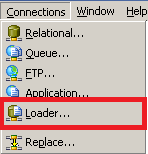
**Increasing the checkpoint interval**

When you run the workflow, the integration service keeps on creating checkpoints at predefined intervals. The checkpoints are used for recovery purposes. Reducing the checkpoint interval will help enhance the performance by storing less checkpoints and storing less data related to checkpoints.

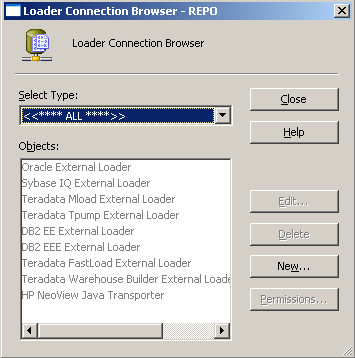
If you reduce the checkpoint interval, even though the performance will increase, it will hamper the recovery time if the system fails with some error.

**Using an external loader**

Informatica PowerCenter supports the usage of multiple external loaders (IBM DB2, Oracle, Teradata, and Sybase IQ), which can help in loading data in the target table faster. To add the external loader, open Workflow Manager, and click on Connection | Loader, as shown in the following screenshot:

****

**In the next screen, select the loader based on your requirement:**

****

**This will help in enhancing the performance by loading the data at a faster pace in the target.**

**Increasing the network packet size :** Every table has a capacity, referred to as Network Packet size, with which it can accept data. If you increase the packet size, the table can accept greater volumes of data.These properties can be enhanced by a database administrator. Consult your database admin team.

**Using bulk load :** By configuring the session properties to use the bulk load, your performance can be significantly enhanced. When you use bulk loading, the database logs are not created; this, in turn, enhances the performance.

However, if you disable the database log, recovery cannot be done as there is no log of the events on the database.As you can notice, performance can be achieved by compromising on some other factors. You need to decide and make a fine balance between all the factors.These were the various ways in which you can find and eliminate the target bottleneck. In the next section, we will talk about the source bottleneck.

**Finding the source bottleneck:** The performance can also be impacted at the source side. Various factors can hamper the performance at the source side as we have discussed at the target side.

**Test mapping:** Using a passthrough mapping can help you identify if the bottleneck is at the source side. Configure a test mapping to have only source, source qualifier, and target. Consider that you have a mapping with one source, one target, and ten transformations; the time taken to process the data through mapping is 60 seconds. This means that combining the source, target, and transformation is taking 60 seconds. Create another mapping by eliminating all the transformations and run the process and check the time. Suppose that now the time taken to complete the process is 50 seconds. As compared to the combined time of target and transformation, which is 10 seconds, 50 seconds is relatively high, which can indicate we have a source bottleneck.

**Using filter transformation :**You can use a filter transformation to check whether the source has a bottleneck. Put a filter transformation in front of the Source Qualifier and set the condition to False; that is, don't allow any record to pass through the filter ahead in the mapping. Using this method, you can compare the runtime of the process with and without the filter condition. This way, you can identify if the source has a bottleneck.

**Checking the database query :**This is another simple method to find the source bottleneck if you are extracting the data from the table. When you use the source qualifier to read the data from the database table, the integration service writes a query to extract the data. Copy the query and run the same query at the database level in SQL developer or a similar tool. Compare the time taken by the query to execute at both places, which can give you an idea if the source has a bottleneck.

**Eliminating the source bottleneck**

**T**here are various ways in which you can optimize the source. They are discussed as follows.

# Increasing the network packet size :As discussed under the Eliminating the target bottleneck section, increasing the network packet size of the table will allow greater volumes of data to pass the network at the particular instance.

These properties can be enhanced by the database administrator. Consult your database admin team to increase the network packet size.

# Optimizing the database query: When you read the data from the database table, the integration service generates a query to extract the data from the table. You can fine-tune the query to extract only the required data. If you extract all the data and then add filter or other transformations, it will hamper the performance. You can tune the query to extract only the required data, which will save time and help in performance enhancement. This is called SQL override.

These were the various ways in which you can find and eliminate the source bottleneck. In the next section, we will talk about the mapping and transformation bottleneck.

# Finding the mapping bottleneck

If you don't have a source or target bottleneck, you can have mapping or transformation bottlenecks.

# using filter transformation

You can use a filter transformation to check whether transformation or mapping has a bottleneck. Put a filter transformation before the target and set the condition to False; that is, don't allow any record to pass to the target. Using this method, you can compare the runtime of the process with and without the filter condition. This way, you can identify if the transformations have bottlenecks.

# Eliminating the mapping bottleneck

There are various ways in which you can optimize the transformations and mapping.

# Using single pass mapping

Consider a scenario where you have multiple targets to load from the same source; in such a scenario, avoid creating multiple mapping. You can save significant time by loading multiple targets in the same mapping. Use a single source and pass the data to different pipelines and then to multiple targets. This way, you can save the time of reading the same data multiple times in multiple mappings.

# Avoiding data type conversions

Avoid changing the data types across the transformations in the mapping. When you change the data type, the integration service takes time for the processing. It is always recommended that you should not change data types wherever not required.

# Processing numeric data

The integration service processes numeric data faster compared to other data. Try to process as much numeric data as possible.

# Using operators instead of functions

The integration service processes operators faster compared to functions. For example, consider using || (pipe) in place of the CONCAT function to concatenate the data.

# Using Decode in place of multiple IIF

If your logic contains multiple **Informatica Inbuilt function** (**IIF**) functions, try replacing them using DECODE. The decode function is faster compared to multiple IIFs.

# Tracing level

The tracing level defines how much detailed information you wish to write to the session log. When you run the process, the integration service writes the information about the run in the session log. Setting a proper tracing level will help in improving the performance.

# Using variable ports

If you are performing the same operation multiple times in a transformation, consider calculating the value in a variable port and use the variable port value multiple times in the transformation. Suppose you need to convert the first name and last name to uppercase, concatenate them, and need to cut part of the data. In place of using the UPPER function every time, use the variable port to convert data to uppercase and use the variable port to perform other operations. This way, you save time of performing the save operation multiple times.

# Optimizing filter transformation

You can use filter transformation as early as possible in the mapping to avoid processing unnecessary data. If you filter unwanted records early in the mapping, you can enhance the performance.

Similarly, using Router transformation in place of multiple filter transformations will have help save time.

# Optimizing Aggregator transformation

Always pass sorted data to Aggregator transformation to enhance the performance. When you pass the sorted data, the integration service needs to save less data in the cache, which helps in enhancement of performance.

You can also improve performance of Aggregator transformation by doing group by numeric columns. For example, consider grouping the data on department ID in place of location. It is possible to do this only as per your business requirement.

Use incremental aggregation whenever possible in the session properties to enhance the performance. When you use incremental aggregation, the performance is improved as aggregator transformation now needs to calculate lesser records.

# Optimizing joiner transformation

It is recommended to assign the table with lesser number of records as master while using joiner transformation. A table with lesser number of duplicates should be used as the master table.

It is also recommended to perform joining in the source qualifier using SQL override as performing joins on the database is sometimes faster compared to performing in Informatica.

Additionally, pass the sorted data to joiner transformation to enhance the performance as this utilizes less disk space compared to unsorted data.

# Optimizing lookup transformation

Lookup transformations are one of the complex transformations in Informatica PowerCenter. Optimizing lookups will significantly help in improving the performance.

When you use lookup transformation in the mapping, use the concurrent cache. When you use concurrent cache, the integration service caches the lookup table data before it starts processing the data from the source; otherwise, lookup performs cache on row-wise basis, which utilizes more time. So it is recommended that you enable caching when you use lookup.

If your mapping contains multiple lookups that look up on the same lookup table, it is suggested you share the cache in order to avoid performing caching multiple times.

You can reduce the processing time if you use lookup SQL override properly in the lookup transformation. If you are using lookup to look up on the database table, you can use the lookup SQL override to reduce the amount of data that you look up. This also helps in saving the cache space.

If you are using more than one lookup condition in lookup transformation, it is recommended that you place the conditions in an optimized order; that is, place the equal to (=) condition first, then less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=), and at last, Not equal to (!=). This enhances the performance.

These were the various ways in which you can find and eliminate the mapping and transformations bottlenecks. In the next section, we will talk about the session bottleneck.

# Eliminating the session bottleneck

If you do not have source, target, and mapping bottlenecks, you can check for session properties for bottlenecks.

# Optimizing the commit interval

Commit interval is the number of records after which the integration service commits the data into the target. Selecting an appropriate commit interval will help in enhancing the performance. If you select a low value as the commit interval, it will make the integration service commit data more number of times, which will hamper the performance.

# Buffer memory

When you run the workflow, the integration service needs to allocate blocks of memory to hold the data at various stages of processing, including cache if required. Make sure that you have sufficient buffer memory available for the processing; otherwise, the integration service fails the process because of lack of memory.

# Performance data

Session properties allow you to store the performance-related details in the repository. If you select to save the performance details, the integration service writes the log to the repository. This will consume processing time. Make sure that you are not checking the option if you do not require to save the performance details.

# Eliminating the system bottleneck

The last step in performance enhancement you can try is to find the bottlenecks in the system. Eliminating the system bottleneck may not be in your control; you can contact your admin team to improve the system capabilities to enhance the system performance.

You can add multiple CPUs to make the process run faster or make the session run in parallel.

You can check with the admin team if the network is working properly at the optimized speed to confirm if the processing is optimized.

Contact your admin team to add extra memory if the buffer memory or cache memory is not sufficient. Adding extra space may save processing time if your cache memory requirements are more.

Using these performance rules, you can make your process optimized. After taking care of all these rules, if you feel your system is not utilized fully, you can make use of partitioning.

# Working on partitioning

Before we discuss partitioning, make a note that partitioning is a high availability feature that you need to purchase separately from Informatica. If you enable high availability features, you can make use of the partitioning functionality.

By default, a mapping containing source, target, and transformations has a single partition. A single partition means that a single record can flow from the source to target at a time. By adding multiple partitions, you logically divide the mapping into multiple sections-each section can pass a record at a time. So if you make three partitions in the mapping, three records can pass through the mapping, making your runtime reduced by one third. When you add a partition at any stage of the mapping, the integration service adds partitions at other stages of the mapping. You need to make sure that you have sufficient memory space and system capacity to handle the processing of multiple records at a time.

If you have 1,000 records to process and you created four partitions, the integration service will process four records at a time and the total time required to process 1,000 records will be reduced to a fourth.

To enable partitions, you need to set the partitioning properties in the Session task.

# Partitioning properties

To enable partitioning, you need to define the following attributes.

# Partition points

You can define the partition points in a pipeline. By default, the integration service sets partitions at various transformations. You can define the partition at any stage in the mapping.

# Number of partitions

Based on your system capability, you can increase or decrease the partitions. When you add a partition at any stage of the pipeline, the integration service adds the same number of partitions at other stages of the mapping. The number of partitions in a mapping should be equal to the number of database connections at the source and target side. When you create partitions, the integration service processes the data concurrently. Suppose you create three partitions; the integration service reads three records from the source, passes three records to transformations, and concurrently loads three records to the target.

# Partition types

Informatica supports multiple types of partitions to distribute the data. Partition types control how you wish to divide the data among the partitions that you created in the mapping. If you have high availability features, you can define the type of partitions at different stages of the mapping. You can define the type of partitioning in Session properties. Different types of partitions are mentioned as follows:

* **Pass-through:** In the pass-through type, the integration service does not distribute the data among partitions. The data in the particular partition stays in the partition after passing through the partition point.
* **Round-robin:** In round-robin partitioning, the integration service distributes the data evenly among the partitions. This makes equal amounts of data pass through each partition.
* **Key range:** In key range partitioning, the integration service distributes the data on the basis of ports or set or ports defined. You also define the range of value for each port. When the source and target are partitioned by key range, select this type of partitioning.
* **Database partition:** This type of partitioning is possible with Oracle or DB2 database. When you select database partitioning, the integration service reads the partitioning information from the nodes in the database.
* **Hash auto-keys:** In hash auto-key partitioning, the integration service divides the data based on the partition key using the hash function. All the grouped and sorted ports in transformations are used as partition keys. This type of partition can be used in Rank, Sorter, and Aggregator transformations.
* **Hash user keys:** Similar to hash auto-keys, the integration service in this partitioning uses the hash function to partition the data. You need to manually define the number of ports for the partition key.

# Pushdown optimization

Pushdown optimization is a concept using which you can push the transformation logic at the source or target database side. When you have the database table as the source, you can make use of SQL override to remove the logic written in the transformation. When you use SQL override, session performance is enhanced as processing the data at the database level is faster compared to processing the data in Informatica. You cannot remove all the transformations from the mapping. The part of transformation logic that can be pushed at source or target level is referred to as pushdown optimization.

Consider that you have a mapping with a sequence indicated as follows:

Source - Source Qualifier - Filter - Sorter - Aggregator - Expression - Lookup - Rank - Target

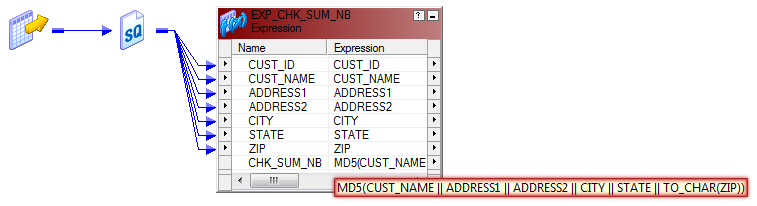
In filter transformation, we are filtering the data on a particular location. In sorter transformation, the data is sorted on a particular department ID. In aggregator, we are grouping the data on department ID. In expression transformation, the unconnected lookup transformation is called using the :LKP function, and finally, rank is used to get the top salaried employee in the target.

We can remove the filter transformation, sorter transformation, and aggregator transformation by adding the WHERE clause, ORDER BY clause, and GROUP BY clause, respectively, in the SQL override in the source qualifier transformation. We cannot remove expression transformation as we cannot write the :LKP function in SQL override.

So our mapping becomes simple after using SQL override indicated as follows:

Source - Source Qualifier - Expression - Lookup - Rank - Target

Pushdown optimization will help in saving the processing time by extracting lesser numbers of records of data from the source and also by processing lesser number of records in the transformations.



What is the MD5 Function in Informatica?

MD5 (Message Digest Function) is a hash function in Informatica which is used to evaluate data integrity.  The MD5 function uses Message-Digest Algorithm 5 (MD5) and **calculates the checksum of the input value**.  MD5 is a one-way cryptographic hash function with a 128-bit hash value.

MD5 returns a 32 character string of hexadecimal digits 0-9 & a-f and returns NULL if the input is a null value.

Example:

When you wish to write changed data to a database.  Use MD5 to generate checksum values for rows of data that you read from a source.

**When you run a session, compare the previously generated checksum values against the new checksum values.**  Then, write the rows with an updated checksum value to the target.  You can conclude that an updated checksum value would indicate that the data has changed.

Change data capture (CDC) can be done in many ways.  There are methodologies such as *Timestamp, Versioning, Status indicators, Triggers and Transaction* logs and *Checksum*.  The advantage of using MD5 function is to reduce overall extract-transform-load (ETL) run-time and the cache memory usage, by caching only the required fields which are of utmost importance.

When To Use MD5 Function in Informatica?

**Only update the changed records (any column change) in the target.**  Instead of passing all existing records to the target for update, ( whether changed or unchanged ) it's always recommended to compare the records.

Column-to-column comparison is little painful if your column counts are high.  Concatenate all columns and use the MD5 Function (source and target records for the same key) and then compare the output from the MD5 Function.  The changed records can be identified and only those records can be updated into the target.  This calculation is done really fast and the output of MD5 Function can be used as a unique key to differentiate records

MD5 will help in improving performance when compared to lookups only if the comparison columns are more than 10.  MD5 function enhances the performance as compared to lookups only when the comparison columns are more than 10.

Limitations

If we have non-string columns (SMALLINT, INT, NUMBER, etc.) we need to convert them into characters using TO\_CHAR function because the MD5 Function only validates character strings.  If you don't use the TO\_CHAR, the output port using MD5 may become invalid.  Sometimes it does validate the output port using MD5, but the warning message generated may read validated and non-string data type (e.g. SMALLINT) columns are converted to string.  This warning has a great performance impact.

Normal Approach

To identify records for updates and inserts, we use a lookup transformation.  The cache built by lookup depends on two factors:

1. The number of columns in the comparison condition
2. The amount of data in the lookup table.

When there is not a primary key column to identify the changes, there are two options:

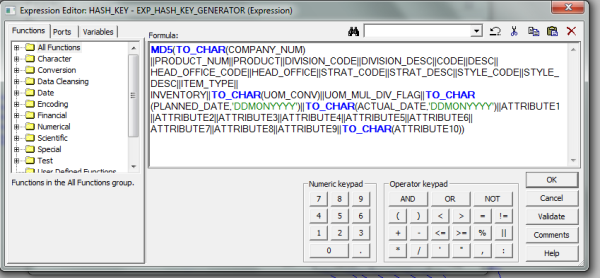
1. Compare all the columns in the lookup or...
2. Compare the data using the concept of power exchange change data capture.

Beware that this can degrade the performance.  In this scenario, using the MD5 Function is our best option.

Scenario

Consider a scenario where the incoming Product records consist of PRODUCT\_NUM, PRODUCT\_DESC, and address fields which have no primary key are associated with them.  In such a scenario, it is imperative that a unique identifier be assigned to these records on-the-fly which is immutable.  This on-the-fly unique identifier can also be used in future loads as a key to identify whether an incoming customer record is a potential update or an insert.

The MD5 Function generates a unique hexadecimal string 32 which is character-wide for a given input string.  In this example, the source table of a customer include customer details such as PRODUCT\_NUM, PRODUCT\_DESC, and address fields.  The MD5 Function in an expression is used to assign a 32 character-wide key to each of these records and load them in target file.



Conclusion

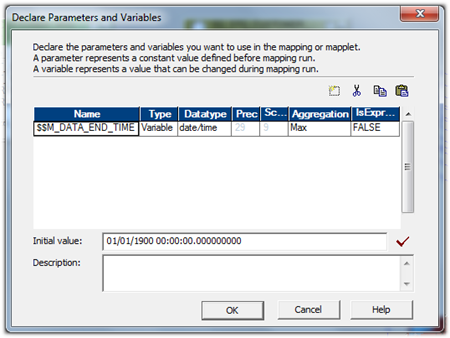
By using the MD5 values, we can identify whether the data is changed or unchanged without the performance being degraded and data is handled in the most efficient way possible. The MD5 value is always recommended for scenarios with many comparison columns and no primary key columns in the lookup table.   There is limitation, however; the input to the MD5 values needs to be a string by data type and it returns a 32 bit hexadecimal.

**Change Data Capture Implementation**

Now we understand the mapping variables, lets go ahead and start building our mapping with [Change Data Capture](http://www.disoln.org/2013/03/Change-Data-Capture-Implementation-for-Multi-Sourced-ETL-Process.html). Here we are going to implement [Change Data Capture](http://www.disoln.org/2012/12/Change-Data-Capture-CDC-Implementation-Using-CHECKSUM-Number.html) for CUSTOMER data load. We need to load any new customer or changed customers data to a flat file. Since the column UPDATE\_TS value changes for any new or updated customer record, we will be able to find the new or changed customer records using UPDATE\_TScolumn. 

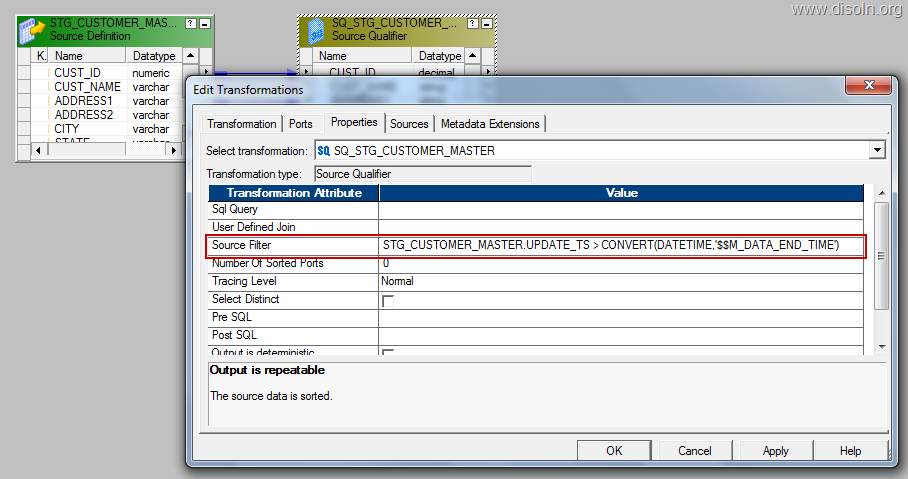
As the first step lets start the mapping and create a mapping variable as shown in below image.

* + $$M\_DATA\_END\_TIME as Date/Time

[](http://lh6.ggpht.com/-_sHvDtOQOIM/UHes3ESSEjI/AAAAAAAAFp8/cA8W_xx8ELE/s1600-h/image%5B11%5D.png)

Now bring in the source and source qualified to the mapping designer workspace. Open the source qualifier and give the filter condition to get the latest data from the source as shown below.

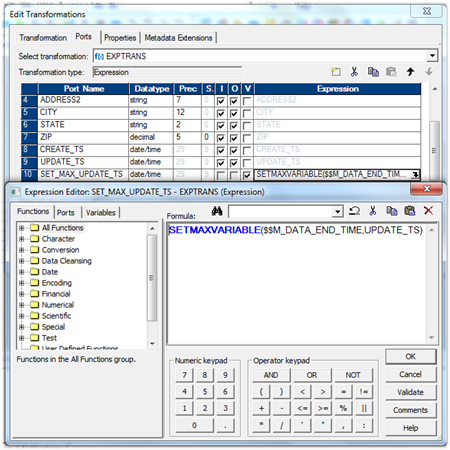
* + STG\_CUSTOMER\_MASTER.UPDATE\_TS > CONVERT(DATETIME,'$$M\_DATA\_END\_TIME')



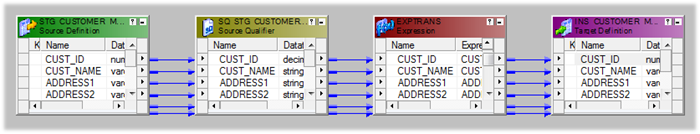
Note : This filter condition will make sure that, latest data is pulled from the source table each and every time. Latest value for the variable $M\_DATA\_END\_TIME is retrieved from the repository every time the session is run. 

Now map the column UPDATE\_TS to an expression transformation and create a variable expression as below.

* + SETMAXVARIABLE($M\_DATA\_END\_TIME,UPDATE\_TS)

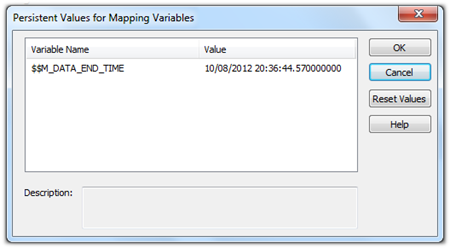
[](http://lh5.ggpht.com/-wkXWeSOXjU0/UHes30RqY7I/AAAAAAAAFqM/tK8ZTh6CFnU/s1600-h/image%5B20%5D.png)

Note : This expression will make sure that, latest value from the the column UPDATE\_TS is stored into the repository after the successful completion of the session run.Now you can map all the remaining columns to the down stream transformation and complete all other transformation required in the mapping.

[](http://lh3.ggpht.com/-bqC0GTJxHxU/UHes4_NIJfI/AAAAAAAAFqc/IHzZOZq_gcY/s1600-h/image%5B25%5D.png)

That’s all you need to configure Change Data Capture, Now create your workflow and run the workflow. Once you look into the session log file you can see the mapping variable value is retrieved from the repository and used in the source SQL, just like shown in the image below.[http://lh6.ggpht.com/-boEI6Hod5wo/UHes6ZXxL9I/AAAAAAAAFq0/vFnBgzKrMmc/image_thumb%25255B15%25255D.png?imgmax=800](http://lh4.ggpht.com/-f79dyLcritU/UHes5y55zII/AAAAAAAAFqs/lKfOaVYlUfs/s1600-h/image%5B29%5D.png)

You can look at the mapping variable value stored in the repository, from workflow manager. Choose the session from the workspace, right click and select 'View Persistent Value'. You get the mapping variable in a pop up window, like shown below.

[](http://lh5.ggpht.com/--J6ZUQepFhM/UHes6txLX7I/AAAAAAAAFq8/cTRXl0Fs7Wk/s1600-h/image%5B7%5D.png)

**Data Masking in Informatica**

**Data Masking** is one of the latest features introduced  by [Informatica](http://dwhlaureate.blogspot.in/2012/08/informatica-9-features.html" \t "_blank)and is big advantage considering the security of sensitive data stored in [Datawarehouse](http://www.blogger.com/%20http:/dwhlaureate.blogspot.in/2012/07/datawarehouse.html" \t "_blank).For eg

* SSN number of a customer
* Credit Card number
* Phone number etc…

This is done by masking the original data using various masking techniques   i.e. converting the original data into some other readable format. This is done without affecting the granularity of the table ,all the integrity constraints are kept the same.The objective behind using data masking transformation is to restrict the outside world from accessing the Customer Information.

Since data ,masking is an integrated option available in **@Informatica Power centre** it greatly reduces the Development effort.

**Why we need data masking?**

Normally in projects we have non production environments like development, testing and training . Often the IT companies are careless about the security  in the developments phase. While moving the code into actual production environment they rarely notice that the information can be leaked out.So we need to be careful here.In order to avoid this situation we can do masking at the development phase itself and then outsource the data.That we will be more safer.

Another advantage is that we can avoid the use of Dummy values in the testing and development.Instead of using Dummy values we can use the actual and then mask it and the successfully promote to the Live Environment.

Now we will go through the different algorithms used for data masking

* **Nondeterministic randomization**:Replace a sensitive field with a randomly generated .This can be some either any number or date field.
* **Blurring:** Add a random variance to the original value .Variance can be small or larger. For example adding 10 percent increase in the salary of employee.
* **Repeatable masking:** Maintain referential integrity by using values that are unique and different.Eg Replace the SSN number 1/45/89 with 12/22/44
* **Substitution:**  Randomly substitute original values with entirely different but real valuesE.g., substitute “Wayne Smith” with “Don Kelly”)

[Data quality](http://www.disoln.org/2012/10/User-Defined-Error-Handling-in-Informatica-PowerCenter.html) is very critical to the success of every data warehouse projects. So ETL Architects and Data Architects spent a lot of time defining the [error handling](http://www.disoln.org/2012/07/error-handling-made-easy-using.html)approach. Informatica PowerCenter is given with a set of options to take care of the error handling in your ETL Jobs. In this article, lets see how do we leverage the PowerCenter options to handle your exceptions.

**Error Classification**

You have to deal with different type of errors in the ETL Job. When you run a session, the PowerCenter Integration Service can encounter fatal or non-fatal errors. Typical error handling includes:

* + **User Defined Exceptions**: Data issues critical to the data quality, which might get loaded to the database unless explicitly checked for quality. For example, a credit card transaction with a future transaction data can get loaded into the database unless the transaction date of every record is checked.
  + **Non-Fatal Exceptions**: Error which would get ignored by Informatica PowerCenter and cause the records dropout from target table otherwise handled in the ETL logic. For example,  a data conversion transformation error out and fail the record from loading to the target table.
  + **Fatal Exceptions**: Errors such as database connection errors, which forces Informatica PowerCenter to stop running the workflow.

**I. User Defined Exceptions**



Business users define the user defined user defined exception, which is critical to the data quality. We can setup the user defined error handling using;

* + - 1. [Error Handling Functions](http://www.disoln.org/2012/10/User-Defined-Error-Handling-in-Informatica-PowerCenter.html).
      2. User Defined Error Tables.

**1. Error Handling Functions**

We can use two functions provided by Informatica PowerCenter to define our user defined error capture logic.

**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.

You can use ERROR in Expression transformations to validate data. Generally, you use ERROR within an IIF or DECODE function to set rules for skipping rows.

Eg : IIF(TRANS\_DATA > SYSDATE,ERROR('Invalid Transaction Date'))

Above expression raises an error and drops any record whose transaction data is greater than the current date from the ETL process and the target table.

**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.

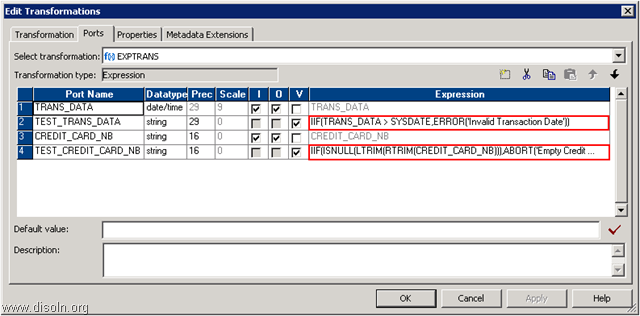
You can use ABORT in Expression transformations to validate data.

Eg : IIF(ISNULL(LTRIM(RTRIM(CREDIT\_CARD\_NB))),ABORT('Empty Credit Card Number'))

Above expression aborts the session if any one of the transaction records are coming with out a credit card number.

**Error Handling Function Use Case**

Below shown is the configuration required in the expression transformation using ABORT() and ERROR() Function. This transformation is using the expressions as shown in above examples.

[](http://lh4.ggpht.com/-Ps9JKWcCkZ8/U0JGFXzYFyI/AAAAAAAAJS4/y5P3ShnPgoU/s1600-h/image%5B36%5D.png)

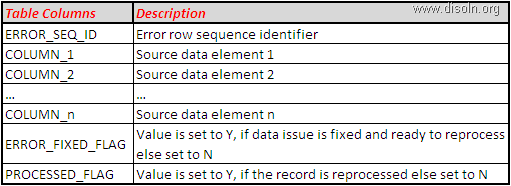
**Note**:- You need to use these two functions in a mapping along with a session configuration for [row error logging](http://www.disoln.org/2012/07/error-handling-made-easy-using.html) to capture the error data from the source system. Depending on the session configuration, source data will be collected into Informatica predefined PMERR error tables or files.

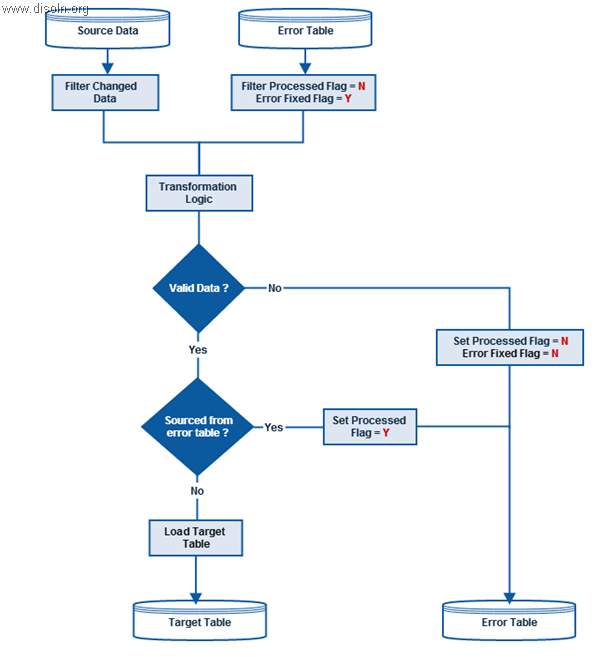
Please refer the article "[User Defined Error Handling in Informatica PowerCenter](http://www.disoln.org/2012/10/User-Defined-Error-Handling-in-Informatica-PowerCenter.html)**"**for more detailed level implementation information on user defined error handling.

**2. User Defined Error Tables**

Error Handling Functions are easy to implement with very less coding efforts, but at the same time there are some disadvantages such as readability of the error records from the PMERR tables and [performance](http://www.disoln.org/search/label/Performance%20Tips?max-results=8) impact. To avoid the disadvantages of error handling functions, you can create your own error log tables and capture the error records into it.

Typical approach is to create an error table which is similar in structure to the source table. Error tables will include additional columns to tag the records as "error fixed", "processed". Below is a sample error table. This error table includes all the columns from the source table and additional columns to identify the status of the error record.

[](http://lh3.ggpht.com/-SXg7jFaTW9I/U0Ixcb-DGxI/AAAAAAAAJR8/LOamayJqQ0E/s1600-h/image%5B12%5D.png)

Below is the high level design.[](http://lh5.ggpht.com/-eVoD2A6KJec/U0JCLWmtkAI/AAAAAAAAJSU/XqiOP7reDZM/s1600-h/Error%20Processing%5B12%5D.png)

Typical [ETL Design](http://www.disoln.org/search/label/ETL%20Design?&max-results=15) will read error data from the error table along with the source data. During the data transformation, data quality will be checked and any record violating the quality check will be moved to error tables. Record flags will be used to identify the reprocessed and records which are fixed for reprocessing.

**II. Non-Fatal Exceptions**



Non-fatal exception causes the records to be dropped out in the ETL process, which is critical to quality. You can handle non-fatal exceptions using;

* + - 1. Default Port Value Setting.
      2. Row Error Logging.
      3. Error Handling Settings.

**1. Default Port Value Setting**

Using default value property is a good way to handle exceptions due to NULL values and unexpected transformation errors. The Designer assigns default values to handle null values and output transformation errors. [PowerCenter Designer](http://www.disoln.org/2012/08/Understand-Informatica-PowerCenter-Mapping-Designer.html) let you override the default value in input, output and input/output ports.

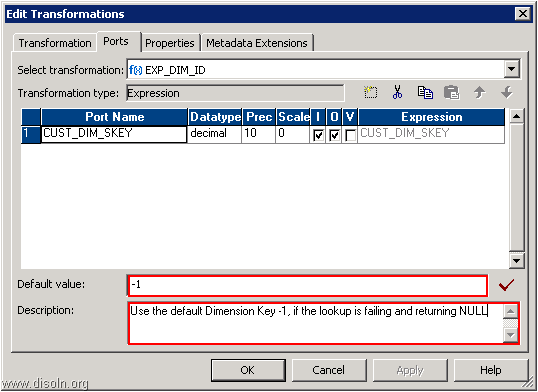
Default value property behaves differently for different port types;

* **Input ports** : Use default values if you do not want the Integration Service to treat null values as NULL.
* **Output ports** : Use default values if you do not want to skip the row due to transformation error or if you want to write a specific message with the skipped row to the session log.
* **Input/output ports** : Use default values if you do not want the Integration Service to treat null values as NULL. But no user-defined default values for output transformation errors in an input/output port.

**Default Value Use Case**

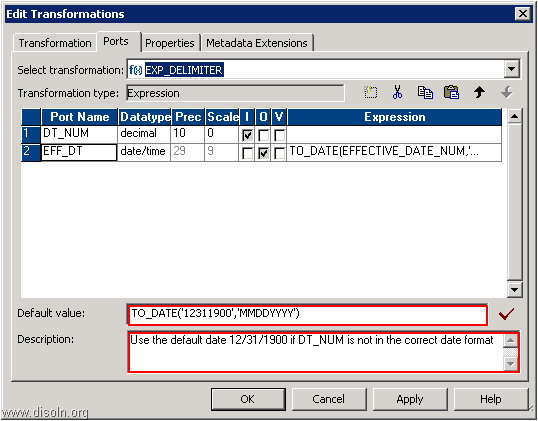
**Use Case 1**

Below shown is the setting required to handle NULL values. This setting converts any NULL value returned by the [dimension](http://www.disoln.org/2012/08/slowly-changing-dimension-type-2-implementation-using-informatica.html) lookup to the default value -1. This technique can be used to handle [late arriving dimensions](http://www.disoln.org/2013/12/Design-Approach-to-Handle-Late-Arriving-Dimensions-and-Late-Arriving-Facts.html)

[](http://lh3.ggpht.com/-iAbkIOrccS8/U0JCMQsTnNI/AAAAAAAAJSg/t8MWICQXrKs/s1600-h/image%5B25%5D.png)

**Use Case 2**

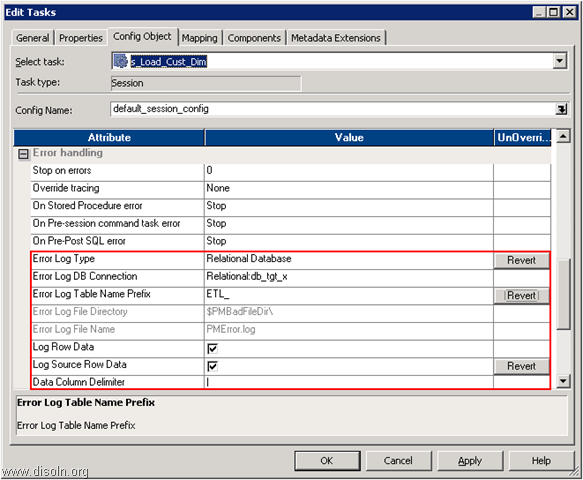
Below setting uses the default expression to convert the date if the incoming value is not in a valid date format.

[](http://lh4.ggpht.com/-Py56aSkSqLg/U0JSu_emPiI/AAAAAAAAJTQ/L-y8fPez8Vo/s1600-h/image%5B47%5D.png)

**2. Row Error Logging**

Row error logging helps in capturing any exception, which is not consider during the design and coded in the mapping. It is the perfect way of capturing any unexpected errors.

Below shown session error handling setting will capture any un handled error into PMERR tables.

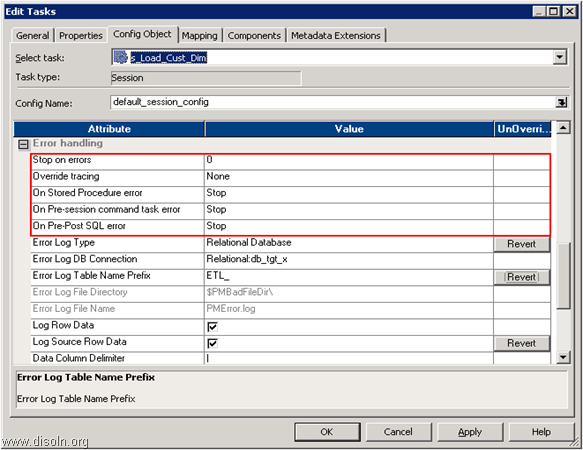
[](http://lh5.ggpht.com/-faZ4fR4Q-X8/U0JVzfnxdtI/AAAAAAAAJTk/txvtOHH0LQY/s1600-h/image%5B62%5D.png)

Please refer the article [Error Handling Made Easy Using Informatica Row Error Logging](http://www.disoln.org/2012/07/error-handling-made-easy-using.html) for more details.

**3. Error Handling Settings**

Error handling properties at the session level is given with options such as Stop On Errors, Stored Procedure Error,  Pre-Session Command Task Error and Pre-Post SQL Error. You can use these properties to ignore or set the session to fail if any such error occurs.

* **Stop On Errors** : Indicates how many non-fatal errors the Integration Service can encounter before it stops the session.
* **On Stored Procedure Error** : If you select Stop Session, the Integration Service stops the session on errors executing a pre-session or post-session stored procedure.
* **On Pre-Session Command Task Error** : If you select Stop Session, the Integration Service stops the session on errors executing pre-session shell commands.
* **Pre-Post SQL Error** : If you select Stop Session, the Integration Service stops the session errors executing pre-session or post-session SQL.

[](http://lh3.ggpht.com/-qwTYuhVWnmA/U0JV0fNHNaI/AAAAAAAAJT0/YENBCiMO7dc/s1600-h/image%5B73%5D.png)

**III. Fatal Exceptions**

A fatal error occurs when the Integration Service cannot access the source, target, or repository. When the session encounters fatal error, the PowerCenter Integration Service terminates the session. To handle fatal errors, you can either use a [restartable ETL design](http://www.disoln.org/2013/02/Restartability-Design-for-Different-Type-ETL-Loads.html) for your workflow or use the [workflow recovery features](http://www.disoln.org/2013/07/Workflow-Recovery-Configuration-for-Informatica-PowerCenter-Workflows.html) of Informatica PowerCenter

* + - 1. [Restartable ETL Design](http://www.disoln.org/2013/02/Restartability-Design-for-Different-Type-ETL-Loads.html)
      2. [Workflow Recovery](http://www.disoln.org/2013/07/Workflow-Recovery-Configuration-for-Informatica-PowerCenter-Workflows.html)

**1. Restartable ETL Design**

Restartability is the ability to restart an ETL job if a processing step fails to execute properly. This will avoid the need of any manual cleaning up before a failed job can restart. You want the ability to restart processing at the step where it failed as well as the ability to restart the entire ETL session.

Please refer the article "[Restartability Design Pattern for Different Type ETL Loads](http://www.disoln.org/2013/02/Restartability-Design-for-Different-Type-ETL-Loads.html)**"**for more details on restartable ETL design.

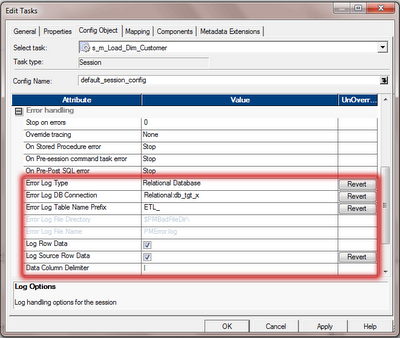
**2. Workflow Recovery**

Workflow recovery allows you to continue processing the workflow and workflow tasks from the point of interruption. During the workflow recovery process Integration Service access the workflow state, which is stored in memory or on disk based on the recovery configuration. The workflow state of operation includes the status of tasks in the workflow and workflow variable values.

Every time we start with a Data Warehouse or Data Integration project we spent lot of time defining our error handling approach. Still there are good chances that we might miss some scenarios because of unexpected data issues. Here in this article, lets us discuss an approach to handle the unexpected error.

Error handling Configuration.

We do not have any error configuration required during the mapping development. So after the mapping is created, during the session configuration set the session properties as shown in below image.

[](http://3.bp.blogspot.com/-EBtdaXQ-iNs/UANftysNu5I/AAAAAAAAEd4/bk_M9YC_940/s1600/error-handling-config.png)

Here is the details on what each property means.

Error Log Type :- Specifies the type of error log to create. It can be relational database or flat file.

Error Log DB Connection :- Database connection for a relational log.

Error Log Table Name Prefix :- Specifies the table name prefix for relational logs.

Log Row Data :- Specifies whether or not to log transformation row data.

Log Source Row Data :- Specifies whether or not to log source row data.

Data Column Delimiter :- Data will be delimited by the specified character in DB column.

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 configuration, we are done with the setting required to capture any error during the session execution. Now lets see how do we retrieve the data from the error log tables and report it to Business Users or IT Department.

Report the Error Data.

Now we have the error data stored in the error table, it is important to share the error data to the Business Users or to IT Department. Lets see how we can pull data from these tables.We can pull the basic error report using the SQL. We can be more fancy with the SQL and get more information from the error tables.

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>

The above SQL can be converted to an Informatica mapping and can be scheduled to run after all your ETL jobs are completed to get a daily error report and the same can be scheduled for an email delivery to the data owners to take relevant action on the errors.

Pros and Cons of this Approach.

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

Pros.

Out of the box Solution Provided by Informatica.

Captures all the Error, Including the unthought error.

Less Coding and Testing efforts required by the development team.

Cons.

If an error occurs after an active transformation, Integration service will not be able to capture source data into the erro table.

In sessions with multiple non-pass through partitions, Integration service can not capture source data into error table.

Added overhead to the Session performance (Which is expected and acceptable).

Please leave us a comment below, if you have any difficulties implementing this error handling approach. We will be more than happy to help you.

I have 3 delimiter flat files, one file is comma separated, 2nd file is tab separated, 3rd file is pipeline separated. Can we load it single target using Indirect load??

2)I have 1 flat file with multiple delimiters.

Ex..X1234A4 ,234543$ 297 | Y

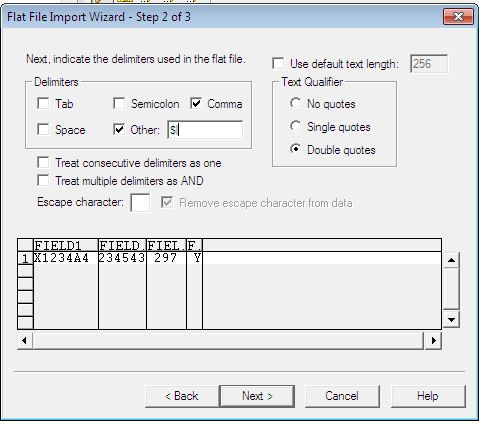
Here $| are the delimiters. How to handle it?

For scenario 1, you can create 3 flows in 1 mapping. 1 for each type of file, and you can control the load order of the flows from 'Mappings -> Target Load Plan'

Now in the session properites of each source you can define the delimiter for respective file

For case 2 with files having multiple delimiters, one option is to place ,$| in the column delimiters box and uncheck'Treat multiple delimiters as AND' option.

Other option is to do a sed replace both $ and | to , in a pre session command and use , as the delimiter



**->Use the input in quoted strings. It will avoid the interpretation of comma in value as delimiter. Go to Advanced properties check treat multiple delimeter as AND.**

**->A simple thing that you can try out, create a UTF-8 (unicode) file with all the special characters as inputs in field1 on separate lines of the file. You can do this by saving a notepad file with Encoding as UTF-8.   
  
Next in an expression transformation, use the ASCII function like ASCII(field1) in an output port. Pass the field1 and ASCII values to another flat file that will give you the ASCII values for each special character that you get. So, whenever you get an new special character, you can do this exercise of generating the ASCII value for it.   
  
Also. the Integration Service requires that the code page environment variable be set to UTF-8. In session properties for the source input file, go to Set File Properties > Click Advanced > Select Codepage as UTF-8 encoding of Unicode. Run this session and it will produce the output file with ASCII equivalent of the special characters.   
  
eg: ASCII for '9' is suppose 57 and I need to replace the character '9' in one of the fields with a blank string (''), I can use the function as   
REPLACESTR(1, input\_string, CHR(57), '').   
  
Similarly, to replace all the special characters, you can write a nested REPLACESTR function, based on the ASCII values you get in the output file. You first need to decide how to handle these special characters i.e. whether you want to pass them or replace them with something else. If you just want to pass them, setting the code page at the different places I have mentioned should work, else if you want to replace them, with a blank string, an example I shown below:   
  
REPLACESTR(1,   
REPLACESTR(1,   
REPLACESTR(1,   
REPLACESTR(1,   
input\_string,   
CHR(193), 'A'),   
CHR(225), 'a'),   
CHR(198), 'AE'),   
CHR(230), 'ae'))))   
  
where CHR(value) -- value is the ASCII equivalent of the special characters.**