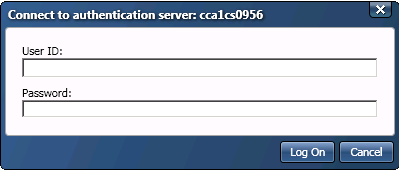
Documentation for Data Management studio (dataflux)

This document is intended to highligth the information needed in order to create and implement monitoring business rules.

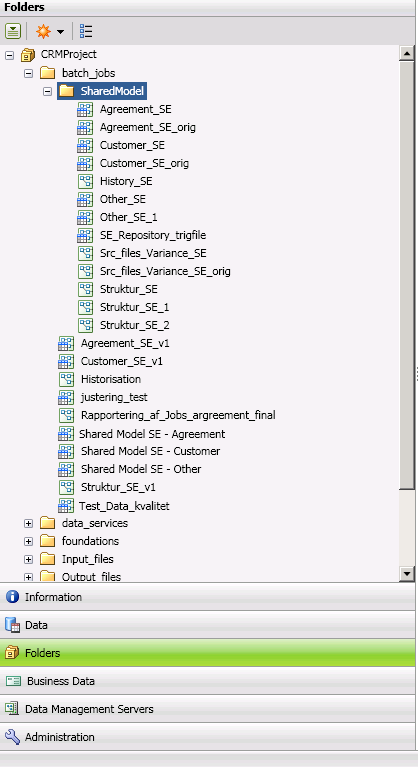
# DM Studio

Start by logging into Data Management studio using your windows credentials without the domain.



## A look into the environment:

The work area when creating new jobs is located in the CRMProject folder within the Folders tab highlighted with red circle. From here you would have the possibility to create process jobs and data jobs.



### The process job:

Implementing new monitor business rules, you need to work with process jobs.

Process jobs per definition: “A process job combines data processing with conditional processing. Data job nodes can beaded to a process flow to encapsulate all of the data processing power of a data job into a node in the process flow.”

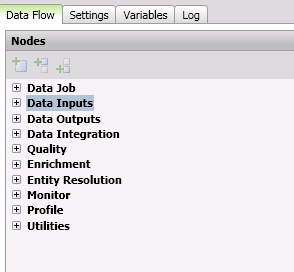
In process jobs you can call data jobs nodes or reference to data jobs nodes, call SAS code nodes or reference SAS code, setup event listeners or use IF-Then clauses to mention a few.

In some of the jobs already created the process jobs include both SAS code nodes as well as data jobs.

### The Data job:

Data jobs are the main approach to process data in Dataflux data management studio. Each data job specifies a set of data-processing operations from source to target.

The data-processing operations are presented below. In the data tools you will primarily be focusing on monitoring nodes together with the data input, - outputs and – Integration nodes.



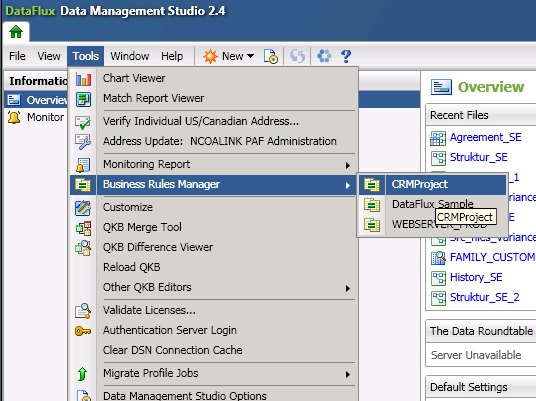
# Business Rules Manager:

A feature in data management studio is to easily incorporate Business rules and make them accessible to the entire organisation in the form of dashboards and monitoring view of data triggers that was acceptable to the business rules defined.

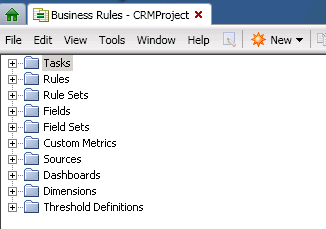
In dataflux data management studio, a business rule is a reusable object that specifies a dataflux expression. The Expression evaluates the value in one or more table fields. Event can be triggered based on the results that are turned by the rule. A business rule can be used in one or more data jobs.

A task specifies one or more rules and one or more events that can be triggered based on the results that are returned by a rule.

In order to access, the business Rules Manager go to Tools->Business Rules Manager and choose CRMProject as shown below.



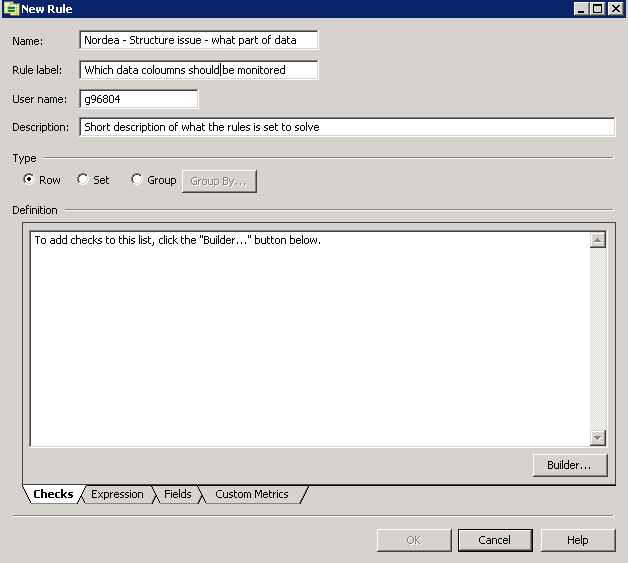
Opening the CRMProject you are presented with a set of opportunities. In the start you should focus on Tasks, Rules and fields.



## Create a Rule:

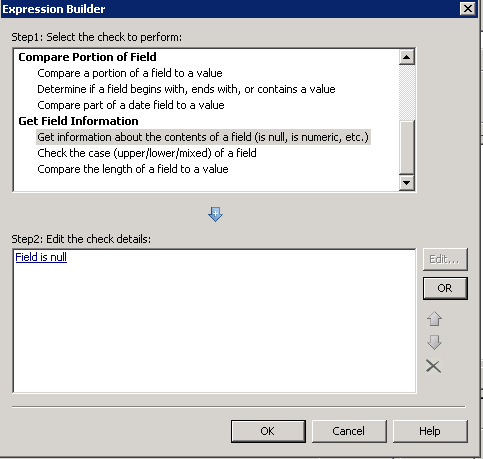
The first part of creating a new business rule is to go under Rules and create a new Rule. When creating a new Rule provide as many information’s as possible since this will make the process of identifying a business rules in the future easier. Below are examples of a useful description.

The Type -> see blue circle always has to be set to Row.



The next part is to define a Builder. The builder contains a kind of IF-Then and where functionality without having to write any code. The builder is divided into three areas

1. Compare Entire field
2. Compare a portion
3. Get a field information

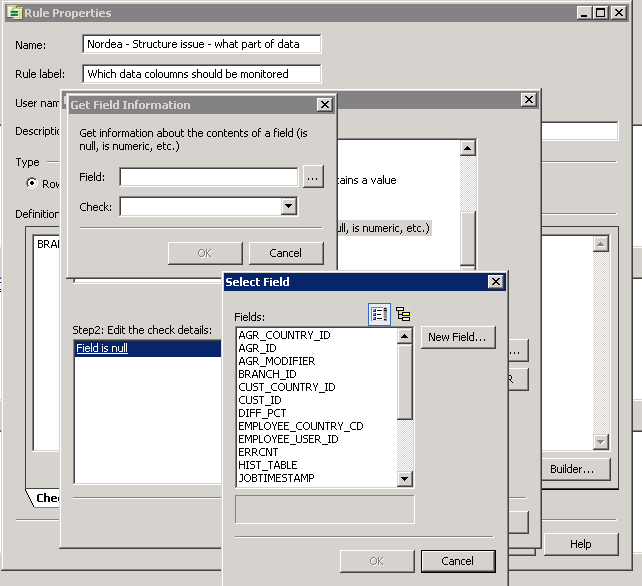


First step: The first step is to define which of the three areas you are going to use. Next the check that you would perform ex. “Get Information about the contents of a field” from the “Get Field Information” area.

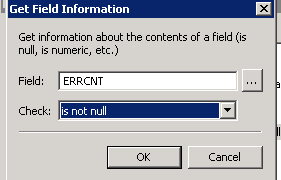
Second step: Edit the check “Field is Null” and setup the check needed to be performed. First, however you would need to create a new field or use an existing field name. This is done by pushing the button (red circle).

A field is the same as a SAS Variable or a Column name in T-SQL.

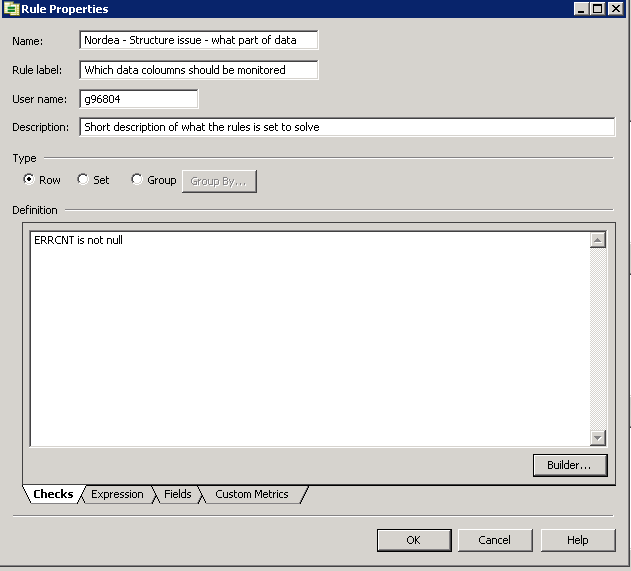
The field should equal the column from the table you use as input to the business rule. The field name should be defined as broad as possible in order to be used in more than one business rule.



When a field name has been created, the condition needs to be setup. Below an example of how a condition is created presented.

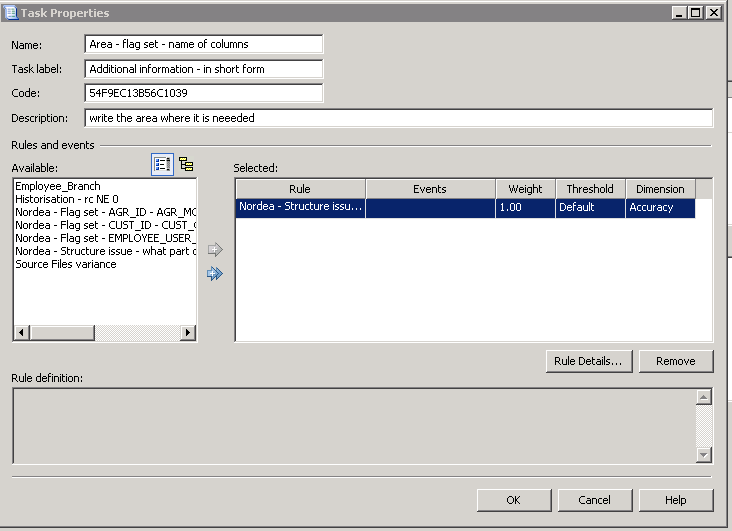


The Business rule now contains all the needed information and is finished.

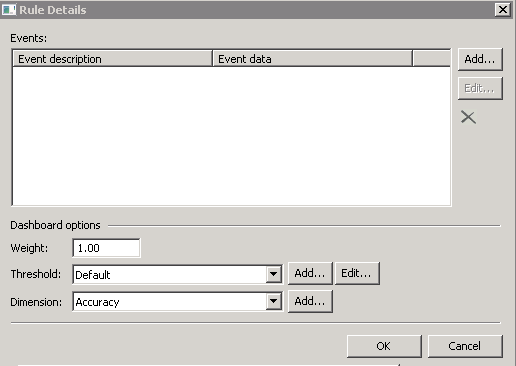


## Create a Task:

In order to use the rule created you need to create a task. The task is used to define which event needs to be triggered. Supply information’s as shown below except for the code line green circle - this information is automatically generated.

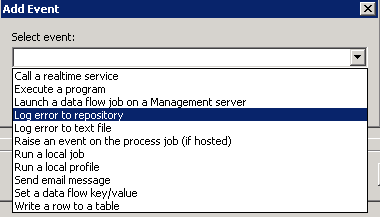


Move one or more available rules – so that the rule is displayed to the right – red circle. Next go into the Rule Details – yellow circle to setup an event for the given rule.

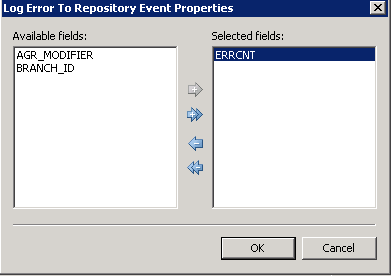


The “Add” bottom in the rule details window is related to dashboard definitions and is described in the document for DataFlux Web Studio.

For this example just click the add button.



The events presented above are the possible events to choose. You should choose the highlighted event “Log error to repository”. Next select the field/s you wish to monitor triggers for. In the example below the ERRCNT has been selected.



The business rule is now complete and needs to be referenced in the data job. This is archived by creating or referencing a data job with a monitoring node that reference the business rule, which is presented in the next chapter.

## Incorporate the business rules into the data job

In order to incorporate the rules into monitoring views and dashboard a data job needs to be setup. In this section you will see an example of how this can be managed.

Start by accessing the data job Guideline\_for\_monitoring in the folder tab under SharedModel

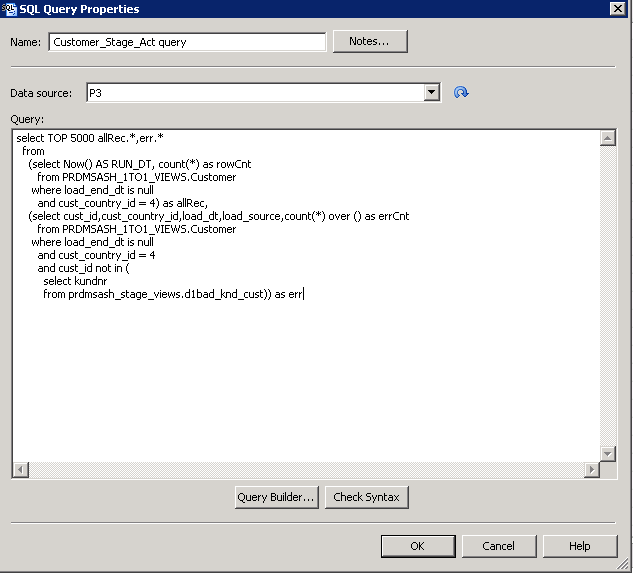


This job consists of four nodes:

1. Data input. (SQL query to access the Teradata)
2. Expression node (EEL (code language) query used for defining variables in an output dataset
3. Data output. (Create a new dataset to output the trigger files in SAS)
4. Monitoring node (creates the monitoring view and dashboards for the business rules)

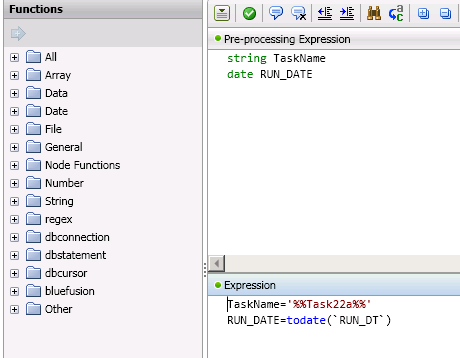
### Data input:

IT and analytics will provide the query for the data input needed to be monitored. An example of a query for data input is shown below.



### Expression node:

The expression node is the node for scripting EEL code in order to add extra functionality to.

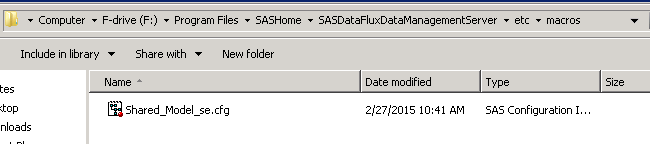


First: Notice the code %%Task22a%%. The %% indicate that a macro value is present with the name Task22a. The content of Task22a is set to “Customer\_Stage\_ACT” which is the taskname that is going to be monitored.

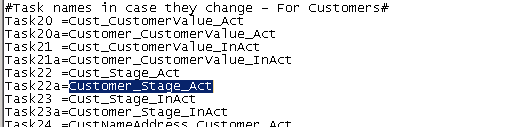
The reason for using a macro variable is the possibility to easily make changes to the taskname if needed.

Second: the code for RUN\_DT is to filter only the trigger files generated today.

All macro files gathered in the following location – see blow:



The content of Shared\_Model\_se.cfg is showned below. Please notice that there are two macro variables for each tasks. The macro Task22 is used to name the SAS dataset in the node below.



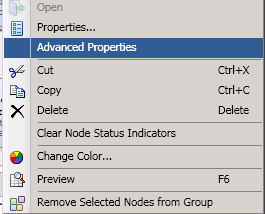
### Data output node:

The data output node is used to write all trigger files out to a dataset. The reasons for creating a SAS dataset are:

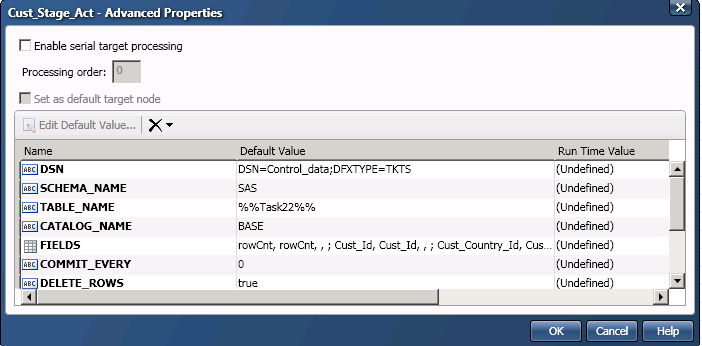
1. Generate historical information.
2. Possibility to view extra columns not visible in the Repository primary.

The node uses the macro file %%Task22%%, as shown below. In order to see if macro values are used on a node - right click the node and choose Advanced Properties.





The macro is shown in the blue circle below. The reason for using macro for naming dataset is due to the length limitation on the dataset name.

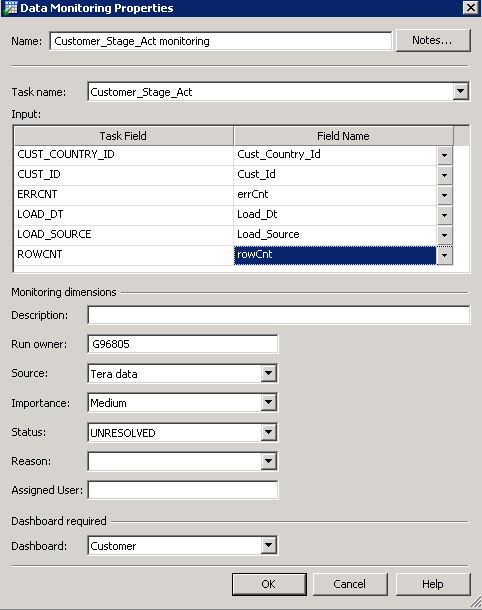


### Monitoring node:

The monitoring node for the task Customer\_Stage\_Act is the last node.



Clicking into the node – you will be presented with the following view:

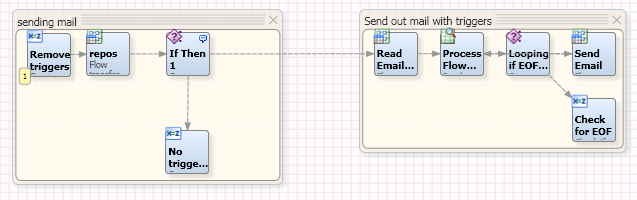


Task name is a dropdown menu where all task created are displayed. The Task field (the blue circle) contains all the input variables and the Field Name (red circle) contains all the fields you have created.

The monitoring dimensions are defined in the web studio document.

# Mail output template (part 1):

A template for sending out mail has been created and will be possible to use for varies jobs. The template for this documentation is named “Guideline\_mail\_output” and is a process job referencing data jobs and if then functionality.

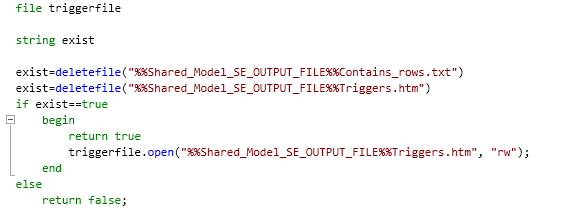


The first part of the process job is checking if triggers are present. If there are trigger files a html file is created and is sent to the second part of the process node if no triggers are found no email is sent.



### Expression node:

The expression node cleans up old trigger- and history files in order for the process flow to succeed.

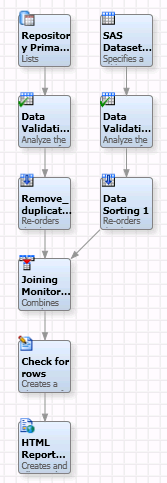


### Data job node:

Next node in process job is a data job containing its own node functionality.



The data job consist of the following nodes. For this job only the the most important nodes will be explained and examined.



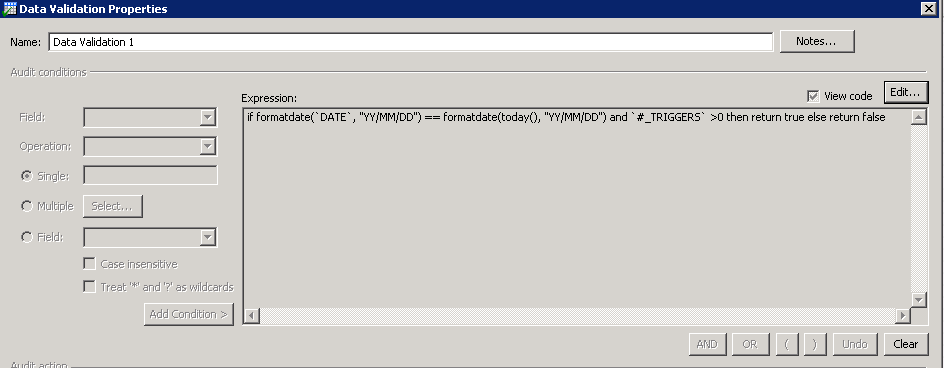
### Repository Primary:

The repository primary (yellow circle) contains all the information generated from varies monitoring nodes.

### Validation nodes:

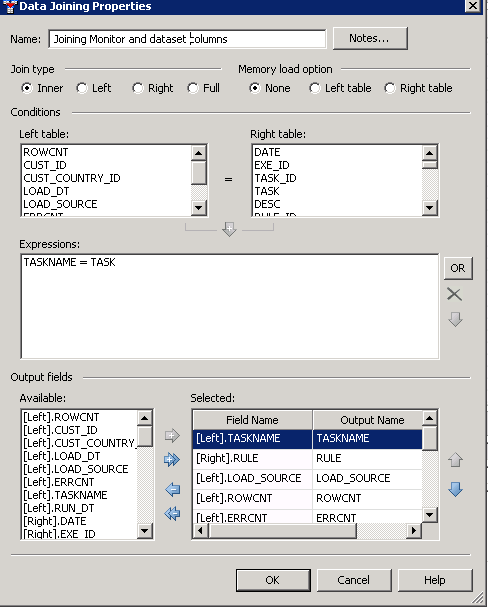
The validation node (green circle) is an interface for applying “if-then” and “where” clauses either in the form of drop boxes (the left side) or to write code as shown it the right side of the screen.

In this example the validation is used to view only rows that contain trigger values and have been discovered today.



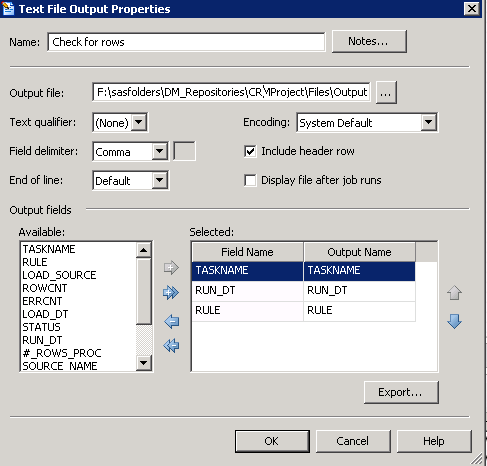
### Joining node:

In order to join data from the repository primary and the SAS dataset created earlier a joining node is needed (blue circle).



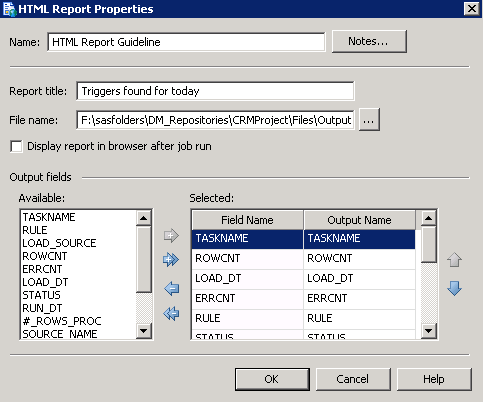
### Text file output node:

The text file output (red circle) is used to create a text file. The text file is only populated with content in the case that trigger files appears. This functionality is used for determining if a mail should be sent out or not.



### HTML Report node:

The HTML report node (black circle) is used for presenting the trigger files. Please notice that you can use a macro file for the report title and filename.



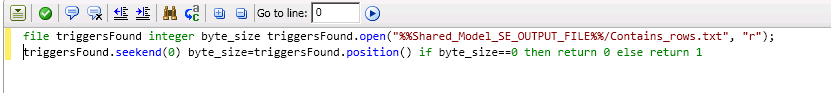
### IF then node:

The last part of the mail output template (part 1) is to determine if a mail should be sent out or not. This is determined within the IF Then 1 node viewed below.

1. If no trigger files records are found in the text file.
   1. No emails are sent out and the process stops in the expression node (green circle)
2. If trigger files are found in the text file.
   1. The process continues to part two of the mail output template.

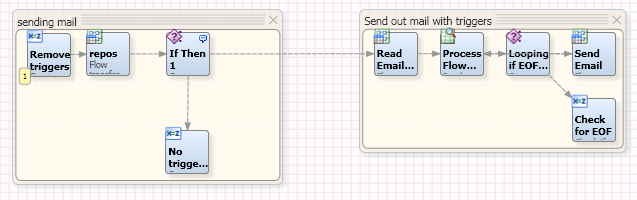


The code for declaring the IF Then node is presented below. Please notice that the text file in the data job node is the cause for determining if the job process should continue.



# Mail output template (part 2):

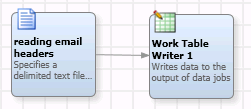
The second part of the template for sending out mail (green square) is for sending a mail to the recipients.



For this template I will only present the part regarding reading email information since this is the only part that needs to be changed when new mail recipient should receive trigger files.

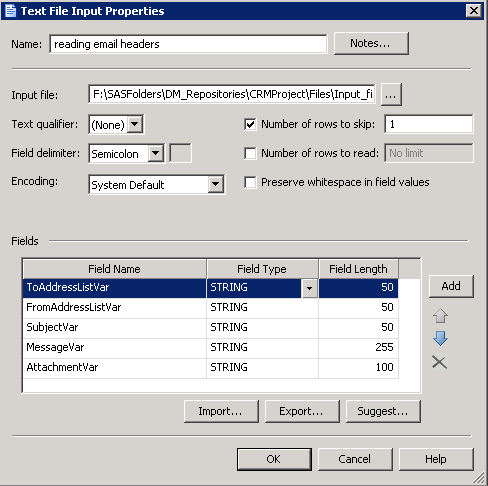
### Data job node:

In the data job (blue circle) you´re inputting a text file and outputting the content of the text file to a work table that resides in the memory and is used by the rest of the nodes in the process job.

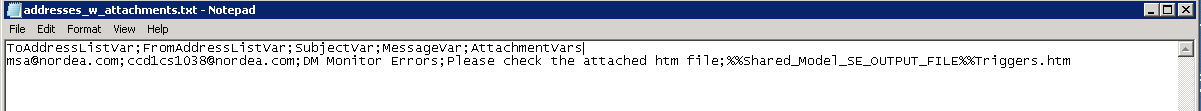


### Text file node:

The text file input used for reading the email header and content of the input files defined the property file. Be advised you have to add the field name, - type and – length yourself.



The corresponding text file is shown below.



### Work table writer node:

In order to make use of the mail information to the rest of the process job it needs to upload the content of the text file into worktable in memory as shown below.

