Talend User Component tFileInputTextFlat

cimt
let your work flow

Purpose **http://www.cimt-ag.de**

This component extracts field from flat text files with two different methods:

1. separates fields between delimiters
2. separates fields at absolute position

The advantages are:

- configure field position according to the names in the header line (also by regex)
- reads only the fields needed (unlike tFileInputDelimited)
- tolerant against enclosure chars within the content
- all error messages contains the line number and the field name or index
- after extracting field content you can use regex to do some post processing

Talend-Integration

This component can be found in the palette under File->Input
This component provides an output and an reject flow and several return values.

Parameters

Property	Content
File name	Full path of the file to import ***required***
Encoding	File encoding
Lines to Skip at Start	Number of lines to skip (In case you specify later you have an header line, don't count it here)
Skip empty lines	Detect empty lines and skip them
Schema	The schema. For date and timestamps you should specify here the pattern. ***requiered***
Has Column Header	Check it if your file provides an column header. This enables following options. The header will be skipped in the main output flow.
Use Header line to find position	The position of a delimited field will be set according to the position of the field name in the header line.
Find column position in header by regex	This option enables you to use a regularly expression in the field configuration “Name in Header”.
Ignore Not Null Constraints	Avoid throwing Exceptions if a value is null but the schema defines it as not null. This works for all columns.
Field Extraction	See explaining below
Field Separator	Char to separate the fields (only for field extracting by method Delimited Fields)
Text Enclosure	Char to enclosure the field content. It is helpful to read fields with line breaks as content. Don't escape double quotas here! E.g. for a double quoted field set “”” here. It works even if not all fields are putted in quotas.

Split Row before Field	If it is not allowed to have line breaks in field content, check this. This option helps to check the correct file structure. The performance of the line separation can be increased.
Allow Enclosure within Content	If the enclosure char can occurs in the content (not as enclosure) check this to avoid parsing problems. If switched off the file structure can be checked strict.
Locale For Number Format	To parse the number different then the local pattern, specify here the locale. The default is the English format. E.g. in German we have the following pattern: 999.999,99 In this case set the locale to “de”. Quotation required!
Default Date Pattern	For all date schema column without a defined pattern, this pattern here will be applied. If none of the provided date formats are matching, the component test for it self typical English, French and German formats and applying them. Only if this last attempt fails, the component throws and exception.
Die On Error	If true all errors stops the processing. If false all malformed data sets are send to the reject output flow (if added).

Field Extraction

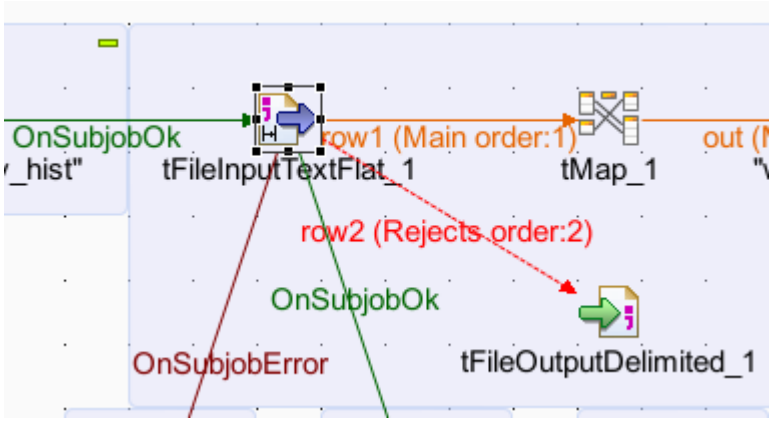
Column	Meaning
Column	Schema column name
Positioning	Field positioning method (extraction method)
Position	Position within delimiters or absolute position (depends on the chosen Positioning method) **<u>If blank for Delimited</u>**: field position comes from field index in schema **<u>If blank for Absolute Positioning</u>**: field position is the next after the last (it is a relative position to the previous field)
Length	Length of the field. It is required for absolute positioning. If a length is provided, the content will be trimmed to this length.
Regex	Regularly expression to post processing the field content. (no quotation needed)
Name in Header	Specify a different name for the column in the header. Field names often are not given according to the Java identifier naming rules. Therefore, you can specify the real name here (without quotation). If the option “Find column position in header by regex” is switch on you can write here regex (case insensitive). The regex expression must describe the whole possible column name in the header. It is not enough to declare only a pattern like contains. The regex expression will be surrounded from the component with ^ and $. Example: column name in the header line can vary from: Payment_Product to P_Product. To match a schema column you can write as Name in Header this regex: `P[a-z]*_Product` to match both (and more) possibilities
Ignore If Missing	Sometimes we get fields which does not have all columns expected. You can specify here this column can be missed without problems. If missing the values remains blank.

Return values

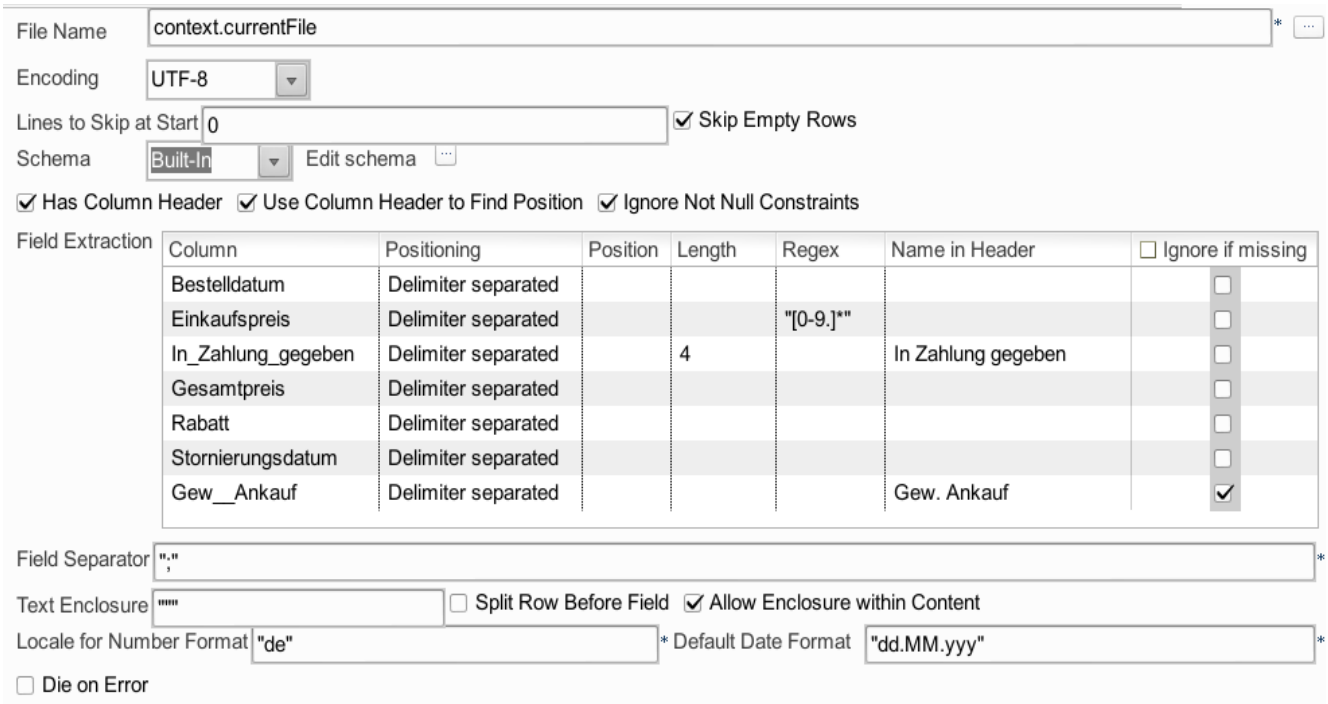
Return value	Content
ERROR_MESSAGE	Last error message
NB_LINE	Number of delivered lines
NB_REJECTED	Number of rejected lines

Szenario 1:

Reading a delimited file and write malformed line into a file.



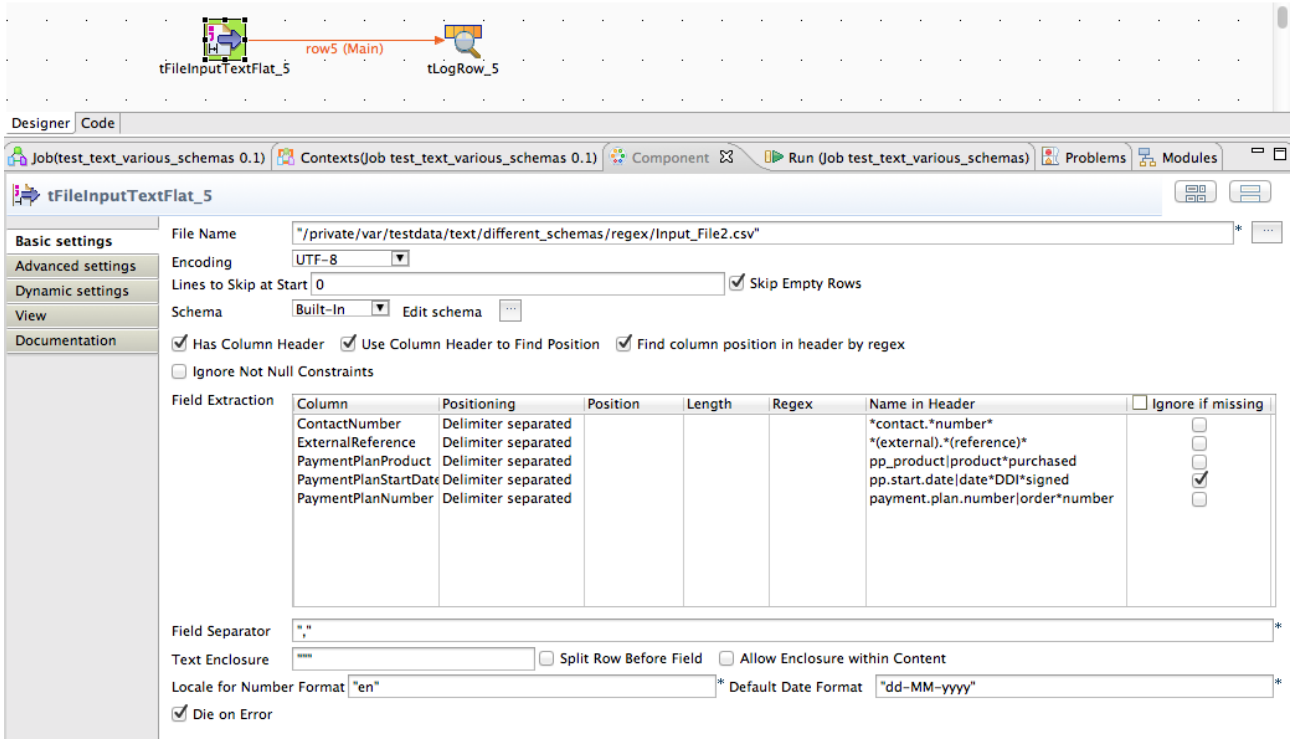
Configuration with examples of different header names.
It is an example of a file which has as first line a header line.



It is not necessary to specify the position if the position is identical to the schema column index.

Scenario 2:

Using regularly expression to find the correct field position by the header line of the file.
In case the input files are provided by a system or organization which cannot be motivated to a more fixed interface design. This should be avoided but sometimes it happens.



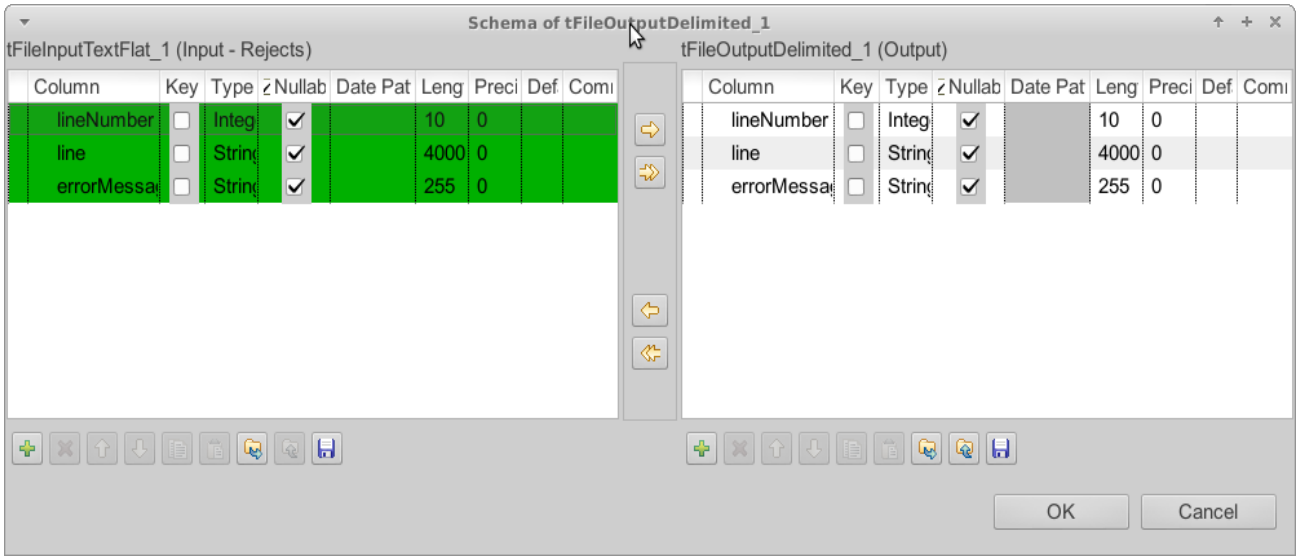
The current example as a header line like this:

```
ContactNumber,ExternalReference,Source,DateDDISigned,Title,ProductPurchased,InterestedInBeingACampaigner,PaymentFrequency,ChangeOfPaymentFrequency,StartDate,NextPaymentDue,AutoPaymentMethodStartDate,MailingDate,CallOutcome,WelcomeCallDate,PaymentMethod,AccountName,PaymentPlanDetailBalance,OrderNumber
```

Only 4 columns are needed and the names can be vary from file to file.
This matching can only be done by regex – see the screenshot above.

Reject flow:

The reject flow will only be filled if the option “Die On Error” is switched off.
If you add the reject flow, at first it contains all columns from the schema and these additional columns. Only the green columns will be filled, all other columns can be erased here.



The line number is the number of the extracted lines and can differ from the number in the file if fields contains line breaks.
The line is the input for the field extraction. You can find here the probably malformed content.
The error message contains the description of the problem occurred while parsing the line.

Possible error messages:

If a column is missing in the header line (assuming we digger for it) an exception will be thrown which says the missing column.

If a column matches to the an already used delimiter position the read of the first line fails with exceptions like this:

```
java.lang.Exception: nextDataRow failed in line 0:Extract field pp.start.date|
date*DDI*signed failed:Current field index 8 is lower then last field index:9
```

This is a failed check of the parser to avoid reading the content with a wrong configuration.
The component sorts all field internal by the position. It is a design decision to avoid reading a content unwanted twice.