Talend Component: tJasperReportExec

cimt
let your work flow
http://www.cimt-ag.de

Purpose

This component creates (compile + fill + export) reports based on Jasper Report designs (jrxml files).
Making reports in the ETL system provides multiple advantages:

- In the ETL system the knowledge exists when the necessary data are ready to use
- Mostly ETL systems provides very powerful scheduling and monitoring capabilities
- The file based output of the reports can be processed in nearly unlimited way (send per email, transfer to any location or embed in existing content – what ever the ETL system provides as possibilities

Advantages of this component apart from the Talend build-in components:

- Can use Talend database connections as data source (works in the same way as in JasperStudio)
- Can use XML files as data source or an empty data source (in case of the report actually layouts sub reports)
- Is not limited to provide parameters to the report
- Allows the various export option for the formats PDF and XLS/XLSX
- Detects automatically if the report or its sub reports must be compiled and compiles them.
- Includes the directory of the report and its sub reports to class path, though resources will be found automatically (e.g. resource bundles, images etc.)
- At least it is build on top of the current Jasper Library releases

Talend-Integration

This component can be found in the palette under Business Intelligence/Jasper
This component provides several return values.

Basic Parameters

Property	Content
Datasource	Choose the data source and the depending option **Database Connection**: **Connection**: Choose the database connection in the next chooser **XML file**: **XML file**: Choose the input data xml file **Use select expression from report**: **Select expression**: in case of the option above if false: XPath expression to the loop element **Date pattern**: pattern to interpret date strings to a Date type **Number pattern**: pattern to interpret a number string to a Number type **Dummy Records**: It is actually the Empty Data source of JasperStudio **Number records**: number of empty records to be created
Jrxml File	The report design file of the top report. The embed sub reports and resources will be found relative to this report file. Please refer to the chapter *Report Deployment*
Output Type	The report will be exported as file based output. Choose the type of output. Based on this type in the advanced settings it is possible to setup additional options related to the output type.
Output locale	The locale to be used to create the output file. This affects currencies, number- and date formats.
Target directory	Where to place the generated output files.
Output file name	The name of the output file. The file extension can be leafed out; it will be added based on the chosen output type.
Overwrite output	If true: an existing file will overwritten, otherwise the component fails if the file already exists.
Create directory if necessary	If true: the component checks if the target directory exists and creates all necessary directories to the final target directory. Otherwise the component fails if the target directory does not exist.

Sequential file names	If true the component adds to the file name (before the extension) a configurable time stamp.
Timestamp pattern	If option Sequential file names is true: setup here the timestamp pattern
Report Parameters	Here gets the reports it parameters. To setup the parameters you need to know the Java types of the parameters in the jrxml file. The same type is necessary here. If you have the value of the parameter only as String, the component can convert it into the necessary type. **Name:** Name of the report parameter (case sensitive – take it from the report design!) **Value:** Value of the parameter. If possible use an expression, which returns the same Java type of the parameter in the report (take it from the report design!). **Convert to type:** If the value can only be provided as String typed value, here you can setup a type conversion. Choose here the type of the original parameter in the report. E.g. A report expects a date typed value, you could set this value as string and the component converts it into a Date object using the **Date parameter pattern** component parameter. **… and Collection:** works only if a type conversion tales place. If the report parameter is a collection typed parameter, this option converts the value into a collection. Use the value separator to separate the String typed value in the single values of the collection.
Delimiter for parameter value collection	If the String typed parameter value has to be separated into multiple values for a collection, this char will be used as delimiter.
Date format for parameter values	If the String typed parameter value has to be converted into a Date type value, this pattern will be used.

Advanced parameters in case of output type is PDF

Property	Content
Compressed	The PDF will be compressed
Create bookmarks	For a better navigation through the generated PDF file, bookmarks can help
Encrypt document	The generated PDF file can be password protected
128-bit-key	This is causes an 128-bit encryption (otherwise a 64-bit encryption will be used)
Password	The password as simple String

Advanced parameters in case of output type is XLS or XLSX

Property	Content
Detect cell type	The data type of the cell will be set according to the written value
Use white page background	The background if the cell will be set to white, otherwise the background will be determined by the document default (typically none opaque)
Remove empty space between rows	A space (also if it only a very small space) causes cells in Excel. If you want to have cells only for the widgets of the report, you can check this option and only the designed cells will appear.
Remove empty spaces between column	Same issue like the option above but for columns. It is strongly recommended to check both options (it is the default).
One sheet per page	Every page of the report will be rendered into its own sheet. The sheet names will be set automatically of taken from the report.
Template file	Instead of creating a complete new document, you can read in a preconfigured document as template.
Keep template sheets	The sheets from the template document can be remaining in the target document. If false at start all sheets from the template document will be deleted. This is on most cases not useful. The only reason the delete all sheets could be the workbook contains e.g. custom colour palettes and the sheets are needed.

Return values of the component:

Value	Content
OUTPUT_FILE	The generated output file which is written to the local file system
ERROR_MESSAGE	Error message if something went wrong
NUMBER_OF_PAGES	Number of generated pages (or sheets in case of Excel). If a no-data page is created this page does not count.
REPORT_QUERY	In case of using a database source the used query can be obtained here. Please note that this is the query of the main report.

Report deployment

Sub reports and compiling

The component expects the jrxml file from the main report. Often reports contain sub reports and these sub-reports can also contain further sub-reports.
The component does not need the compiled reports (jasper files). The jasper files will be created if they do not already exist or if the jrxml file is newer than the corresponding jasper file.

Resource bundles

If a report uses resource bundles typically these files (.properties files) will be loaded via the class loader.
Because of this behaviour the component adds the directory of the main report and of the sub reports also to the job class-path. The best way is to place the resource bundles in the same directory as the report.

Images

At the end the path to the image must be an absolute path. The necessary flexibility for this path can be achieved by using parameters for the base path to the images.
Example for a path expression in a report:

$P{imageBasePath}+"/products/product_" + $F{product_id} + ".jpg"

This parameter (or more than one if needed) can be set via the parameter interface of the component and should be read from Talend context variables.

Fonts

JasperStudio packages fonts as jar file. These jar file can be added to the job with the tLibraryLoad component.
I suggest using the Dynamic Libs feature in the advanced settings of tLibraryLoad.
Please take care the libraries are loaded before the component tJasperReportExec starts.

Additional renderers

Because of the limitation of the maximum component size (currently 30MByte) it does not contain the necessary libraries to render bar codes or SVN images (only to name a few). Nevertheless it is possible to use these features in your report.
You have to add the necessary libraries to the job with tLibraryLoad (as already explained in the Font section).

Libraries for barcode

JasperStudio supports two different bar code implementations: Barcode4j and Barbecue

Barcode4j widgets

Homepage of the Barcode4j project: http://barcode4j.sourceforge.net/
Libraries:
barcode4j-2.1.jar
avalon-framework-4.2.0.jar

Barbecue widgets

Homepage for the Barbecue project: http://sourceforge.net/projects/barbecue/
Libraries:
barbecue-1.5-beta1.jar
jdom.jar

SVN

Libraries for SVN

batik-anim.jar
batik-awt-util.jar
batik-bridge.jar
batik-css.jar
batik-dom.jar
batik-ext.jar
batik-gvt.jar
batik-parser.jar
batik-script.jar
batik-svg-dom.jar
batik-svggen.jar
batik-transcoder.jar
batik-util.jar
batik-xml.jar
xml-apis-ext.jar

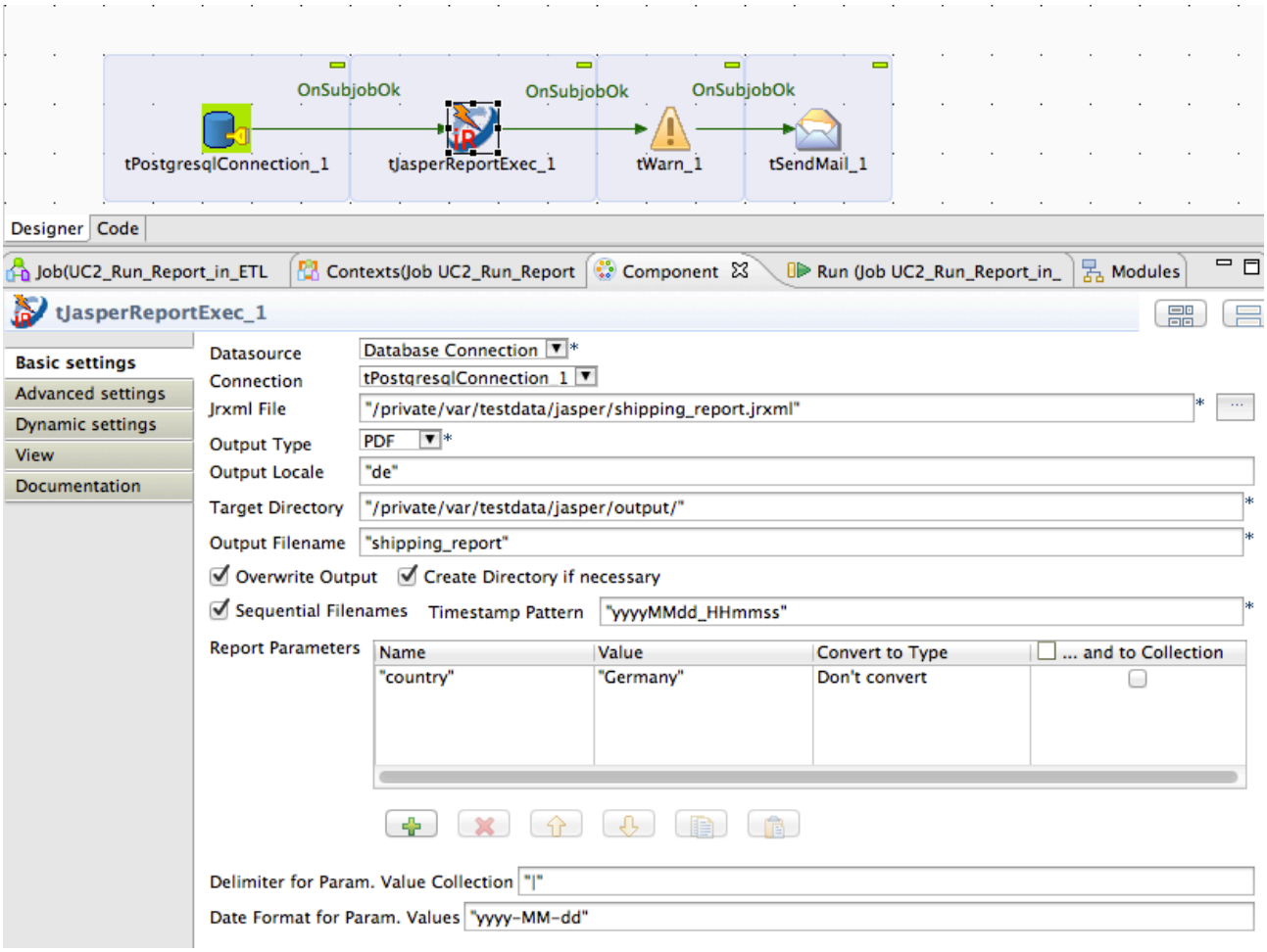
Where can you get these libraries:

All necessary libraries can be extracted from an archive in JasperStudio (e.g. here release 5.6.0).
Unpack plugins/net.sf.jasperreports_5.6.0.final.jar (perhaps copy it to a different location and rename it to .zip)
In this unpacked jar (or zip) file you will find a folder with the name lib and within are all mentioned libraries.

Compatibility settings in JasperStudio

If the component is not based on the latest JasperLibrary (e.g. the component supports an older release as the studio could provide) you have to set the supported release of the component in JasperStudio in:
Preferences -> Jasper Studio -> Version

Scenario 1: Build a PDF file within the ETL process



This scenario is one of the most typical scenarios. The component creates a report similar JasperStudio would do that. And the report is from the type PDF.

Example for the advanced settings for a particular PDF report. It is also possible to use here context variables (e.g. for the year/month in this example).