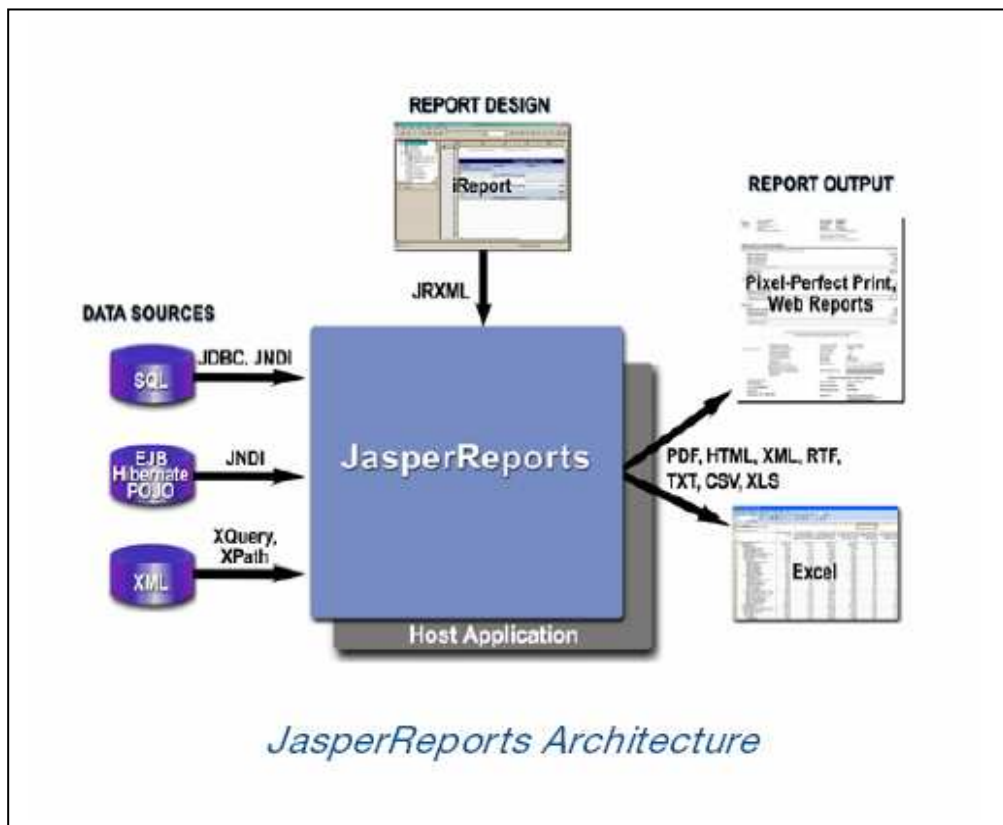

Report Editor

LYNX

Editing and Exporting Reports with JasperReports

The LYNX Reporting feature is based on the JasperReports report building engine.

JasperReports is the most powerful reporting tool currently available in Open Source. Based on Java technologies, it is used by many famous companies around the world. Its Open Source development ensures that it is and will be widely supported and enhanced.

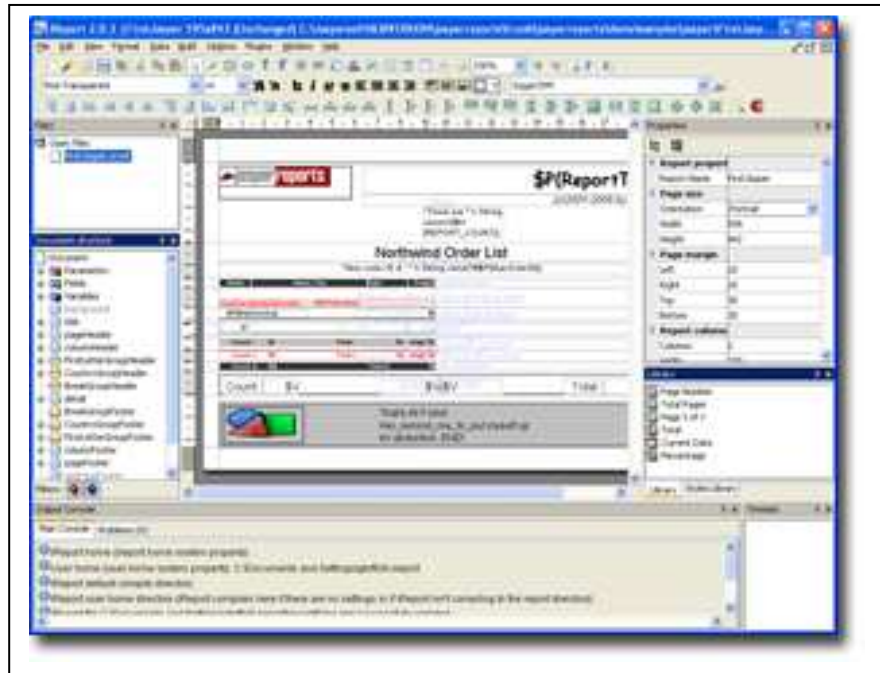


With a connection to the LYNX Web server, it is possible to select a report that has been prepared beforehand using the graphical tool (iReports, see below); rendering will then be done directly on a Web page, in a PDF file, to a desktop application, etc.

This option does not involve any setup on the client stations that will connect to the Web server to have the reports that are maintained by the SCADA LYNX manager.

These reporting options may be used by LYNX applications or by batches to generate outputs automatically (e.g. daily reports).

In the context of the SCADA LYNX, reports are built using a very powerful graphical application: iReport.



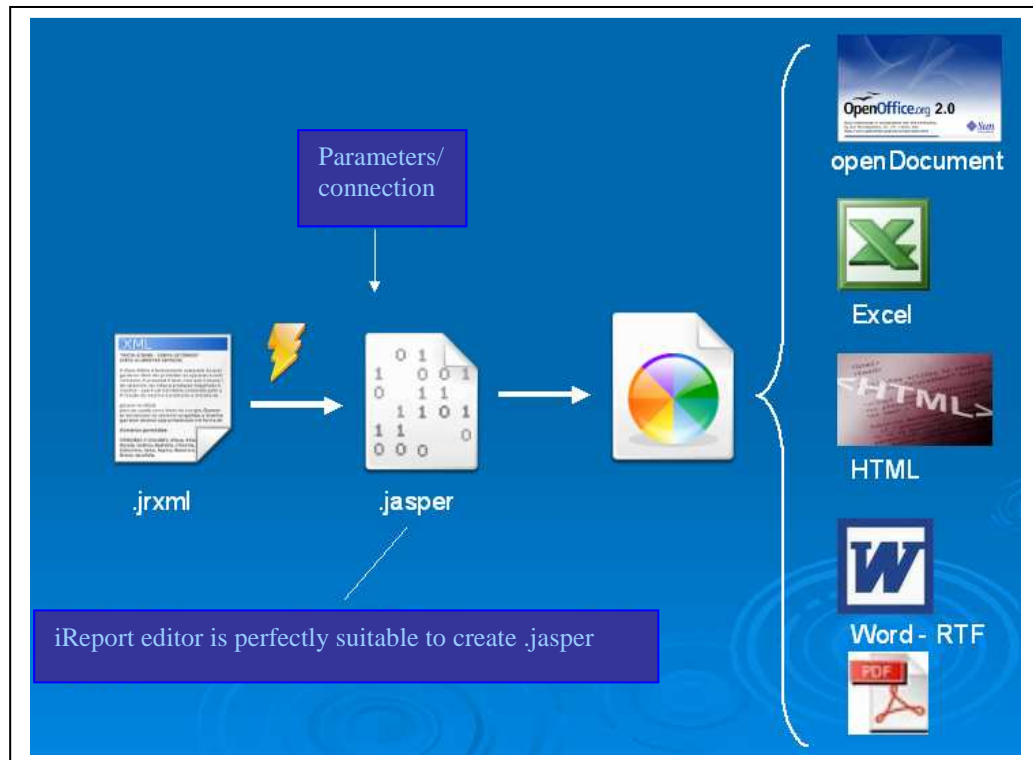
Reports can be designed on machines that are separate from the LYNX farm (a remote SQL connection allows to do most of the presentation).

Files that result from preparing report templates (XML format) are stored on the server. These files will then be available (mainly, but not only) from the Web interface. Reports that are prepared in iReport will be (re-)generated on demand with the current (or archived) data, in the format that is selected by the user.

Principle for Creating Reports

In a first stage, reports should be created in XML format, through the iReport graphical editor or using any text editor (XML format). These files usually have the .jrxml extension (JasperReport XML).

The JRXML file will then be "compiled" so that it can be viewed. The compile process can be done in iReport, in the code for an application or a batch, but dynamically on the Web server (the .jrxml file that is designed in the editor is actually transferred to the LYNX processing server). In the LYNX environment, the compile stage will not be visible to final report users.



From the same JRXML file, JasperReports can then generate several file types:

- PDF
- XLS (Excel)
- ODF (OpenOffice)
- RTF (Word)
- CSV
- Text
- Etc.

JasperReports will accept several data source types. In the context of LYNX, data from the PostresSQL database engine will mainly be used.

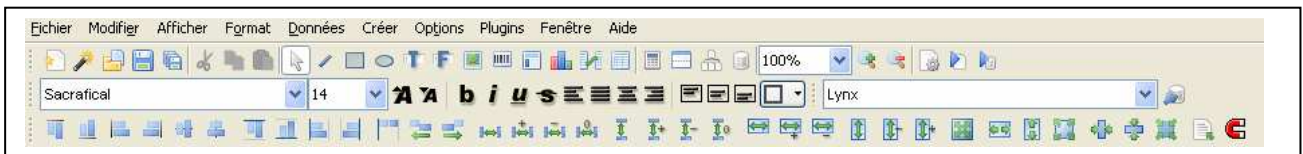
A report can have only one data source, and can make one SQL request only. A report can however consist of subreports, each of which has its own request. A subreport can be set by the main report.

A report can consist of multiple visual elements:

- Static text
- Dynamic text (from base data, reprocessed or not)
- Graphics (histograms, pie charts, etc.)
- More or less complex geometric shapes
- Images
- Hyperlinks
- Etc.

Text fields can be formatted according to the needs and the data type, just like a visual tool. Static/dynamic data concatenation is also possible.

The different presentation items are available from the main toolbar:



These tools are described in the Online Help for iReport. Many Internet sites offer example documents that are created with these tools.

A Practical Example

Suppose a list of companies should be displayed. The SQL request is in the following form: **"SELECT * FROM COMPANIES WHERE ... ORDER BY..."**

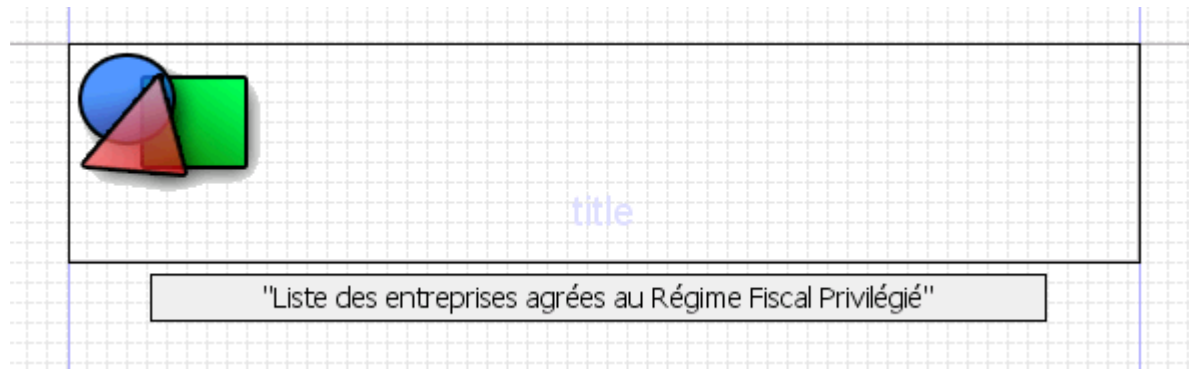
The report logo/title should be on the first page only, and each page will have a page number and footer.

The goal is to get a report that looks as follows:

RID	Raison	SIGLE	N° arrêté	Date
864597	ENTREPRISE 1	ENT 1	AGREMENT 1	
95242	ENTREPRISE 10	ENT 10	AGREMENT 10	
80329	ENTREPRISE 11	ENT 11	AGREMENT 11	21/07/2007
871141	ENTREPRISE 12	ENT 12	AGREMENT 12	
173247	ENTREPRISE 13	ENT 13	AGREMENT 13	08/12/2002
29423	ENTREPRISE 14	ENT 14	AGREMENT 14	01/01/2004
10	ENTREPRISE 15	ENT 15	AGREMENT 15	02/09/2007
79056	ENTREPRISE 2	ENT 2	AGREMENT 2	
89106	ENTREPRISE 3	ENT 3	AGREMENT 3	01/01/2003
431124	ENTREPRISE 4	ENT 4	AGREMENT 4	08/02/2007
819819	ENTREPRISE 5	ENT 5	AGREMENT 5	
88940	ENTREPRISE 6	ENT 6	AGREMENT 6	
84733	ENTREPRISE 7	ENT 7	AGREMENT 7	01/01/2003
76705	ENTREPRISE 8	ENT 8	AGREMENT 8	
118861	ENTREPRISE 9	ENT 9	AGREMENT 9	

Page 1/ 1
Pied de page
Bla bla bla bla

Title:



The big box that contains a drawing is the logo. The title is static in a greyed box.

Column headers

RID	Raison	SIGLE	N° arrêté	Date
-----	--------	-------	-----------	------

Static column headers (this is often the case), greyed through properties.

Detail

\$F	\$F{DENOMINATION}	\$F{SIGLE}	\$F	\$F
-----	-------------------	------------	-----	-----

Dynamic content: Fields returned by the SQL request are used (note the syntax: `$F{FIELDNAME}`).

These elements will be repeated according to the number of records that are returned by the SQL request.

Footer

"Page " + String.valueOf(\$V) + String.
"Pied de page blah blah blah"

Here the page number is inserted using variables (see below), together with the total number of pages and static text at the end.

Subreports

There can be only one data source and one SQL request per report; this limitation can be bypassed by inserting subreports.

Subreports are literally reports within reports. To create a subreport, a regular report is created, then called in another report that will become the parent report.

A report can contain as many subreports as required.

The required information can be transferred to the child report from the parent report. The parameters of the child report will then be used.

Example:

If the operating results are needed for a company of which the ID is known, a subreport will be created with a request in the following form:

```
"select * from results WHERE company_id=${MYID}"
```

The parameter `${MYID}` should then be passed from the parent report to the child report with the requested value.