Reporting Manual

ADITO Academy

Version 2.9 | 15.09.2023

Version	Changes
2.9	New opening methode for the Inspector and the report
2.8	Flag for execution of the reporting field
2.7	Bold texts in reports
2.6	Adding note for new parameter in neon.openReport()
2.5	Implementation of the logo and address, adding SQLBuilder to sample code
2.4	Seperate optional chapters, changing file name, adding dynamic pictures parameter
2.3	New ViewTemplate description and Subreports in Subreports
2.2	Revised version for ADITO 2020
2.1	Reviewed version
2.0	New, reviewed version
1.0	First version

Index

1. Introduction

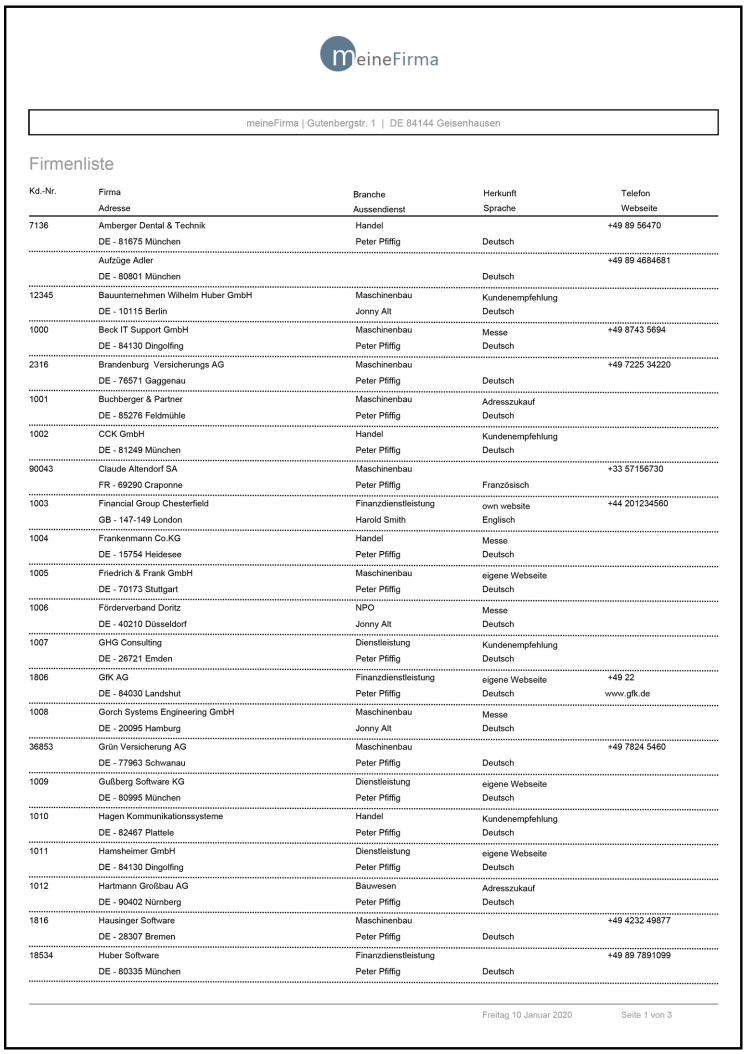
1.1. Purpose of this document

This document explains the basics of Jasper reporting in conjunction with ADITO. More in-depth information about JasperReports can be found on the official Jaspersoft website.

1.2. What is a report?

Reports are a systematical accumulation of data. They can cover a wide range of topics at once, but usually focus on transmitting specific information with a clear purpose, to a specific audience. Good reports are documents that are accurate, objective and complete. They are often used for analysis purposes or for structured overviews.

An good example for a report is a structured list of all companies:



meineFirma

meineFirma | Gutenbergstr. 1 | DE 84144 Geisenhausen

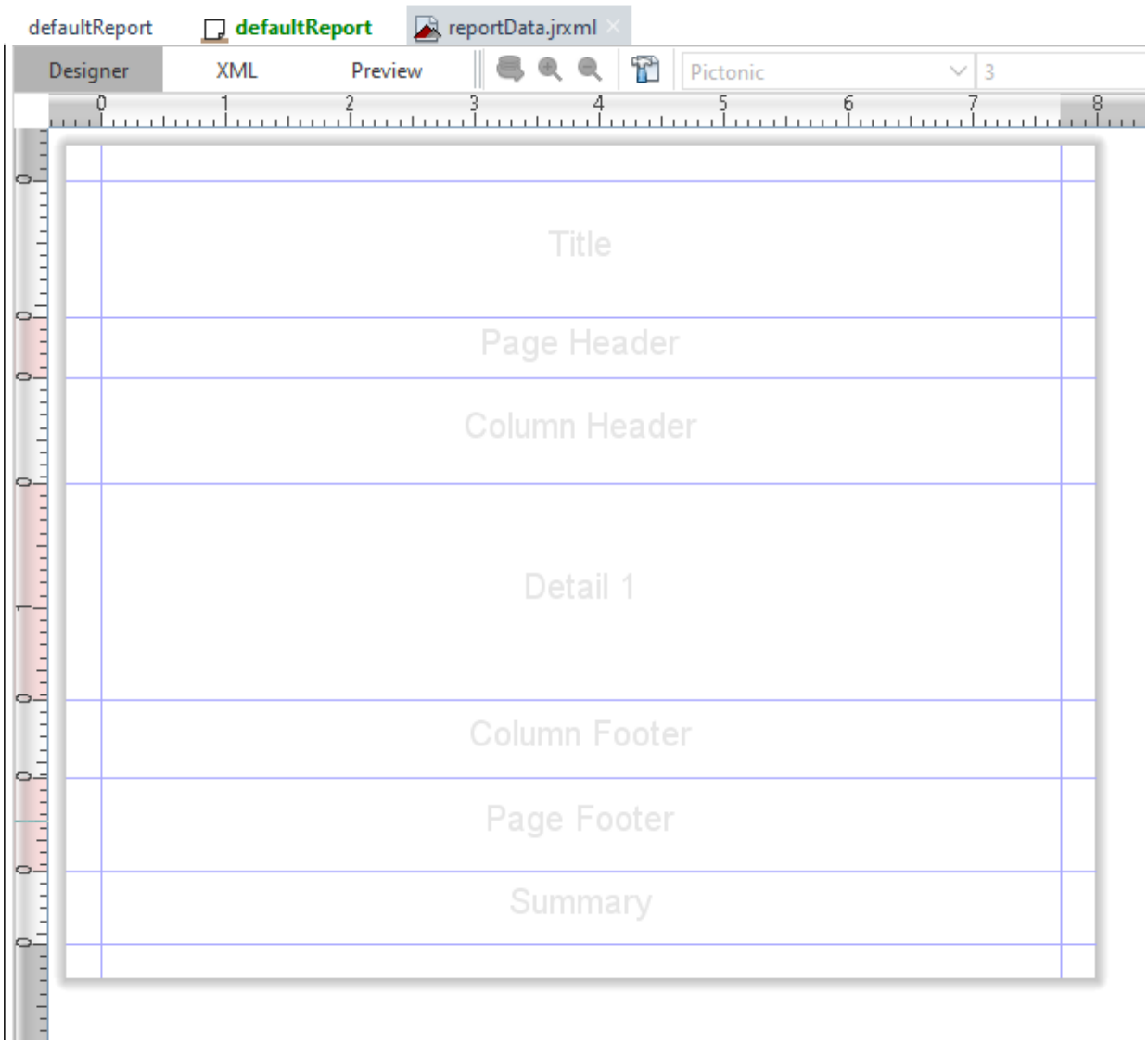
Firmenliste

Kd.-Nr.	Firma Adresse	Branche Aussendienst	Herkunft Sprache	Telefon Webseite
7136	Amberger Dental & Technik DE - 81675 München	Handel Peter Pfiffig	 Deutsch	+49 89 56470
	Aufzüge Adler DE - 80801 München		 Deutsch	+49 89 4684681
12345	Bauunternehmen Wilhelm Huber GmbH DE - 10115 Berlin	Maschinenbau Jonny Alt	Kundenempfehlung Deutsch	
1000	Beck IT Support GmbH DE - 84130 Dingolfing	Maschinenbau Peter Pfiffig	Messe Deutsch	+49 8743 5694
2316	Brandenburg Versicherungs AG DE - 76571 Gaggenau	Maschinenbau Peter Pfiffig	 Deutsch	+49 7225 34220
1001	Buchberger & Partner DE - 85276 Feldmühle	Maschinenbau Peter Pfiffig	Adresszukauf Deutsch	
1002	CCK GmbH DE - 81249 München	Handel Peter Pfiffig	Kundenempfehlung Deutsch	
90043	Claude Altendorf SA FR - 69290 Craponne	Maschinenbau Peter Pfiffig	 Französisch	+33 57156730
1003	Financial Group Chesterfield GB - 147-149 London	Finanzdienstleistung Harold Smith	own website Englisch	+44 201234560
1004	Frankenmann Co.KG DE - 15754 Heidesee	Handel Peter Pfiffig	Messe Deutsch	
1005	Friedrich & Frank GmbH DE - 70173 Stuttgart	Maschinenbau Peter Pfiffig	eigene Webseite Deutsch	
1006	Förderverband Doritz DE - 40210 Düsseldorf	NPO Jonny Alt	Messe Deutsch	
1007	GHG Consulting DE - 26721 Emden	Dienstleistung Peter Pfiffig	Kundenempfehlung Deutsch	
1806	GfK AG DE - 84030 Landshut	Finanzdienstleistung Peter Pfiffig	eigene Webseite Deutsch	+49 22 www.gfk.de
1008	Gorch Systems Engineering GmbH DE - 20095 Hamburg	Maschinenbau Jonny Alt	Messe Deutsch	
36853	Grün Versicherung AG DE - 77963 Schwanau	Maschinenbau Peter Pfiffig	 Deutsch	+49 7824 5460
1009	Gußberg Software KG DE - 80995 München	Dienstleistung Peter Pfiffig	eigene Webseite Deutsch	
1010	Hagen Kommunikationssysteme DE - 82467 Plattele	Handel Peter Pfiffig	Kundenempfehlung Deutsch	
1011	Hamsheimer GmbH DE - 84130 Dingolfing	Dienstleistung Peter Pfiffig	eigene Webseite Deutsch	
1012	Hartmann Großbau AG DE - 90402 Nürnberg	Bauwesen Peter Pfiffig	Adresszukauf Deutsch	
1816	Hausinger Software DE - 28307 Bremen	Maschinenbau Peter Pfiffig	 Deutsch	+49 4232 49877
18534	Huber Software DE - 80335 München	Finanzdienstleistung Peter Pfiffig	 Deutsch	+49 89 7891099

Freitag 10 Januar 2020 Seite 1 von 3

2. Design Tab

Once you open a report, the first thing you'll notice is the Design area in the center of your development environment (here: the ADITO Designer). This is called the Design tab, which is used to design a report.



3. Bands

The Design tab is divided into different horizontal portions, named bands, where you can place report elements (explained in next chapter Elements). When the report design is combined with the data to generate the print, each band is printed multiple times based on its function. For instance, the Page Header band is repeated at the beginning of every page, while the Detail band is repeated for each record.

The width of a band is as big as the page width (right and left margins excluded).
The height, however, should always be considered “the minimal height” of the band. That’s due to the characteristic that most bands adapt to the height of the elements, that are placed within them. Exceptions are the Column Footer, Page Footer and the Last Page Footer band. Their size is fixed.

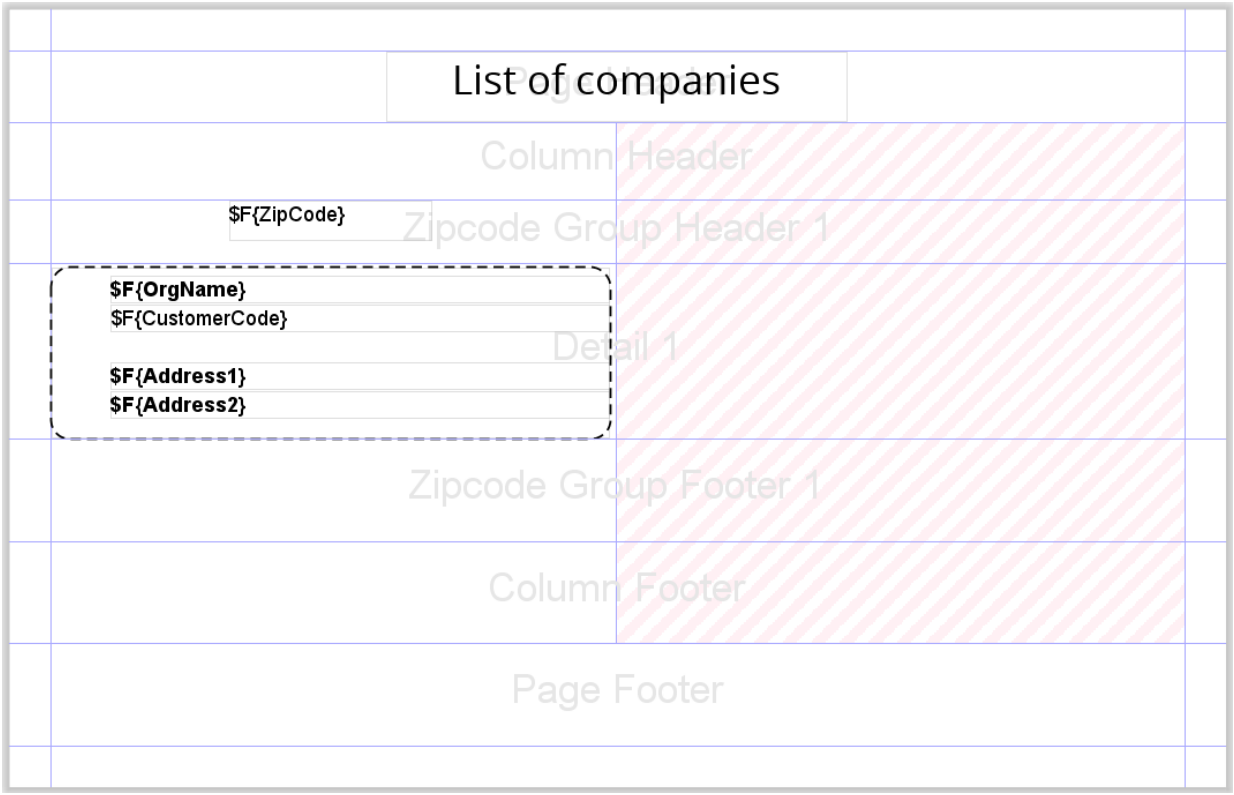
The sum of all band heights (except for the Background band) always needs to be less than the page height (minus the top and bottom margins) or equal to it.

Band Type	Description
Title	The Title band is the first band and usually contains the title of the report. It is shown only once (at the very beginning of the report). It can optionally be printed on a separate page. In case the content of the band exceeds the report page height, it will just cut the band off.
Page Header	The Page Header band appears at the top of all printed pages as a header, except for the Title and Summary bands when printed on a separate page.
Column Header	This band is printed at the beginning of the first Detail band. It usually holds labels containing the column names of a tabular report, e. g., "Company name".
Group Header	This band is placed before the Detail band (and after the Column Header) and usually holds information referring to a group (explained in chapter Groups). This band isn’t shown from the beginning, it only appears if a group is added to the report. A report can contain zero or multiple Group Header bands. It is possible to force a Group Header on a new page or in a new column.

Band Type	Description
Detail	A report can contain zero or multiple Detail bands, in which fields (explained in chapter Fields) containing record information are usually placed. A Detail band shows the first value of each Field before repeating itself vertically until all Field values have been displayed.
Group Footer	This band is placed after the Detail band (before the Column Footer) and usually contains fields to view subtotals or graphic elements for separation, such as lines. It appears, once you added a group to your report. A report can contain zero or multiple Group Header bands. A Group Footer is usually accompanied by a Group Header, but both can separately be set visible or invisible.
Column Footer	The Column Footer band appears after the (last) Detail band. Its height is always exactly as defined (it won't resize even if it contains resizable elements such as subreports or text fields with a variable number of text lines).
Page Footer	The Page Footer band appears on the very bottom of every page. It can be used to show the page number. Like the Column Footer band, its height is always exactly as defined.
Last Page Footer	If you want the Footer band of the last page to be different from the other footers, you use the Last Page Footer band. Its height is also always exactly as defined.
Summary	The Summary band allows you to display data once at the end of the report, below the Footer bands. This can be used to summarize the report data or to show total calculations or average values, etc.
Background	The Background band stretches from the first to the last band, but is located underneath those. This enables you to create watermarks and other effects, such as a frame around each page.

4. Columns

A Column separates the Detail bands and the Detail related bands (Group and Column bands) into horizontal sections. The data you then insert in the first Column will be repeated in the other Columns. All configurations concerning Columns can be found in the report's Property window.



5. Elements

An element is a graphical object, such as a rectangle or text block. Everything in a report is created with elements. You can add them by dragging them from the Palette window (in the right upper part of your Designer) and dropping them on the Design tab in a certain band.

In the following the most used elements will be explained.

5.1. Text Elements

Two elements are specifically designed to display text in a report: `Static Text` and `Text Field`.

By default, both of them are styled transparently with no border and in black text color. The most used text properties can be modified using the text tool bar above the Designer Tab that is displayed when a text component is selected. More complex text properties can also be modified using the Property window (by default in the lower left corner of your ADITO Designer).

Static Text

The Static Text element is used for labels or, in general, to print text that is not meant to change, e.g., a simple label `Company name`.

Text Field

The Text Field element allows you to print text using an expression, not only static text. For example, you'd type `$F(ORGNAME)` (= Organisation name/Company name) in the Text Field and the element translates that expression with the values stored in the Field `ORGNAME`.
Also, in some cases, you should use a Text Field instead of a Static Text for labels, too. That's when you need your texts to be translated into different languages because not only German users use your ADITO software for example. That's simply not possible with a Static Text.

Text can be translated into the ADITO Client's language. Therefore you need to insert a Text Field and change the $F in the property `Text Field Expression` to $R.

5.2. Breaks

The Break element offers two different kinds of breaks: A page and a column break. They are used to force the report engine to start with the next page or column. A column break in a report with only one column has the same effect as a page break. The type of break is initially set in a selection window that appears when you drop the element on the Design tab, but it can easily be changed in the Property window afterwards.

In the Designer Tab, a small line symbolizes the Break element. The line is resizable in theory, but as it only symbolizes a break at a specific vertical position, there's no use to do that.

5.3. Graphic Elements

Graphic elements like lines and shapes are used to make reports more attractive and readable.

Line

You can add lines to your report to structure the content visually. In the Designer, the Line element is defined by a rectangle of which the line represents the diagonal. That means if you resize the line vertically, you'll have a diagonal line and not a straight one. You can customize the style of the Line element in its Property window.

Rectangle and Ellipse

The Rectangle and Ellipse element are mainly used for clarity purposes by acting as frames around other elements.

You can also create extraordinary frames around Images and Text Fields elements via the "Padding and Borders" option, which can be opened by right-clicking on the element.

Image

An image is always filled with the value of a Parameter which name starts with `adito.image.`, otherwise ADITO doesn't recognize the passed data as an image.
The Parameter needs to be filled with a base64-String of a picture.

You can access the logo of your system by using the method "project.getPreferenceValue("custom.myCompany.logo");" as well as the address "project.getPreferenceValue("custom.myCompany.addr");".
This replaces the old JDito method `getMyASYS_ICONSdata()`. You can change your logo and address in the preferences of your local project.
But the formatting for the logo is important. The prefix of the data has to be changed. To include a new logo in the preferences you have to decode your picture to a base64-string and change the prefix from "data:image/png;base64:" to "base64:". In order to use your image again you have to undo this step in your report code.

5.4. Subreports

The Subreport element lets you nest one report (the subreport) inside another (the master report).

Subreporting is one of the most advanced features of JasperReports. They allow you to design very complex reports with a lot of information.

The subreport receives its data via a Parameter from the master report. The Parameter's property `Class` needs to be set to `Object` and its name always starts with `adito.datasource.` and may not include special characters.

When do I use subreports?

- For modularizing reports – you can create a subreport with your preferred data Fields and layout, and then use the subreport in multiple master reports.
- Displaying multiple data sources with its relations correctly in a report (E.g., a company with all its employees)

6. Groups

Groups allow you to organize the records of a report into a more organized structure. A group is defined by an expression containing Fields (and Variables) that define the criteria for inclusion in that group. JasperReports evaluates this expression and puts all the elements that meet those criteria into the group. You can compare Groups with the SQL command GROUP BY.

They group data by certain criteria, most of the time by IDs like ORGID or PERSID. Adding a group adds a Group Header and Group Footer band around the Detail band.

It is possible to have nested groups, i.e., the first group groups the data by ORGID and then a second group groups the data by PERSID. But it is not possible to have separate groups on the same level, i.e., one group that groups data by the ORGID and then another group that groups data by PERSID. That's because a report can only have one set of Detail bands and the groups are always established around all Detail bands.

7. Value Storage

There are three different kinds of objects that can store values in a report: Fields, Parameters and Variables

7.1. Fields

A print is usually created starting with a data source that provides a set of records divided into several fields. This behavior is exactly like getting the results of an SQL query. You've got several database columns, and several data records. You can think of report fields as database columns (one for the name, one for the street, etc.).

Fields will contain a one dimensional array that is passed on by our JDito code later on (which contains the data records).

7.2. Parameter

Report parameters are used in many different ways inside a report. They can be used to provide additional data to the report (i.e. the value for a title or the name of the user that executed the report).

Whatever you decide to use the parameters for, they represent the best communication channel between the report engine and the execution environment (which is the process with your JDito code).

7.3. Variables

Variables are generally used to store partial results or to do complex calculations with the data extracted from a data source. E.g., you've got a Field containing a price and a Field containing a

discount. To calculate and store the final price, you can use a Variable.

You can also find predefined Variables in your report. They contain a page or column counter for example.

8. Getting started

In the following chapters we will create and configure our first own report.

8.1. Prerequisites

To understand the following chapters, knowledge about JDito, as well as basic JavaScript skills, are required. Special reporting terms will usually be explained briefly when used, but all of them can also be found in detail in the above chapters.

8.2. Setting Up Your Computer

To be able to begin with reporting, you open the ADITO Designer and your project. Then start the local server or connect to the service.

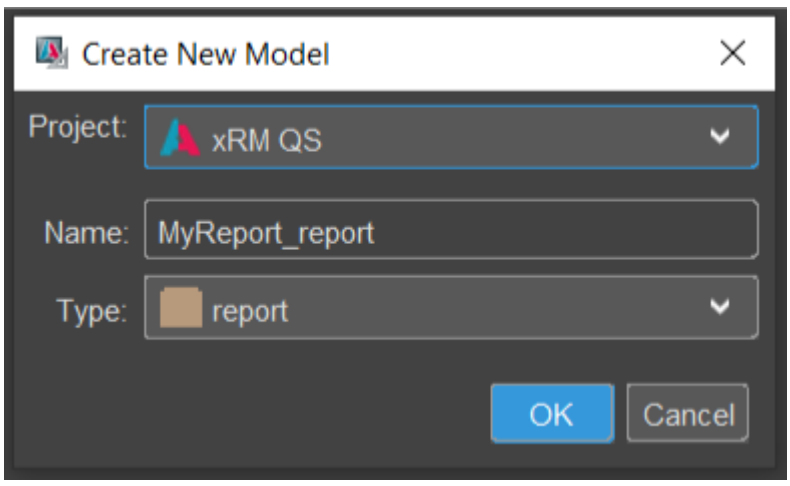
8.3. Creating a basic report

To be able to use a report, we need two things: The report itself, and a place that the report will be opened from.

We start with the report:

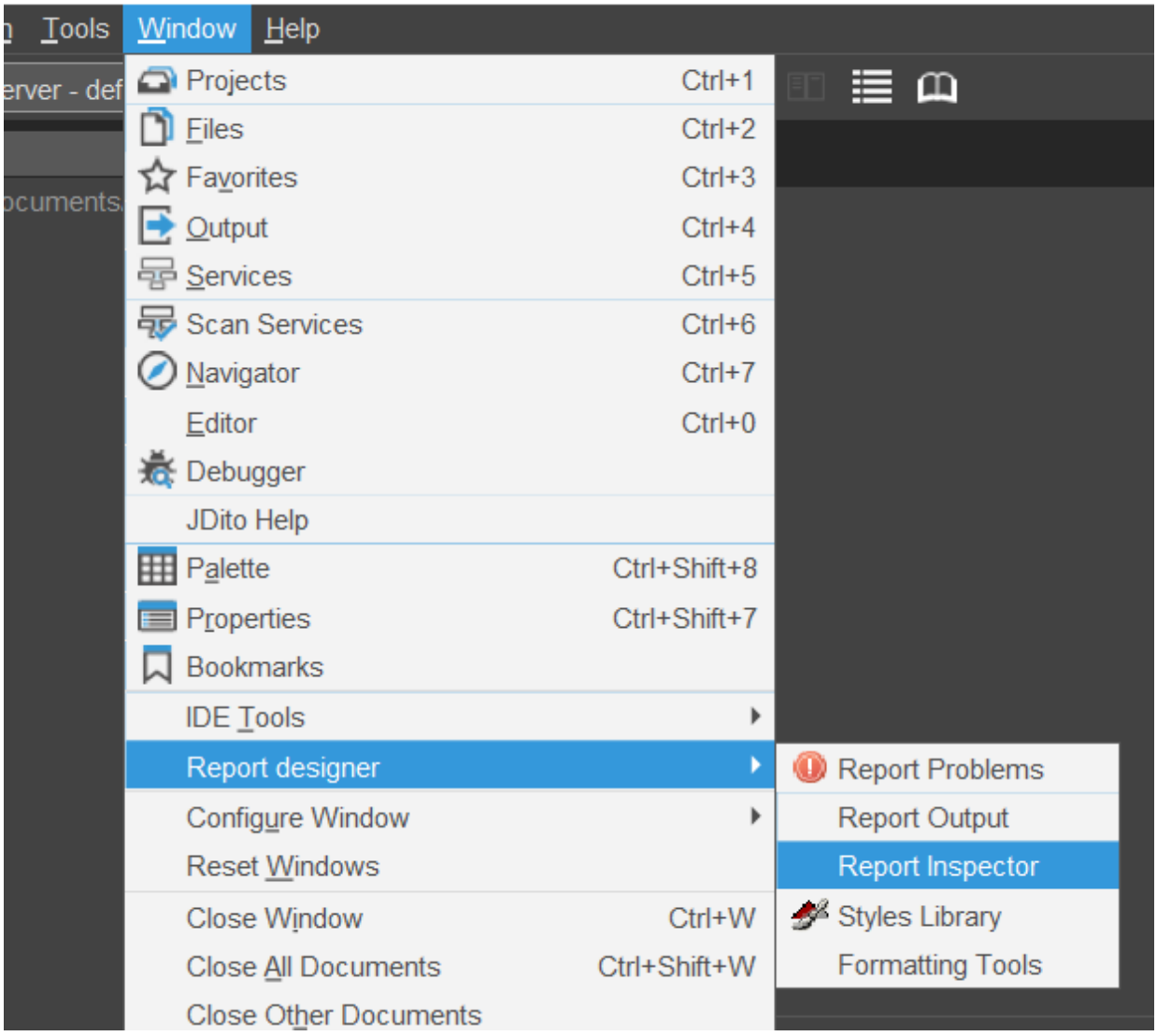
Press the New button on the upper left side of your Designer. A dialog will open in which you

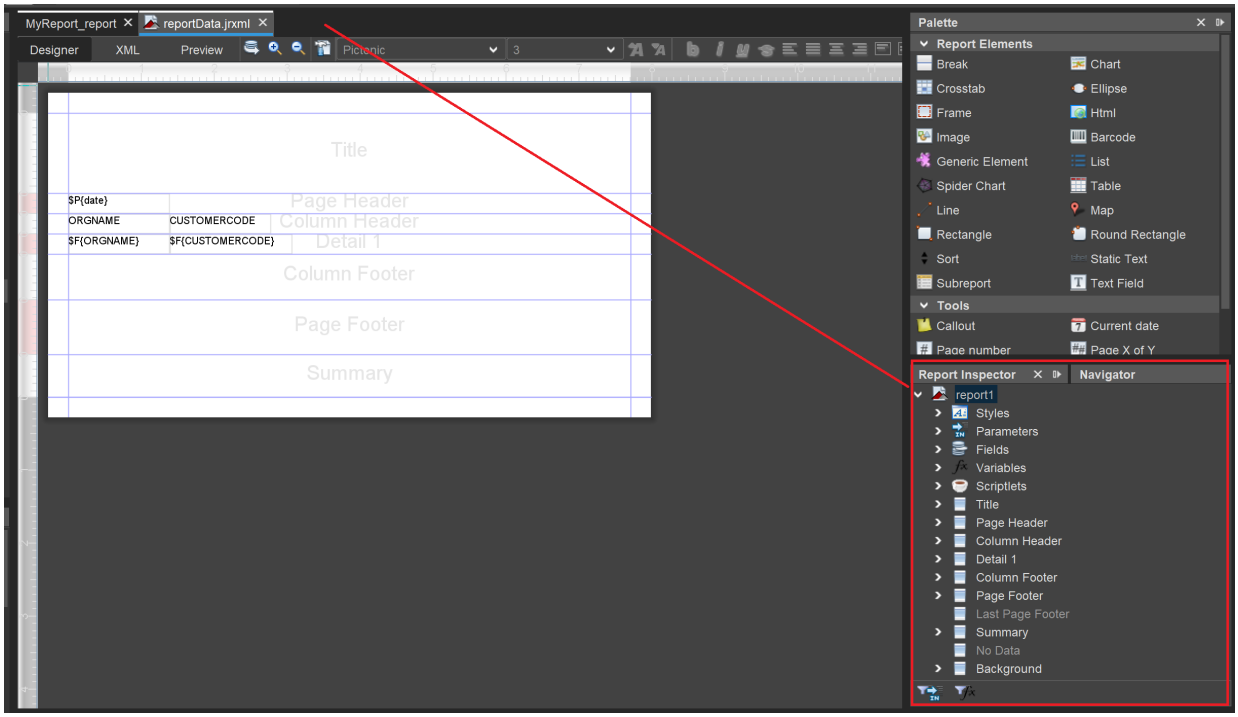
1. Choose the project in which the report should be created.
2. Type in the name of the report, ending with `_report` , e.g. MyReport_report (In versions older than ADITO 2019 the name of the reports start with RPTJ_).
3. Change the type to `report`.
4. Press OK. In versions older than ADITO 2019 another dialogue will open in which you can choose between the report type `Jasper` and `inet`. There you would choose `Jasper` and press OK.



i-net is an alternative way of creating reports which won't be supported at all in ADITO 5.0 and later versions. Therefore, this document will only focus on how to create a JasperReport.

Now, in the Designer Tab, you see an empty report with all of its bands. In addition to the Designer tab, the Report Inspector Window is essential to work with reports. You might have to add it to your Designer in the beginning ("Window" → "Report" → "Report Inspector"), for the last step just drag and drop the Inspector to the navigator place on the right side of the designer.





The next step is to create a place where the report will be opened from. As our report will be based on company data, the most logical thing to do is to create a button in the `Filter_View` of the "Organisation" entity.

For that we have to do a couple of things:

1. Create a new Action Group
2. Press the `Open` button in the upper left part of your Designer
3. Type "Organisation_entity" and press `OK`
4. Right-Click on `Actions` in your Navigator window and select `New Action Group`.
5. Enter the name "myActionGroup".
6. Press `Ok`.
7. Give the Action Group a title, e.g., "My action group"

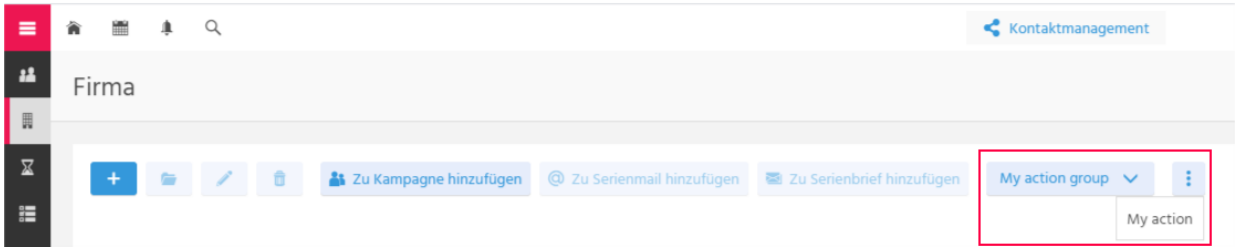
Create a new Action

1. Right-Click on the Action Group in your Navigator window and select `New Action`.
2. Enter the name "myAction"
3. Press `Ok`.
4. Give the Action a title, e.g., "My action".

Integrate the Action Group in the Filter_view

1. Open the "OrganisationFilter_view".
2. Select the tableViewTemplate so that its Property window is shown (in ADITO 2019 you then switch to tab "Others")
3. Go to the property `favoriteActionGroup3` and select the created Action Group.

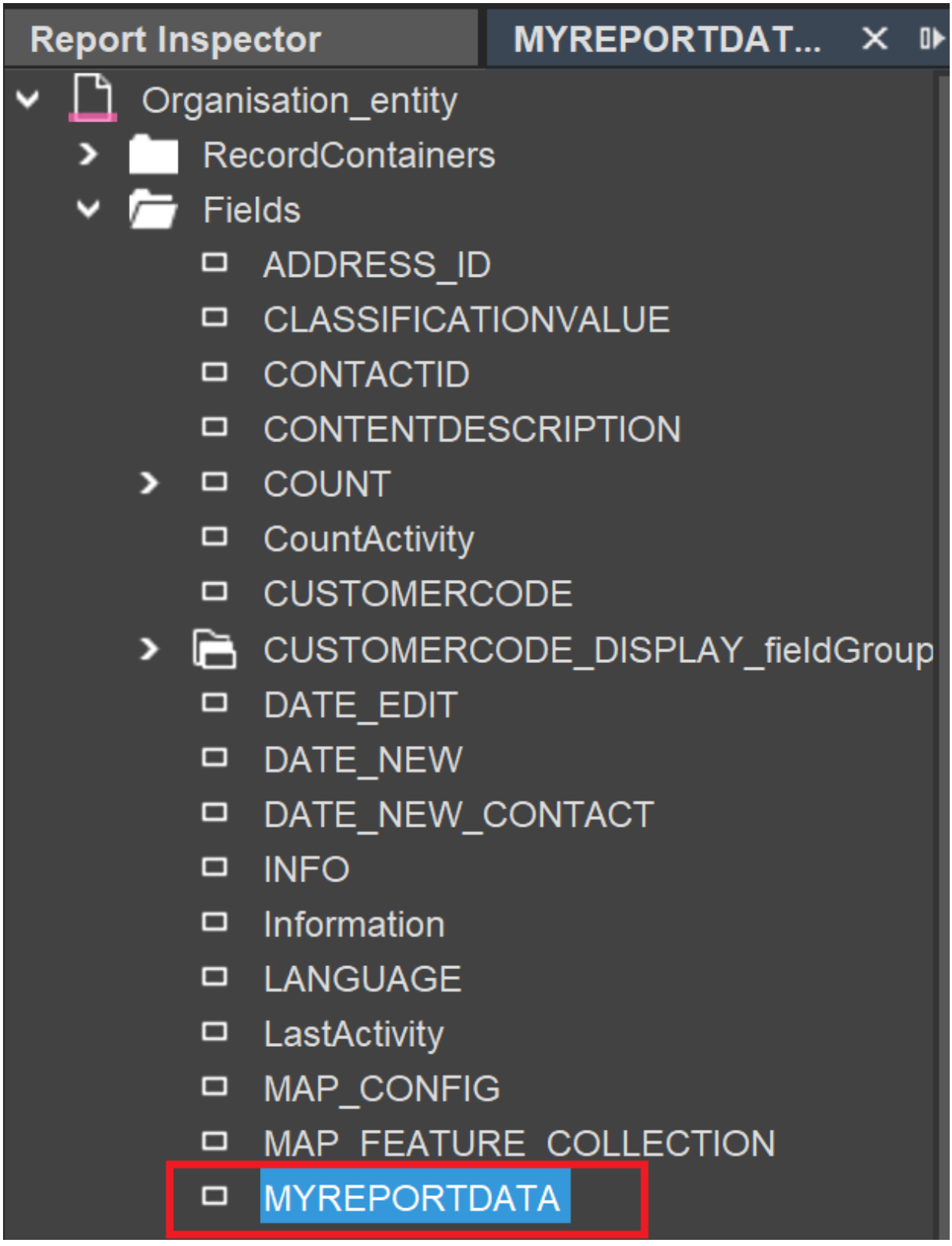
Now deploy your project. Open the Web Client and the Companies view. here you can see your Action Group and Action.



After that we have to create a field that will have our code that will call the data for our report.

For that we do the following steps:

1. Double-Click on the Entity on projects menu
2. Right-Click on the Fields in your Navigator window and select `New Field`.
3. Enter the name "MYREPORTDATA"
4. Press `Ok`.



At last we need a place to show or view or report, and for that we need to do the following:

1. Right-Click on the context in the Project menu on the left side of the designer

2. Chosse "New"
3. Then enter the name of your view, e.g.,"OrganisationMyRerpot_view".
4. Set the type to view and then press "ok".
5. After that open the view
6. Right-click in the naviagtor menu and then choose "Add view Tamplet".
7. Choose the Report Tamplet and give it a name
8. Then in the properties menu on the bottom left, set the reportData propertie to the field we created earlier("MYREPORTDATA").

Next, we'll design our report and fill it with data.

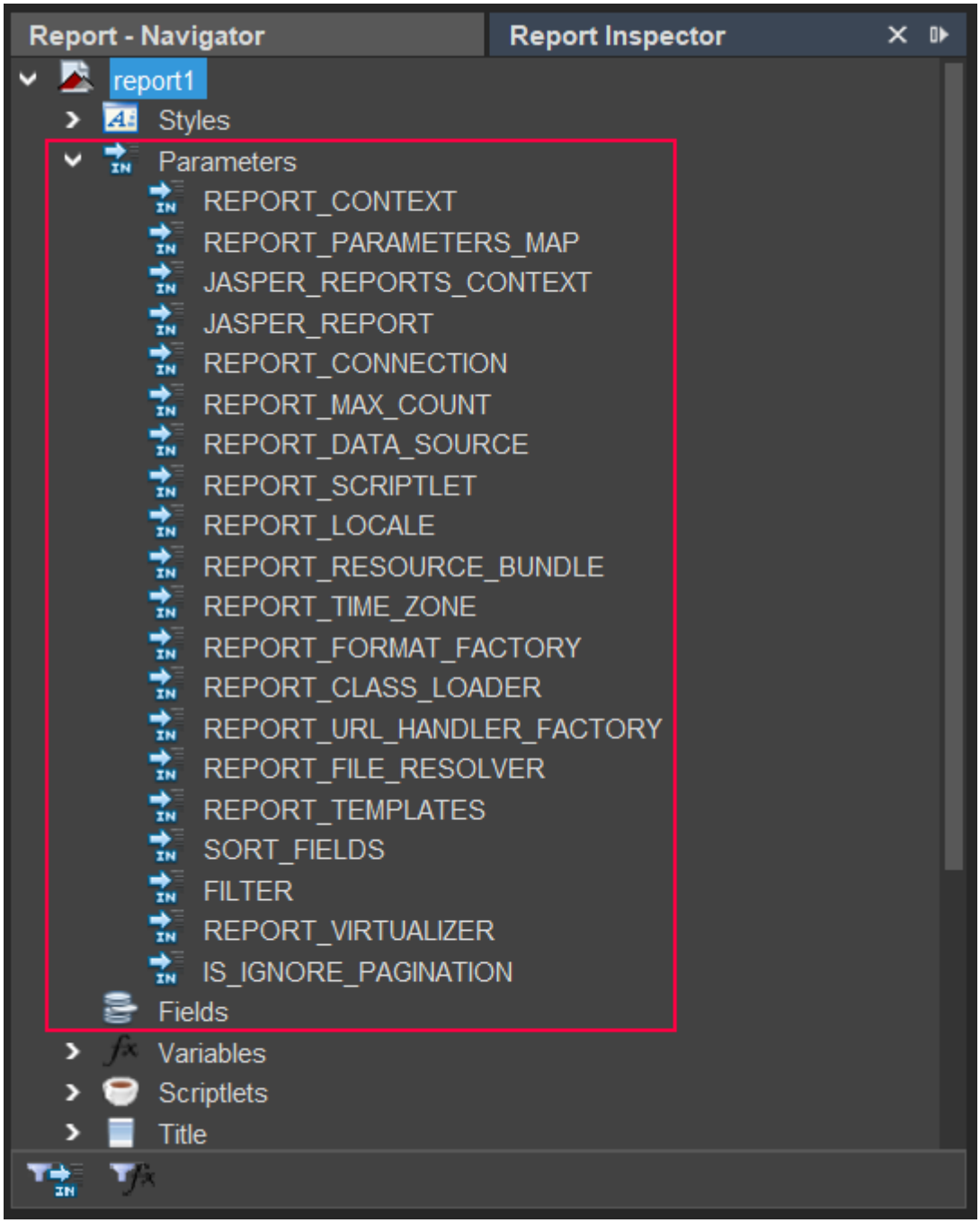
What we need:

- Parameters and Fields in which the data will be stored
- Elements that display the values of the Parameters and Fields in the report
- Code to fill the Parameters and Fields with data

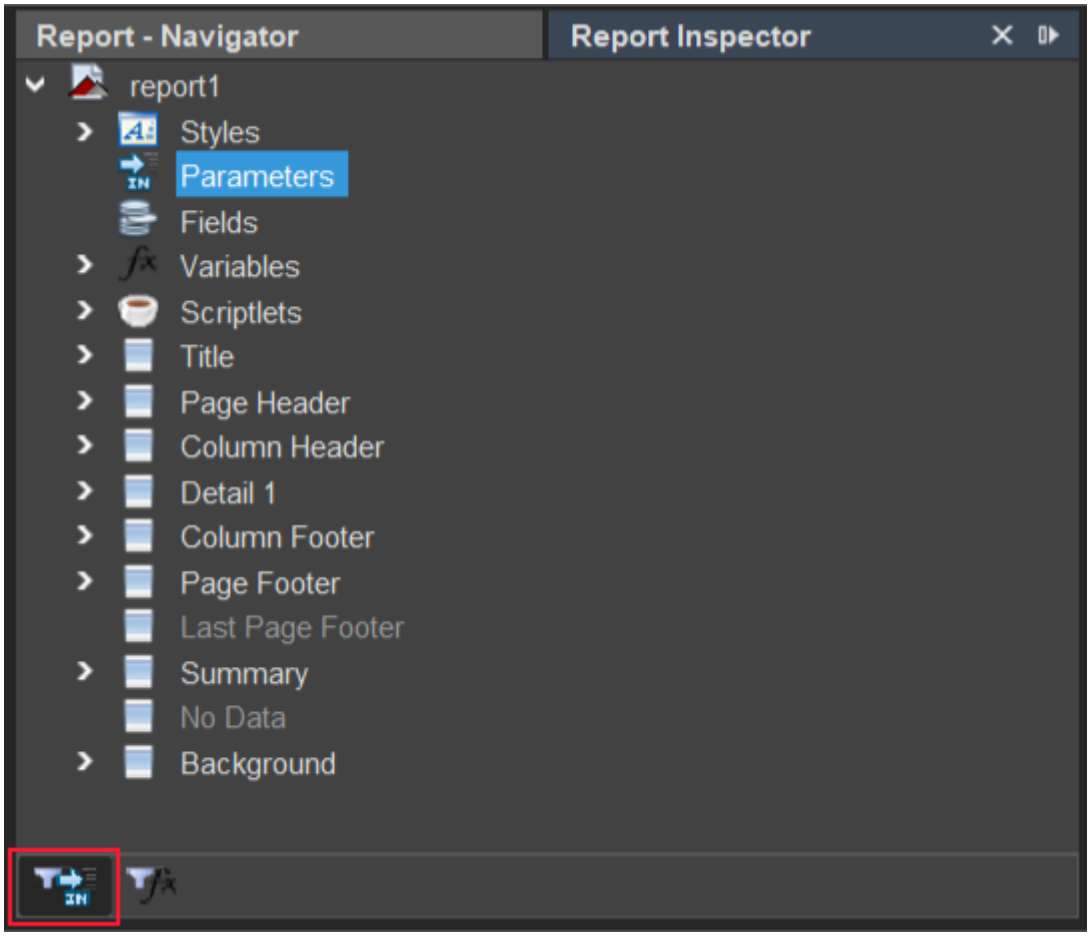
This three steps can generally be done in any order. Yet sticking to the same pattern every time simplifies things. The following order is a suggestion.

8.4. Creating Fields and Parameters

Let's switch back to the report. You can find Fields and Parameters in the Report Inspector window, which is on the lower right part of your Designer by default.



As you can see in the picture, a quite a few Parameters always already exist in the report, but these can be completely ignored. The reporting engine needs them to create the report, yet we won't need them at all. When working with Parameters, those already existing Parameters can be quite confusing. For that reason, there is a button to hide them, which is located on the bottom left part of the Report Inspector window.

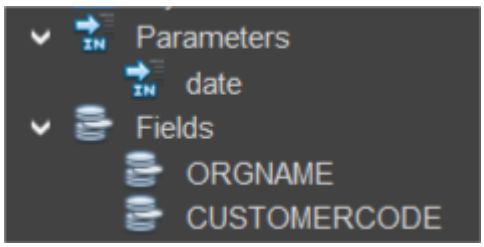


Now we will create our own Parameter, by right-clicking on `Parameters` and selecting `Add Parameter`. Then you can rename the Parameter to "date" (via right-click), as we will be filling it with the current date.

When we're done with that, we also add two new Fields exactly the same way: We right-click on `Fields`, choose `Add Field` and we rename them to "ORGNAME" and "CUSTOMERCODE". This works exactly like it does for the Parameters.

In our Coding Guidelines it is set, that the Fields need to be named exactly like their corresponding database column or by the pattern "TABLENAMECOLUMNNAME". For understanding purposes, we won't do that in the following example.

When you are done, it should look like this in your Report Inspector window:



8.5. Creating elements

We successfully created our Fields and Parameters, now their (later generated) value needs to be displayed inside our report.

There are two different ways of achieving this:

1. Add and configure a Text Field
    1. Look for the `Text Field` element inside the Palette window (in the upper right part of your Designer)
    2. Drag and drop the Text Field into any band you want (e.g., the "Page Header" band)
    3. Open the Text Field Expression in the Property window
    4. Refer to a Parameter by typing `$P{parametername}` or to a Field by typing `$F{fieldname}`
2. Drag and drop the Field or Parameter
    1. Choose the Parameter or Field you want to display
    2. Drag and drop it to the band in which you want to place it (e.g., the "Page Header" band)
    3. A Text Field with that Field/Parameter in its `Text Field Expression` will be generated automatically

The second option is the faster and easier way, yet knowing how the first option works or what exactly the second option generates, is good to know.

Which band to choose, is explained above in chapters Elements and Bands. But in short: Fields are usually displayed inside the Detail band, while Parameters are usually displayed anywhere but the Detail band.

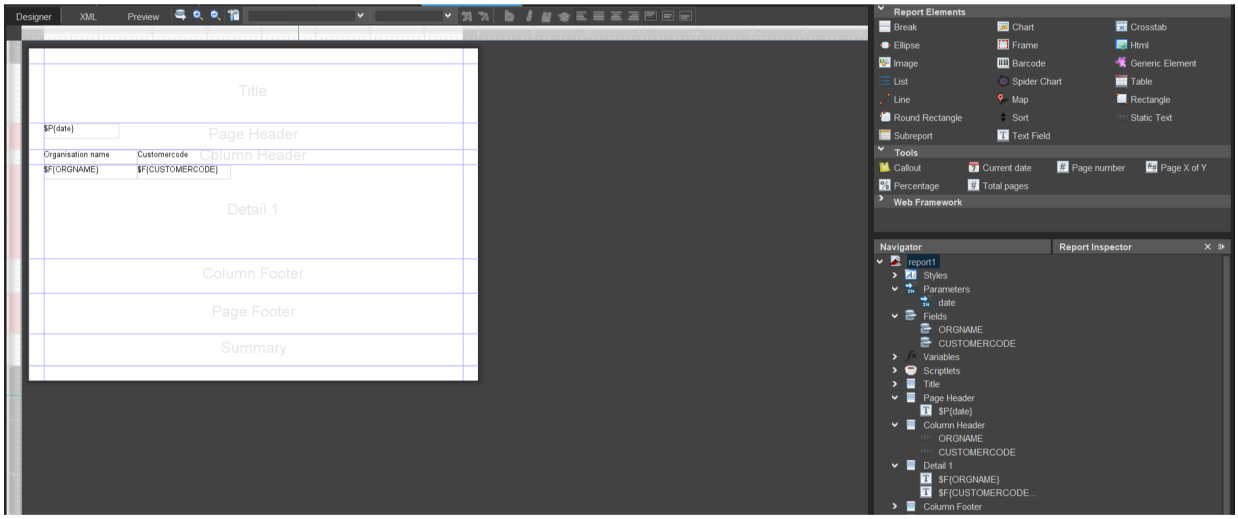
The Detail band results in the display of repetitive lines, one for each dataset passed in the value array corresponding to the Field. E.g., if our Field "ORGNAME" is inside the Detail band and it contains an array with a length of five (five company names), then the Detail band will be displayed five times.

As soon as a Field is placed inside the Detail band, a static label is generated inside the "Column Header" band. This is a feature that is supposed to spare you some work, but in most cases the Field names aren’t really user friendly terms and therefore have to be changed anyway.

Parameters only contain one single value and thus can be displayed anywhere. They can also be displayed inside the Detail band, even though it does not often make sense to display the exact same value over and over again.

Let’s place our Fields and Parameter onto the report, following this rules.

After that your report should roughly look like this:



8.6. Writing the required code

Our report is now ready to receive the right data. Now we need to fill the `valueProcess` process of the "MYREPORTDATA" field that we created in the Organisation_entity.

This is done in 3 steps:

1. The report objects

   We need to create an instance of the Class "Report" and "ReportData", because every method we need to call later on referres to an instance of those classes.

   When creation the `Report`-Object, one argument containing the report's name needs to be specified.
   When creation the `ReportData`-Object, one array-typed argument need to be specified. It contains the names of the Fields in the report.

```
//Report objects
var myReport = new Report("MyReport_report");

//                    0             1
var rptfields = ["ORGNAME", "CUSTOMERCODE"];
var myReportData = new ReportData(rptfields);
```

1. Data for the Parameters

   Create a new array called "params".

Fill this array referring to the Parameter names used in the report (case sensitive!) with the required data.

Call `.addReportParams` on the `Report`-Object, to map the array to the object.

```
//Parameter
var params = new Array();

var currentDate = datetime.toDate(datetime.date(), "dd.MM.yyyy");
params["date"] = currentDate;

myReport.addReportParams(params);
```

2. Data for the Fields

Here we already created the `ReportData`-Object with the one dimensional array that contains the names of our report Fields.

Now we need to pass on a two dimensional array to the `ReportData`-Object containing the data for the Fields. The report will then split up the array in multiple one dimensional arrays, one for each Field.

With the function `.table()` within the `SqlBuilder` (can be found in the library `Sql_lib`), a two dimensional array is being created.

We then need to map the data to the `ReportData`-Object. We do that with calling the method `.add`.

At last we need to map the `ReportData`-Object to the `Report`-Object. That's done by the method `.setReportData`.

```
//Fields
//                      0         1
var data = newSelect("NAME, CUSTOMERCODE")
               .from("ORGANISATION")
               .table();

myReportData.add(data);

myReport.setReportData(myReportData);
```

- The names in the array `rptfields` and the names of the `params` -array elements need to match the names of the report's Fields and Parameter precisely.
  For example, array entry ["ORGNAME"] and Report Field name "ORGNAME" or params["date"] and Parameter name "date" (case sensitive!).
- The order within the array `rptfield` and the array `data` needs to match.
  If, for example, "ORGNAME" is at position 0 in the array `rptfields`, the database column containing all organisation names needs to be at position 0 of the array `data`.

3. Method to open the report

   All that is missing now, is the function with which the report is opened.

```
myReport.openReport();
```

Finally, our code should roughly look like this:

```
import("system.db");
import("system.datetime");
import("Report_lib");

//Report objects
var myReport = new Report("MyReport_report");

//                    0            1
var rptfields = ["ORGNAME", "CUSTOMERCODE"];
var myReportData = new ReportData(rptfields);

//Parameter
var params = new Array();

var currentDate = datetime.toDate(datetime.date(), "dd.MM.yyyy");
params["date"] = currentDate;

myReport.addReportParams(params);

//Fields
//                         0        1
var data = newSelect("NAME, CUSTOMERCODE")
                .from("ORGANISATION")
                .table();

myReportData.add(data);
myReport.setReportData(myReportData);

result.string(myReport.exportReport()[1]);
```

There's a handy function in JDito's library `Sql_lib` called `concat`. With this function you can easily pair various database columns with only one report Field. For example:

```
var myReportData = new ReportData(["DATEOFBIRTH", "NAME"]);

var sqlutils = new SqlMaskingUtils();

var data = newSelect("DATEOFBIRTH, " + sqlutils.concat(["PERSON.SALUTATION","PERSON.TITLE",
"PERSON.FIRSTNAME", "PERSON.LASTNAME"]))
                .from("PERSON")
                .table();
```

With this solution you can be sure there's always the perfect amount of spacing between several Fields in the report as the function always inserts one space between them by default.

After we finished our "valueProcess", we now need to tell the Action to open our report, to do that we

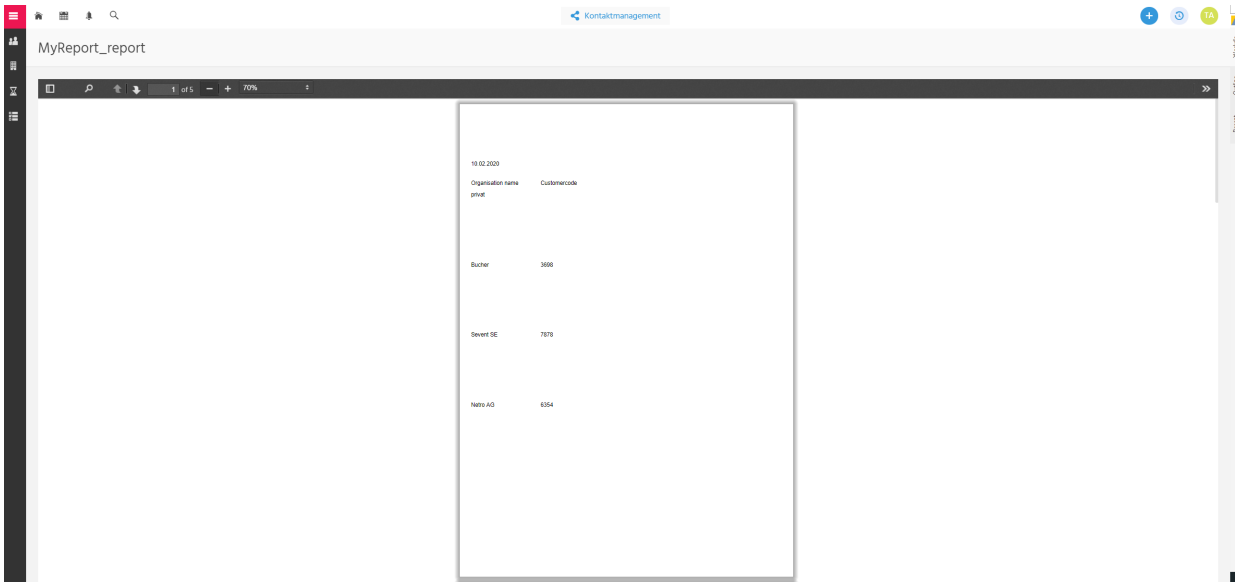
need to go to our Action again, and then insert the following code in the propertie "OnActionProcess" in the Propertie menu on the bottom left of our designer.

```
var Recipt = vars.get("$param.OrganisationReportData_param");

neon.openContextWithRecipe("Organisation","OrganisationMyRerpot_view",Recipt,neon.OPERATINGSTATE_VIEW,null,true);
```

8.7. Testing the report

Our report is now finished and can be deployed and tested. Once opened, the report should look like this in the client:



8.8. Beautifying the report

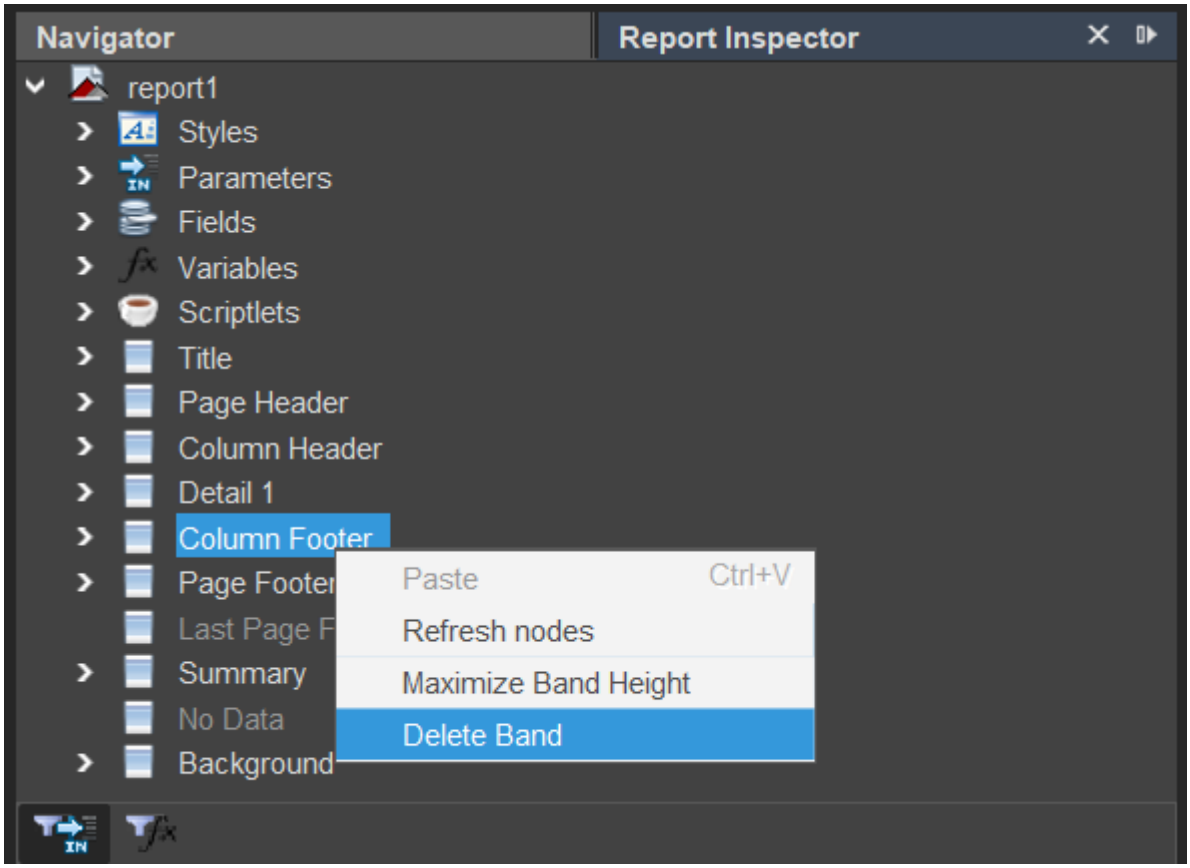
Now that the functionality of the report is given, we can work on its appearance.

8.8.1. Spacing

First of all, we want to get rid of the huge spacing between every dataset. Two things can be done about that:

1. We delete unused bands.
2. We clip the bottom of a band to the lowest component in that band.

Deleting bands is very easy, all you do is to open your report in the Designer, right-click the band in the Report Inspector or in the Designer window itself, and click `Delete Band`.



As you can see, the band is still there in the Report Inspector, it is just gray now. Whenever you need the band again, you can just right-click on the band in the Report Inspector and select Add Band.

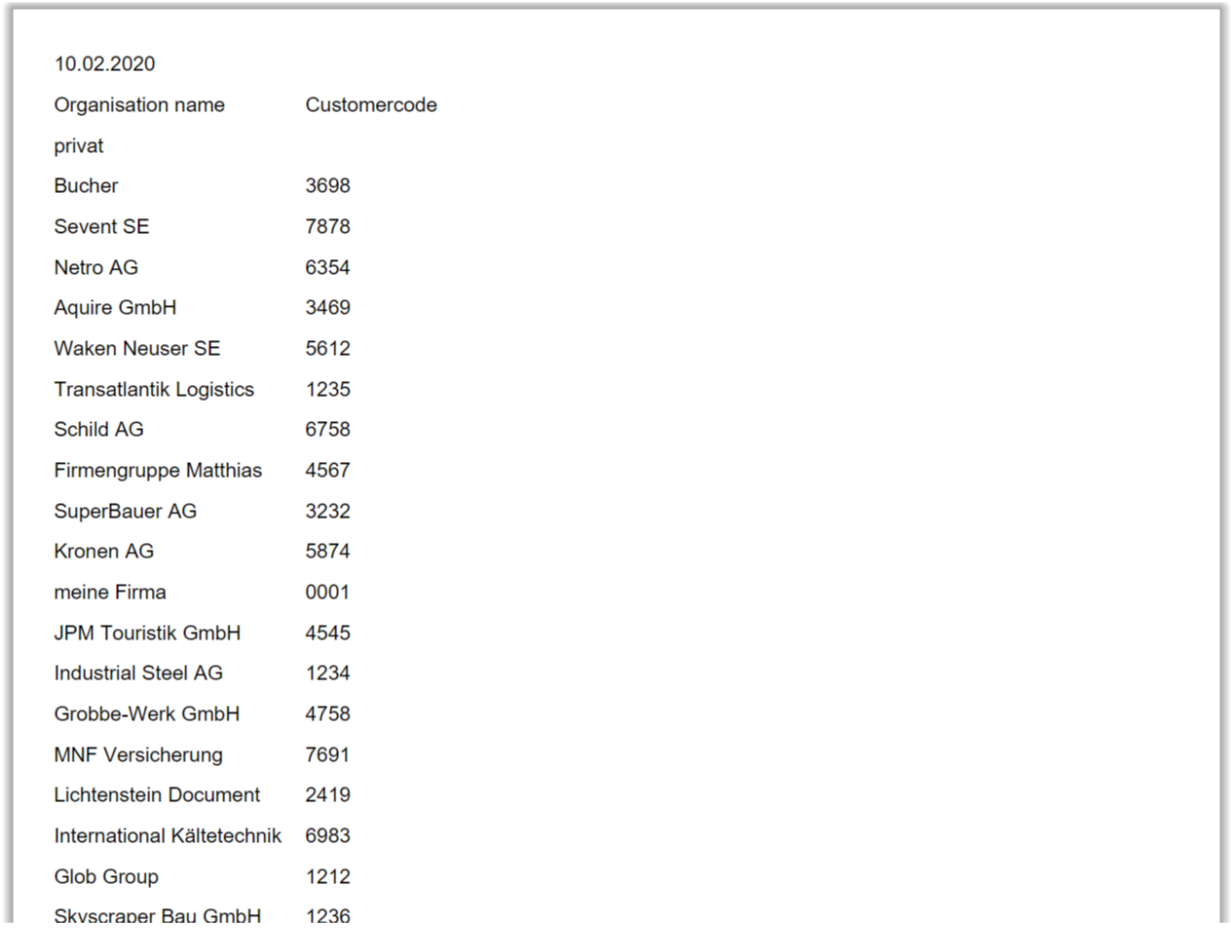
We'll delete every band except the Page Header, Column Header, and Detail band.

After we have deleted all bands we do not need, we need to adapt the size of the used bands. This can be done by hovering with your mouse over the bottom line of the band, holding the left mouse button, and dragging the line up or down, making the band smaller or bigger.

Our goal is to change the band size, so that the lower end of the band clips onto the lowest component in that band. For this, the easy way is to hold your mouse over the bottom border of the band and to double-click.

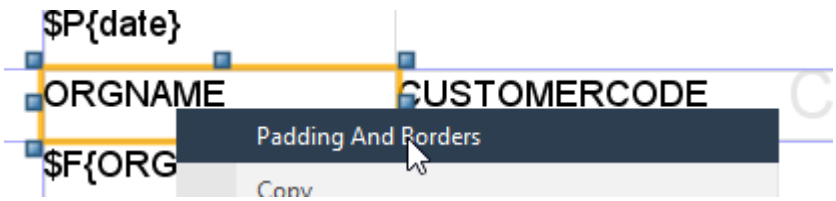
By double-clicking, the band's lower border will automatically clip itself to the lowest component in that band.

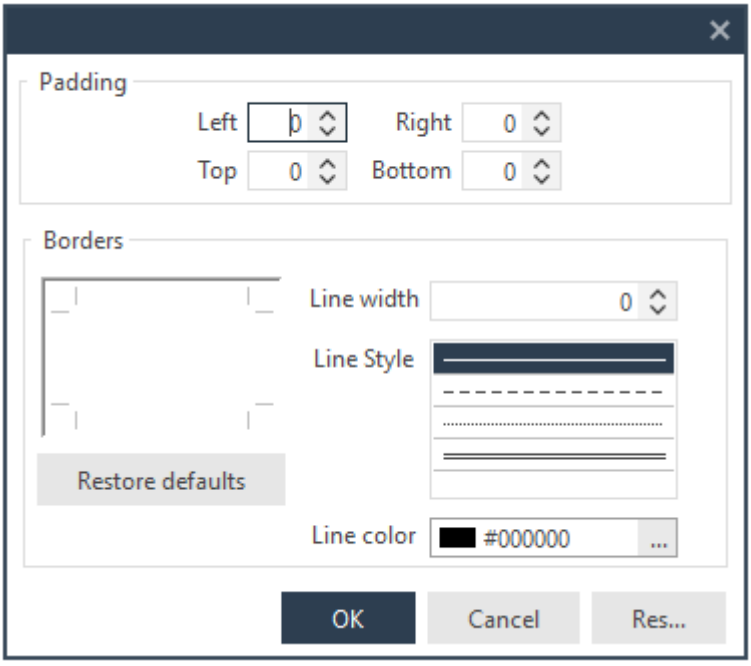
If we deploy, reload the entity and reopen our report now, it should already look a lot better.

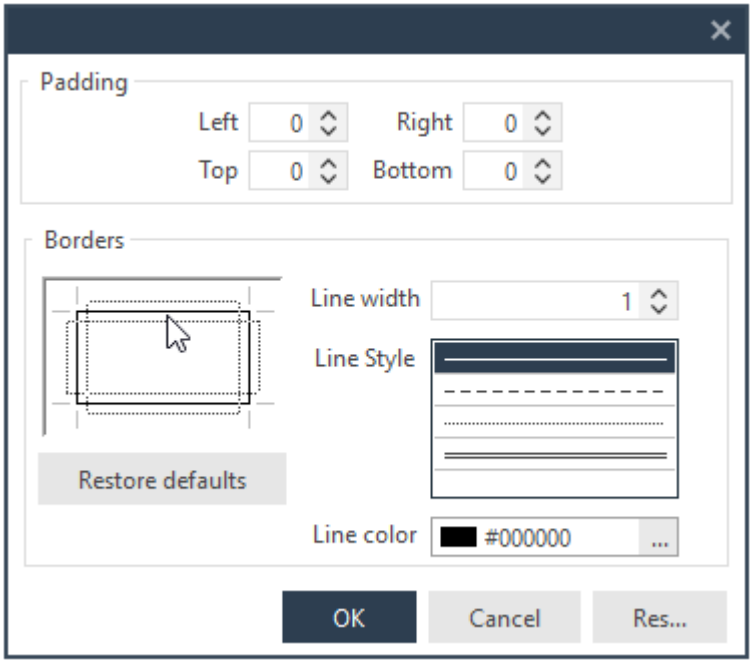


8.8.2. Padding and borders

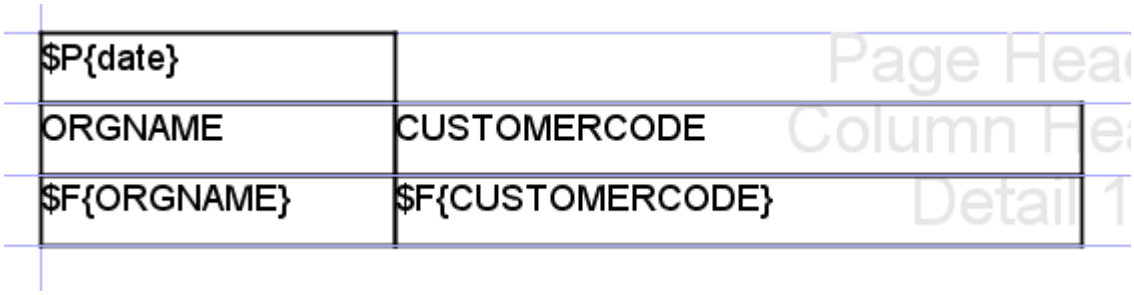
Another way to improve the report's appearance is to put lines/boxes around the Text Fields. You can do that with right-clicking on the Text Field and choosing `Padding And Borders`.
Here you choose the borders that should have a line by clicking in the little box area on the bottom left, and finally change the line width to more than 1.



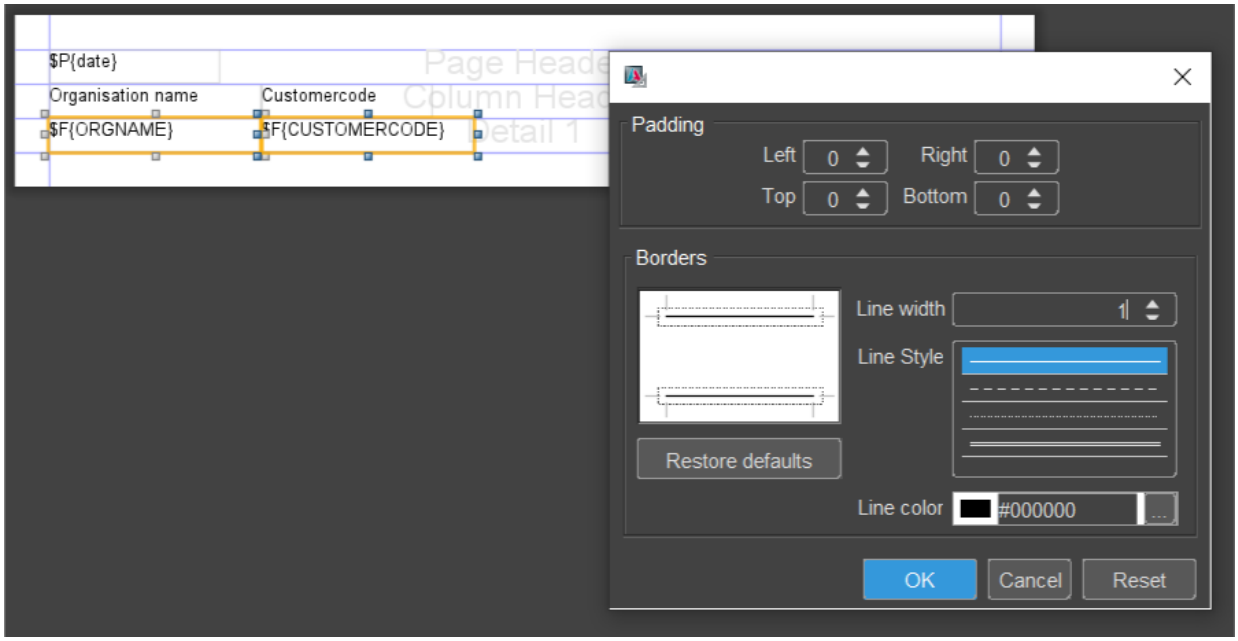




You can also do this while having selected more than one Text Field element at a time.



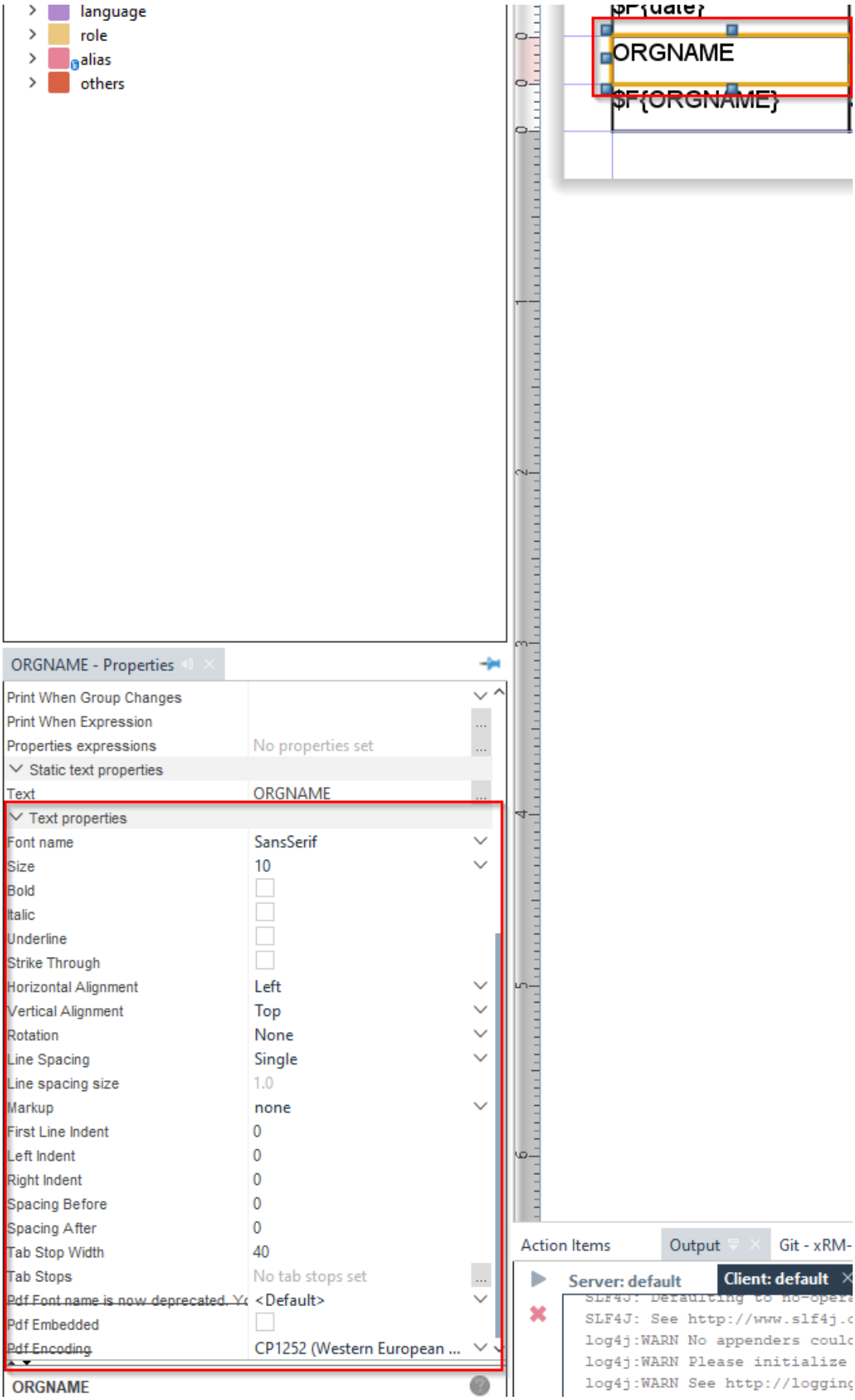
For our report, let's give our Fields a top and bottom border.



8.8.3. Text properties

Another way of customizing your report is to format the text in your Text Field element, e.g., by altering the font size, left margin, etc.

These properties can be configured in the Property window, after you have selected a Text Field element.



Here you have many different options to format the text inside your component, according to your requirements.

You can simply copy a specific format to other Text Fields, so you don't have to repeat configurations over and over again. This can save you a lot of time and potential mistakes.

You right-click on the Text Field element with the correct text format and select `Copy format`. Then you right-click the Text Field element that should be formatted, and select `Paste format`. This will copy all of the text properties. You can also use multiselection for pasting the format to multiple Text Field elements at a time.

If you have trouble with text not being displayed bold, even though it's configured as bold, refer to chapter Troubleshooting.

8.9. View-template for Reports

With ADITO you are able to display different views for different purposes. Another way to display the report is to implement a `ContextNameReport_view` . Therefore you have to create a new view in your context. Right click on the property of the new created view, in the navigator, and `Add View Template` … to your view. Select `Report` and write a name.

Now create a field in your entity that has the JDito code. You can use the same code as displayed above but you have to change the last line of the JDito code in order to display the report correctly.

```
//the second postion of the report is the data, the first is the
name
result.string(myReport.exportReport()[1]);
```

To improve performance you can check if the current field should be displayed.

onActionProcess-process that opens the reportview

```
import { neon, neonFilter, vars } from
"@aditosoftware/jdito-types";

let params = {
    "yourParameterInYourEntityThatIsExposed_param" :
true
}

let recipe = neonFilter
.createEntityRecordsRecipeBuilder()
  .parameters(params)
  .uidsIncludelist([vars.get("$sys.uid")]);

neon.openContextWithRecipe("CONTEXT", "VIEW", recipe
.toString(), neon.OPERATINGSTATE_VIEW, null, true);
```

valueProcess

```
if(Utils.toBoolean(vars.get("$param.yourParameterInYour
EntityThatIsExposed_param")))
{
  //generate the reportdata here
}
```

Always export your code to libraries. You can reuse the logic later and the code will be easier to maintain later.

Go back to your view and connect the created field as the `reportData` property. You can select a action if you already have one. Include the new view in a already existing one.

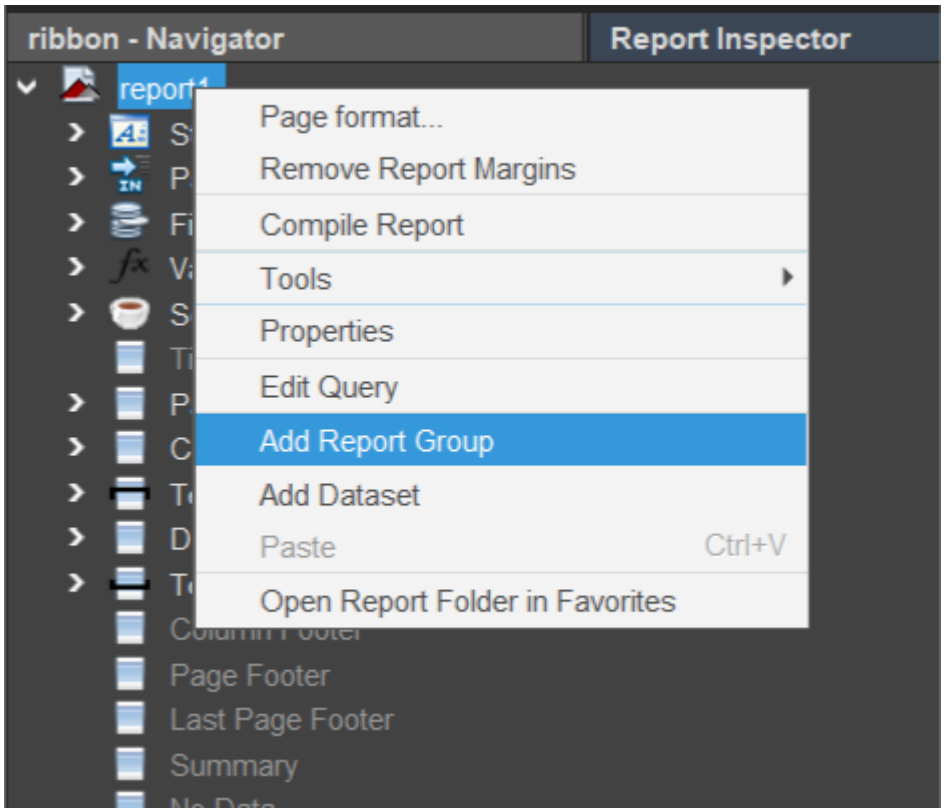
After your deploy the report should be visible.

8.10. Groups

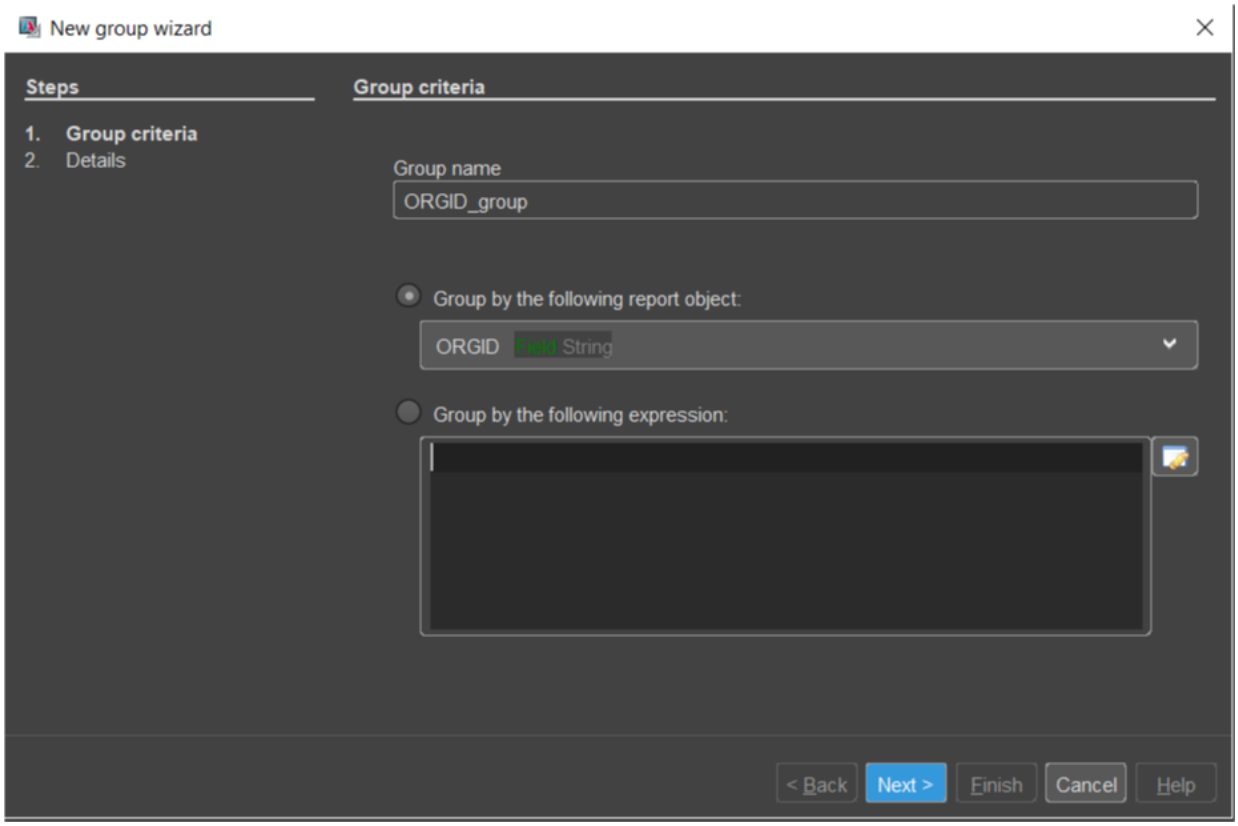
We’re now going to display all employees to our companies. To get that right connection between companies and employees in the end, we’re adding a Group.

For the following example we first add the report Field `ORGID`. Also add this field to the SQL statement as well as to the `rptfields` array in the `onActionProcess` process.

To add a group, you right-click on the report in the Report Inspector and choose `Add Report Group`.

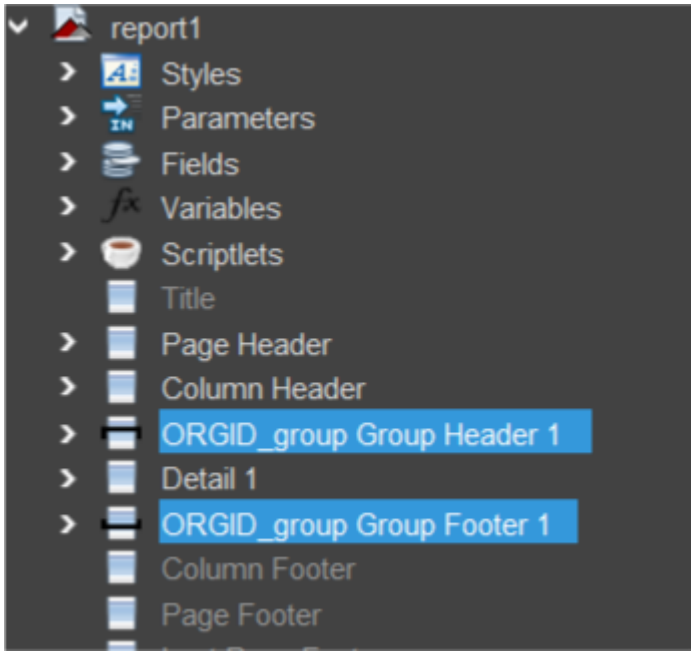


A dialog opens, in which you enter a name for the group ending with `_group`, e.g., `ORGID_group`, and the associated report object.



After clicking `Next`, you'll be asked if you want to add the Group Header and the Group Footer, which we both accept.

By clicking on `Finish`, the group is established and the two new bands are created around the Detail band.



Now everything in the Detail bands is being grouped by the ORGID.
The Group Header and Group Footer bands are repeated for every group (every ORGID).

Now we will add four more Fields to the report:

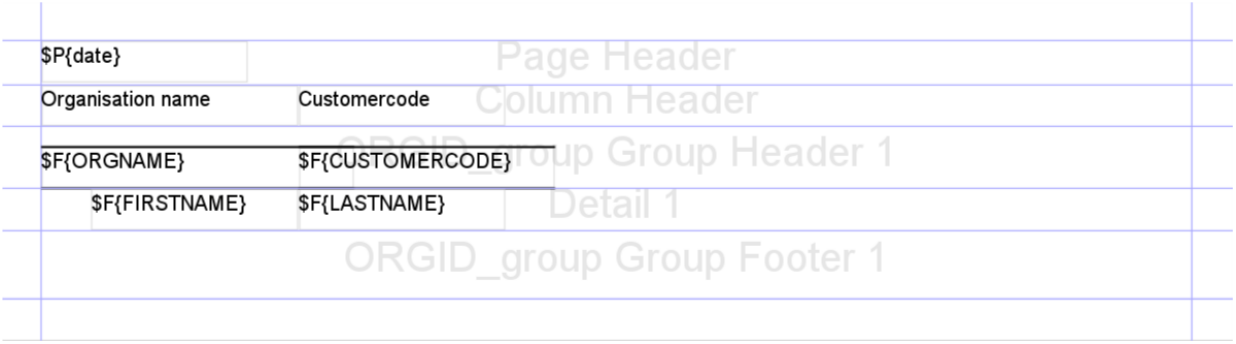
1. PERSID
2. FIRSTNAME
3. LASTNAME
4. CONTACTID (This one is needed in the next chapter)

Add those to the SQL statement (join the database tables PERS and CONTACT) as well as to the `rptfields` array. It should look like the following code:

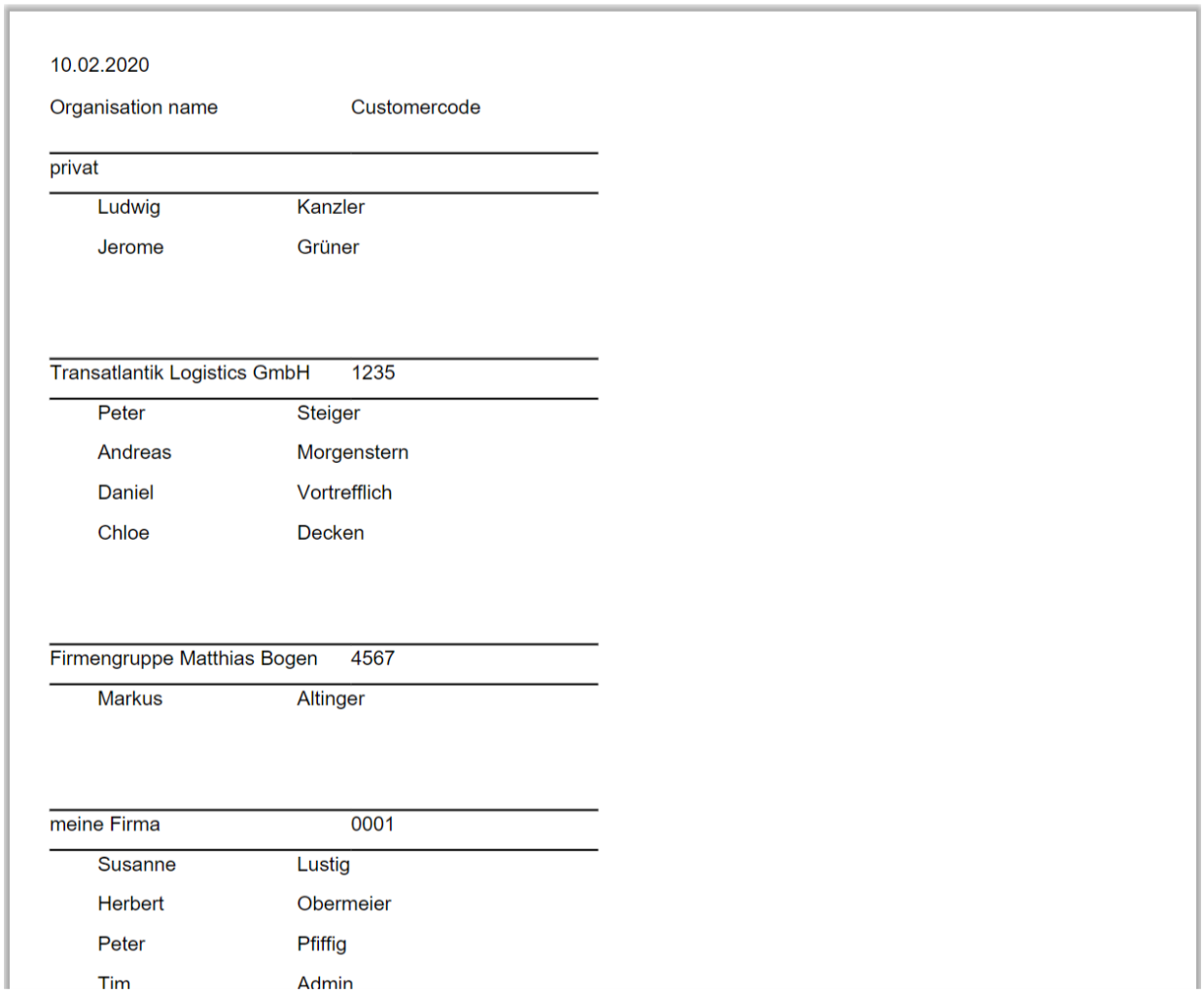
```
var rptfields = ["ORGID", "ORGNAME", "CUSTOMERCODE", "PERSID", "FIRSTNAME", "LASTNAME",
"CONTACTID"];

var data = newSelect("ORGANISATIONID, NAME, CUSTOMERCODE, PERSONID, FIRSTNAME,
LASTNAME, CONTACTID")
.from("ORGANISATION")
.join("CONTACT", newWhere("ORGANISATION.ORGANISATIONID = CONTACT.ORGANISATION_ID"))
.join("PERSON", newWhere("PERSON.PERSONID = CONTACT.PERSON_ID")
.table();
```

Now you can put all Fields corresponding to the company into the Group Header and the Fields corresponding to the person into the Detail band.



If you deploy and open your report, you should get a list of all companies with their associated persons.



8.11. Subreports

We now want to display all communication data for every person listed. Therefore we're going to add a subreport.

Subreports are used, if you want to display data from a separate, but related database table to your main data. E.g., if you want to display all communication data or addresses for one person.

8.11.1. Creating a subreport

To add a subreport to your report, you have to drag the Subreport component from the Palette window onto your Design tab and position it within a band. For our example we'll choose the Detail band.

When putting a Subreport element onto a report, a wizard for configuring the subreport is shown.

- The first page asks if you want to create a new report, to choose an existing report or to only create the element, which then can be configured later on. We'll go for "Create a new report".

- On the second page you can choose the layout of your subreport. You can find quite a few presets here. We'll go for Blank A4.
- Then the wizard asks about choosing a datasource, where `Empty Datasource` is typically chosen, because it does the basic configuration for the data, we will provide later. You could also create a new datasource, if your report should directly connect to a database. *But: Connecting a report directly to a database is not recommended, because all data should be provided to the report using JDito.*
- In the next two pages you're asked about which Fields you want to use and by which Fields to group by. Those pages are empty in our example, because we used an empty datasource.
- On the sixth page you are asked for a report name and where to store it. Here you just have to choose a file name, e.g., `subreportCOMM`. The wizard automatically integrates it into the standard paths within the ADITO project directory. With the radio buttons in the lower part of the wizard, you can choose if you want to store the relative directory path within a parameter or if you want to use the fixed absolute path. *The latter is not recommended, as it can lead to compatibility issues, if more than one developer works with the report*
- Lastly the wizard asks you, which source is to be used. Here we choose `Empty Datasource` again.

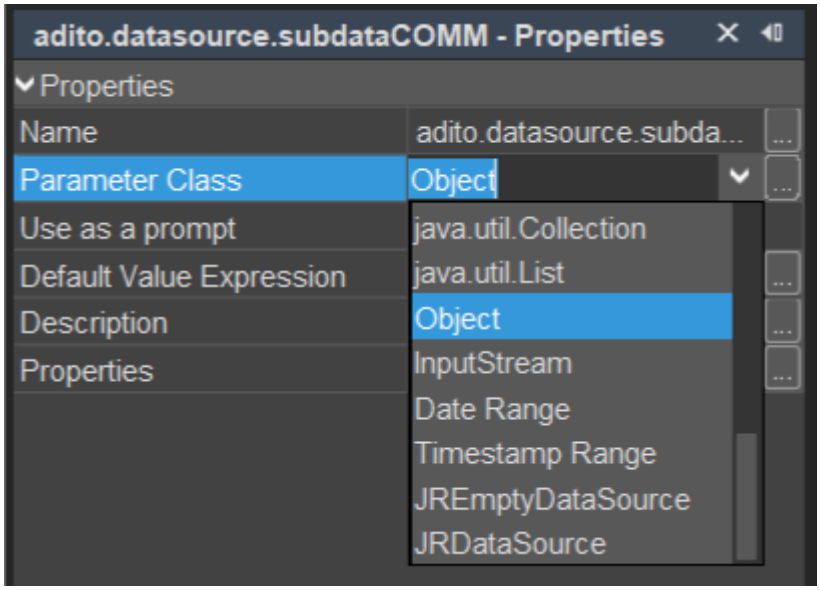
After clicking `Finish`, the Subreport element is placed within the master report and the subreport is opened in a new Editor tab.

Now we have to prepare our datasource in the master report, which then gets passed on to the subreport by a Parameter. Right click onto the Parameters node in the Report Inspector of the master report and choose `Add Parameter`. Then select the newly created Parameter and rename it to `adito.datasource.subdataCOMM`.

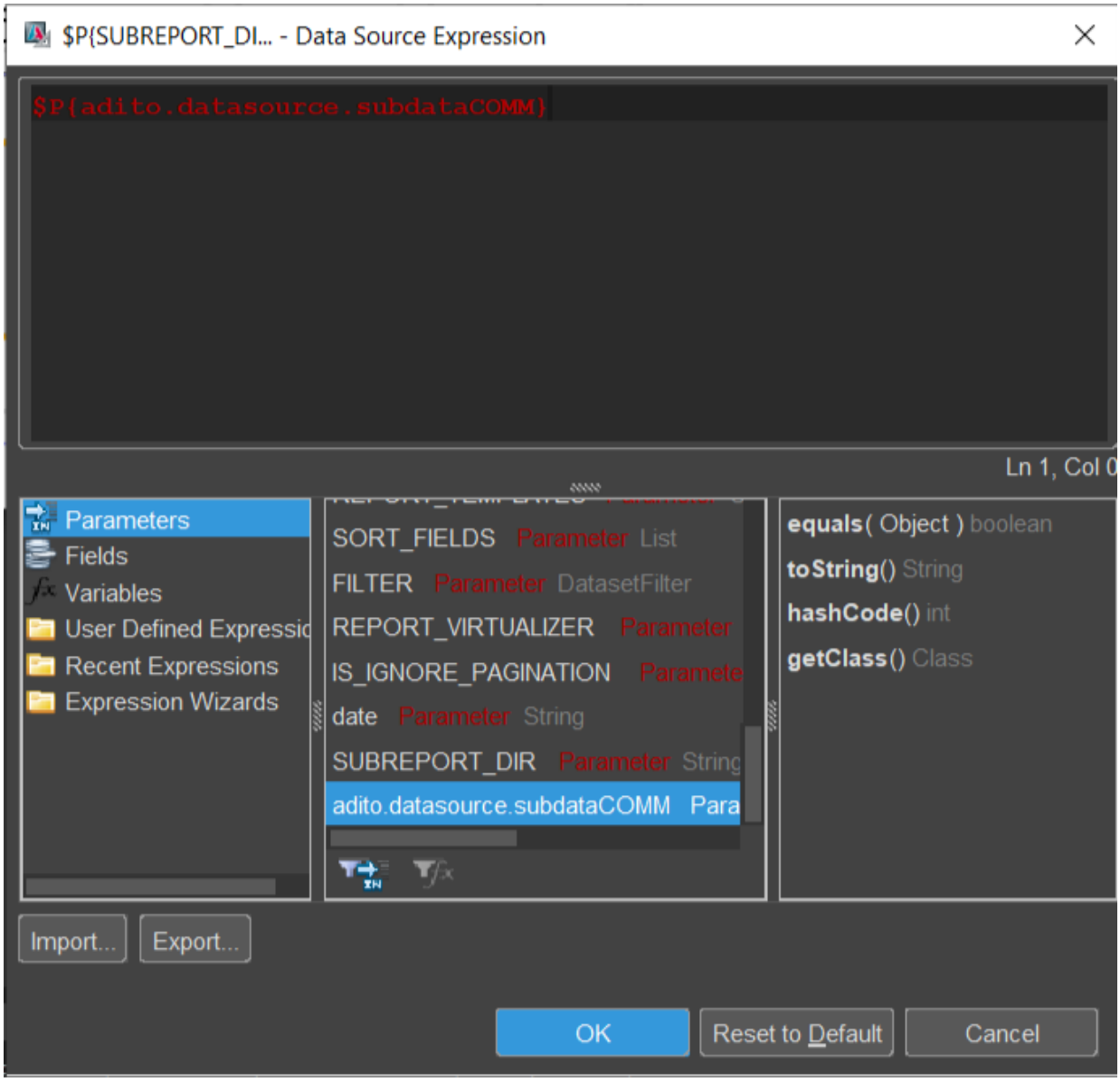
The name of the datasource Parameter has to follow a certain pattern.
It is always named beginning with `adito.datasource.`. Then after that you can add another word or numbers to describe it further. E.g. `adito.datasource.subdataCOMM` or `adito.datasource.sub1`. It also may not include special characters.

But the `adito.datasource.` part has to be there, otherwise ADITO doesn't recognize the passed on data as a subdataset!

After that you have to set the type of the Parameter to `Object` by selecting the Parameter in the Report Inspector and changing the property `Parameter Class` from `java.lang.String` to `Object` in the Property window.



After creating and configuring the Parameter, we select the subreport component in the Report Inspector of the master report to get its properties shown. Then we scroll down to the property `Datasource Expression` and click the … button. Here we choose the Parameter from the list and double click on it to add it to the expression or simply type the expression `$P{adito.datasource.subdataCOMM}` ourselves.



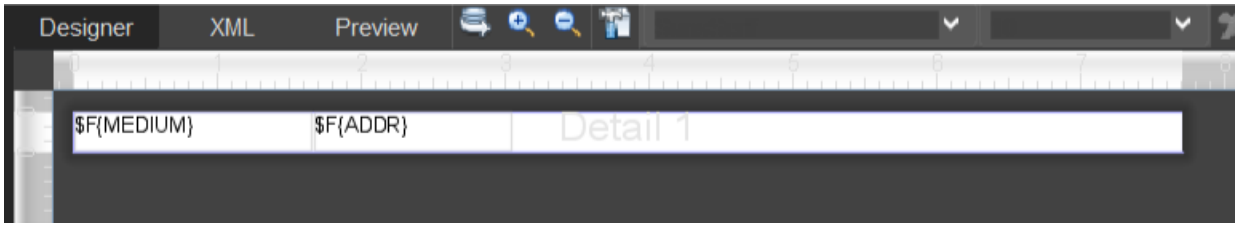
In the subreport, we want all communication data for a person to be listed.
To achieve this, we'll start with adding the needed Fields to the subreport.

1. COMMID
2. CONTACTID
3. MEDIUM
4. ADDR

Now delete every band but the Detail band, because that's the only band we need.

Then add the Fields "MEDIUM" and "ADDR" to the Detail Band and minimize the height of the band.

You report should look like this.



8.11.2. Passing data to the subreport with JDito

Now we need to pass on the data from the master report to the subreport. For that we need to expand the Parameters that are passed on in the `onActionProcess` process of our Action. The parameter in JDito has to have the exact name as the one in the master report. The following code gathers all communication data and gives it to the subreport.

```
var mySubreportData = new ReportData(["COMMID","CONTACTID","MEDIUM","ADDR"]);
var commData = newSelect("COMMUNICATION.COMMUNICATIONID,
COMMUNICATION.CONTACT_ID, COMMUNICATION.MEDIUM_ID,
COMMUNICATION.ADDR").from("COMMUNICATION").table();

mySubreportData.add(commData);
myReport.addReportParams(params);

myReport.addSubReportData("subdataCOMM", mySubreportData);
```

The function `addSubReportData` adds the data for a subreport. It concatenates the passed on subreport name with `adito.datasource.` (which sums up to the Parameter's name) and creates a "SubReportMap", that is passed on to that Parameter.

After this is done, you should be able to open the report with the subreport filled.

But you'll notice that there's way too much communication entries for every person. If you have a closer look, it'll get clear that every communication entry of every person and company is displayed for every person.

That's because the correct connection of the master report's data and the subreport's data hasn't been determinated yet.

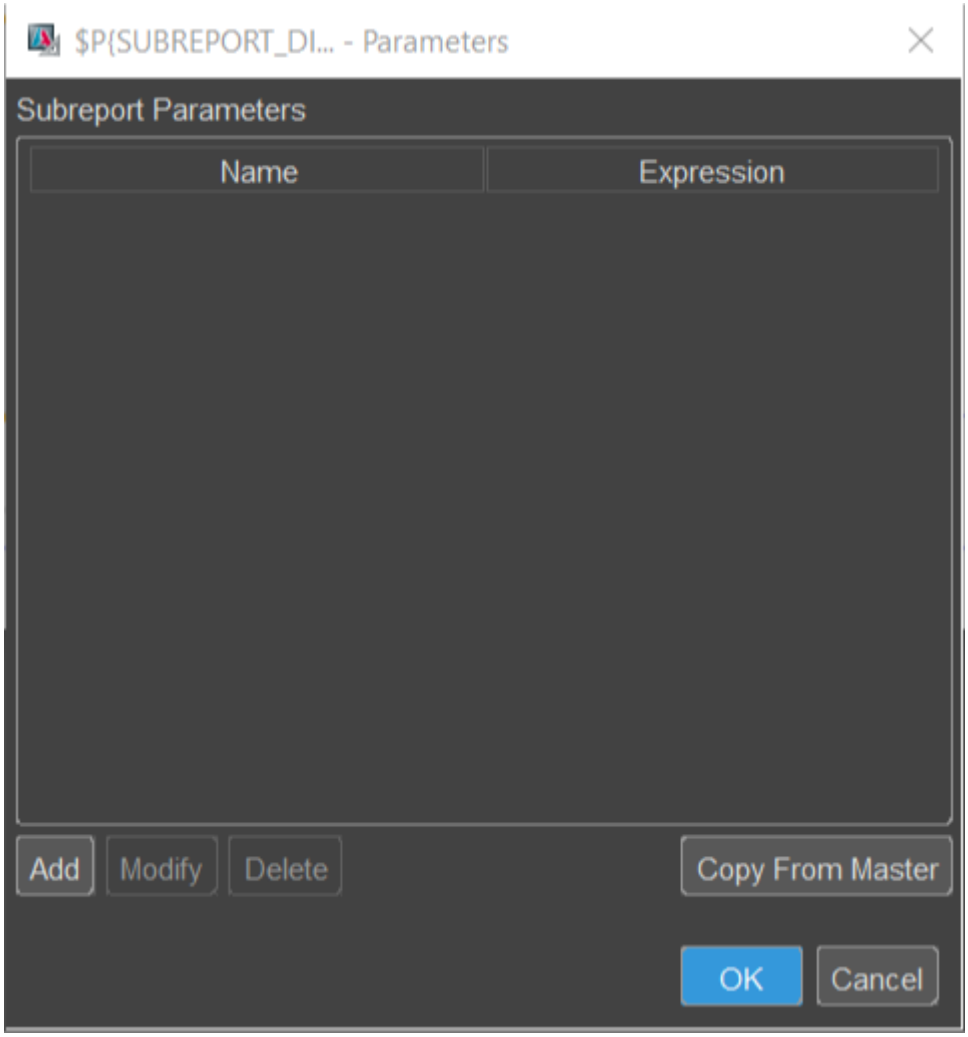
This can be fixed by using an Filter Expression.

8.11.3. Passing data to the subreport using Parameters

Subreport data in general is always passed on to the subreport via JDito code. But in case you want to pass on data of the master report to the subreport, you need to use Parameters. How exactly you do that, will be explained in the following in combination with the Filter Expression.

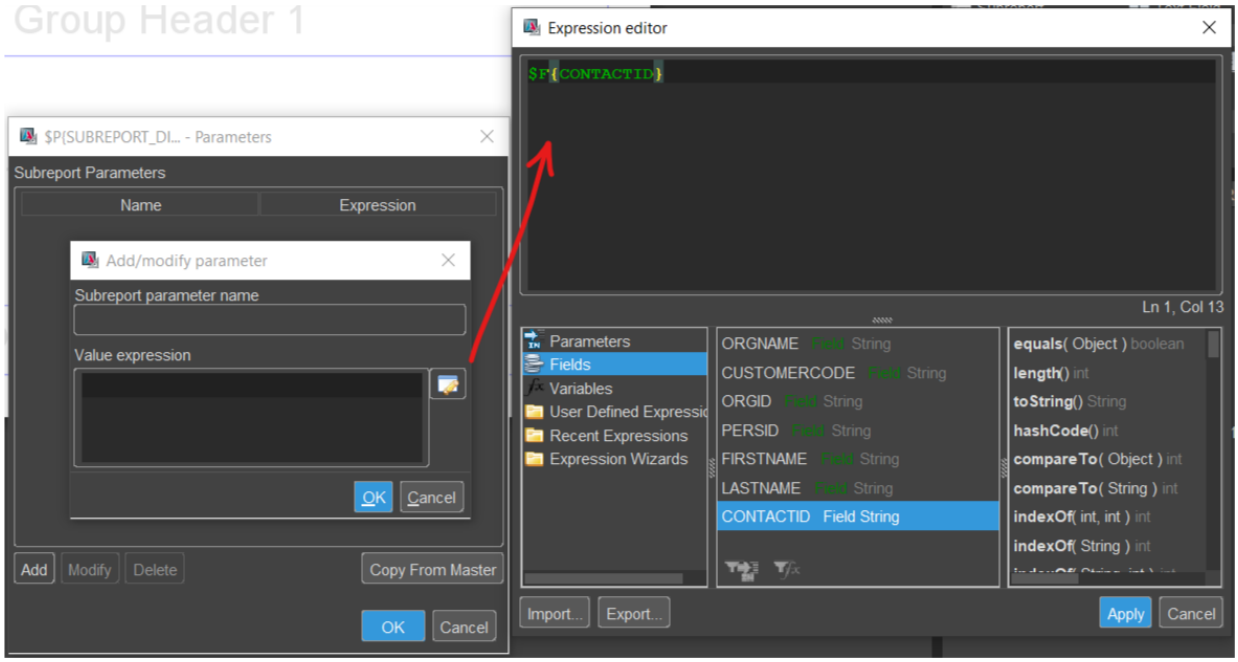
Back to our example: To filter the data of the subreport correctly, we need to define the relation between the two datasets. Here it is the `CONTACTID`. This is saved in the Field `CONTACTID`, which we added earlier to the master report. The value of this Field needs to be passed on to the subreport via a Parameter.

First we need to add a new Parameter to the subreport, e.g., `FILTER_CONTACTID`. Now we go back to our master report and select the subreport component to show its properties. Within the properties we scroll down to `Parameters` and click on the … button, which opens a wizard.



There we click on `Add`.
Here we enter the name of the Parameter which is `FILTER_CONTACTID` and then click on the

button next to the value expression list to open the Expression editor.



In this editor we choose the Field CONTACTID and double click on it to add it to the expression. Now we can press `Apply`,OK and OK to add the Parameter.

Now the value of the Field gets passed on to the subreport and can be used to only display the correct information. As a last step we switch back to the subreport and select it in the Report Inspector to show its properties. In the properties we can find `Filter Expression` and open the Expression editor by clicking onto the three dots.

As expression we add the condition, that the Parameter `FILTER_CONTACTID` and the Field `CONTACTID` need to be equal. For that we add the code `$P{FILTER_CONTACTID}.equals($F{CONTACTID})`. Now press OK to add the filter expression.

After you deploy the report and open it again, you'll see that only the right communication data is shown for each person.

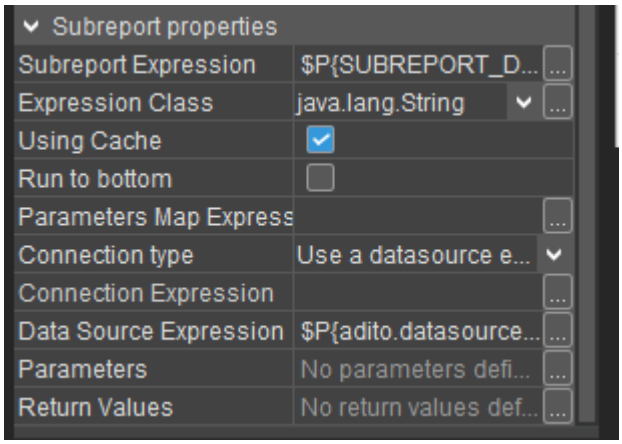
9. Optional: Subreports in Subreports

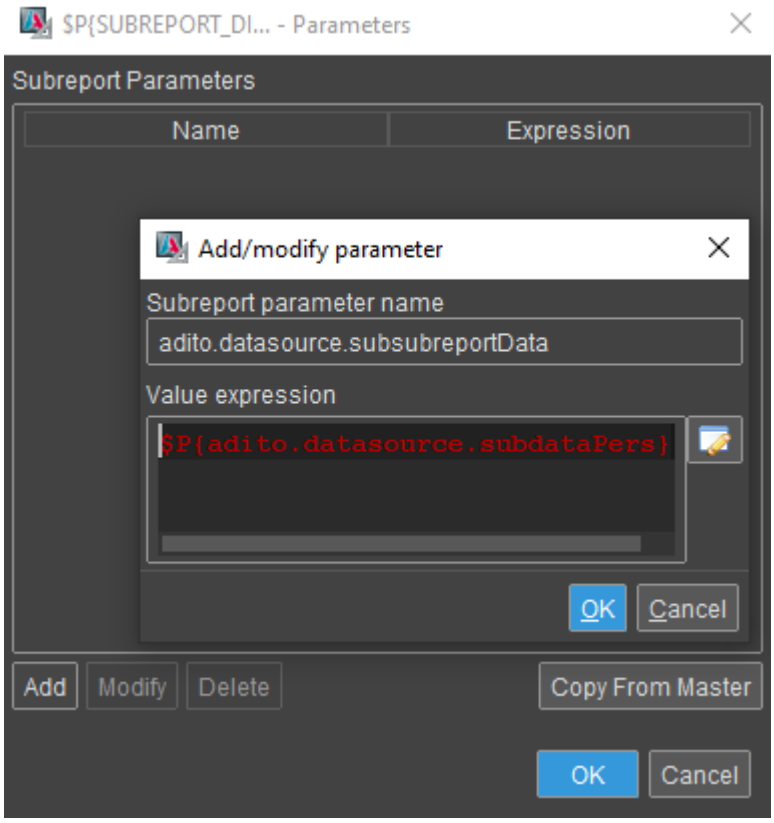
There are usecases where you need a Subreport in another Subreport. This will help you to list details of a detail. E.g. a report with all your companies and the corresponding employees with each of their Histories.

9.1. Subreport parameters

In order to receive data of the upper report you need to create a parameter just like you did with the normal subreport. Click on the upper/main subreport and fill the property "parameters" with the content of the other subreport.

This can look like this:





If you open the sub-subreport you can access this data via a parameter with the same name as above. Therefore you have to create and name the parameter like you already filled with the subreport.

10. Optional: Dynamic pictures

Dynamic pictures can be used to display e.g. different colors or the companys logo. Therefore it needs to be dynamic with each data set. There is a cheaky way to do just that.

10.1. Creating the image

To be able to use this feature you need to add a import to your report. Open your report and scroll down the properties of the report itself. The last property should be "Imports". Add an import with the name "org.apache.commons.codec.binary.Base64" and save it.

Grab a image-element in your desired canvas and leave it there select the properties of this element and fill/select the following two properties:

- image Expression: new ByteArrayInputStream(new Base64().decodeBase64($F{fieldName}.getBytes("UTF-8")))
- Expression Class: java.io.InputStream

With these you are allowed to interpret a Base64-String as a picture in the generated report. You also have to create a field with the same name as above.

10.2. Filling the image via code

Add a new field in the array and in each row for the values. This straight up includes the raw base64-String in there.
This **does not** include anything before the raw string, like "" which is often saved in the database.

10.3. Best Practice

Sometimes the image should not be displayed and you can simply give the code an empty string, but you have to change the "On Error Type" property to blank. Otherwise your report won’t open.

11. Additional information

This document gets you started on how to configure a basic report. Details about customizing your report can also be found online on the official Jaspersoft website. Selected elements are also explained further above in this document in chapter Elements.

12. Exercises

12.1. Exercise: Label printer

The goal of this exercise is to create a report that presents the standard address of **every person** in a certain way, so that we can print them on labels.

The difficulty of this exercise is, that exact positioning is essential for the outcome. Otherwise the content won't fit to the label paper, that it will be printed on.

12.1.1. Requirements

- Two columns
- The order of the contacts needs to be by lastname ascending
- A frame around each dataset
- Styling, that makes the report look like the example shown below
- Eight datasets (= eight addresses) per column need to fit on one page

The height of the Fields needs to be 14.

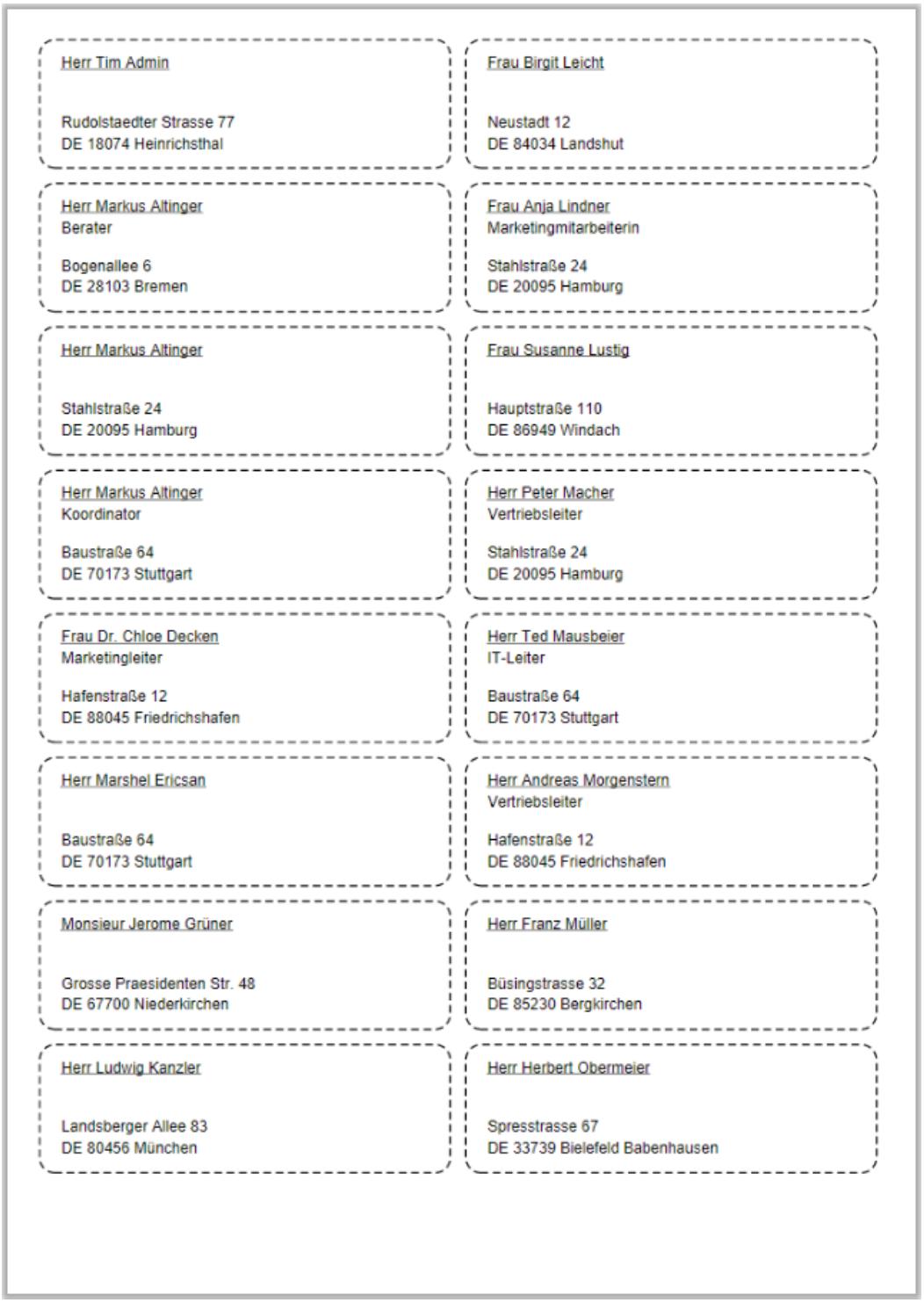
12.1.2. Database information

You can find all the necessary information in the following DB-tables/-columns:

- Person: PERSON > SALUTATION + TITLE + FIRSTNAME + LASTNAME
- Contact role: CONTACT > CONTACTROLE
- Address line 1: ADDRESS > ADDRESS + BUILDINGNO
- Address line 2: ADDRESS > COUNTRY + ZIP + CITY

12.1.3. End result

In the end your report should look as close as it gets like this.



Are your contacts on the first page of your report the same as the ones you see in this picture? If not, there might be something wrong with your SQL statement.

12.2. Exercise: Customer sheet

The goal of this exercise is to create a report that shows all companies with their employees.

In this exercise there will be a lot of different aspects to work on.

12.2.1. Requirements

- The icon and address of our company (meineFirma) at the beginning of **every new dataset (company)**
- The company's name and customer code
- All the company's employees with salutation, name, department and contact role
- At the very bottom of every the page the current date and the page number
- All Text Fields should translate to the user's language, where reasonable
- Styling, that makes the report look like the example shown below

- Have a closer look at chapter `Groups` in your reporting manual.
- Remember that there are predefined variables.
- Base64-Strings of pictures are stored in the database table ASYS_BINARIES. **Watch out**: The function `getMyASYS_ICONSdata()` is not implemented in ADITO 2020 yet. Ask your trainer for a workaround.

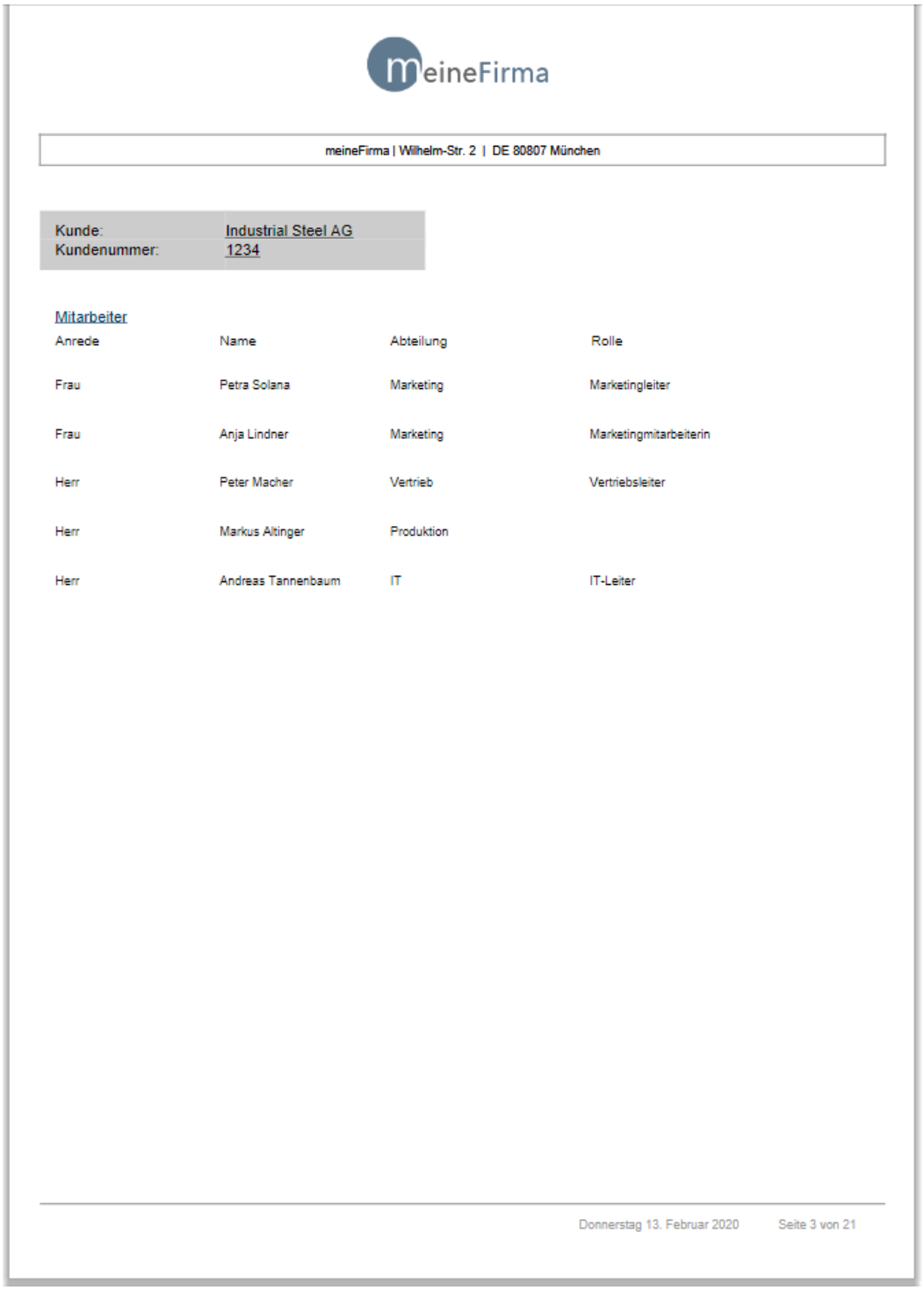
12.2.2. Database information

You can find all the necessary information in the following DB-tables/-columns:

- Company name: ORGANISATION > NAME
- Customer code: ORGANISATION > CUSTOMERCODE
- Employee salutation: PERSON > SALUTATION + TITLE
- Employee name: PERSON > FIRSTNAME + LASTNAME
- Employee department: CONTACT > DEPARTMENT
- Employee relation title: CONTACT > CONTACTROLE

12.2.3. End result

In the end (the third page of) your report should look as close as it gets like this.



meineFirma

meineFirma | Wilhelm-Str. 2 | DE 80807 München

Kunde:	Industrial Steel AG
Kundenummer:	1234

Mitarbeiter

Anrede	Name	Abteilung	Rolle
Frau	Petra Solana	Marketing	Marketingleiter
Frau	Anja Lindner	Marketing	Marketingmitarbeiterin
Herr	Peter Macher	Vertrieb	Vertriebsleiter
Herr	Markus Altinger	Produktion	
Herr	Andreas Tannenbaum	IT	IT-Leiter

Donnerstag 13. Februar 2020 Seite 3 von 21

12.2.4. Add-on exercise

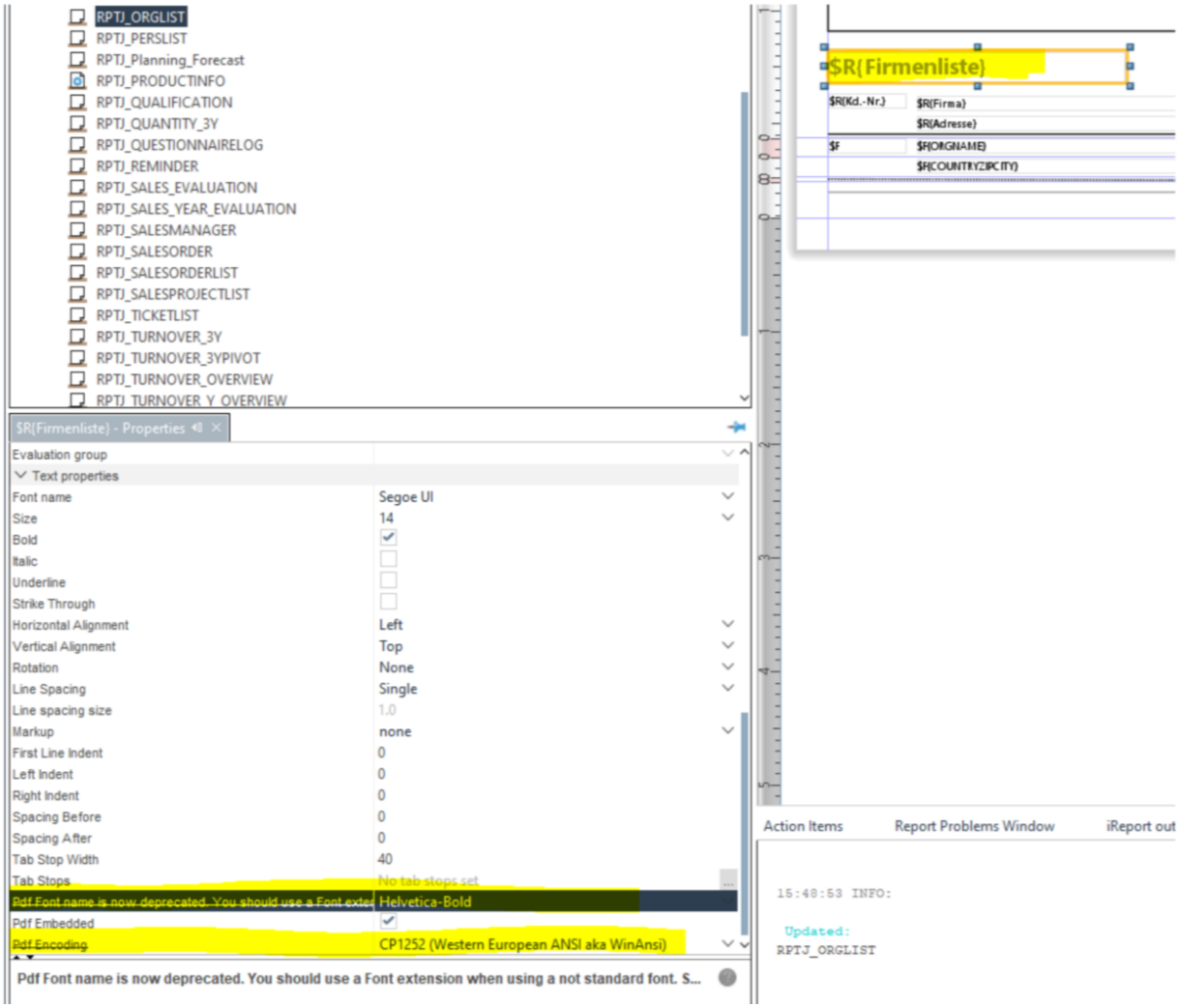
Show the company's histories after the employees.

13. Troubleshooting

13.1. Text not being displayed as bold

If you configured Text in your Jasper Report as bold, but in the exported pdf-File it's not bold anymore, this is what you have to do:

- Look for the Text Field property `Pdf Font name is now deprecated. You should use a Font…`
- Enter one of the following values
    - Helvetia-Bold → bold
    - Helvetia-Oblique → cursive
    - Helvetia-BoldOblique → bold & cursive
- In property `Pdf Encoding` enter "CP1252 (Western European ANSI aka WinAnsi)"



13.2. Adding Fonts to a ADITO Cloudsystem

In order to add/change existing fonts for you report you need to add them via the filesystem of your cloudsystem.

These Fonts have to be licensed!

Download your tunnel via "Download Tunnel" in your SSP System. Execute the .bat-file and now your tunnels should be opened locally. You can access the filesystem via "localhost:80" (Port 80 might not work, simply change the resulting port in the .bat-file that you want to use). Place your .jar-File, which include the fonts, in the "ext"-Folder. After restarting each pod you should be able to generate a reporting and have e.g. bold fonts available.

You can add new fonts in your designer too. Open your ADITO Designer and open the settings. Go to the tab "iReport" and open "Fonts". Now you can install your desired font there. Follow the given instructions.

13.3. Get a String with milliseconds to Date-format in the report

If you pass a millisecond value as string in the data array to a Jasper report (e.g. as result of a timestamp or date query in a SQL) and want to convert it into a readable date value for display, this can be done by the Text Field Expression "new java.util.Date(Long.parseLong($F{DATEFIELD}))" and the Expression Class "java.util.Date" at the corresponding Field in the Property's window.

So that the report does not throw an error if the date value is empty, " $F{DATEFIELD} != "" " should be entered in the Print When Expression.

13.4. Jasper Report Bars Diagram Axes Label

To display only integers in the value axis in a Jasper Report it is sufficient to insert the property `net.sf.jasperreports.chart.range.axis.integer.unit` with the value "true" in the Properties Expressions of the diagram.
In order to prevent the unattractive behaviour of the axis display at 0 values, it is sufficient to set the value 0 in the Range Axis Min Value Expression with Double.parseDouble("0").

13.5. Incorrect sorting of the quotation items in the report

If you have the following sorting of the quotation items in the quotation report:

1
10
11
2

3

…

Here’s what you have to do to get the order right:

Disable sorting in the report for the Pos field.
- Open the affected report (here "RPTJ_OFFER").
- Switch to the XML view
- Remove the corresponding "sortField" tag (here:<sortField name="OFFERITEM_ITEMPOSITION"/>), if present.

The array with the offer positions is usually sorted correctly by default, otherwise you have to sort the array correctly in the JDito code.

13.6. Jasper Reports seems to load forever

When opening a report, it may happen that it seems to load forever and the client then has to be shot down/reopened.
This can happen if there is an overflow in a subreport: A Text Field is set with "Stretch with Overflow" and "Position Type" = "Fix relative to Bottom".
To fix the problem, the Position Type must then be set to "Fix relative to Top".

13.7. Errors from missing Subreports

Sometimes when you delete subreports it could happen, that the report displays an error while trying to open it. This problem could be caused of the missing files or dead references in the AOD of the report.

To solve this problem, you have to navigate to your project and open the folder `report`. Now you can see every report in your system. Go into your problematic report and you should see all `.jrxml`. You can delete all your deleted subreports, which you already tried to delete the designer. Now open the AOD-file and navigate to each dead reference and delete the tag `<subreport>` from the file:

The part that should be deleted (example values)

```
  <subReports>
    <subreport>
      <name>subreportCOMM1</name>

<reportData>%aditoprj%/report/MyReport_report/subreportCOMM1.jrxml</
reportData>
    </subreport>
  </subReports>
```