Visma Reporting - Report Design Guidelines

Table of Contents

Introduction 4

Chapter 1: Drill-Through Reports 5
- Drill-Through Reports 6

Chapter 2: Report Images 7
- Embedded Images 8
- Company Logo 8
- External Images 8
- Globalization of Local External Images 8
- Adding File Provider Images (internal use only) 9
- Add Image Data Set Manually 10

Chapter 3: Parameters 11
- Available Values 12
    - Reserved Parameter Values (Non-queried mode) 12
    - Parameter Values (Non-queried mode) 12
    - Dynamically Queried Values 13
    - Static Parameter Values 15
- Parameter Visibility 16
- Extended Parameter Visibility 20
- Default Values 20
- Formulas 21
- Select All Values 21
- Parameter Description 22
- Parameter Grouping 22
- Display Selected Values 25
- Visualize Report Filter 26
- User, Company and Language Parameters 27
- Parameter Value Cache Cross Report 28
- Datasets Used in Parameters Only (performance options) 28
- Dataset Named Parameters 28
- Dataset Unnamed Parameters 29
- Forcing Server Update 30
- Filtering report data using group filters and multi-value parameters 31
- Report Scaling 33

Chapter 4: Translation of Report Texts 36
- Textbox Values 37
- Report Parameters 37
- Syntax of the Translation Tag 37
- Language Text Files 38

Chapter 5: Merging Data From Multiple Data Sources 39
- Merging Data From Multiple Data Sources 40

Chapter 6: Multi-Application Reporting 41
- Multi-Application Reports 42
- Application Type 42
- Parameters 42
- Dataset Naming 43
- Database Datasets 43
- Merge Dataset 44

Chapter 7: Multi-Company Reporting 46

Multi-Company Reports....................47
Column-Based Datasets....................47
Fixed Company Query....................47
Dataset Naming....................48
Database Dataset....................48
Merge Dataset....................49
Raw-Based Datasets....................50
Fixed Company Query....................51
Merge Dataset....................51
Drill Through to Company Report....................52

Chapter 8: Electronic Reporing....................54
Electronic Reporting....................55
Electronic Reports....................55
Electronic Report Examples....................55
Parameter Setup....................56
Sending the Electronic Report....................57
Downloadable Reports....................58
Examples....................58
Parameter Setup....................59

Chapter 9: Deploying Report Packages....................61
Deploying Report Packages....................62
How to authenticate to report server to use deployment of reports....................63

Chapter 10: Visma Business Custom Functions....................65
Organizational Unit Parameters....................66
Organizational Unit in Dynamic SQL....................67
Date Handling, Visualize Company....................67

Chapter 11: Visma OnDemand Reports....................68
How to Activate the Data Extension in MS Report Designer....................69
Data Source Definition....................69
Dataset Query Definition....................70
Passing Date Time Parameters to a Stored Procedure....................71
Send a Report as PDF to Email Recipients....................71
Export a Report for Sending to Visma Print Service (Maventa)....................75
Export a Report for Sending to Itella Print Service....................76
Run-time Translations in a Report....................79
’GlobalizationString’ field....................81

Chapter 12: SQL Server Analysis Services Reports....................83
Working With Parent-Child Hierarchies....................84

Chapter 13: Subscriptions....................85
Dynamic auto-update date parameters....................86

Chapter 14: Data Providers....................87
Available Data Providers....................88

Chapter 15: Tips and Tricks....................90
Report Design Tips and Tricks....................91
Summary....................93

Introduction

This document describes special features and functionality developed by Visma, which can be used together with Microsoft Report Designer when you design or modify reports for Visma Reporting.

It is required that you are familiar with Microsoft Report Designer and have a certain understanding of T-SQL (select statements) to enjoy this document. For documentation on Microsoft Report Designer see the SQL Server online help.

You will need to use Microsoft Report Designer in Visual Studio 2008 to design reports, since the viewer in our application only supports MSSQL Server 2008 Reporting Services.

You don’t need this document to create standard reports, but in some cases you will fine helpful functionality that makes your reports easier to create and easier to standardize. Hopefully you will also find some helpful hints.

Chapter

1

Drill-Through Reports

Topics:

- *Drill-Through Reports*

Drill-Through Reports

A drillthrough report is a report that a user opens by clicking a link within another report.

To achieve this, drill-through reports must reside in the same folder as their *parent* reports when they are deployed into Visma Reporting.

Chapter

2

Report Images

Topics:

- *Embedded Images*
- *Company Logo*
- *External Images*
- *Globalization of Local External Images*
- *Adding File Provider Images (internal use only)*
- *Add Image Data Set Manually*

Embedded Images

Embedded images are part of the RDL file. No special treatment is needed.

Company Logo

Company logos can be stored in the Visma User Directory to be shown in reports. See VUD online help, for instructions on how to add the logofile.

When a company has a logo it can be shown in the report by:

1. Add an image to the report
2. Name the image VR_LogoImage
3. Set the image source to External

The logo of the selected company will now show in the report.

External Images

The 'Value' property of the image must contain the URL for the image file. External images located on the local file system of our own application server (file://) will be embedded automatically in the RDL file when running the report in Visma Reporting. External images located on a remote web server (http://) will be linked and shown run time (not embedded).

The URL of a local external image must be on the format:
`file://<FilePath>/<FileName>`, or `file://<FilePath>\<FileName>`

The URL of a remote external image must be on the format:
`http://<WebSite>/<FileName>`

Example of an image URL in which a path is built up using a field in the product table of the ERP system:

```
file://c:\visma\sysadmin.bmp
http://www.vg.no/logo.gif
=”file://” & FieldsPictNo.value
In this example you build up a path using a field in the product table of the
 ERP system
```

If you use a local external image the report server will collect the images and embed them into the report before it is sent to the report viewer.

Globalization of Local External Images

If you want to have different versions of a report image dependent on the selected language, put the different versions in the 'Resources' folder on the server. We use the *Visma.Core.Globalization* component for finding the correct image file for the selected language.

The file name of the image file must be on the format:
`<LanguageCode>-<CultureCode>_<Domain>.<FileName>`

<LanguageCode>-<CultureCode> can be one of the followings:

en-GB	English (U.K.)
no-NO	Norwegian
sv-SE	Swedish
da-DK	Danish

fi-FI	Finnish

<Domain> can be Reporting - Image files for Visma Reporting.

<FileName> must correspond to the image URL of the image in the report. Example: en-GB_Reporting.sysadmin.bmp.

Note:

All language codes, culture codes and domains are case sensitive. Take this in regard when setting up image files for globalization.

Adding File Provider Images (internal use only)

About this task

In addition to using Embedded and External images it is possible to display images that are loaded from a database or a file archive. If the Application Type = Visma Business the described functionality is used/reserved to collect images from Visma Document Center.

This can be done by using Visma Reporting's File Provider. The file provider must be implemented to support your file storage, before you try to display File Provider images in your report. This is done automatically when Visma Reporting is installed. The File provider will collect all image files on the server.

The following must be performed to add an image to your report that is read from the File Provider:

Procedure

1. Create image data source
2. Create a new data set for images. The data set must be named **dsReportImages** to be recognized as a report image data set
3. Add a data source field that represents your image. For instance a data source field named "MyReportImage"
4. Create a dummy query that populates the columns if you want to be able to compile the report in the Designer. For instance:

   ```
   SELECT      TOP (1) ID AS MyReportImage
   FROM        SmallTable
   ```

   MyReportImage is the Dataset field name. This is only done to avoid error messages when previewing the report in Microsoft Report Designer.
5. Add a report parameter of type string to the report and give it the same name as the data source field (e.g. "MyReportImage")
6. Mark the parameter as Hidden to avoid displaying it in the UI
7. Set the parameter value to be your **file id**. (This depends on the file provider of your product, but will normally be a file path or an id that can be used by your document archive to retrieve the file)
8. Add image control to report
9. Set the image Source property to **Database**
10. Set the image Value property to `=First(Fields!MyReportImage.Value, "dsReportImage")`

Results

The report should now display the image provided by your File Provider for the given file id.

Add Image Data Set Manually

An alternative to the instruction list above is to add the data source into the report definition file (.rdl) file in a text editor.

Copy the followings to the *DataSets* section of your .rdl file, below the last *DataSets* section:

```
<DataSet Name="dsReportImage">
  <Query>
    <CommandText/>
    <DataSourceName>yourDataSourceName</DataSourceName>
  </Query>
  <Fields>
    <Field Name="MyReportImage">
      <rd:TypeName>System.Byte[]</rd:TypeName>
      <DataField>MyReportImage</DataField>
    </Field>
  </Fields>
</DataSet>
```

The ”yourDataSourceName” value above must be an existing data source.

Make sure your logo image control properties look like this:

```
<Image Name="yourLogoImageControlId">
  <Sizing>Fit</Sizing>
  <MIMEType>image/jpeg</MIMEType>
  <Source>Database</Source>
  <Value>=First(Fields!MyReportImage.Value, "dsReportImage")</Value>
</Image>
```

Chapter
3

Parameters

Topics:

- *Available Values*
- *Parameter Visibility*
- *Extended Parameter Visibility*
- *Default Values*
- *Formulas*
- *Select All Values*
- *Parameter Description*
- *Parameter Grouping*
- *Display Selected Values*
- *Visualize Report Filter*
- *User, Company and Language Parameters*
- *Parameter Value Cache Cross Report*
- *Datasets Used in Parameters Only (performance options)*
- *Dataset Named Parameters*
- *Dataset Unnamed Parameters*
- *Forcing Server Update*
- *Filtering report data using group filters and multi-value parameters*
- *Report Scaling*

This chapter describes how to correctly setup a report parameter.

Available Values

To add available values to a report parameter you can either add Non-queried values directly to the parameter definition, or create a DataSet and get the values From Query. The DataSet must be reserved for use with parameters only.

Note: You can not use a DataSet which is being used by the report itself.

To be able to get access to the extended features of Visma Reporting the Non-queried solution is recommended. This includes features such as parameter grouping and positioning, and value query directly from an SQL statement instead of using a DataSet.

Reserved Parameter Values (Non-queried mode)

Visma Reporting defines a set of reserved parameter values. These are values with a label name prefixed with **VR_** which will be used to define how and where the parameter is rendered, how the parameter is populated etc.

The value must be prefixed with equal sign "=" and placed within quotation marks **" "** to be accepted by the Report Designer.

Parameter Values (Non-queried mode)

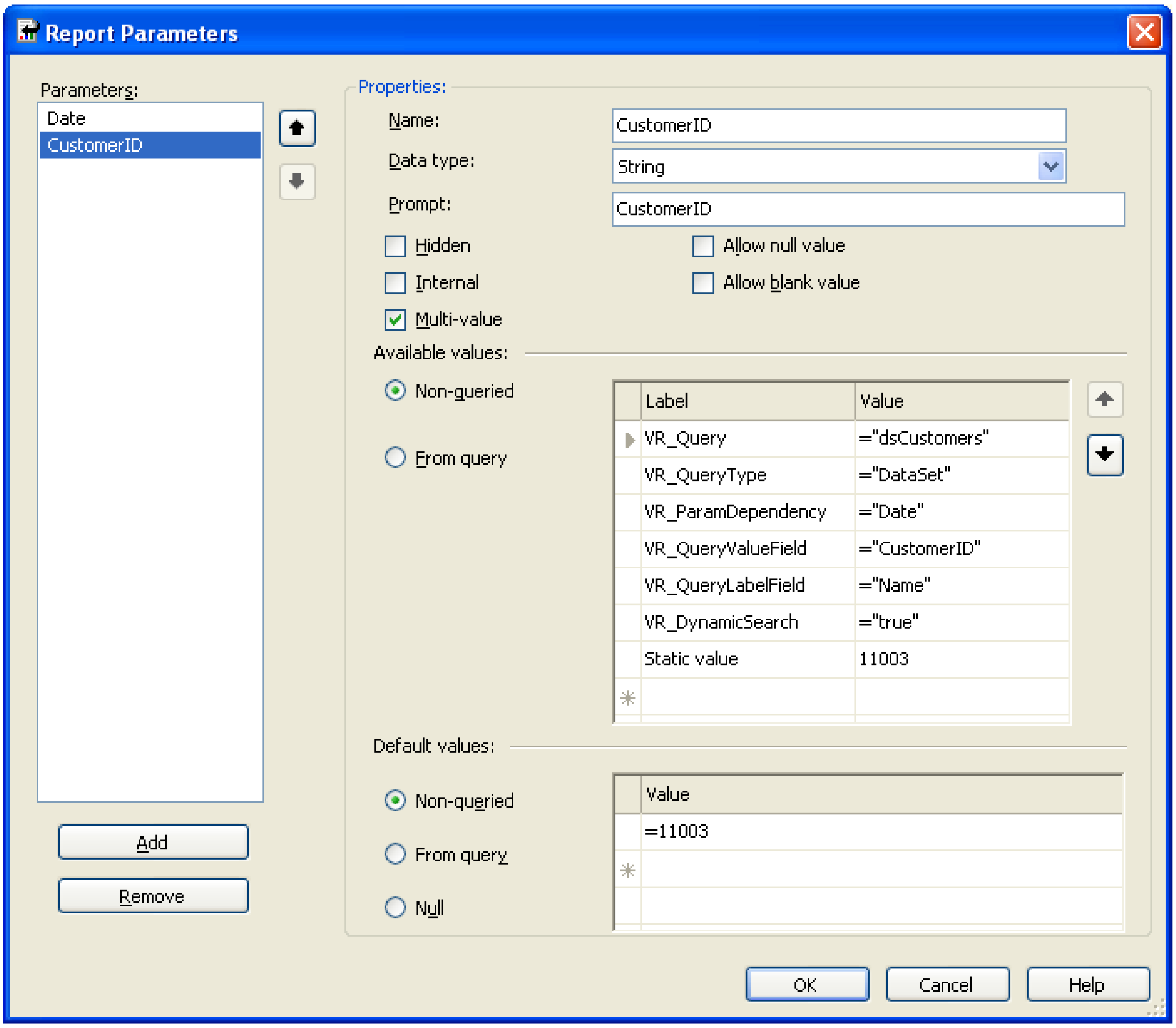
The available values which are defined without a label that starts with VR_ are shown as normal available values for the parameter. You cannot use both predefined available values and queried values in the same parameter.

Queried parameters can be populated using different value labels:

- VR_QueryType: the type of query used to populate parameter values. The type is either DataSet, Query or Command:

  - DataSet: the available parameter values are populated from a DataSet
  - Query: the available values are populated from an sql statement. The sql is written directly into the parameter value field
  - Command: the available parameter values are populated from provider logic (Internal use only)
- VR_Query: the sql, command (dummy dataset name) or dataset name that represents the query
- VR_QueryValueField: the name of the data set column that holds the parameter values
- VR_QueryLabelField: the name of the data set column that holds the parameter label (value is optional)
- VR_DynamicSearch: if value is set to ="true" the values are not pre-fetched, but requires a search to be displayed. Use this feature when the number of parameter values is large. See chapter *Dynamically Queried Values* (see page 13) for more details
- VR_ParamDependency: if the query is dependent of any other parameter values, the parameter names should be listed here as a comma separated list of parameter names. Example: CustomerID, DepartmentID

  Note: If using QueryType DataSet the dependencies are automatically calculated and you can skip this parameter value.
- VR_Visible: used to show or hide parameters based on a parent parameter
- VR_VisibleExtended: used to show or hide parameters based on a parent parameter
- VR_DataType: since one only can set custom parameter values on parameter of type String the actual type should be set. The valid types are: *String, DateTime, Integer, Float* and *Boolean*
- VR_ParentParam: Name of the parent parameter if there are dependencies between the parameters



Name:	CustomerID
Data type:	String
Prompt:	CustomerID

Hidden [ ] Allow null value [ ]
Internal [ ] Allow blank value [ ]
Multi-value [✓]

Available values: [✓ Non-queried] From query

Label	Value
VR_Query	="dsCustomers"
VR_QueryType	="DataSet"
VR_ParamDependency	="Date"
VR_QueryValueField	="CustomerID"
VR_QueryLabelField	="Name"
VR_DynamicSearch	="true"
Static value	11003

Default values: [✓ Non-queried] From query Null

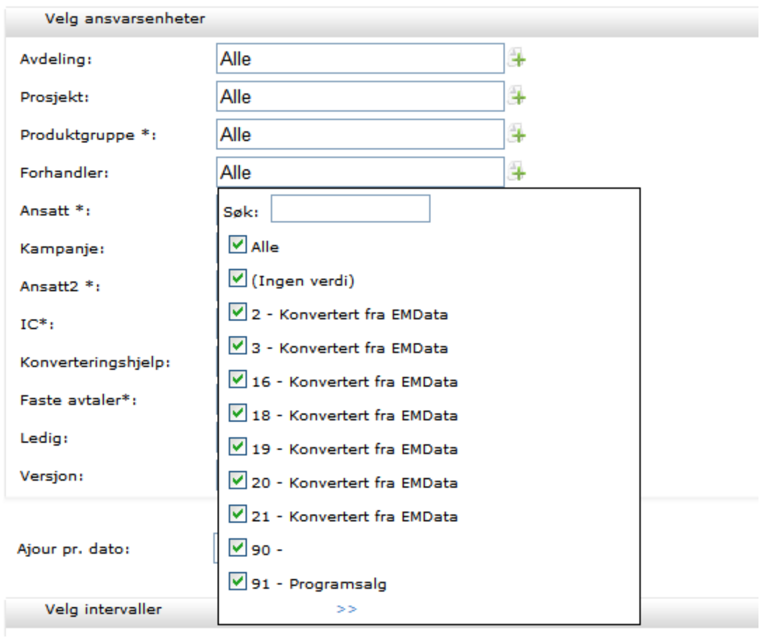
Value
=11003

Note: You cannot use both predefined available values and queried values in the same parameter. This means that if VR_Query is defined all other parameter values are ignored. (in the image above the “Static Value” would be ignored).

Dynamically Queried Values

When a parameter represents a large set of available values to select from, it could be a better approach to let the user search for the wanted value, instead of selecting from a large list. It also gives better performance to avoid pre-fetching to much data. When it is more than 100 available values it is recommended that dynamic search is used.

The image below illustrates how the user interface looks when parameter values are fetched without dynamic search used. All values are pre-fetched once the “+” icon is clicked. It is then possible to search within the available values.



Velg ansvarsenheter

Avdeling:	Alle
Prosjekt:	Alle
Produktgruppe *:	Alle
Forhandler:	Alle
Ansatt *:	
Kampanje:	
Ansatt2 *:	
IC*:	
Konverteringshjelp:	
Faste avtaler*:	
Ledig:	
Versjon:	

Søk:

- ✓ Alle
- ✓ (Ingen verdi)
- ✓ 2 - Konvertert fra EMData
- ✓ 3 - Konvertert fra EMData
- ✓ 16 - Konvertert fra EMData
- ✓ 18 - Konvertert fra EMData
- ✓ 19 - Konvertert fra EMData
- ✓ 20 - Konvertert fra EMData
- ✓ 21 - Konvertert fra EMData
- ✓ 90 -
- ✓ 91 - Programsalg

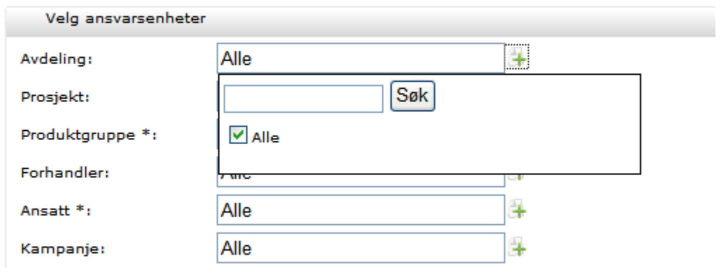
>>

Ajour pr. dato:

Velg intervaller

By adding a static parameter value with label **VR_DynamicSearch** and value =”true” the search functionality is automatically added to the user interface.

As can be seen in the below figure, only a search field and a search button is shown. No values are pre-fetched.



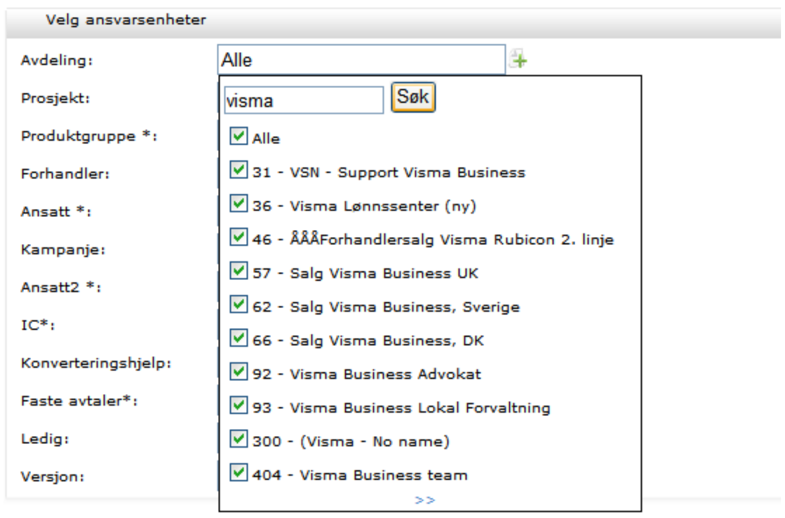
Velg ansvarsenheter

Avdeling:	Alle
Prosjekt:	
Produktgruppe *:	
Forhandler:	
Ansatt *:	Alle
Kampanje:	Alle

Søk

- ✓ Alle

The user then uses the search to narrow the set of returned fields. A smaller result set results in better performance.



Velg ansvarsenheter

Avdeling:	Alle
Prosjekt:	
Produktgruppe *:	
Forhandler:	
Ansatt *:	
Kampanje:	
Ansatt2 *:	
IC*:	
Konverteringshjelp:	
Faste avtaler*:	
Ledig:	
Versjon:	

visma Søk

- ✓ Alle
- ✓ 31 - VSN - Support Visma Business
- ✓ 36 - Visma Lønnssenter (ny)
- ✓ 46 - ÅÅÅForhandlersalg Visma Rubicon 2. linje
- ✓ 57 - Salg Visma Business UK
- ✓ 62 - Salg Visma Business, Sverige
- ✓ 66 - Salg Visma Business, DK
- ✓ 92 - Visma Business Advokat
- ✓ 93 - Visma Business Lokal Forvaltning
- ✓ 300 - (Visma - No name)
- ✓ 404 - Visma Business team

>>

In addition you need to customize your query or dataset to use the search value. Below is an example query for fetching Organization Unit parameter values in Visma Business:

```
SELECT RNo AS Value, Convert(varchar(90), RNo) + ' - ' + Nm AS Label
FROM R2
ORDER BY Value
```

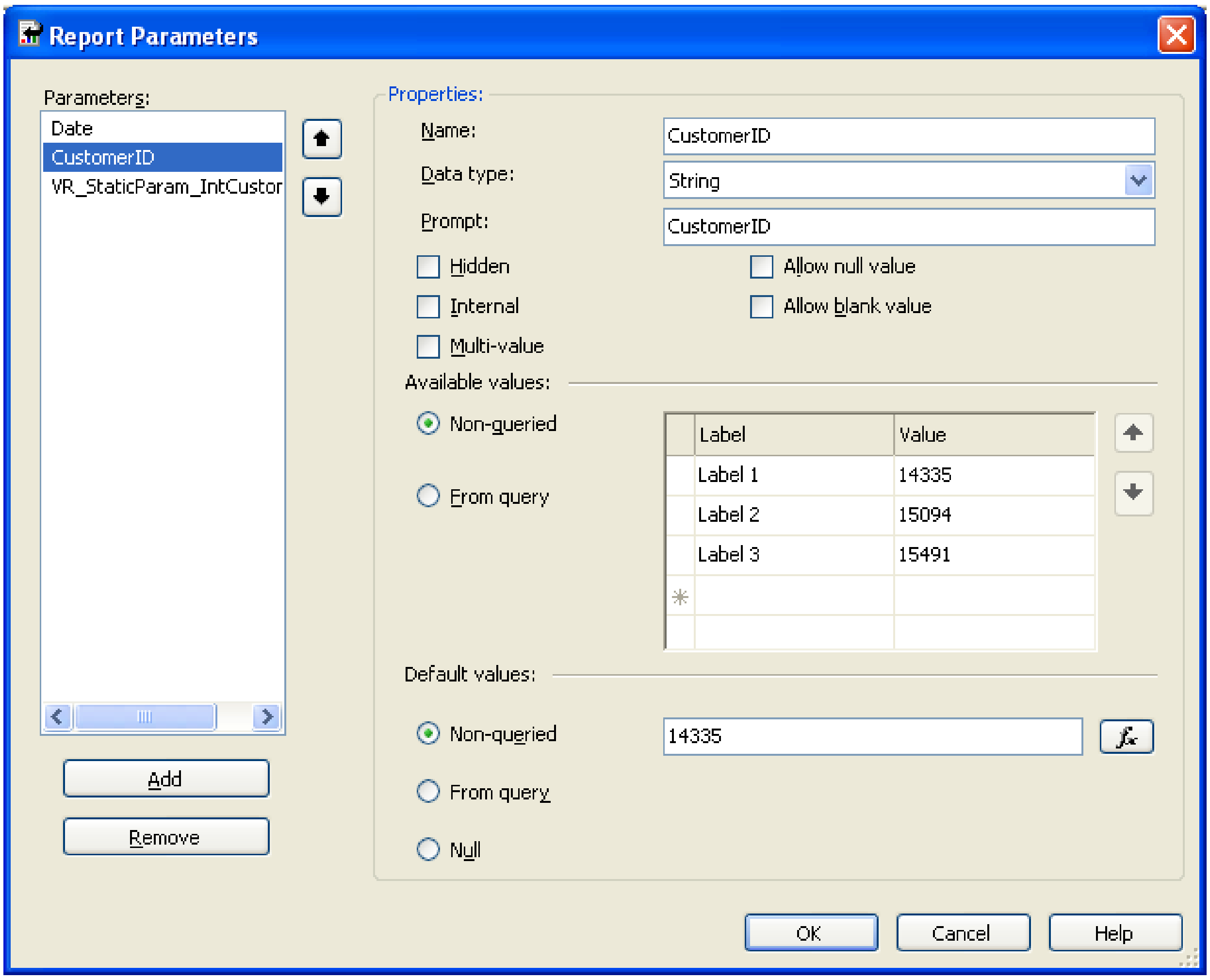
By inserting the tag **/*<DynamicSearchValue>*/** in the query, the tag will automatically be replaced by the search value when values are queried.

```
SELECT RNo AS Value, Convert(varchar(90), RNo) + ' - ' + Nm AS Label
FROM R1
WHERE Nm like '%/*<DynamicSearchValue>*/%'
ORDER BY Value
```

Static Parameter Values

If the set of parameter values are known, i.e. they are part of the parameter definition, the values are shown directly to the user avoiding the search control to be displayed.

Set up the parameter like shown below:



Name: CustomerID
Data type: String
Prompt: CustomerID
[ ] Hidden
[ ] Internal
[ ] Multi-value
[ ] Allow null value
[ ] Allow blank value

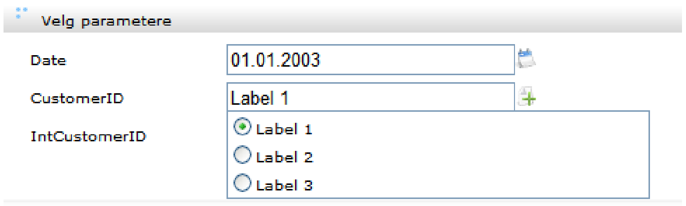
Available values: [✓ Non-queried] From query

Label	Value
Label 1	14335
Label 2	15094
Label 3	15491

Default values: [✓ Non-queried] From query Null

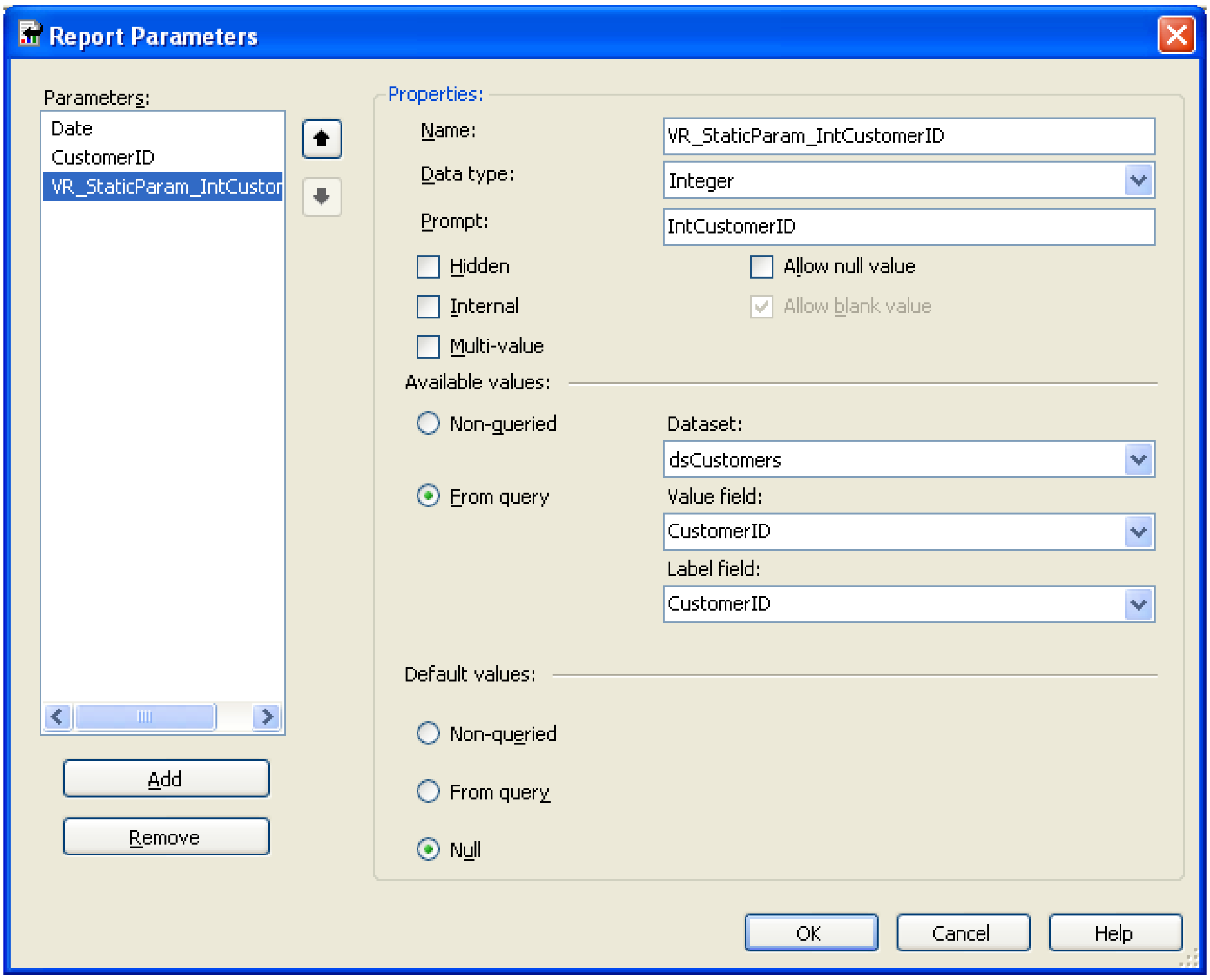
Non-queried: 14335

Result:


Velg parametere

Date	01.01.2003
CustomerID	Label 1
IntCustomerID	[✓ Label 1] Label 2 Label 3

You can also pre-fetch queried values by prefixing the parameter name with **VR_StaticParam_** like shown below. This will give the same result, but we do only recommend this if you know that the query is fast, and that it will not return more values than are viewable in a dropdown like control. Since the values are pre-fetched they affect performance when the report parameters are loaded.


Report Parameters

Parameters: Date, CustomerID, VR_StaticParam_IntCustor

Properties:

Name:	VR_StaticParam_IntCustomerID
Data type:	Integer
Prompt:	IntCustomerID

Hidden [ ] Allow null value [ ]
Internal [ ] Allow blank value [✓]
Multi-value [ ]

Available values: Non-queried [✓ From query]
Dataset: dsCustomers
Value field: CustomerID
Label field: CustomerID

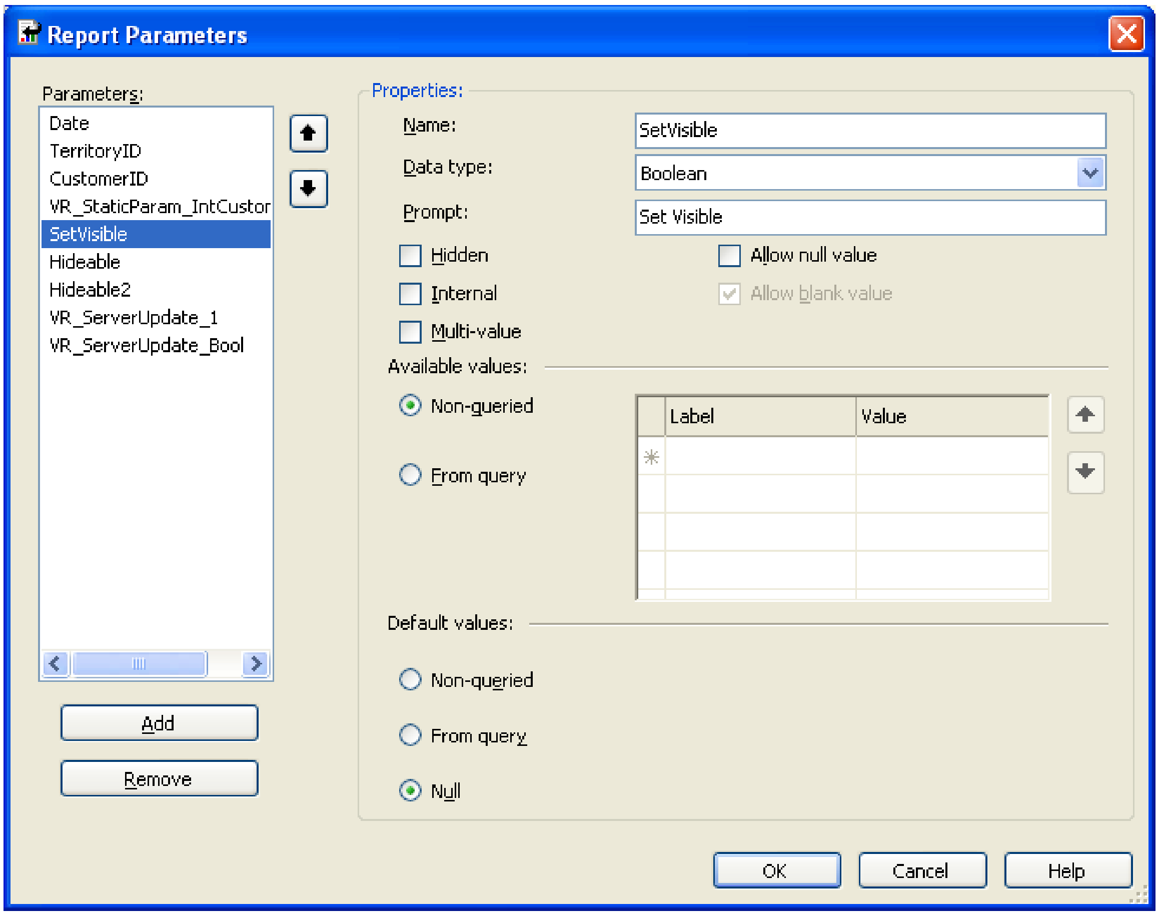
Default values: Non-queried From query [✓ Null]

Parameter Visibility

For some reports it is desired to choose between different sets of parameters, for example view a report based on day or date intervals. By setting a dependency against a *toggle* parameter and telling what value is visible with the VR_Visible parameter value the parameter visibility can be controlled.

The following example has a uses a parameter *SetVisible* to switch between two parameters; *Hideable* and *Hideable2*. Here a checkbox is used to decide. Other types of steering parameter could also be used, but the visibility is controlled with the values "true" and "false".

The toggle:



Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool

Properties:

Name: SetVisible

Data type: Boolean

Prompt: Set Visible

☐ Hidden ☐ Allow null value

☐ Internal ☑ Allow blank value

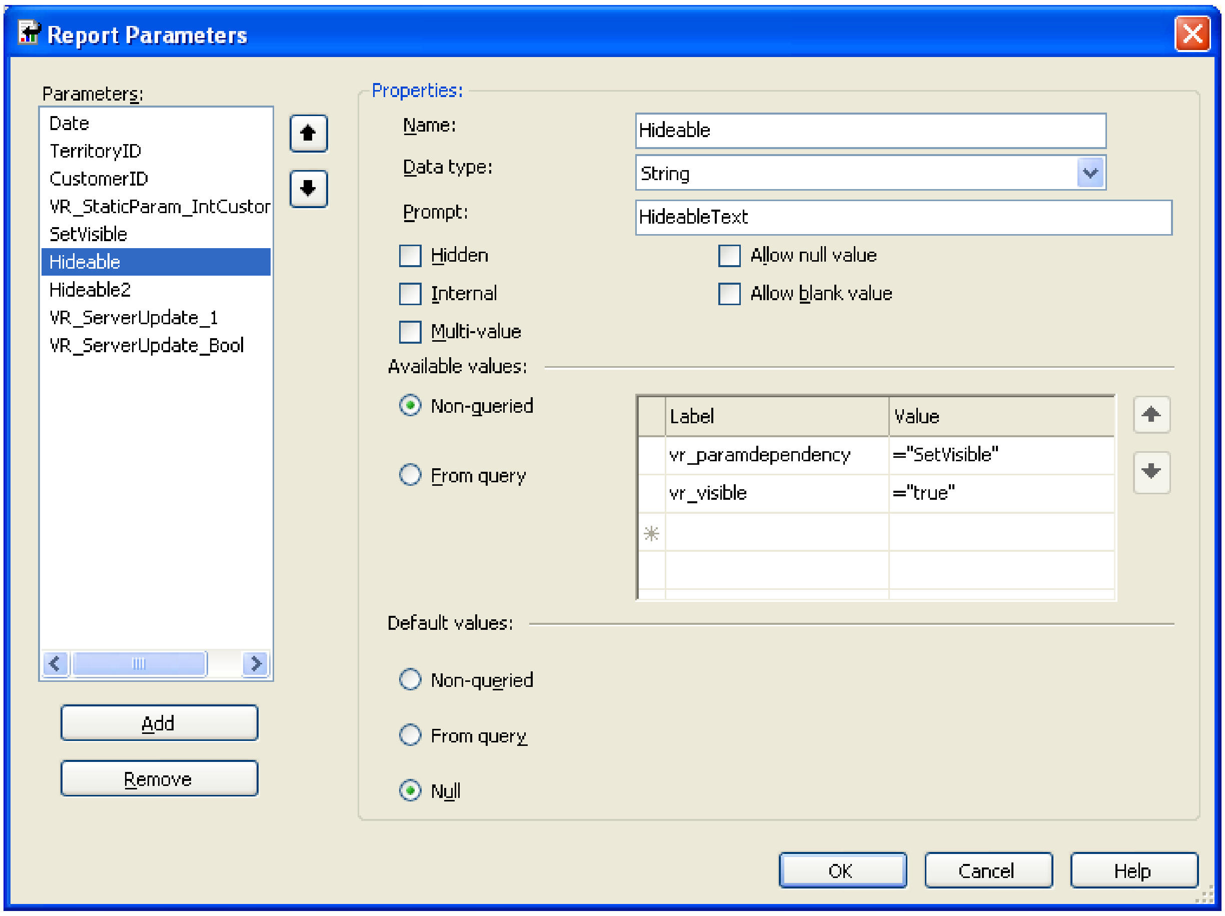
☐ Multi-value

Available values: [✓ Non-queried] From query

Label	Value

Default values: Non-queried From query [✓ Null]

Parameter visible when the toggle value is “true”:



Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool

Properties:

Name: Hideable

Data type: String

Prompt: HideableText

☐ Hidden ☐ Allow null value

☐ Internal ☐ Allow blank value

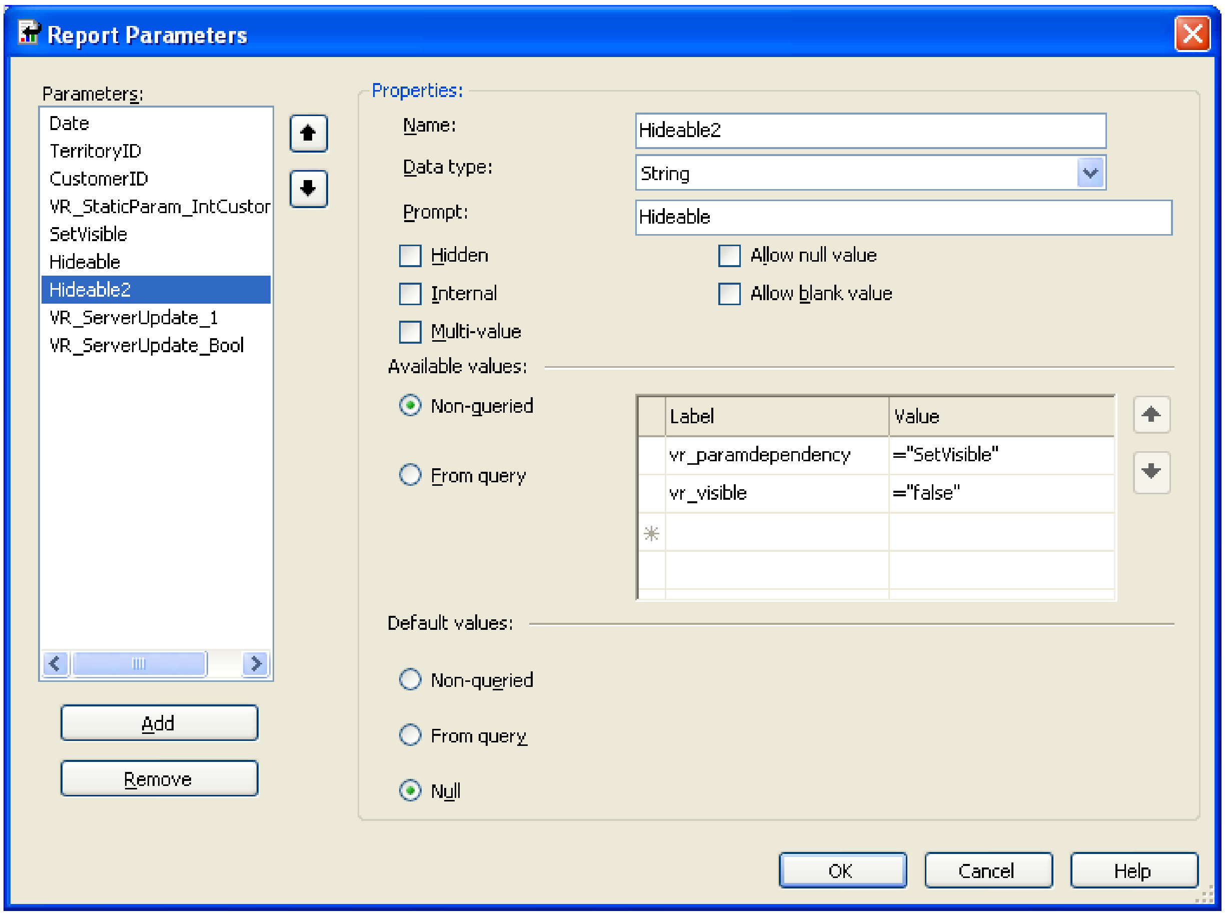
☐ Multi-value

Available values: [✓ Non-queried] From query

Label	Value
vr_paramdependency	="SetVisible"
vr_visible	="true"

Default values: Non-queried From query [✓ Null]

Parameter visible when the toggle value is "false":



Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool

Properties:

Name: Hideable2

Data type: String

Prompt: Hideable

[ ] Hidden [ ] Allow null value

[ ] Internal [ ] Allow blank value

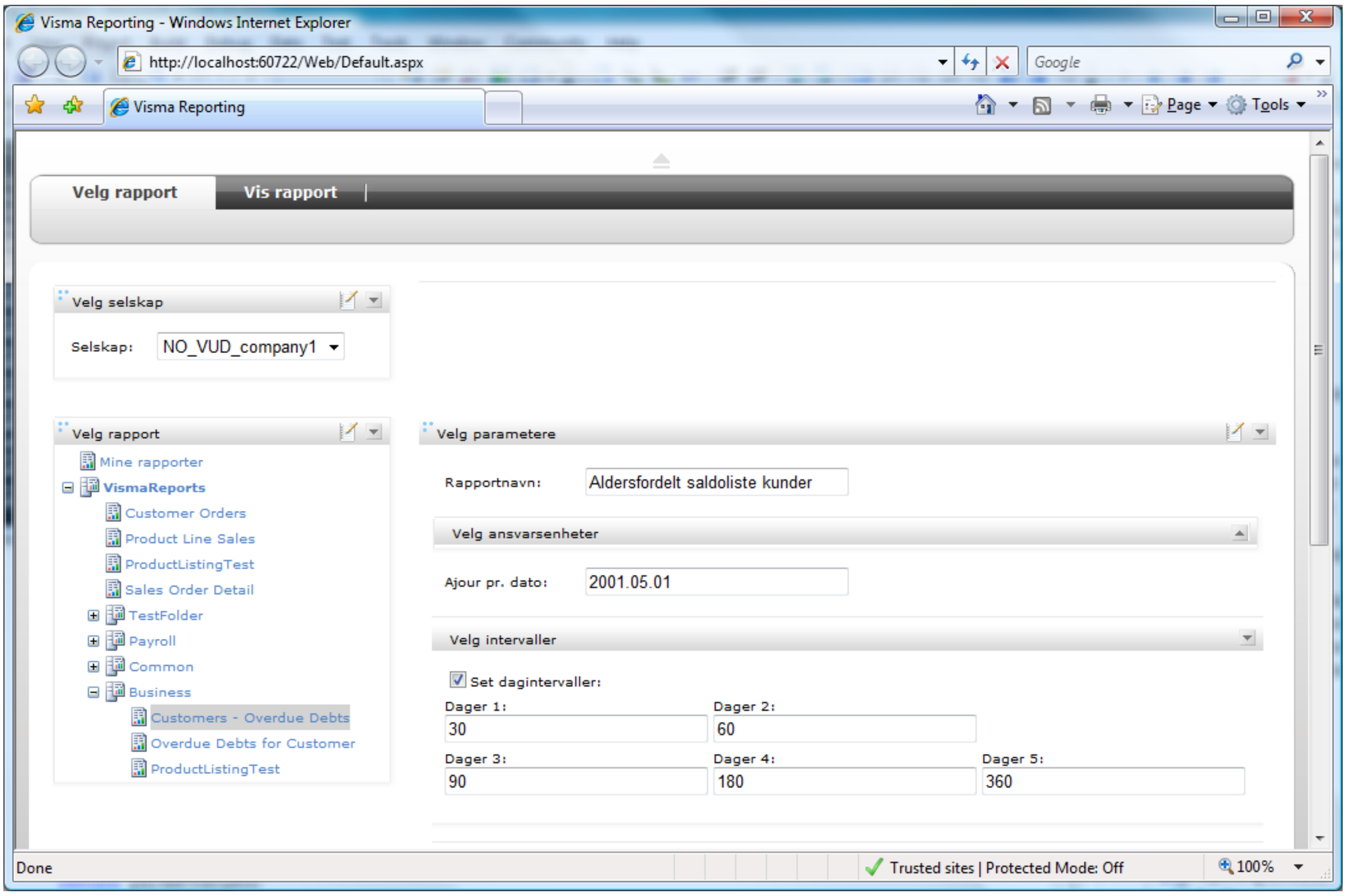
[ ] Multi-value

Available values: [✓ Non-queried] From query

Label	Value
vr_paramdependency	="SetVisible"
vr_visible	="false"

Default values: Non-queried From query [✓ Null]

Below we show the outcome of a real example where the toggle is used to switch between date or day intervals:



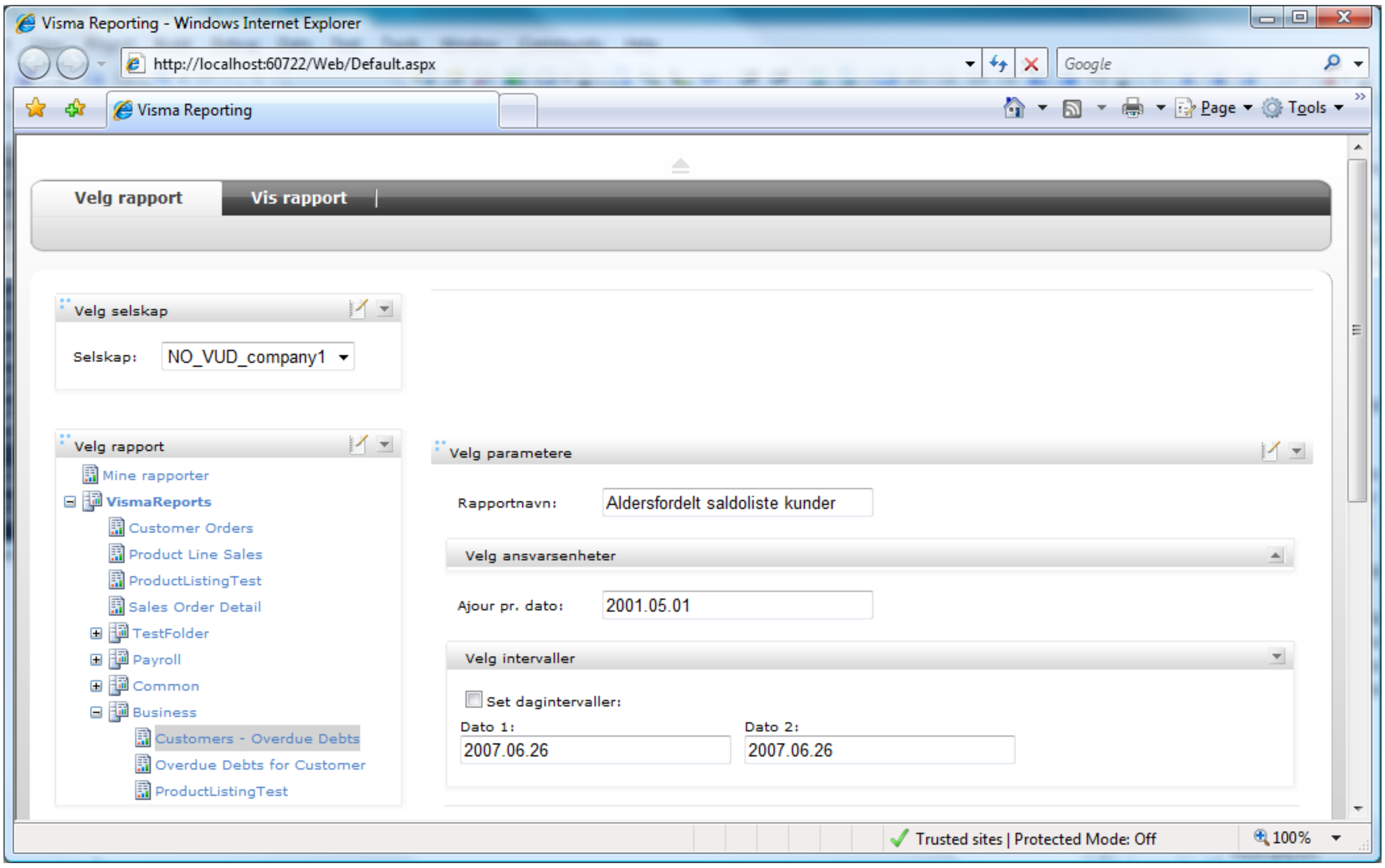
Selskap: NO_VUD_company1

Rapportnavn: Aldersfordelt saldoliste kunder

Ajour pr. dato: 2001.05.01

[✓] Set dagintervaller:

Dager 1:	Dager 2:	
30	60	
Dager 3:	**Dager 4:**	**Dager 5:**
90	180	360



Selskap: NO_VUD_company1

Rapportnavn: Aldersfordelt saldoliste kunder

Ajour pr. dato: 2001.05.01

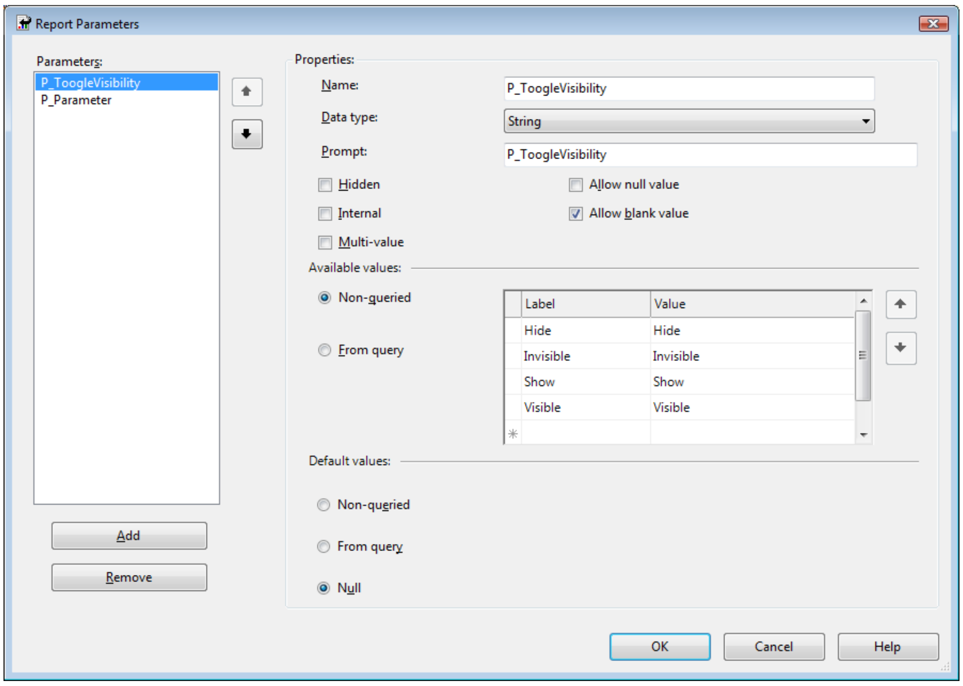
[ ] Set dagintervaller:

Dato 1:	Dato 2:
2007.06.26	2007.06.26

Extended Parameter Visibility

Sometimes you need to control a parameter's visibility based on multiple values from another parameter. The following example makes the parameter *P_Parameter* visible when the user has selected the values *Show* or *Visible* in the parameter *P_ToggleVisibility*. Otherwise, the parameter *P_ Parameter* is hidden.

The toggle control parameter:



Parameters: P_ToogleVisibility, P_Parameter

Properties:

Name: P_ToogleVisibility

Data type: String

Prompt: P_ToogleVisibility

☐ Hidden ☐ Allow null value

☐ Internal ☑ Allow blank value

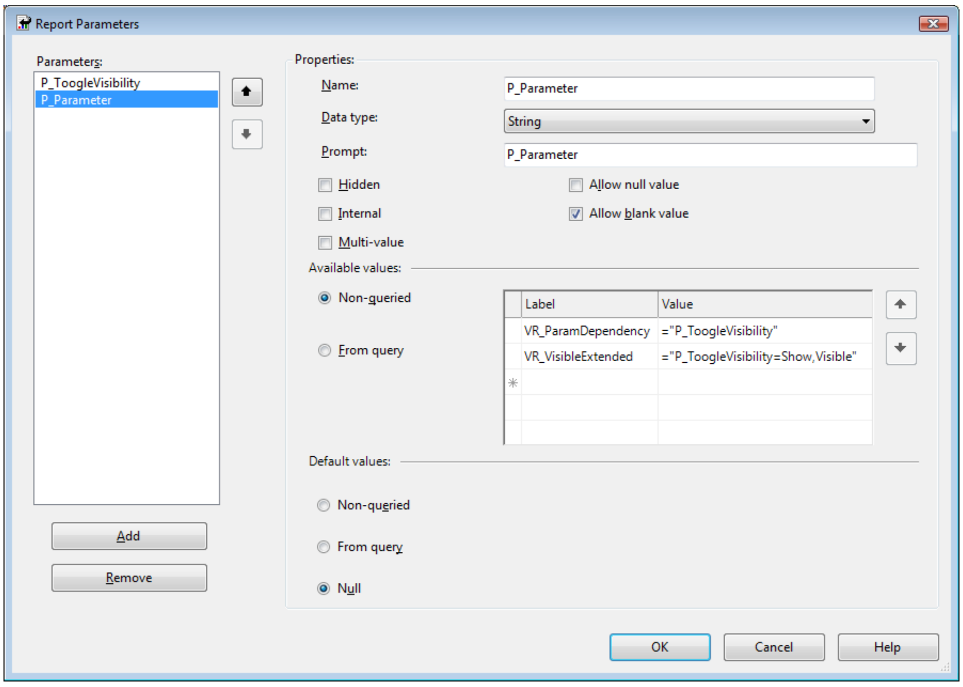
☐ Multi-value

Available values: [✓ Non-queried] From query

Label	Value
Hide	Hide
Invisible	Invisible
Show	Show
Visible	Visible

Default values: Non-queried From query [✓ Null]

The visible/hidden parameter:



Parameters: P_ToogleVisibility, P_Parameter

Properties:

Name: P_Parameter

Data type: String

Prompt: P_Parameter

☐ Hidden ☐ Allow null value

☐ Internal ☑ Allow blank value

☐ Multi-value

Available values: [✓ Non-queried] From query

Label	Value
VR_ParamDependency	="P_ToogleVisibility"
VR_VisibleExtended	="P_ToogleVisibility=Show,Visible"

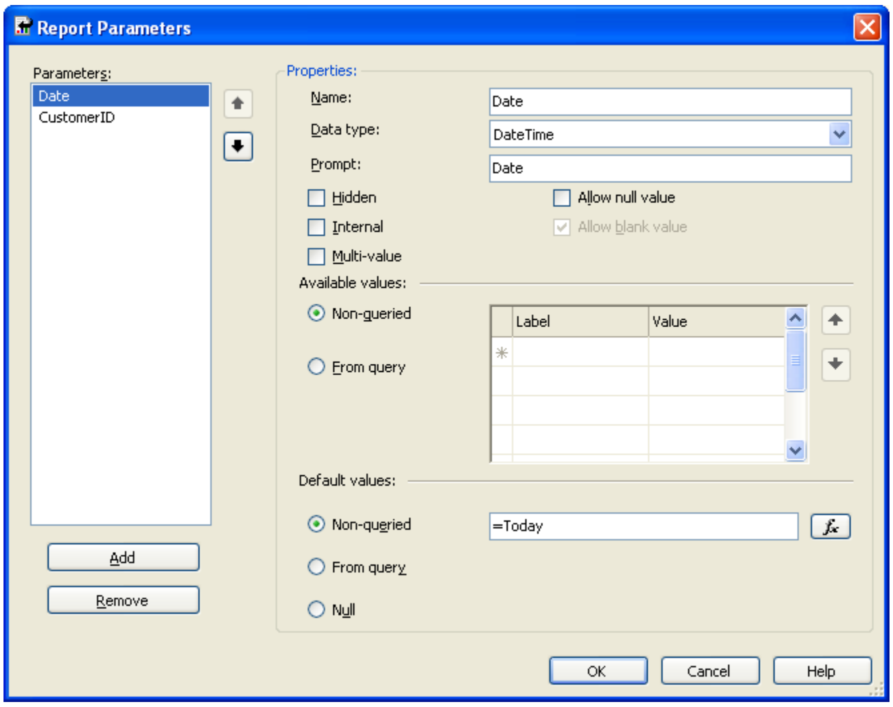
Default values: Non-queried From query [✓ Null]

Default Values

It is possible to use both queried and non-queried values as default value as supported by the Report Designer. In addition it is possible to use formulas and pre-select all values.

Formulas

Most standard formulas are supported. Below is an example of a formula that returns the current date as default value for a date field.



Report Parameters

Parameters:
- Date
- CustomerID

Properties:

Name:	Date
Data type:	DateTime
Prompt:	Date

☐ Hidden ☐ Allow null value
☐ Internal ☑ Allow blank value
☐ Multi-value

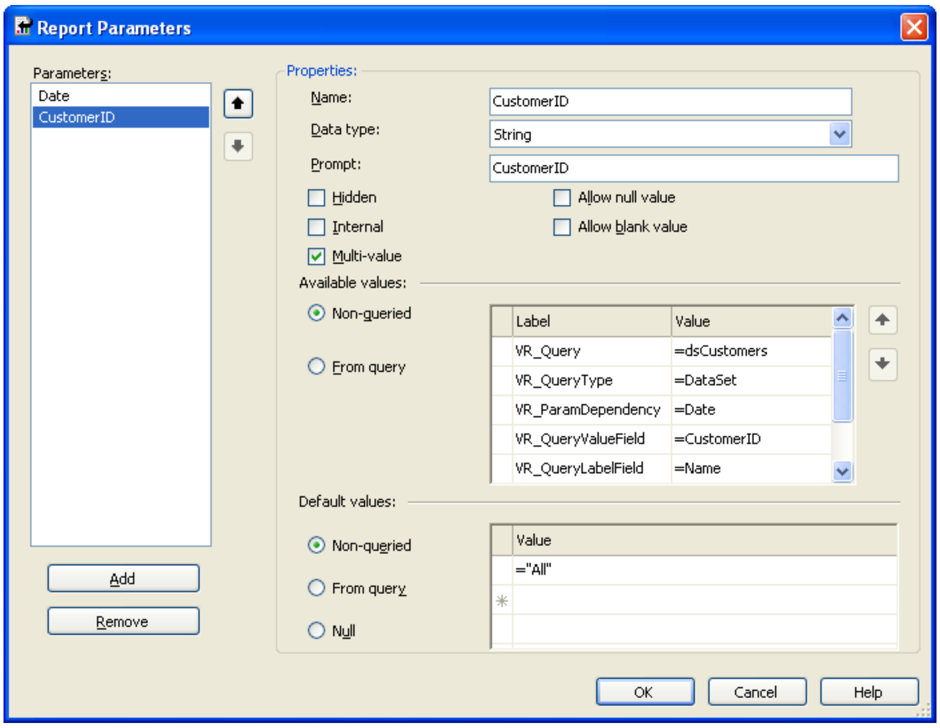
Available values:
(● Non-queried) ○ From query

Label	Value

Default values:
(● Non-queried) =Today
○ From query
○ Null

Select All Values

All values can be pre-selected for a Multi-value parameter by adding the value =”All” as a Non-queried value.



Report Parameters

Parameters:
- Date
- CustomerID

Properties:

Name:	CustomerID
Data type:	String
Prompt:	CustomerID

☐ Hidden ☐ Allow null value
☐ Internal ☐ Allow blank value
☑ Multi-value

Available values:
(● Non-queried) ○ From query

Label	Value
VR_Query	=dsCustomers
VR_QueryType	=DataSet
VR_ParamDependency	=Date
VR_QueryValueField	=CustomerID
VR_QueryLabelField	=Name

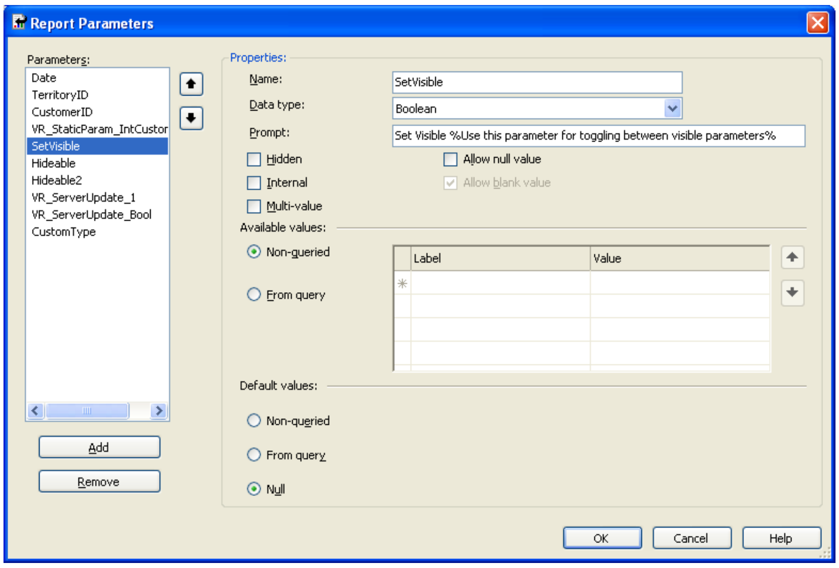
Default values:
(● Non-queried) ○ From query ○ Null

Value
="All"

Parameter Description

You can display additional information on a parameter as a tooltip. This tooltip description can be added to the parameter prompt or as a custom value.

To add the description to the prompt you must enclose the description inside percentage tags, e.g. *%This is my description%*. You can also use translated texts like described in chapter *Translation of Report Texts* (see page 36), e.g. *%#Business!TXT_FromDate#%* . Figure below shows how this is added:


Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool
- CustomType

Properties:

Name:	SetVisible
Data type:	Boolean
Prompt:	Set Visible %Use this parameter for toggling between visible parameters%

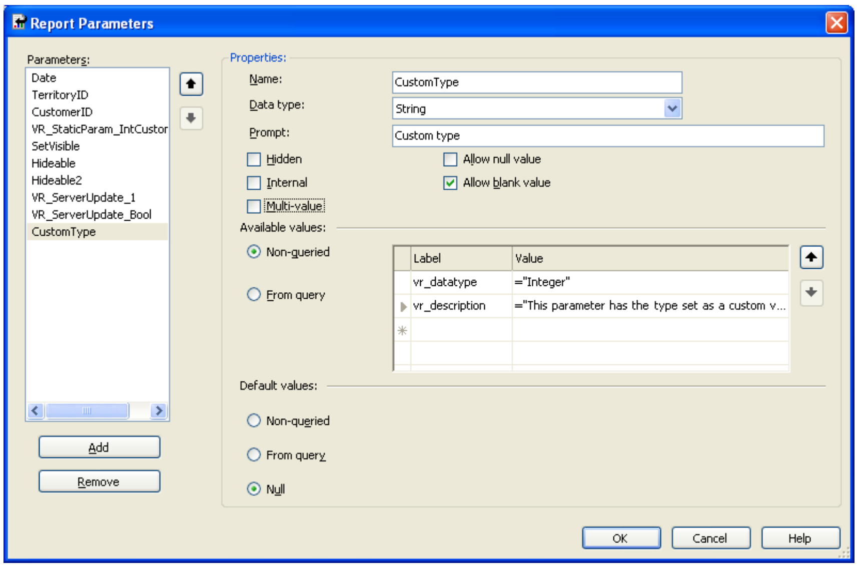
☐ Hidden ☐ Allow null value
☐ Internal ☑ Allow blank value
☐ Multi-value

Available values: (● Non-queried ○ From query)

Label	Value

Default values: ○ Non-queried ○ From query ● Null

To add the description as a custom parameter value, you can add the reserved label **VR_Description** with the description as value, as shown below:


Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool
- CustomType

Properties:

Name:	CustomType
Data type:	String
Prompt:	Custom type

☐ Hidden ☐ Allow null value
☐ Internal ☑ Allow blank value
☐ Multi-value

Available values: (● Non-queried ○ From query)

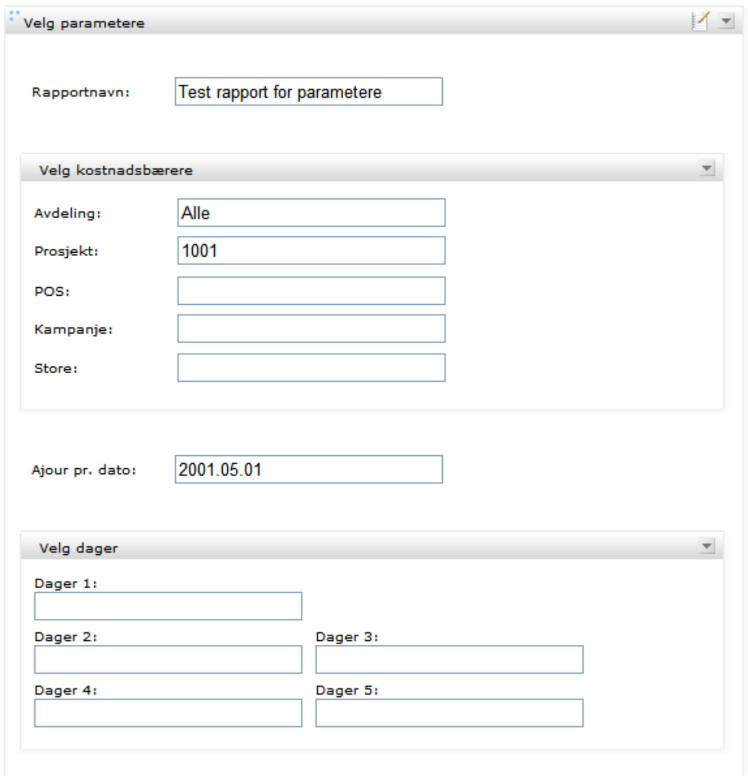
Label	Value
vr_datatype	="Integer"
vr_description	="This parameter has the type set as a custom v...

Default values: ○ Non-queried ○ From query ● Null

Parameter Grouping

It is possible to make parameters appear as grouped in the user interface. This can be seen in the figure below. Cost units and dates are grouped below, while the report name and the update date are not belonging to any group.

The cost units group uses sequential layout, while the date group uses grid layout.



Velg parametere

Rapportnavn: Test rapport for parametere

Velg kostnadsbærere

Avdeling: Alle

Prosjekt: 1001

POS:

Kampanje:

Store:

Ajour pr. dato: 2001.05.01

Velg dager

Dager 1:

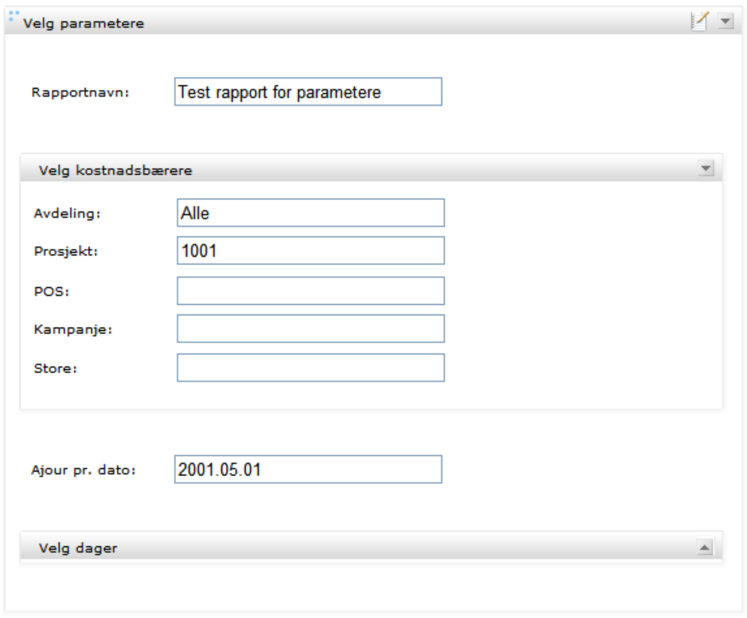
Dager 2:

Dager 3:

Dager 4:

Dager 5:

The parameter groups can be minimized as shown for the date group below. It is also possible to set a group as initially minimized. This is described below.



Velg parametere

Rapportnavn: Test rapport for parametere

Velg kostnadsbærere

Avdeling: Alle

Prosjekt: 1001

POS:

Kampanje:

Store:

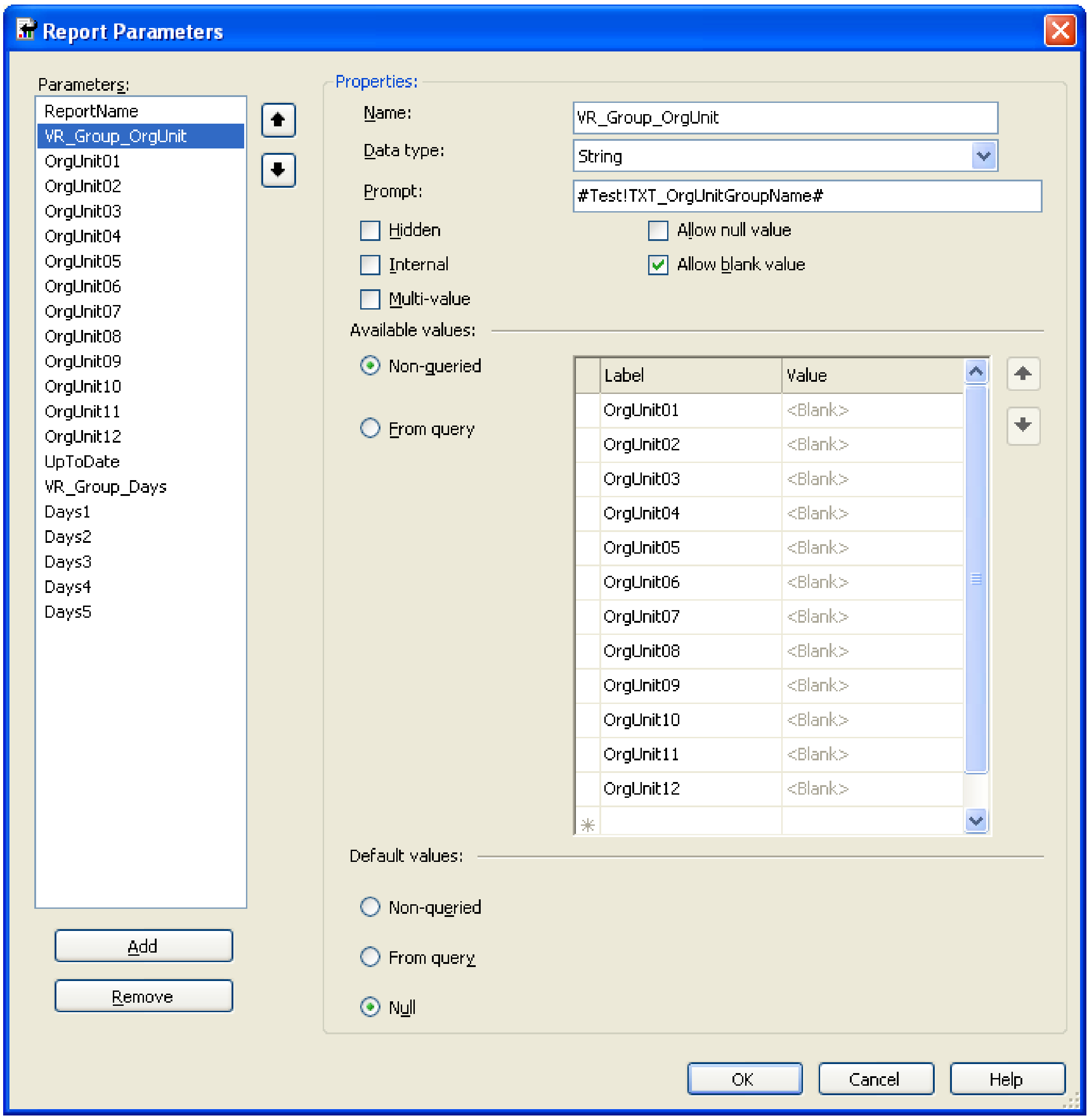
Ajour pr. dato: 2001.05.01

Velg dager

We use tagged parameters to group other parameters. Parameters named with VR_Group_ as prefix are treated as group parameters. The following creates a group:

1. Create a parameter with VR_Group_ name prefix
2. Move the parameter up the list to get the right group sequence. In the example below the org unit group is put after the not grouped ReportName parameter, but before the not grouped UpToDate parameter
3. Give the parameter group a name in the Parameter Prompt field. This is the name as shown with the group. It is possible to use translated texts by using the hash notation as described in chapter *Translation of Report Texts* (see page 36)

4. List the names of all parameters belonging to this group as values as shown below. Here OrgUnit01 to OrgUnit12 are grouped



Parameters: ReportName, VR_Group_OrgUnit, OrgUnit01, OrgUnit02, OrgUnit03, OrgUnit04, OrgUnit05, OrgUnit06, OrgUnit07, OrgUnit08, OrgUnit09, OrgUnit10, OrgUnit11, OrgUnit12, UpToDate, VR_Group_Days, Days1, Days2, Days3, Days4, Days5

Name: VR_Group_OrgUnit
Data type: String
Prompt: #Test!TXT_OrgUnitGroupName#

[ ] Hidden
[ ] Internal
[ ] Multi-value
[ ] Allow null value
[✓ Allow blank value]

Available values: [✓ Non-queried] From query

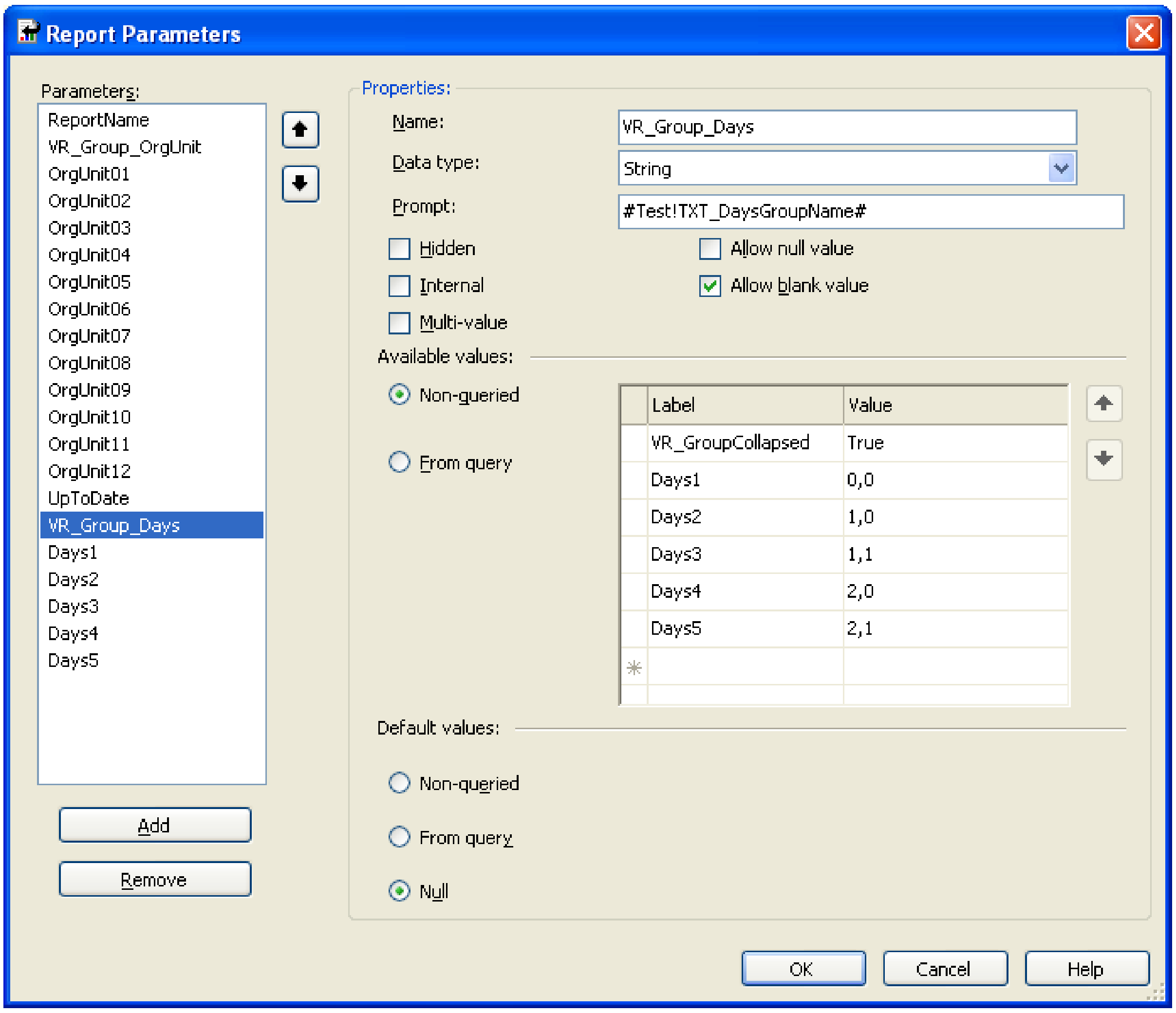
Label	Value
OrgUnit01	<Blank>
OrgUnit02	<Blank>
OrgUnit03	<Blank>
OrgUnit04	<Blank>
OrgUnit05	<Blank>
OrgUnit06	<Blank>
OrgUnit07	<Blank>
OrgUnit08	<Blank>
OrgUnit09	<Blank>
OrgUnit10	<Blank>
OrgUnit11	<Blank>
OrgUnit12	<Blank>

Default values: Non-queried From query [✓ Null]

The figure below has a flow layout of the grouped parameters: They are appearing as a list of parameters. It is also possible to set up the parameters in a grid layout. This is shown in the figure below for the Dates group.

The parameter name labels have a value indicating position. The syntax is **ROW,COLUMN** starting with a zero indexed row and column number. Below we have three rows and two columns.

To set a group as collapsed when the parameter list is first shown, add a reserved parameter value **VR_GroupCollapsed** with a value of true.



Parameters: ReportName, VR_Group_OrgUnit, OrgUnit01, OrgUnit02, OrgUnit03, OrgUnit04, OrgUnit05, OrgUnit06, OrgUnit07, OrgUnit08, OrgUnit09, OrgUnit10, OrgUnit11, OrgUnit12, UpToDate, VR_Group_Days, Days1, Days2, Days3, Days4, Days5

Properties:

Name: VR_Group_Days

Data type: String

Prompt: #Test!TXT_DaysGroupName#

[ ] Hidden [ ] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

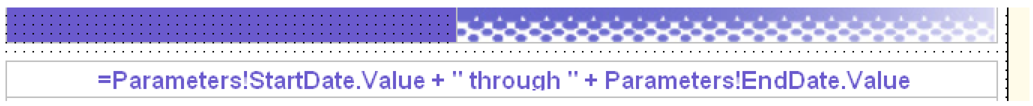
Label	Value
VR_GroupCollapsed	True
Days1	0,0
Days2	1,0
Days3	1,1
Days4	2,0
Days5	2,1

Default values: Non-queried From query [✓ Null]

Display Selected Values

About this task

A common report task is to show a selected parameter **Value** in the report. This works by just using the parameter directly in the report:

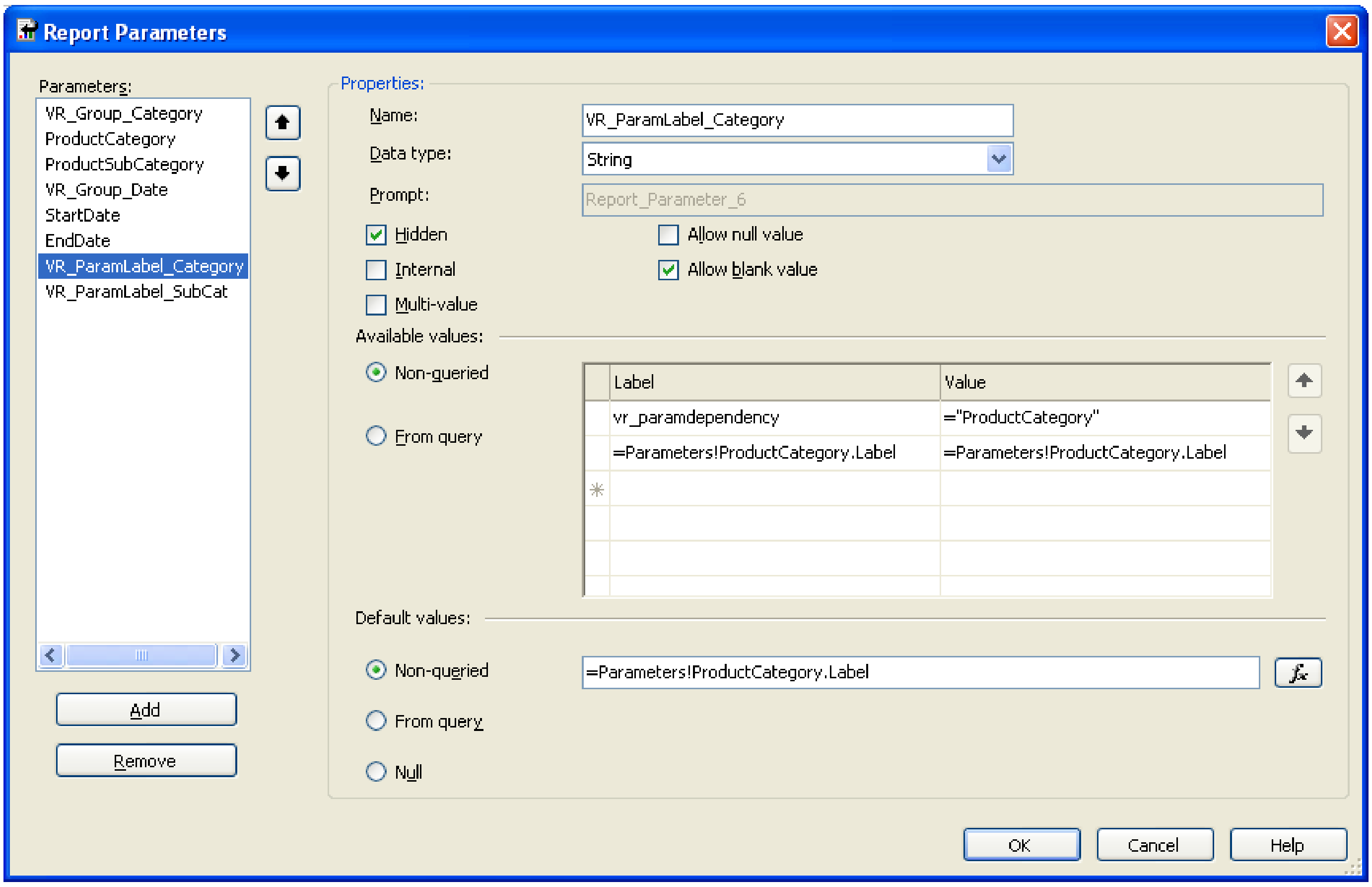


```
=Parameters!StartDate.Value + " through " + Parameters!EndDate.Value
```

But, for some parameters one would like to show the **Label** and not the **Value**. This can be done as described above in the Microsoft Report Designer, but to show it correctly in the report you must do the following:

Procedure

1. Create a label parameter to hold the label. The parameter name must be prefixed with VR_ParamLabel and have type of Text. It is also possible to add a default value to make the selected label appear in the designer as well. You link the label parameter to the original parameter by setting VR_ParamDependency to the parameter. If the original parameter is multi-value and multiple values are selected, the label parameter will contain all the selected values concatenated with comma. Hence, you should **not** define the VR_ParamLabel parameter as multi-value or use Join to concatenate values.



Parameters:
- VR_Group_Category
- ProductCategory
- ProductSubCategory
- VR_Group_Date
- StartDate
- EndDate
- VR_ParamLabel_Category
- VR_ParamLabel_SubCat

Properties:

Name: VR_ParamLabel_Category

Data type: String

Prompt: Report_Parameter_6

[✓] Hidden [ ] Allow null value

[ ] Internal [✓] Allow blank value

[ ] Multi-value

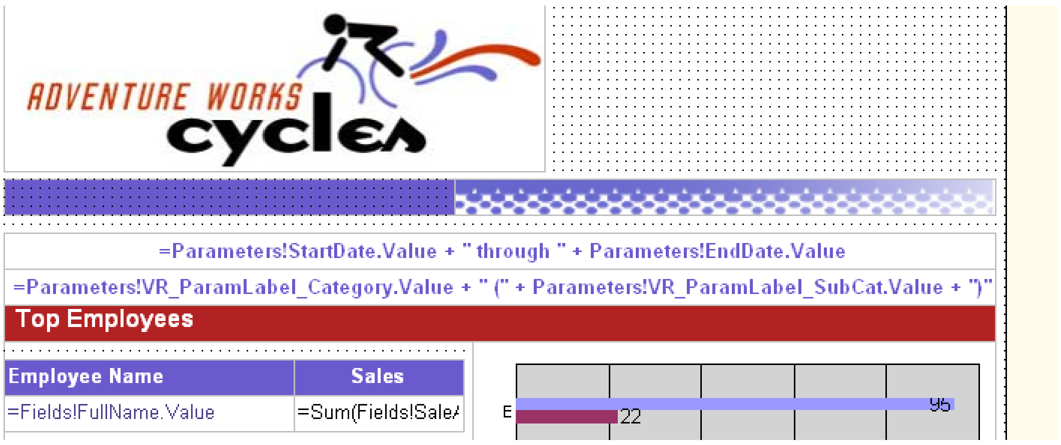
Available values: [✓ Non-queried] From query

Label	Value
vr_paramdependency	="ProductCategory"
=Parameters!ProductCategory.Label	=Parameters!ProductCategory.Label

Default values: [✓ Non-queried] From query Null

=Parameters!ProductCategory.Label

2. Use the Value of the label parameter instead of the Label of the original parameter in the textbox :=Parameters!VR_ParamLabel_Category.Value



=Parameters!StartDate.Value + " through " + Parameters!EndDate.Value

=Parameters!VR_ParamLabel_Category.Value + " (" + Parameters!VR_ParamLabel_SubCat.Value + ")"

=Fields!FullName.Value

=Sum(Fields!Sale/

Visualize Report Filter

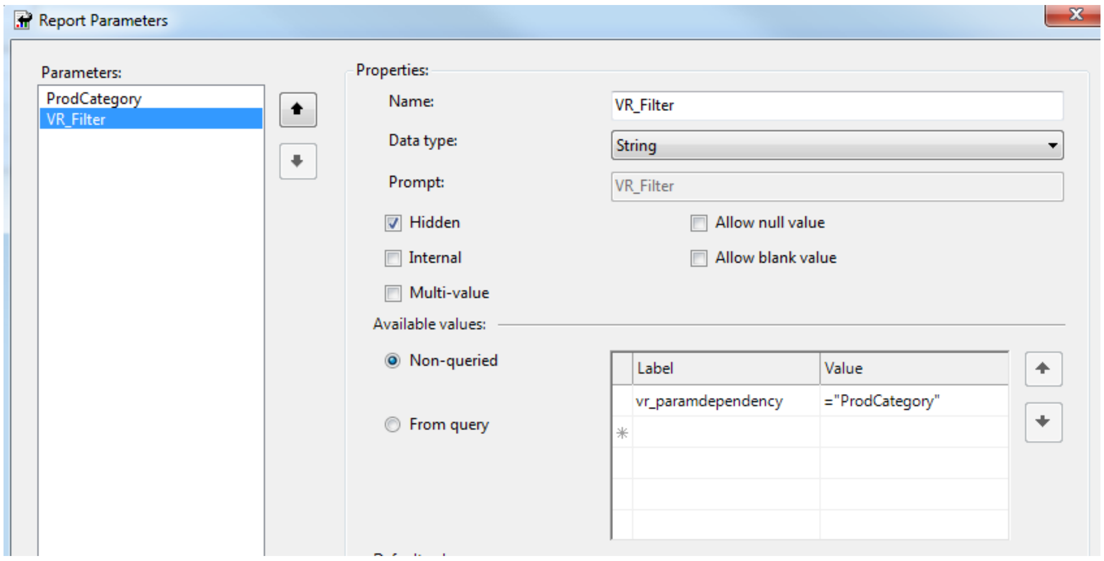
About this task

It is often wanted to show the report filter on the report itself. A parameter named VR_Filter is available for this use.

Procedure

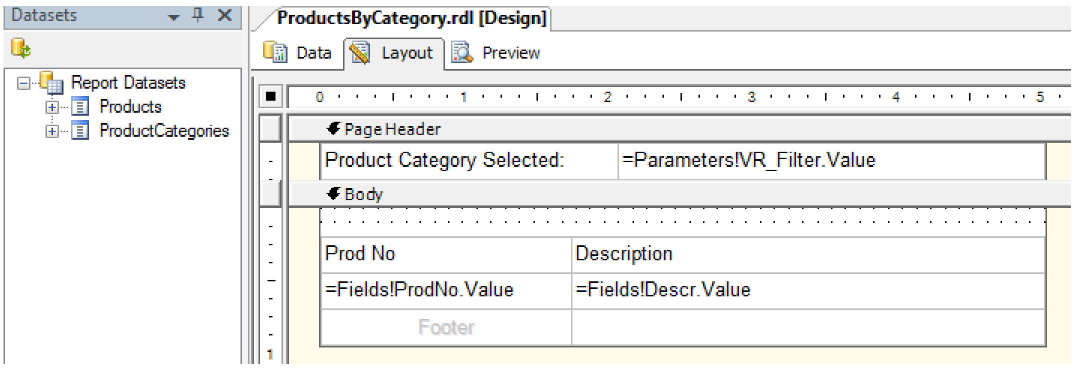
1. Add it to the report. In VR_Filter's Available values table set the label field equal to property VR_ParamDependency and value =”[Parameters]”. [Parameters] is a comma separated list of the names

of the parameters you want to include in the filter. This way you can control which parameters are to be shown in the VR_Filter parameter



Label	Value
vr_paramdependency	="ProdCategory"

2. From a textbox in the report set value to =Parameters!VR_Filter.Value



Product Category Selected:	=Parameters!VR_Filter.Value

Prod No	Description
=Fields!ProdNo.Value	=Fields!Descr.Value

3. On execution of report you will see the desired report filter in the textbox

Results

Note: Beside the "ProdCategory" parameter's DataSet (in this example the ProductCategories dataset) you have to have at least one more DataSet in the report, in order for the VR_Filter to print the filter on the report.

User, Company and Language Parameters

Logged on user

The parameter VR_User shows the username of the logged on user. For instance, set parameter VR_User to hidden and use it in a report SQL statement. It is interesting when wanting to do an SQL on user/customer related data in the database.

Logged on user id

The parameter VR_UserId shows the user id of the logged on user. For instance, set parameter VR_UserId to hidden and use it in a report SQL statement. It is interesting when wanting to do an SQL on user/customer related data in the database.

Logged on user alias

The parameter VR_Alias shows the alias property of the logged on user from Visma User Directory. VR_Alias parameter need to be set as hidden and in case multiple aliases are present it will only display the first one.

Company name

The parameter VR_CompanyName shows the name of the active company. For instance, set parameter VR_CompanyName to hidden and use it in a report SQL statement, or display it in the report.

Company id

The parameter VR_CompanyId shows the id of the active company. For instance, set parameter VR_CompanyId to hidden and use it in a report SQL statement, or display it in the report.

Current language or culture

The parameter VR_Language shows the current language/culture the report is running under. This is based on the language selected by the user. The format is <language>_<country>. For instance nb-NO or en-US.

Parameter Value Cache Cross Report

It is possible to cache selected parameter values between reports.

For instance if you have a Department parameter which you always set with the same value in a large set of reports Visma Reporting will reuse the value in all reports used during a session. E.g. you set the department parameter to “IT department” in the first report. When you open another report also using the Department parameter, the value is already populated with “IT department” and you do not need to set it again.

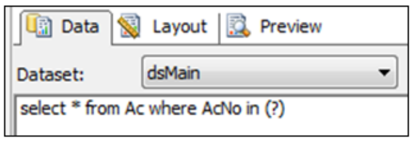
For Visma Reporting to understand that this is a parameter you want to reuse in several reports you need to prefix the parameter name with “VR_”. This means that if you are using a parameter named VR_Department in 2 or more reports the selected parameter value is reused between the reports.

Datasets Used in Parameters Only (performance options)

If you know that a dataset is used to fetch available values for parameter(s) only, you can prefix the dataset name with " VR_ParameterOnly_" to tell Visma Reporting not to populate the dataset when running the report. The dataset is populated only when loading the report parameters. This will result in a performance improvement in cases where the parameter dataset(s) is loaded with large amount of data.

Dataset Named Parameters

Let us say that your query for a dataset looks like this:

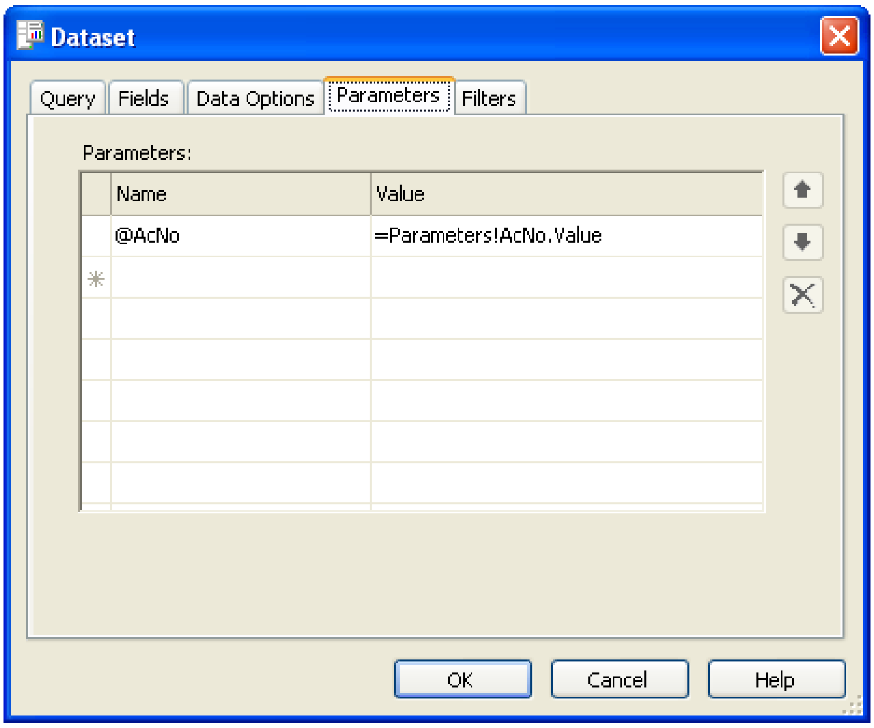


Dataset: dsMain

```
select * from Ac where AcNo in (?)
```

@AcNo is a named parameter for this dataset, and by default it is linked to the report parameter ‘Parameters! AcNo.Value’. If this report parameter does not exist, it will be created. However, you can link the dataset named parameter to any report parameter by clicking the ... button on the right of the dataset combobox and choose the ’Parameters’ tab in the dialog that appears. The value must be on the format ’=Parameters!<ReportParameter>.Value’.

Note: Do not use any other expressions. It is not supported.



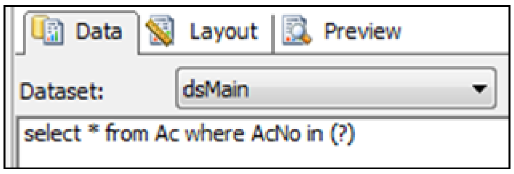
Parameters:

Name	Value
@AcNo	=Parameters!AcNo.Value

Of course, it is not allowed to declare local variables in the query with the same name as any dataset named parameter.

Dataset Unnamed Parameters

OLEDB and ODBC database drivers do not support named parameters in the dataset. In these cases you will have to use unnamed parameters where question marks ('?') in the sql is representing the parameters. Let us say that your query for a dataset looks like this:



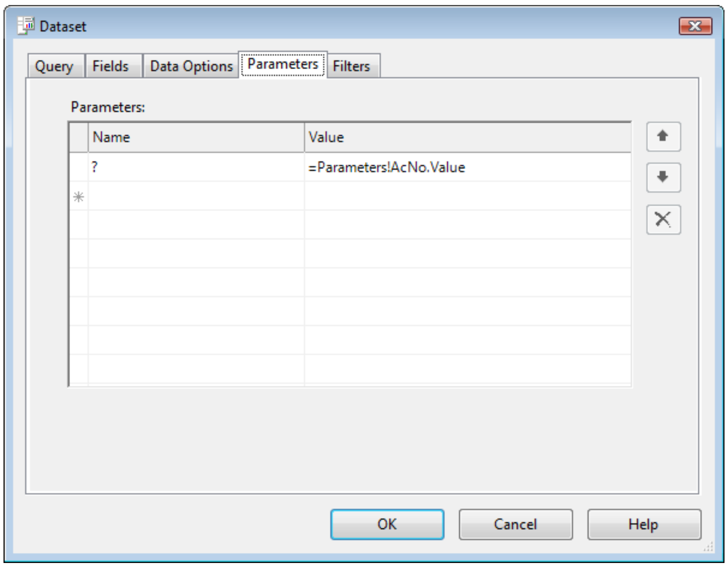
Dataset: dsMain

```
select * from Ac where AcNo in (?)
```

? is an unnamed parameter for this dataset, and by default it is linked to the report parameter ‘Parameters! Parameter1.Value’. If this report parameter does not exist, it will be created.

However, you can link the dataset named parameter to any report parameter by clicking the ... button on the right of the dataset combobox and choose the ’Parameters’ tab in the dialog that appears. The value must be on the format '=Parameters!<ReportParameter>.Value'.

Note: Do not use any other expressions. It is not supported.


Dataset

Query | Fields | Data Options | Parameters | Filters

Parameters:

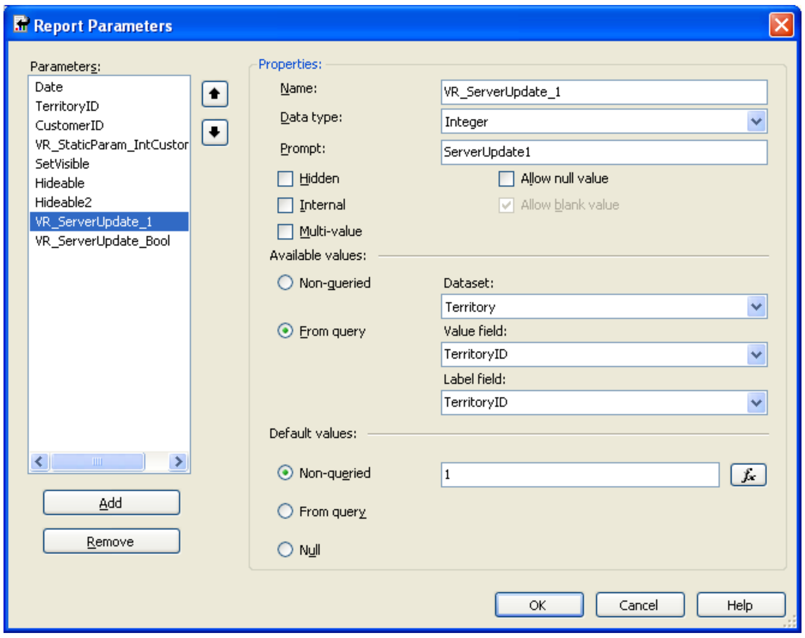
Name	Value
?	=Parameters!AcNo.Value

OK | Cancel | Help

Note: You cannot assign a report parameter to several unnamed parameters. Each unnamed parameter must be assigned to a unique report parameter. The assignments also need to be in the same order in which the unnamed parameters occur in the dataset query.

Forcing Server Update

Some parameters require that other parameters are recalculated server side when their value changes. This server side recalculation typically occurs in a product specific provider. The forcing of server update has been implemented for the default data provider also. To make a parameter force a server update you must prefix the parameter name with **VR_ServerUpdate_** like in the two examples shown below:

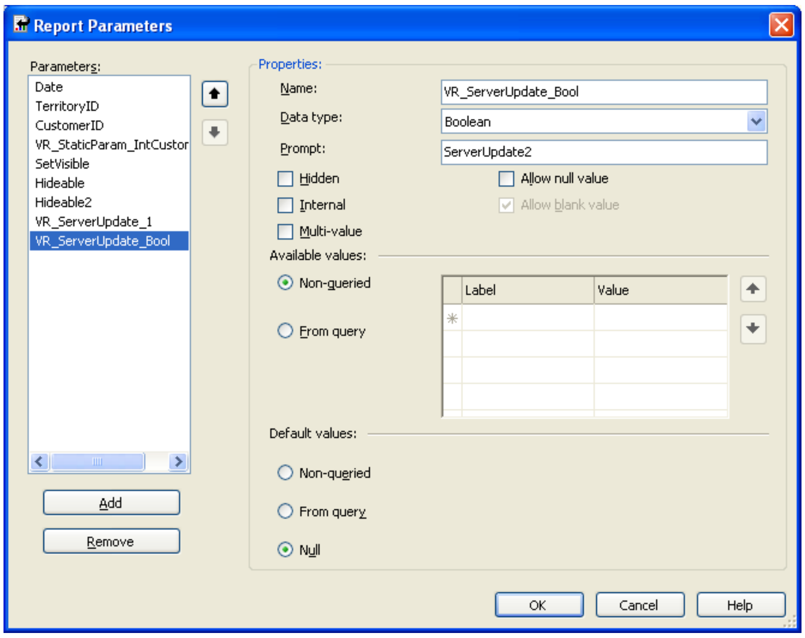

Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool

Properties:
Name: VR_ServerUpdate_1
Data type: Integer
Prompt: ServerUpdate1
Hidden [ ] Allow null value [ ]
Internal [ ] Allow blank value [✓]
Multi-value [ ]

Available values:
Non-queried [✓ From query]
Dataset: Territory
Value field: TerritoryID
Label field: TerritoryID

Default values:
[✓ Non-queried] 1
From query
Null


Report Parameters

Parameters:
- Date
- TerritoryID
- CustomerID
- VR_StaticParam_IntCustor
- SetVisible
- Hideable
- Hideable2
- VR_ServerUpdate_1
- VR_ServerUpdate_Bool

Properties:

Name: VR_ServerUpdate_Bool
Data type: Boolean
Prompt: ServerUpdate2

Hidden, Internal, Multi-value, Allow null value: unchecked; Allow blank value: checked (disabled)

Available values: [✓ Non-queried] From query

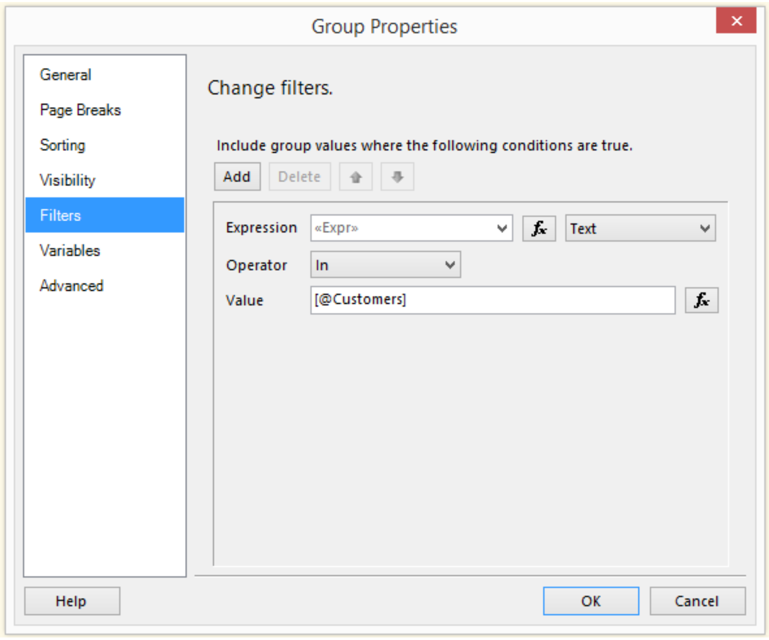
Label	Value

Default values: Non-queried From query [✓ Null]

Filtering report data using group filters and multi-value parameters

The suggested approach for filtering the information displayed in a report applies to filtering inside SQL queries. Filtering inside queries improves the performance of the report.

Sometimes it is needed to do this filtering in the report based on the value of a parameter. This parameter can be single value or multi-value and the filtering can be done by adding a new expression in the “Filters” menu option of a given Group.

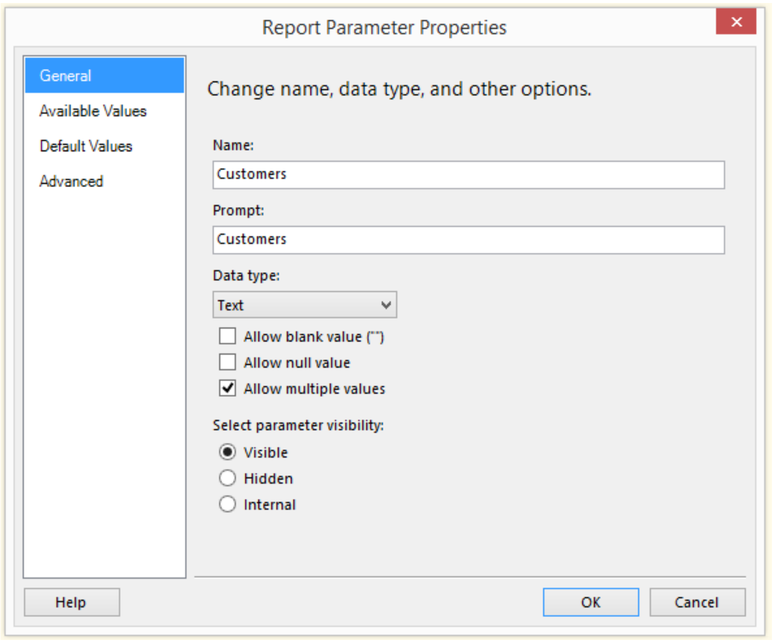

Group Properties

Change filters.

Include group values where the following conditions are true.

Expression: «Expr» Text
Operator: In
Value: [@Customers]

The parameter defined is of type "Text" like the one below.


Report Parameter Properties

Change name, data type, and other options.

Name: Customers

Prompt: Customers

Data type: Text

- [ ] Allow blank value ("")
- [ ] Allow null value
- [x] Allow multiple values

Select parameter visibility:

- ● Visible
- ○ Hidden
- ○ Internal

This filtering works fine inside Visma Reporting for both single and multi value parameters.

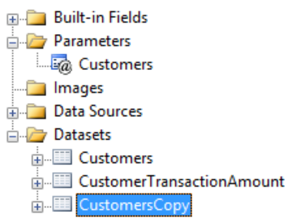
The problem appears when selecting the "All" value for a parameter. In this case the filtering does not return anything.

After processing the data sets of the parameter and the report, Visma Reporting does not send back to the viewer all parameter values, instead of "All" the viewer receives a space " ".

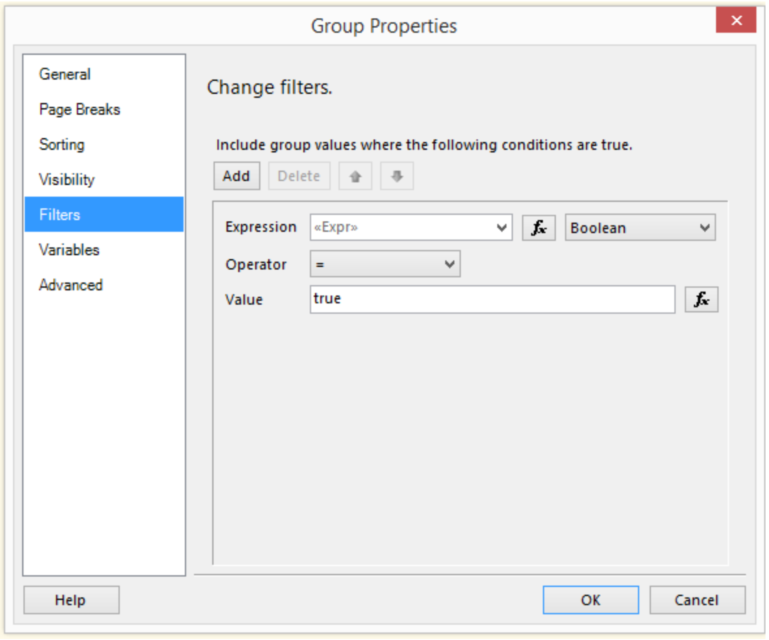
In order to have a working scenario you will need a more complex expression inside the grouping filter.

Below, a step by step example for this scenario:

1. Create a copy of the data set of the parameter and name it differently. This is needed by the Lookup function that will be used inside the expression. The parameter dataset cannot be used by this lookup due to performance reasons; parameter datasets are not send back to the viewer.


```
Built-in Fields
Parameters
    Customers
Images
Data Sources
Datasets
    Customers
    CustomerTransactionAmount
    CustomersCopy
```

2. Open the grouping where you want to add the filtering and create a new filter like the one below.


Group Properties

Change filters.

Include group values where the following conditions are true.

Expression «Expr» Boolean

Operator =

Value true

3. The value of the expression is the following:

   = IIF (Join (Parameters!Customers.Value, ",") = " ", IIF (Lookup (Fields!CustNo.Value,Fields!CustNo.Value,Fields!Nm.Value,"CustomersCopy")= Nothing, false, true), IIF(Array.IndexOf (Parameters!Customers.Value, Cstr (Fields!CustNo.Value)) > -1, true, false))

   The expression first checks if “All” has been selected.

   If this is the case then searches for the current customer number inside the “CustomersCopy” data set. If there is no customer number then returns false - the current line will not be displayed, otherwise display the line.

   If “All” = “ “ has not been selected then just search the customer number between all the values of the customer parameter and display the line accordingly.

Report Scaling

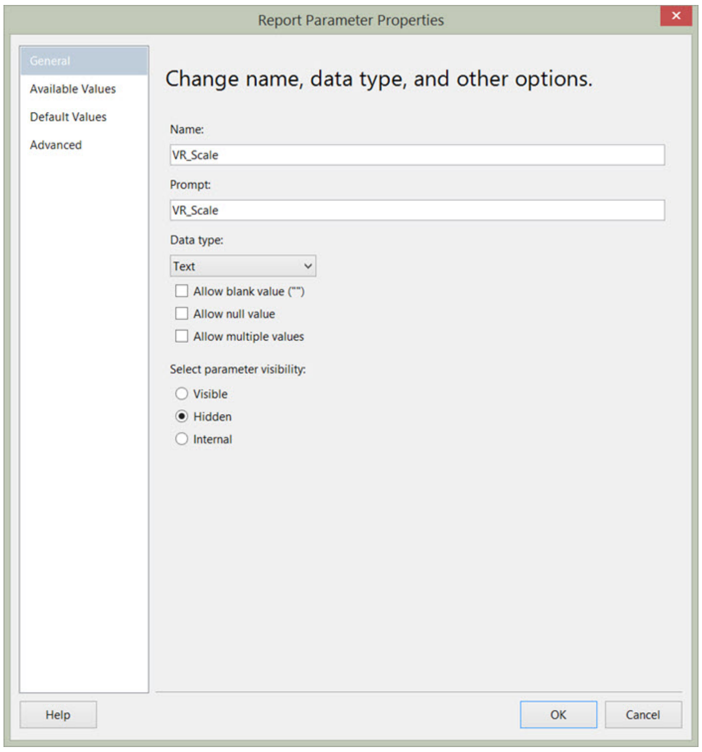
A flexible scaling of the report can be achieved by using the VR_Scale parameter. This is useful especially in case reports need to be designed for different screen sizes.

The available options are:

1. PageWidth – the report is scaled to fit the horizontal screen space
2. PageHeight – the report is scaled to fit the vertical screen space
3. Fit – the report is scaled to either PageWidth or PageHeight depending on which dimension is the largest

The parameter should be:

1. Hidden
2. Have the “Text” data type


Report Parameter Properties

General
Available Values
Default Values
Advanced

Change name, data type, and other options.

Name: VR_Scale

Prompt: VR_Scale

Data type: Text

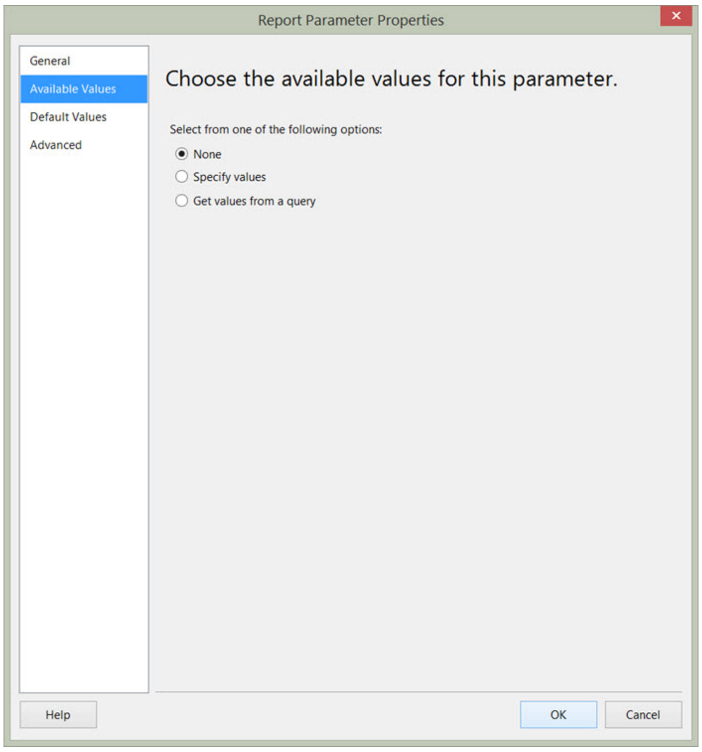
- [ ] Allow blank value ("")
- [ ] Allow null value
- [ ] Allow multiple values

Select parameter visibility:

Visible [✓ Hidden] Internal

Help OK Cancel

3. Have no available values


Report Parameter Properties

General
Available Values
Default Values
Advanced

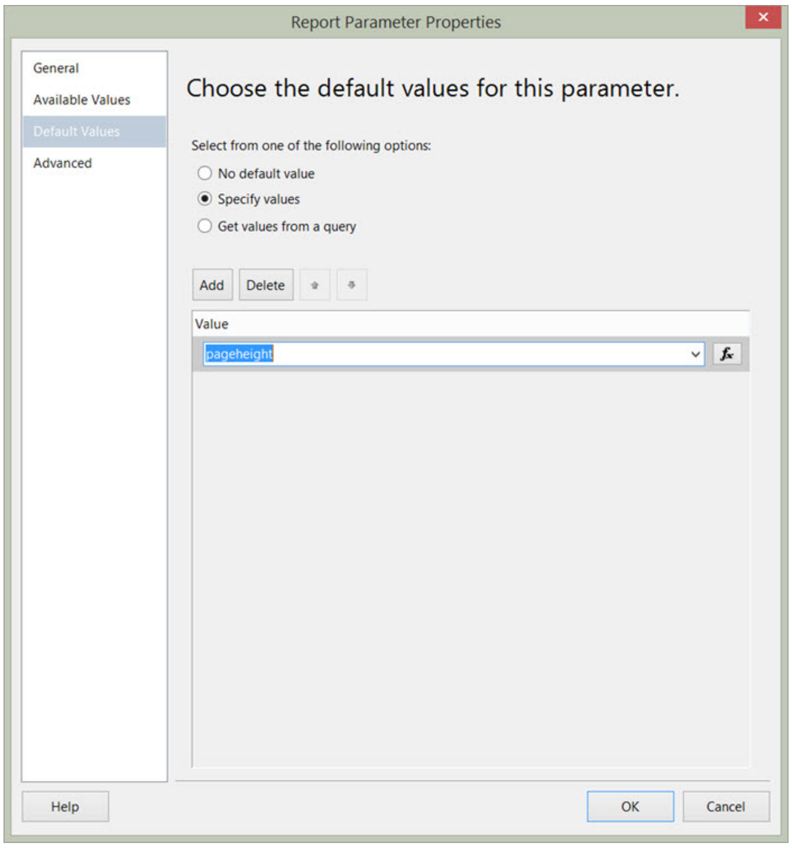
Choose the available values for this parameter.

Select from one of the following options:

[✓ None] Specify values Get values from a query

Help OK Cancel

4. Have a default value equal to one of the options: PageWidth, PageHeight, Fit (they are not case sensitive)



Choose the default values for this parameter.

Select from one of the following options:

- No default value
- [✓ Specify values]
- Get values from a query

Value: pageheight

Chapter 4

Translation of Report Texts

Topics:

- *Textbox Values*
- *Report Parameters*
- *Syntax of the Translation Tag*
- *Language Text Files*

This chapter describes how to correctly setup translation of texts in a report. Use this functionality if you want to create reports with language dependent text as labels, headings and parameters. In short terms this is done by putting the text in external files placed in a specific folder. The file will contain one text for each supported language. Inside the report you point to the external text with the described codes. You can look at Visma's standard report to see examples.

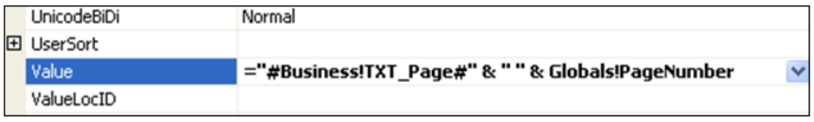
Your language files should be placed in the folder `\Reporting\Server\Resources`.

Textbox Values

The text in the textbox in design mode is entered in the *'Value'* property of the Textbox. If you want translation of the text in the text box when running the report in Visma Reporting, enter a text id surrounded by ' # ' signs in the *'Value'* property of the text box. This is called a translation tag. All the text between the ' # '-signs will be replaced by the translated texts. You can have one or more texts to be translated in the *'Value'* property. Remember that the expression must be valid after the translation. Remember to put the translation tag between two text qualifiers (" ").

Example:

Value is ="#Business!TXT_Page#" & " – " & Trim(Str(Globals!PageNumber))



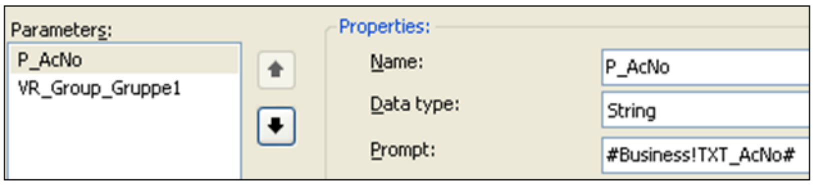
UnicodeBiDi	Normal
⊞ UserSort	
Value	="#Business!TXT_Page#" & " " & Globals!PageNumber
ValueLocID	

Report Parameters

The text for the parameter caption is entered in the 'Prompt' property of the Report Parameter. If you want translation of the text in the text box when running the report in Visma Reporting, enter a text id in the 'Prompt' property instead. You can combine more text ids if you like. Just put them between the ' # ' -signs. In this case you don't need to think of validating the text as an expression.

Example:

Value is #Business!TXT_FromDate# - #Reporting!#TXT_ToDate#



Parameters:
- P_AcNo
- VR_Group_Gruppe1

Properties:
- Name: P_AcNo
- Data type: String
- Prompt: #Business!TXT_AcNo#

Syntax of the Translation Tag

The translation tag must be in the format `#[<Domain>!]<TextID>[.<ValueType>]#` , for instance `#Payroll!TXT_Employee.Value#` , or just `#TXT_Employee#` if the text is to be fetched with default domain and default value type.

Note: The translation tag value is case sensitive and must correspond with the case in the text file.

<Domain> can be one of the followings:

Reporting	Points to the language text file containing common texts for Visma Reporting. **This is the default domain if <Domain> is ommitted**
Payroll	Points to the language text file with common texts for Visma Payroll reports
Business	Points to the language text file with common texts for Visma Business reports
Global	Points to the language text file with common texts for Visma Global reports
CRM	Points to the language text file with common texts for Visma CRM reports

<ReportName>	Points to the language text file with special texts for the current report. <ReportName> is the report file name without the file type extension (.rdl)

<TextID> is a freely chosen text identificator, but we strongly recommend that the text identificator starts with the “TXT_” prefix and do not include any spaces or special characters.

<ValueType> can be one of the followings:

Value	Returns the value of the 'Text' tag for the text in the language text file. **This is the default value type if <ValueType> is ommitted**
Text	Same as 'Value'. Returns the value of the ‘Text’ tag for the text in the language text file
ShortText	Returns the value of the 'ShortText' tag for the text in the language text file
Description	Returns the value of the 'Description' tag for the text in the language text file

Language Text Files

Visma Reporting uses the *Visma.Core.Globalization* component to translate all international texts. All the language text files for the reports must reside in the same resource folder on the server.

The file name of the language text file must be on the format:
`<LanguageCode>-<CultureCode>_<Domain>.language.xml`

<LanguageCode>-<CultureCode> can be one of the followings:

en-GB	English (U.K.)
no-NO	Norwegian
sv-SE	Swedish
da-DK	Danish
fi-FI	Finnish

<Domain> can be one of the followings:

Reporting	Language text files for Visma Reporting
Payroll	Language text files for Visma Payroll reports
Business	Language text files for Visma Business reports
Global	Language text files for Visma Global reports
CRM	Language text files for Visma CRM reports

Example:
nb-NO_Reporting.language.xml
en-GB_Payroll.language.xml

Note: All language codes, culture codes and domains are case sensitive. Take this in regard when designing reports with globalization support.

Chapter 5

Merging Data From Multiple Data Sources

Topics:

- *Merging Data From Multiple Data Sources*

This chapter describes how to merge multiple data sources when creating reports with multi-application or multi-company data.

Merging Data From Multiple Data Sources

When creating reports with multi-application or multi-company data we are fetching data in two steps:

1. First, we fetch data from the different databases
2. Second, we merge the fetched data

Both steps are done with regular SQL queries, but the second step needs some extra tags to make the system understand how the data should be merged. This syntax is described in detail in the next chapters.

The reason we fetch the data in two steps are integration issues:

- When fetching data from multiple databases in the same query, collation problems tend to occur
- You need to configure the databases and link the database servers separately to allow queries across databases
- Fetching data from databases from different vendors (I.e. Oracle, SQL Server) is difficult

By fetching data from one data-source at a time we avoid these problems, and we also get the flexibility we need to fetch data for a varying number of companies.

Conceptually we put the data into temporary tables in the report server memory, which we then query with the merge query. This could be memory intensive so we encourage the designer to do complex querying and filtering in the database queries. That puts more load on the database server and less on the report server. The query on the merged dataset is not provided by the SQL Server, but instead by an internal “SQL component” in the Visma Report Server.

Chapter

6

Multi-Application Reporting

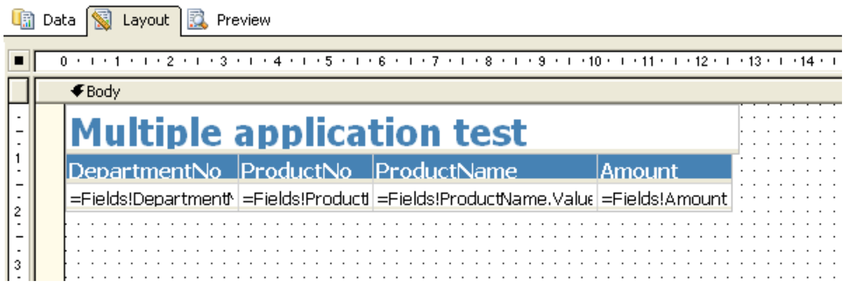
Topics:

- *Multi-Application Reports*
- *Application Type*
- *Parameters*
- *Dataset Naming*
- *Database Datasets*
- *Merge Dataset*

Multi-Application Reports

The following example fetches data from two different systems: Visma Business and Microsoft AdventureWorks example application. These applications are not required to read this chapter. The report definition file MultiAppDepartmentSale.rdl shows the complete example.

The report is a small report that fetches sale information from Visma Business and the product name from CycleShop which is an imaginary application type for on the AdventureWorks database.



Multiple application test

DepartmentNo	ProductNo	ProductName	Amount
=Fields!Department	=Fields!Product	=Fields!ProductName.Value	=Fields!Amount

Overview

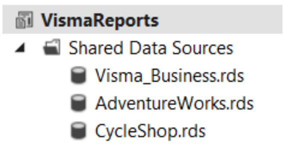
1. Create data-sources for your application types. Make sure the data-source name matches the name of your application type
2. Create the parameter datasets
3. Make sure the dataset references a data-source to the correct application type
4. Create the main database query datasets. Dataset names starts with VR_AppType_
5. Create the main merge query dataset. Dataset name starts with VR_MergeAppType_
6. Design the report
7. Add merge datasets as data-source to data regions

Application Type

The application type is the name of the application defined in Visma User Directory. If the name contains a space (example "Visma Business") it has to be replaced with an "_" (underline) in the RDL file.

For example, for the application types "Visma Business", "Visma Payroll" the naming will be "Visma_Business", "Visma_Payroll".

Application types are used as name of the report data source. This makes the system understand from which database the data should be fetched.



VismaReports
- Shared Data Sources
  - Visma_Business.rds
  - AdventureWorks.rds
  - CycleShop.rds

Parameters

You create and define parameters the same way as you would in a regular report. There are two restrictions that apply:

- The parameter dataset's data-source property must be set to an application type specific data-source
- The parameters only apply to the database query. The merge query will merge what is already fetched into the temporary tables

Note: In case of multi-application reports that have parameters that should be updated on the server-side (see *Forcing Server Update* (see page 30)) you need to take into consideration the following: if a parameter is updated by more than one application type, than the last updated value is going to be taken into consideration.

Dataset Naming

To use datasets for multi-application queries you must use dataset name prefix. The datasets that fetch data from the application databases must have a name that starts with **VR_AppType_** and a dataset that starts with **VR_MergeAppType_** for merging data. For the example used here you will have the following data set names:

- VR_AppType_VB_Test
- VR_AppType_AdWorks_Test
- VR_MergeAppType_Test

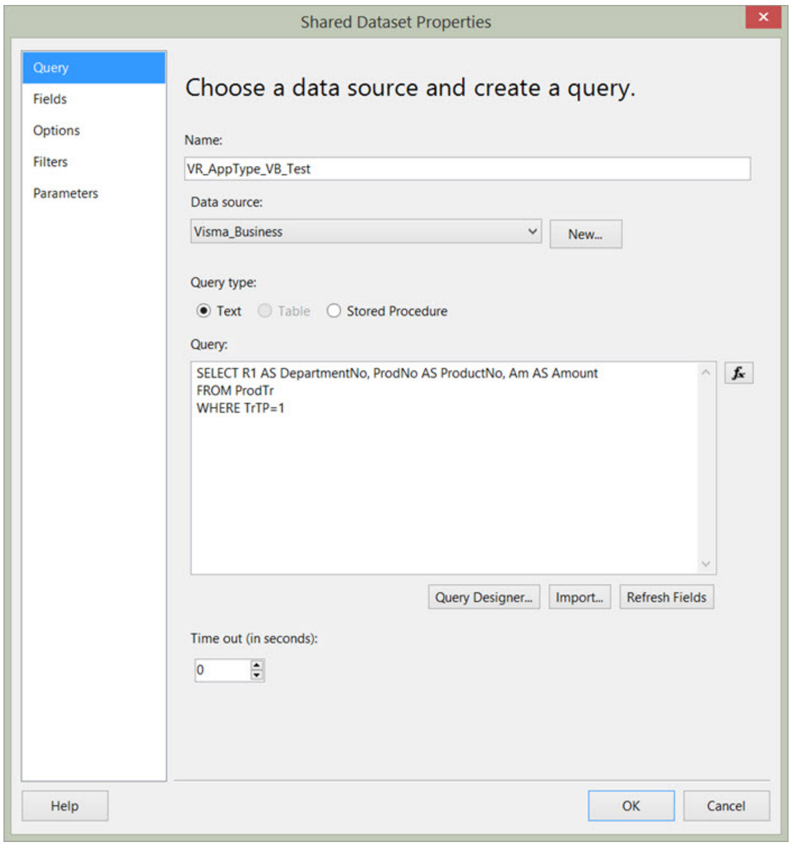
The VR_AppType_ datasets fetch data from the different applications, and the VR_MergeAppType_ dataset merges the data fetch into temporary tables from the VR_AppType_ datasets.

The **_Test** postfix is used to group the datasets together, making the system understand which datasets that belong together. This means you can have several data regions in your report that fetch data from the same applications.

Database Datasets

The datasets fetching data from the different application databases are regular datasets, as any other dataset, the only thing to remember is the naming.

Visma Business:



Choose a data source and create a query.

Name: VR_AppType_VB_Test

Data source: Visma_Business

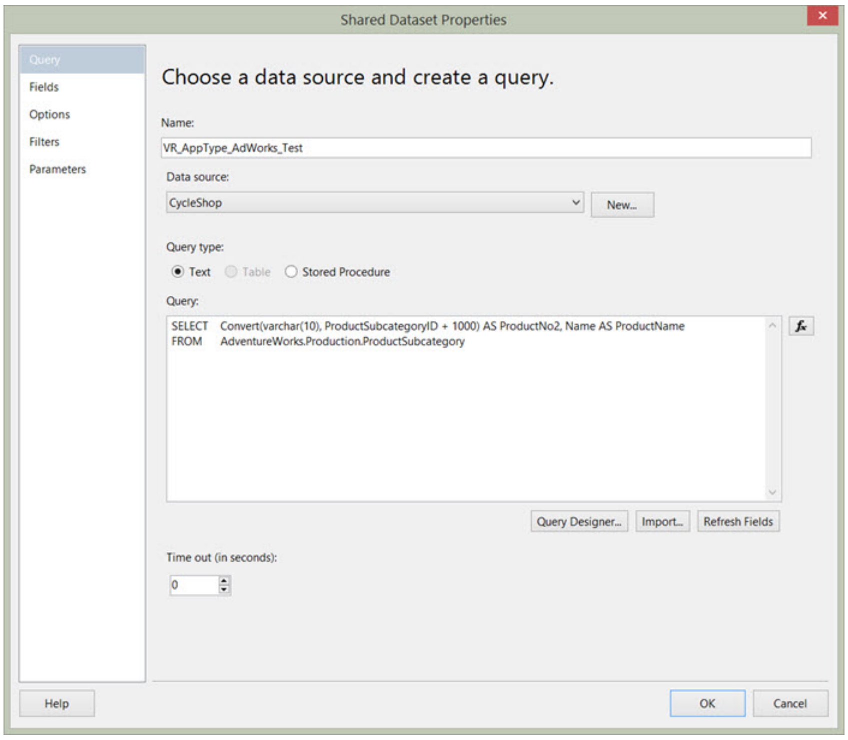
Query type: [✓ Text] Table Stored Procedure

Query:

```
SELECT R1 AS DepartmentNo, ProdNo AS ProductNo, Am AS Amount
FROM ProdTr
WHERE TrTP=1
```

Time out (in seconds): 0

CycleShop:



Name: VR_AppType_AdWorks_Test

Data source: CycleShop

Query type: [✓ Text] Table Stored Procedure

Query:

```
SELECT   Convert(varchar(10), ProductSubcategoryID + 1000) AS ProductNo2, Name AS ProductName
FROM     AdventureWorks.Production.ProductSubcategory
```

Time out (in seconds): 0

Merge Dataset

The merge dataset needs some extra information to it.

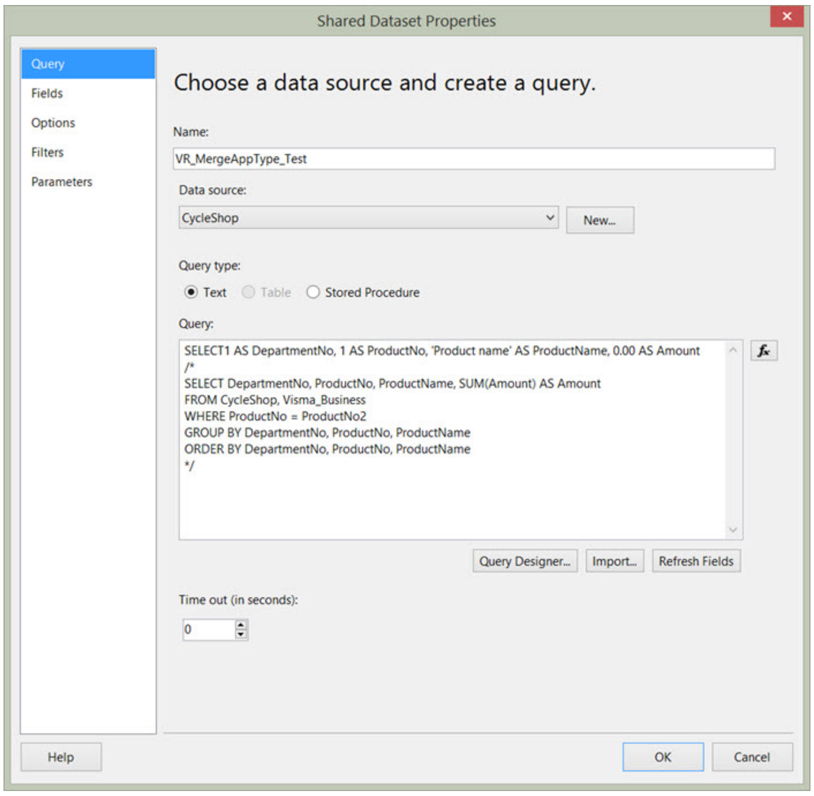
We have a dummy query first that is used in the report designer making the report compile when the dataset is linked to a data-region.

```
SELECT1 AS DepartmentNo, 1 AS ProductNo, 'Product name' AS ProductName, 0.00 AS
Amount
```

The merge query is put in as a SQL comment. This ensures the compiler doesn't complain about the unknown application type names:

```
/*
SELECT DepartmentNo, ProductNo, ProductName, SUM(Amount) AS Amount
FROM CycleShop, Visma_Business
WHERE ProductNo = ProductNo2
GROUP BY DepartmentNo, ProductNo, ProductName
ORDER BY DepartmentNo, ProductNo, ProductName
*/
```

This is as you can see also a regular query, but you need to use the application type as table names in your query. Remember to replace blanks with underscore in the name.



Choose a data source and create a query.

Name: VR_MergeAppType_Test

Data source: CycleShop

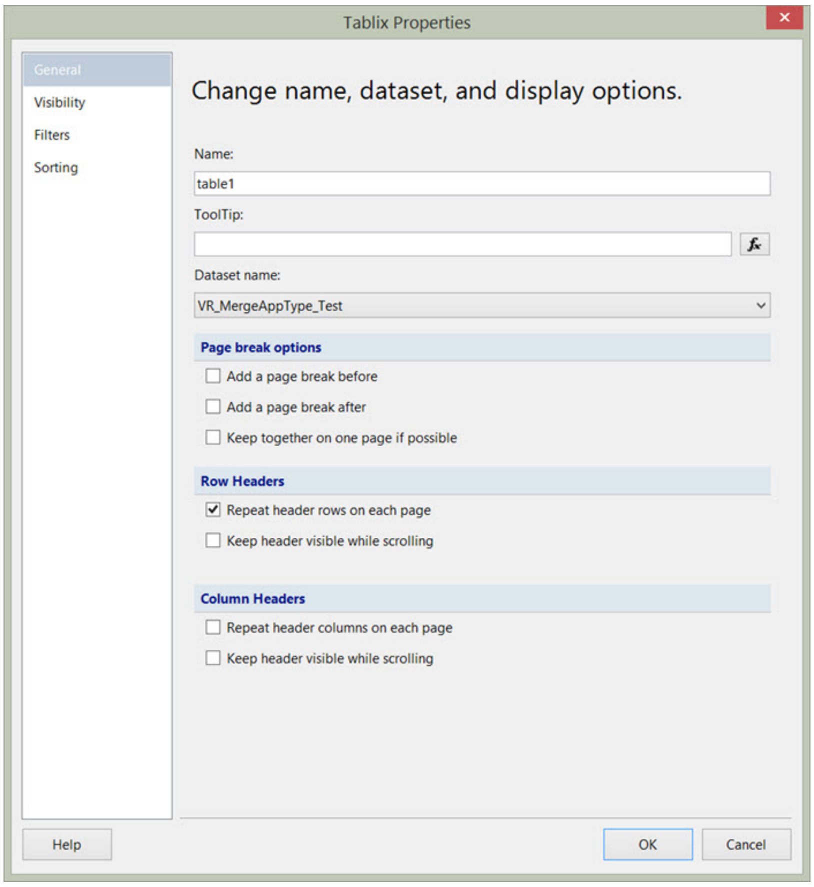
Query type: [✓ Text] Table Stored Procedure

Query:

```
SELECT1 AS DepartmentNo, 1 AS ProductNo, 'Product name' AS ProductName, 0.00 AS Amount
/*
SELECT DepartmentNo, ProductNo, ProductName, SUM(Amount) AS Amount
FROM CycleShop, Visma_Business
WHERE ProductNo = ProductNo2
GROUP BY DepartmentNo, ProductNo, ProductName
ORDER BY DepartmentNo, ProductNo, ProductName
*/
```

Time out (in seconds): 0

The data-region (In this case a table) will then look like this:



Change name, dataset, and display options.

Name: table1

ToolTip:

Dataset name: VR_MergeAppType_Test

Page break options

- ☐ Add a page break before
- ☐ Add a page break after
- ☐ Keep together on one page if possible

Row Headers

- ☑ Repeat header rows on each page
- ☐ Keep header visible while scrolling

Column Headers

- ☐ Repeat header columns on each page
- ☐ Keep header visible while scrolling

Chapter

7

Multi-Company Reporting

Topics:

- *Multi-Company Reports*
- *Column-Based Datasets*
- *Raw-Based Datasets*

Multi-Company Reports

Multi-company reporting differs from the multi-application reporting in that we do not know before runtime how many companies the report should fetch data for. A multi-application report is always built for a fixed set of applications. This means that we have to mark some columns or rows as extendable, meaning that they could be duplicated for each selected company.

In addition, multi-company reports assume that they are run against databases with the same structure and available parameter values, or with a company that is the master. This is because parameter values are only fetched from one of the companies (the master company).

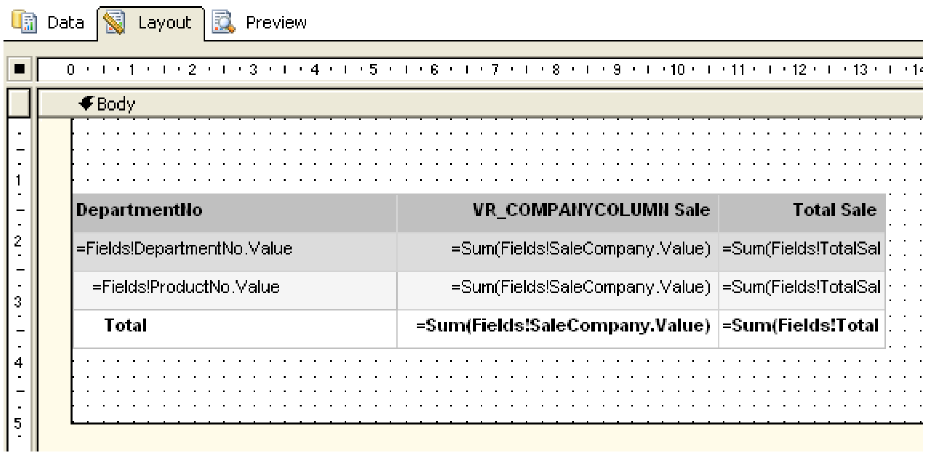
For presenting data use the **Table** component (tablix). The Matrix component (tablix) is not supported for multi-company reports.

Overview:

1. Design the report as a regular report
2. Add a dataset with a query fetching from a fixed set of companies
3. Create the main database query datasets. Dataset names starts with VR_Company_
4. Create the main merge query dataset. Dataset name starts with VR_MergeCompany_
5. Transform the SQL query into a multi-company query.
6. Add merge datasets as data-source to data regions.

Column-Based Datasets

The example report used to show how to create a column-based multi-company report is *Product Sale Column.rdl*. In addition there is a report named *Product Sale Column – Original.rdl* which shows how the report would look with fixed companies. This makes it easier to see the differences.



DepartmentNo	VR_COMPANYCOLUMN Sale	Total Sale
=Fields!DepartmentNo.Value	=Sum(Fields!SaleCompany.Value)	=Sum(Fields!TotalSal
=Fields!ProductNo.Value	=Sum(Fields!SaleCompany.Value)	=Sum(Fields!TotalSal
Total	=Sum(Fields!SaleCompany.Value)	=Sum(Fields!Total

Fixed Company Query

It is easier to first create the query as you would like it to be using hard-coded companies. Examples of such queries could be when fetching from different databases:

```
SELECT  T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, SUM(T.Company1)
 AS SaleCompany1, SUM(T.Company2) AS SaleCompany2, SUM(T.Company1) +
 SUM(T.Company2) AS TotalSale
FROM    (SELECT     R1, ProdNo, Am AS Company1, 0 AS Company2
              FROM       F0001.dbo.ProdTr
              WHERE      TrTp = 1
              UNION ALL
              SELECT     R1, ProdNo, 0 AS Company1, Am AS Company2
              FROM       F8999.dbo.ProdTr
              WHERE      TrTp = 1) AS T
```

```
GROUP BY T.R1, T.ProdNo
ORDER  BY DepartmentNo, ProductNo
```

Or when fetching from a single database (to fake multiple companies) avoiding collation problems etc:

```
SELECT  T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, SUM(T.Company1)
 AS SaleCompany1, SUM(T.Company2) AS SaleCompany2, SUM(T.Company3) AS
 SaleCompany3, SUM(T.Company1) + SUM(T.Company2) + SUM(T.Company3) AS
 TotalSale
FROM  (SELECT     R1, ProdNo, Am AS Company1, 0 AS Company2, 0 AS Company3
              FROM     ProdTr
              WHERE    TrTp = 1
              UNION ALL
              SELECT   R1, ProdNo, 0 AS Company1, Am AS Company2, 0 AS
 Company3
              FROM     ProdTr
              WHERE    TrTp = 1
              UNION ALL
              SELECT   R1, ProdNo, 0 AS Company1, 0 AS Company2, Am AS
 Company3
              FROM     ProdTr
              WHERE    TrTp = 1) AS T
GROUP BY T.R1, T.ProdNo
ORDER BY DepartmentNo, ProductNo
```

Dataset Naming

To use datasets for multi-company queries you must use dataset name prefix. The datasets that fetch data from the company databases must have a name that starts with **VR_Company_** and a dataset that starts with **VR_MergeCompany_** for merging data. For the example used here you will have the following data set names:

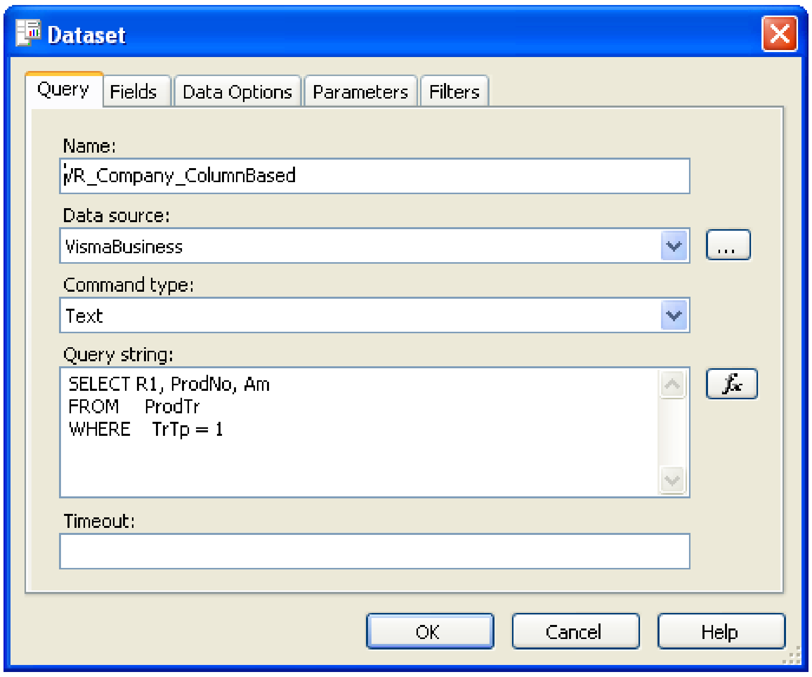
- VR_Company_ColumnBased
- VR_MergeCompany_ColumnBased

The VR_Company_ dataset fetch data from the different companies, and the VR_MergeCompany_ dataset merges the data fetch into temporary tables from the VR_Company_ datasets.

The **_ColumnBased** postfix is used to group the datasets together, making the system understand which datasets that belong together. This means you can have several data regions in your report that fetch data from the same companies.

Database Dataset

The datasets fetching data from the different company databases are regular datasets, as any other dataset, the only thing to remember is the naming. This query will typically be the select used in the UNION statement of your original query.



Name: VR_Company_ColumnBased

Data source: VismaBusiness

Command type: Text

Query string:

```
SELECT R1, ProdNo, Am
FROM    ProdTr
WHERE   TrTp = 1
```

Timeout:

Merge Dataset

About this task

The merge dataset needs some extra information to it:

We have a dummy query first that is used in the report designer making the report compile when the dataset is linked to a data-region.

```
SELECT 1 AS DepartmentNo, 1 AS ProductNo, 0.00 AS SaleCompany, 0.00 AS TotalSale
```

The merge query is put in as a SQL comment. This ensures the compiler doesn’t complain about the unknown company column tags:

```
/*
#BEGIN_QUERY
SELECT T.R1 AS DepartmentNo, T.ProdNo AS ProductNo
         ,SUM(T.#VR_COMPANYCOLUMN$Am#) AS SaleCompany
         ,SUM(T.#VR_COMPANYCOLUMN$Am#)!VR_AGGREGATE AS TotalSale
FROM (#SELECT R1, ProdNo, #VR_COMPANYCOLUMN$Am#!UNION ALL#) AS T
GROUP BY T.R1, T.ProdNo
ORDER BY DepartmentNo, ProductNo
#END
*/
```

Procedure

1. Place the query between the `#BEGIN_QUERY` and the `#END` tags
2. Mark the columns that should be duplicated with the `VR_COMPANYCOLUMN` tag

   The original select:

   ```
   SELECT  T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, SUM(T.Company1)
   AS SaleCompany1, SUM(T.Company2) AS SaleCompany2, SUM(T.Company1) +
   SUM(T.Company2) AS TotalSale
   ```

   Becomes:

   ```
   SELECT T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, SUM(T.#VR_COMPANYCOLUMN
   $Am#) AS SaleCompany, SUM(T.#VR_COMPANYCOLUMN$Am#)!VR_AGGREGATE AS TotalSale
   ```

As you can see instead of listing each company we replace it with the `VR_COMPANYCOLUMN` tag. The tag is used with the T table prefix and Am column name postfix, so replacing it with T.am should work for one company.

Note: VR_AGGREGATE tells that this column should be the aggregation of all company columns. This is used to get at “Total” column.

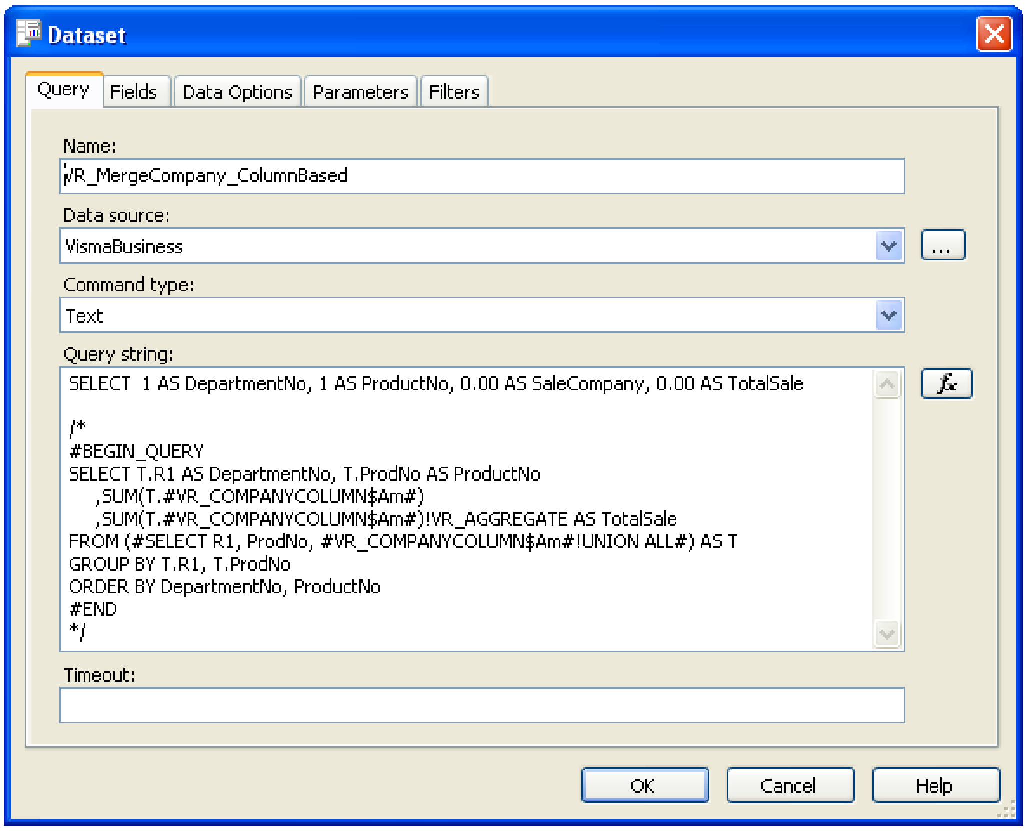
3. Mark the columns in the company query that should be duplicated for each company

The original select: `SELECT R1, ProdNo, Am`

Becomes: `#SELECT R1, ProdNo, #VR_COMPANYCOLUMN$Am#!UNION ALL#`

Note: The !UNION ALL tells what kind of UNION operation should be performed between the company data.

Results



Name: VR_MergeCompany_ColumnBased

Data source: VismaBusiness

Command type: Text

Query string:

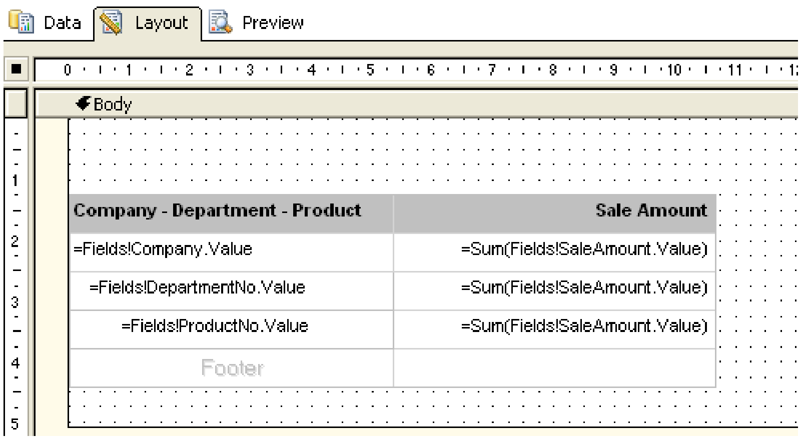
```
SELECT  1 AS DepartmentNo, 1 AS ProductNo, 0.00 AS SaleCompany, 0.00 AS TotalSale

/*
#BEGIN_QUERY
SELECT T.R1 AS DepartmentNo, T.ProdNo AS ProductNo
    ,SUM(T.#VR_COMPANYCOLUMN$Am#)
    ,SUM(T.#VR_COMPANYCOLUMN$Am#)!VR_AGGREGATE AS TotalSale
FROM (#SELECT R1, ProdNo, #VR_COMPANYCOLUMN$Am#!UNION ALL#) AS T
GROUP BY T.R1, T.ProdNo
ORDER BY DepartmentNo, ProductNo
#END
*/
```

Timeout:

Raw-Based Datasets

The row-based multi-company reports are done more or less the same way as the column-based. This chapter just focuses on the differences.

The example report used to show how to create a row-based multi-company report is *Product Sale Row.rdl*. In addition there is a report named *Product Sale Row – Original.rdl* which shows how the report would look with fixed companies.



Company - Department - Product	Sale Amount
=Fields!Company.Value	=Sum(Fields!SaleAmount.Value)
=Fields!DepartmentNo.Value	=Sum(Fields!SaleAmount.Value)
=Fields!ProductNo.Value	=Sum(Fields!SaleAmount.Value)

Fixed Company Query

It is easier to first create the query as you would like it to be using hard-coded companies. The example fetches from a single database (to fake multiple companies) avoiding collation problems etc:

```
SELECT  T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, T.Company, SUM(T.Am) AS
 SaleAmount
FROM    (SELECT   R1, ProdNo, Am, 'Company1' as Company
              FROM     ProdTr
              WHERE    TrTp = 1
              UNION ALL
              SELECT   R1, ProdNo, Am, 'Company2' as Company
              FROM     ProdTr
              WHERE    TrTp = 1
              UNION ALL
              SELECT   R1, ProdNo, Am, 'Company3' as Company
              FROM     ProdTr
              WHERE    TrTp = 1) AS T
GROUP BY T.Company, T.R1, T.ProdNo
ORDER BY Company, DepartmentNo, ProductNo
```

Merge Dataset

About this task

The Database dataset is the same as for the column-based example, but the merge dataset requires some modifications. Instead of having a column that is duplicated, we have only one company name column and a row for each company. We therefore need to indicate that the query should be grouped by company:

```
SELECT 1 AS DepartmentNo, 1 AS ProductNo, 'Company1' AS Company, 0.00 AS
 SaleAmount
/*
#BEGIN_QUERY
SELECT T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, T.#VR_COMPANYCOLUMN# AS
 Company, Sum(T.Am) AS SaleAmount
        FROM (#SELECT R1, ProdNo, Am, #VR_COMPANYCOLUMN#!UNION ALL#) AS T
        GROUP BY T.#VR_COMPANYCOLUMN#, T.R1, T.ProdNo
        ORDER BY Company, DepartmentNo, ProductNo
#END
*/
```

You have to do the followings:

Procedure

1. Place the query between the `#BEGIN_QUERY` and the `#END` tags
2. Mark the company column that should be duplicated with the `VR_COMPANYCOLUMN` tag

The original select: `SELECT  T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, T.Company, SUM(T.Am) AS SaleAmount`

Becomes: `SELECT T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, T.#VR_COMPANYCOLUMN# AS Company, Sum(T.Am) AS SaleAmount`

3. Mark the company column in the company query

   The original select: `SELECT R1, ProdNo, Am`

   Becomes: `#SELECT R1, ProdNo, Am, #VR_COMPANYCOLUMN#!UNION ALL#`

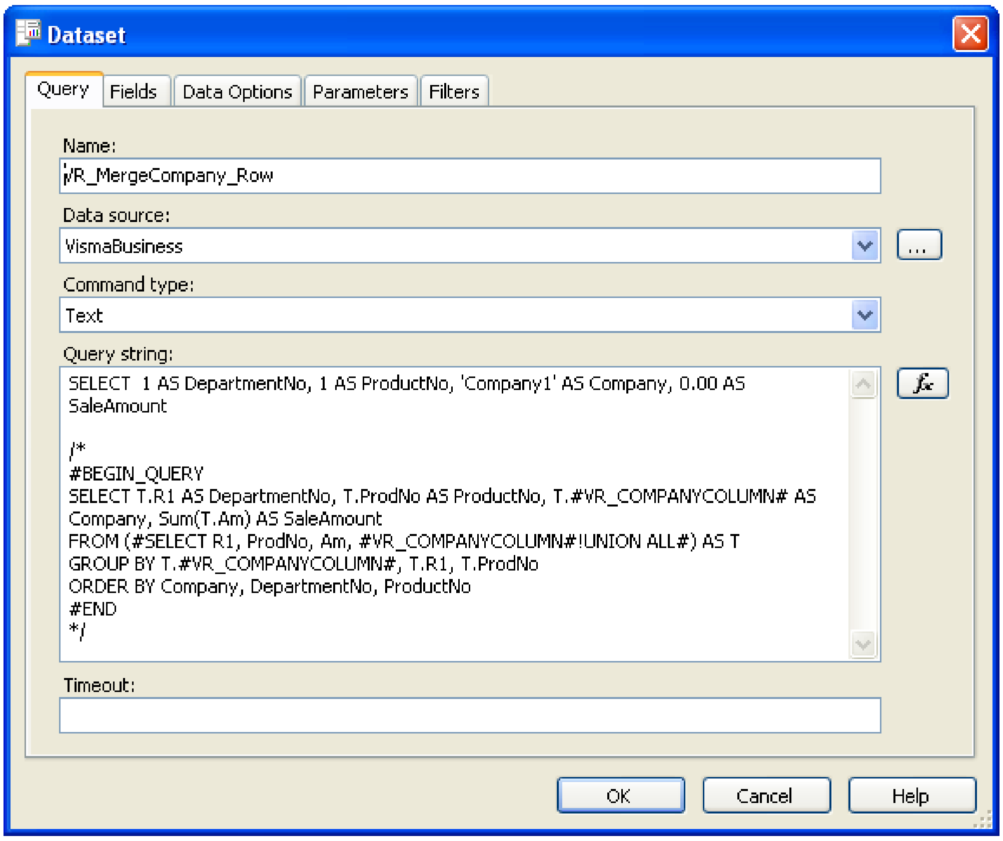
Note: The !UNION ALL tells what kind of UNION operation should be performed between the company data.

4. Add the company column to the GROUP BY clause

   The original group by: `GROUP BY T.Company, T.R1, T.ProdNo`

   Becomes: `GROUP BY T.#VR_COMPANYCOLUMN#, T.R1, T.ProdNo`

Results



Name:
VR_MergeCompany_Row

Data source:
VismaBusiness

Command type:
Text

Query string:

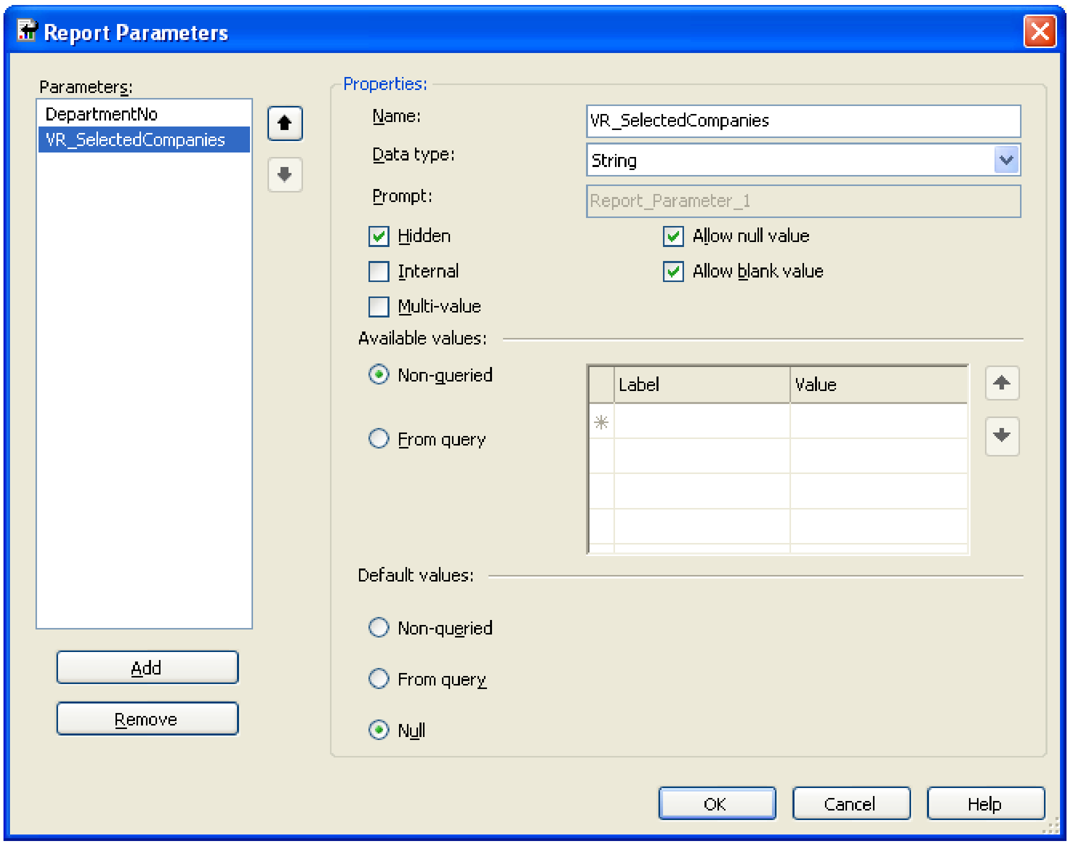
```
SELECT  1 AS DepartmentNo, 1 AS ProductNo, 'Company1' AS Company, 0.00 AS
SaleAmount

/*
#BEGIN_QUERY
SELECT T.R1 AS DepartmentNo, T.ProdNo AS ProductNo, T.#VR_COMPANYCOLUMN# AS
Company, Sum(T.Am) AS SaleAmount
FROM (#SELECT R1, ProdNo, Am, #VR_COMPANYCOLUMN#!UNION ALL#) AS T
GROUP BY T.#VR_COMPANYCOLUMN#, T.R1, T.ProdNo
ORDER BY Company, DepartmentNo, ProductNo
#END
*/
```

Timeout:

Drill Through to Company Report

When you have multi-company reports with drill-through reports that are only showing data for one specific company you have to do two things to send the correct (selected) company from the parent to the child report:

1. Add a parameter named VR_SelectedCompanies to the drill-through report as shown below



Report Parameters

Parameters:
- DepartmentNo
- VR_SelectedCompanies

Properties:

Name:	VR_SelectedCompanies
Data type:	String
Prompt:	Report_Parameter_1

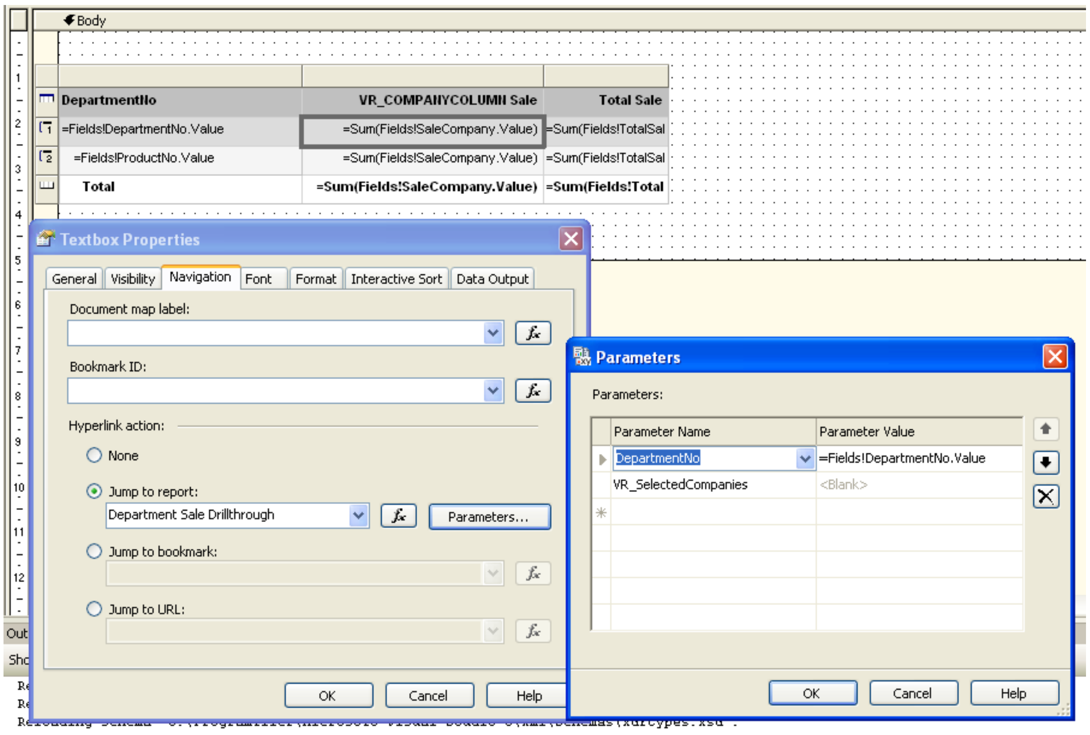
[✓] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

Label	Value

Default values: Non-queried From query [✓ Null]

2. Add VR_SelectedCompany to the list of parameters in your multi-company (parent) report. Make sure the Parameter value is empty, as shown below



DepartmentNo	VR_COMPANYCOLUMN Sale	Total Sale
=Fields!DepartmentNo.Value	=Sum(Fields!SaleCompany.Value)	=Sum(Fields!TotalSal
=Fields!ProductNo.Value	=Sum(Fields!SaleCompany.Value)	=Sum(Fields!TotalSal
Total	=Sum(Fields!SaleCompany.Value)	=Sum(Fields!Total

Textbox Properties — Navigation

Hyperlink action: None [✓ Jump to report:] Jump to bookmark: Jump to URL:

Jump to report: Department Sale Drillthrough

Parameters

Parameter Name	Parameter Value
DepartmentNo	=Fields!DepartmentNo.Value
VR_SelectedCompanies	<Blank>

This would ensure that the system are able to set the correct selected company to the drill-through report.

Chapter

8

Electronic Reporing

Topics:

- *Electronic Reporting*
- *Electronic Reports*
- *Downloadable Reports*

Electronic Reporting

By electronic reporting we mean reports that are created with the purpose of being saved locally for later to be sent to an electronic system that can read and process this file. Examples of this are other ERP systems or an Internet portal for public reporting (E.g. AltInn in Norway). Visma Reporting also offers the possibility to send this automatically, by implementing integration logic in the Reporting File Provider.

Example data

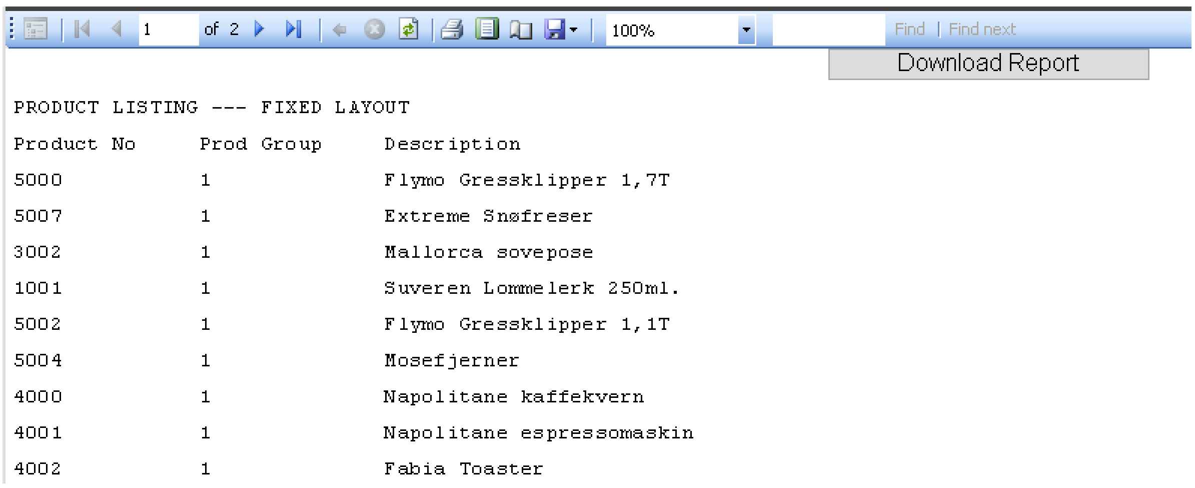
All files from the electronic reporting examples are available in the attachment “Electronic Reporting.zip”.

Electronic Reports

Electronic Report Examples

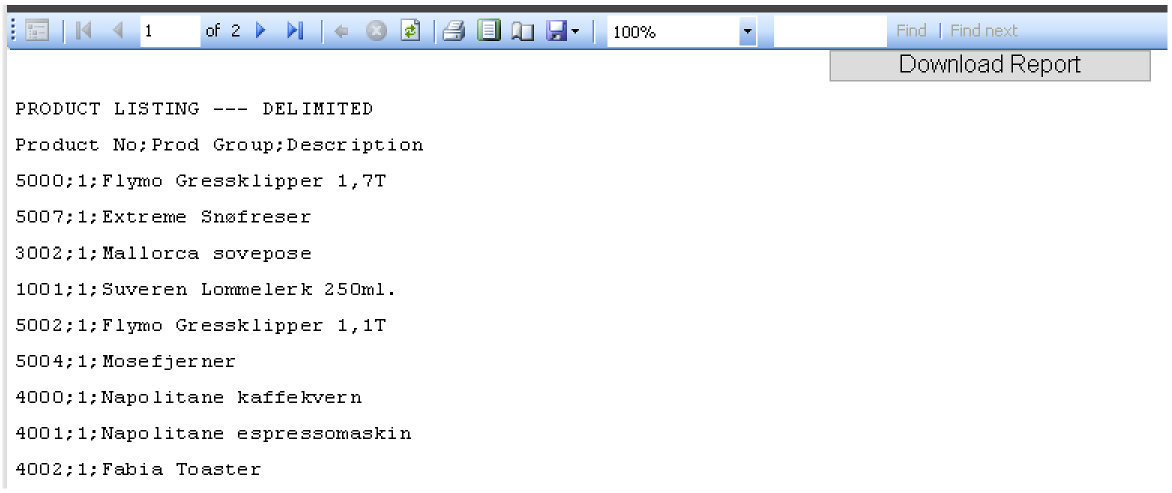
Two examples of electronic report are provided as an appendix. These are simple reports made for Visma Business, but they explain the concepts.

One using a fixed layout:



```
PRODUCT LISTING --- FIXED LAYOUT
Product No     Prod Group     Description
5000           1              Flymo Gressklipper 1,7T
5007           1              Extreme Snøfreser
3002           1              Mallorca sovepose
1001           1              Suveren Lommelerk 250ml.
5002           1              Flymo Gressklipper 1,1T
5004           1              Mosefjerner
4000           1              Napolitane kaffekvern
4001           1              Napolitane espressomaskin
4002           1              Fabia Toaster
```

And another using a delimiter:



```
PRODUCT LISTING --- DELIMITED
Product No;Prod Group;Description
5000;1;Flymo Gressklipper 1,7T
5007;1;Extreme Snøfreser
3002;1;Mallorca sovepose
1001;1;Suveren Lommelerk 250ml.
5002;1;Flymo Gressklipper 1,1T
5004;1;Mosefjerner
4000;1;Napolitane kaffekvern
4001;1;Napolitane espressomaskin
4002;1;Fabia Toaster
```

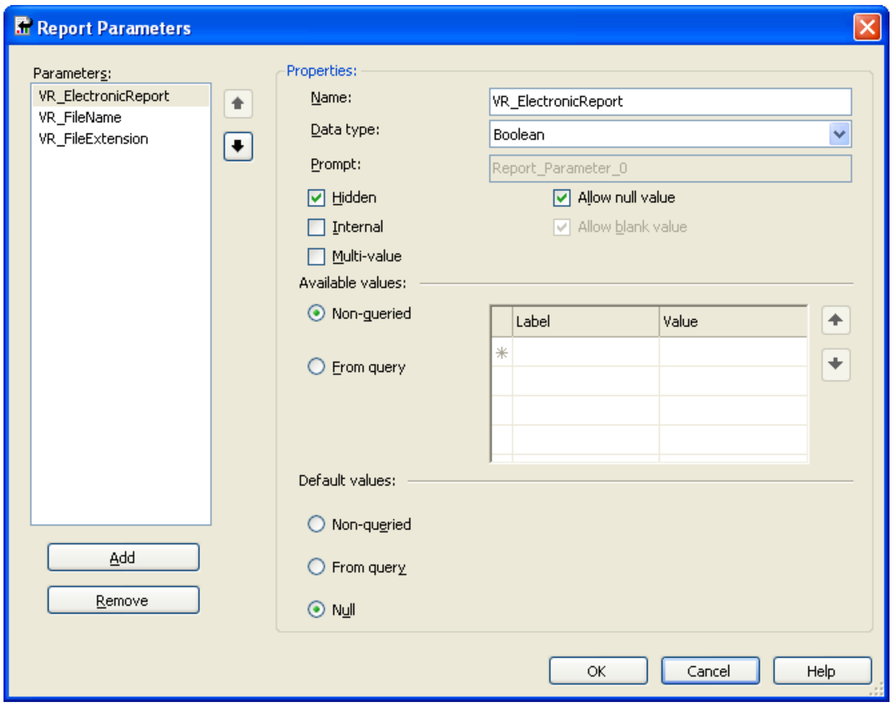
The example reports contain string functions for getting the layout that could be reused in other reports. The “Download Report” button toggles a parameter used to indicate that the report content should be downloaded.

Parameter Setup

To tag a report as an electronic report you must add some special parameters.

VR_ElectronicReport - Mandatory

This parameter marks the report as an electronic report. It does not require a value and is merely a hidden marker parameter. Setup as shown below:


Report Parameters

Parameters:
- VR_ElectronicReport
- VR_FileName
- VR_FileExtension

Properties:

Name: VR_ElectronicReport
Data type: Boolean
Prompt: Report_Parameter_0

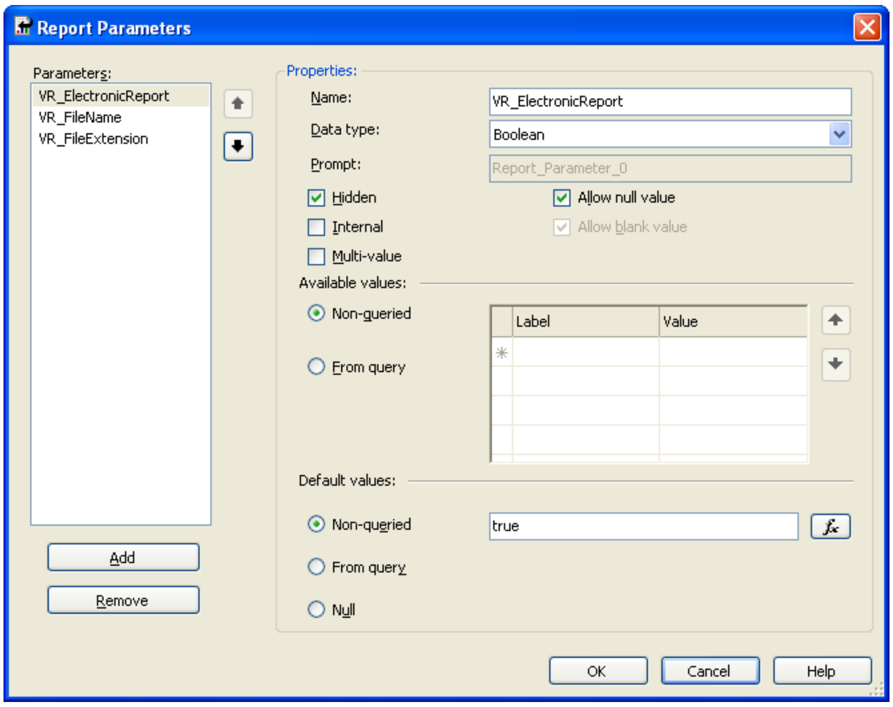
[✓] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

Label	Value

Default values: Non-queried From query [✓ Null]

You can also use the same parameter with a default value of “true” to download the content directly:


Report Parameters

Parameters:
- VR_ElectronicReport
- VR_FileName
- VR_FileExtension

Properties:

Name: VR_ElectronicReport
Data type: Boolean
Prompt: Report_Parameter_0

[✓] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

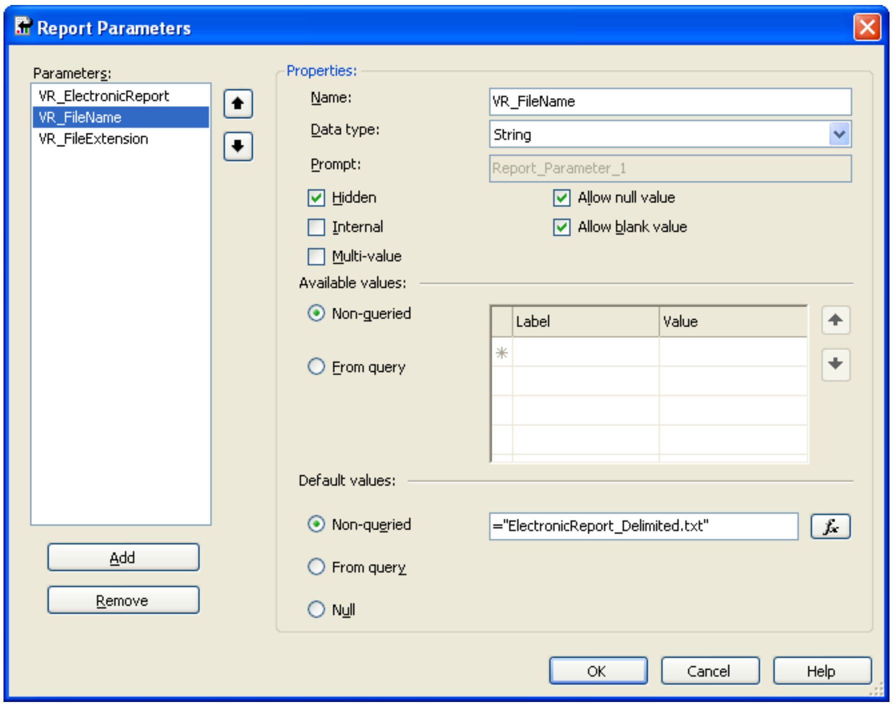
Label	Value

Default values: [✓ Non-queried] true From query Null

The provided sample reports both have a “Download report” button that toggles the value of this parameter, and a report that uses the default value “true” to download the report directly.

VR_FileName – Optional

This parameter can be used if you want to give your output file a specific name. In this example it is used as a hidden variable with a default value, but it could also be used as a parameter set by the user. The use is optional. Setup as shown below:



Report Parameters

Parameters:
- VR_ElectronicReport
- VR_FileName
- VR_FileExtension

Properties:

Name:	VR_FileName
Data type:	String
Prompt:	Report_Parameter_1

[✓] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

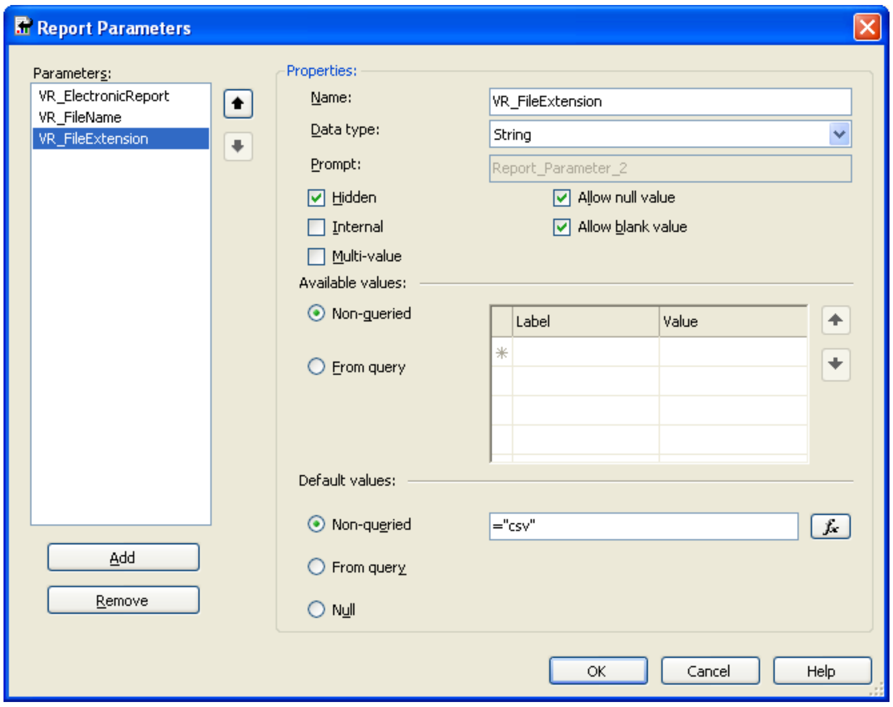
Available values: [✓ Non-queried] From query

Label	Value

Default values: [✓ Non-queried] ="ElectronicReport_Delimited.txt"   From query   Null

VR_FileExtension – Optional

This parameter can be used if you want to give your output file a specific file extension. In this example it is used as a hidden variable with a default value, but it could also be used as a parameter set by the user. The use is optional. Setup as shown below:



Report Parameters

Parameters:
- VR_ElectronicReport
- VR_FileName
- VR_FileExtension

Properties:

Name:	VR_FileExtension
Data type:	String
Prompt:	Report_Parameter_2

[✓] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

Label	Value

Default values: [✓ Non-queried] ="csv"   From query   Null

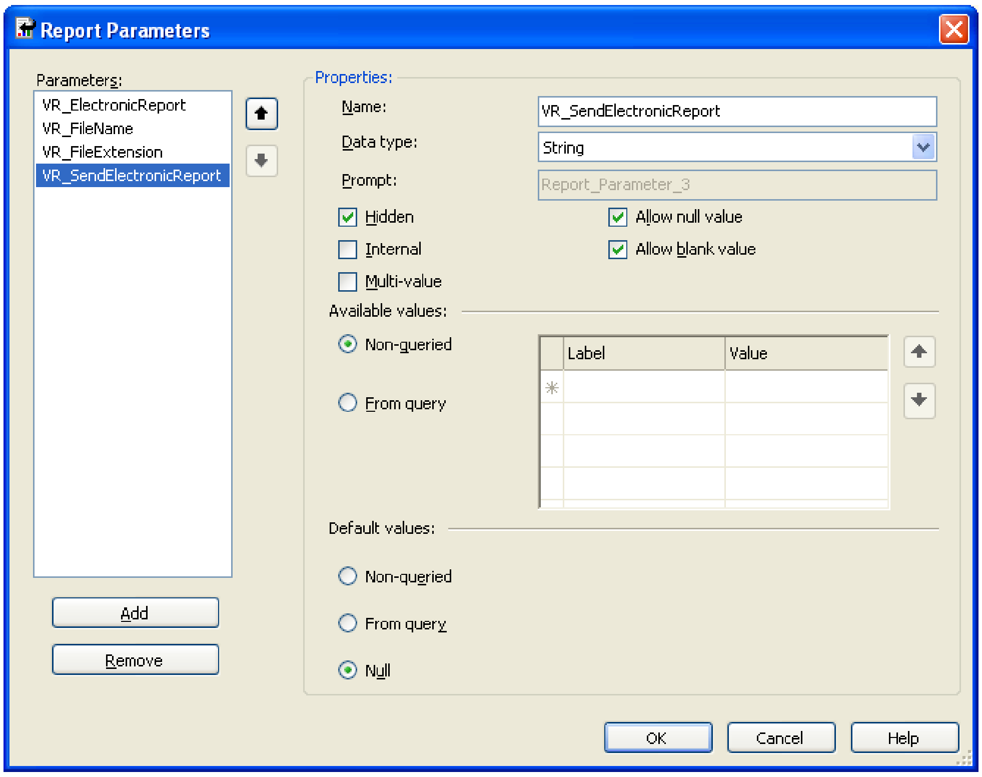
Sending the Electronic Report

It is possible to send the electronic report to an external electronic system or automatically save it to a file share etc.

There are two ways of sending the electronic report. Either you send the downloaded file directly by using the VR_SendElectronicReport parameter or you send the report content by using the dsEReportResult dataset.

VR_SendElectronicReport Parameter

This parameter marks the report as an electronic report that automatically should be sent to an electronic system. If this parameter is available the downloaded file is given to the File Provider. Default behavior is to do nothing, but it could be overridden for product specific actions. The parameter has no value and is merely a hidden marker parameter. Setup as shown below:



Parameters:
- VR_ElectronicReport
- VR_FileName
- VR_FileExtension
- VR_SendElectronicReport

Properties:
Name: VR_SendElectronicReport
Data type: String
Prompt: Report_Parameter_3
[✓] Hidden
[ ] Internal
[ ] Multi-value
[✓] Allow null value
[✓] Allow blank value

Available values: [✓ Non-queried] From query

	Label	Value
*		

Default values: Non-queried / From query / [✓ Null]

dsEReportResult DataSet

By adding a dataset named dsEReportResult to your report the report data is given to the FileProvider. Default behavior is to do nothing, but it could be overridden for product specific actions.

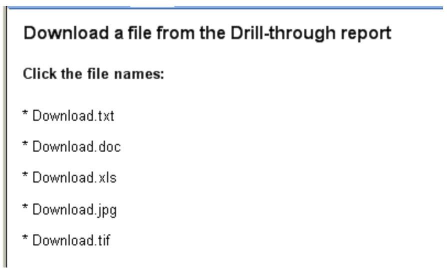
This dataset has two purposes. It tells Visma Reporting that the report should be sent, and it could be used to return status information about the send process to the user.

Downloadable Reports

A downloadable report is a report that could be used to download a file directly. The file is then uploaded to the client without being shown in the Report Viewer window. The user can then choose to save or view the file. This report can be used to fetch documents directly from a file archive. The default implementation is to read a file from disk using the provided file path, but this can be overridden to read the file from a database, web service, etc.

Examples

The provided example is using a regular report listing some documents, and a drill-through report that downloads the files. The drill-through report is the downloadable report:



Download a file from the Drill-through report

Click the file names:

* Download.txt

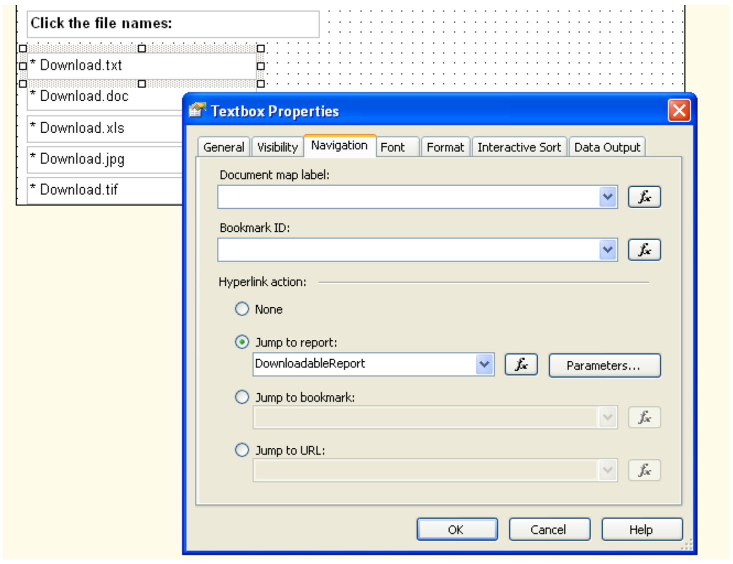
* Download.doc

* Download.xls

* Download.jpg

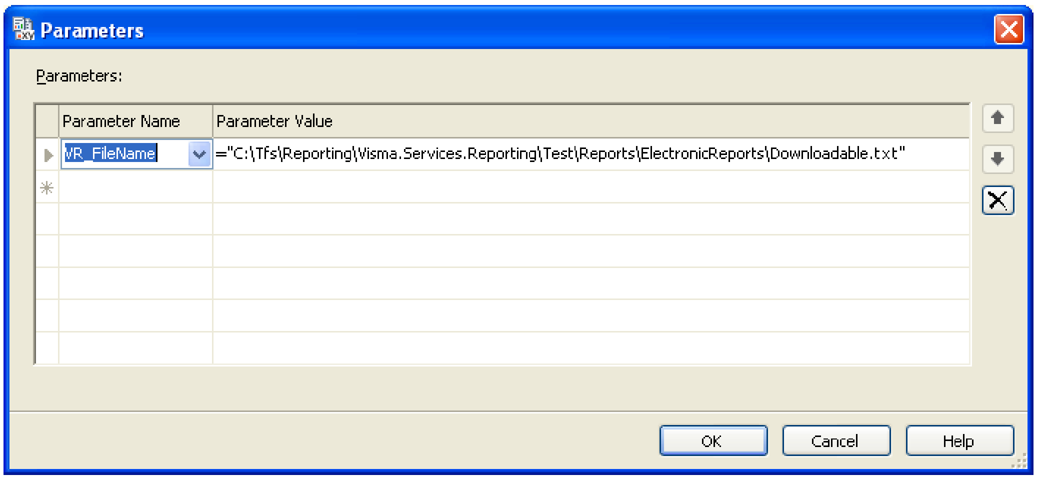
* Download.tif

Each file name has navigates to the downloadable report:



Jump to report: DownloadableReport

The navigation provides the VR_FileName parameter to the drill-through report:



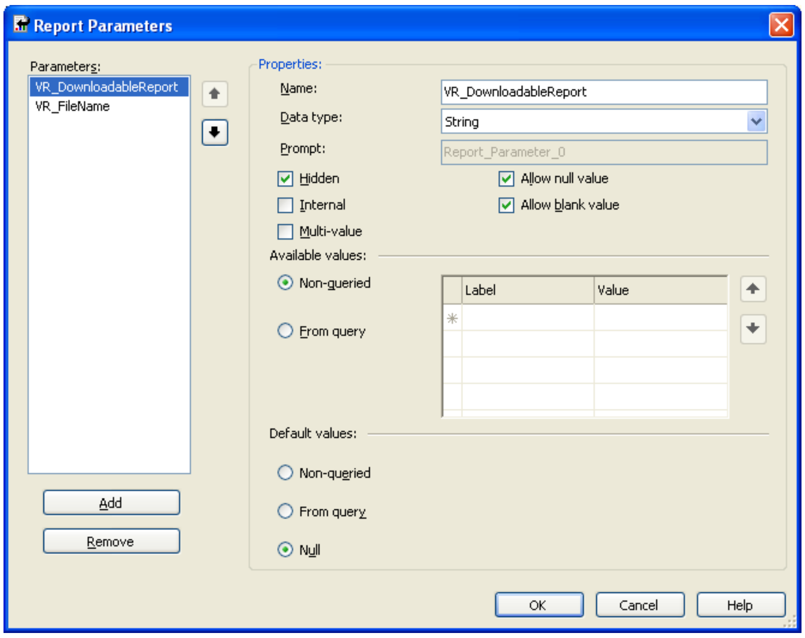
Parameter Name	Parameter Value
VR_FileName	="C:\Tfs\Reporting\Visma.Services.Reporting\Test\Reports\ElectronicReports\Downloadable.txt"

Parameter Setup

To tag a report as a downloadable report you must add some special parameters.

VR_DownloadableReport - Mandatory

This parameter marks the report as a downloadable report. It has no value and is merely a hidden marker parameter. Setup as shown below:



Report Parameters

Parameters:
- VR_DownloadableReport
- VR_FileName

Properties:

Name: VR_DownloadableReport
Data type: String
Prompt: Report_Parameter_0

[✓] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

Label	Value

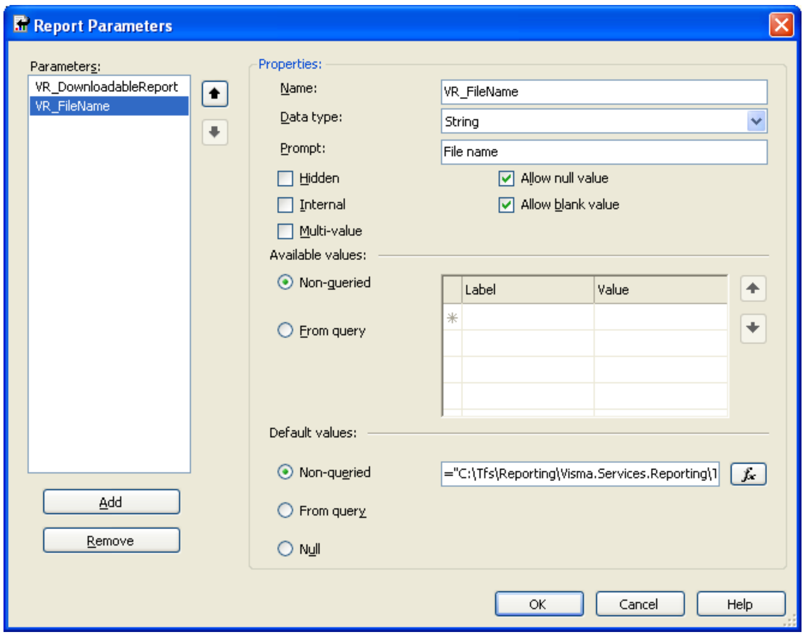
Default values: Non-queried From query [✓ Null]

The VR_Downloadable parameter can also be set up with Boolean as data type. In that way you can toggle the behaviour by setting the value to True or False.

If you attach a dataset to this parameter, the behaviour would be like if you prefixed the parameter with VR_StaticParam_ (the available values will be prefetched).

VR_FileName - Mandatory

This parameter provides the downloadable report with the path to the file. It is mandatory for the default implementation, but it is possible to override this functionality to fetch the report from any datasource. Setup as shown below:



Report Parameters

Parameters:
- VR_DownloadableReport
- VR_FileName

Properties:

Name: VR_FileName
Data type: String
Prompt: File name

[ ] Hidden [✓] Allow null value
[ ] Internal [✓] Allow blank value
[ ] Multi-value

Available values: [✓ Non-queried] From query

Label	Value

Default values: [✓ Non-queried] ="C:\Tfs\Reporting\Visma.Services.Reporting\T From query Null

Chapter 9

Deploying Report Packages

Topics:

- *Deploying Report Packages*
- *How to authenticate to report server to use deployment of reports*

Deploying Report Packages

The following section makes specific references to the Visma Business Report Package which is to serve as an example if you are creating your own package of reports to deploy.

When deploying reports they may either be deployed one at a time or all together in a package. A package is defined by an XML document, with the extension .rdlconfig, describing:

- Roles to be attached to the reports

  - If this is specified at package level, the roles are going to be used for all reports
  - If this is specified at report level (using "roles" attribute of the Report) then these roles are going to be used for that report, in addition to the roles specified at package level. Multiple roles can be specified by separating them with ";"
  - If role is specified at the report group level, then all roles in the group will inherit the roles set for the report group
- Application type – name of product the reports fetch data from

  - If this is specified at package level, this application type is going to be used for all reports
  - If this is specified at report level (using "applicationtypes" attribute of the Report) then this application type is going to be used for that report, overwriting the application type specified at package level. Multiple application types can be specified by separating them with ";"
- Resource files (e.g. language text files)
- Which documents the package consists of
- Structure of the deployed reports including ReportGroups (catalogs for reports)
- Description of Report or ReportGroup
- If report is visible or not

Please take into account that for successful deployment, your xml structure should reflect your folder structure. This is a requirement for deploying from the web client but should always be observed as best practice. Referring to the relevant parts of the example above, please consider the restriction:

Structure of rdlconfig and folders need to match
```
<Reports>
       <ReportGroup id="Business" name="Visma Business" >
            …
            <ReportGroup id="Customer" name="Customer" >
                …
                <Report id="AgeDistrList" name="Age Distribution List
 Customers"
                    filepath="Visma Business\Customer\Age Distribution List
 Customers.rdl" visible="true" />
…
```

For the example above you will need to have a folder structure that is having the main folder: Visma Business. Inside Visma Business folder you should find the Customer folder and in this folder you should see the Age Distribution List Customers. The filepath for this report must also match with the folder structure.

For version 9.0 of Reporting we are supporting giving roles to specific reports or report groups in addition to giving roles at the top of the rdlconfig, that is the roles that will be given to all reports in the package. Below is an example of a report package configuration file.

Example of rdlconfig file
```
<?xml version="1.0" standalone="yes" ?>
<ReportPackage>
   <!-- Optional. The roles of the users that has access to this report -->
   <Roles>
      <Role>TestRole</Role>
   </Roles>
   <!-- Optional. The application types the report belongs to -->
   <ApplicationTypes>
      <ApplicationType>Visma Business</ApplicationType>
   </ApplicationTypes>
```

```
    <!-- Optional. Resource files for localized reports -->
    <Resources>
       <Resource>en-uk.language.xml</Resource>
       <Resource>nb-no.language.xml</Resource>
    </Resources>
    <!-- Required. The report structure -->
    <Reports>
    <ReportGroup id="MainReportGroup" name="Main Report Group"
  displayname="MainReportGroup" >
       <ReportGroup id="ReportGroup1" name="Logistics Group"
  displayname="ReportGroup 1" roles="Logistics">
          <Description>Bla bla...ReportGroup1</Description>
          <Report id="Report1" name="Logistics" displayname="Report 1"
  filepath="LogisticsTest.rdl" visible="true" >
             <Image>background.jpg</Image>
          </Report>
          <Report id="Report2" name="LogisticsSub" displayname="Report 2"
  filepath="SubLogisticsTest.rdl" visible="false" >
             <Image>background2.jpg</Image>
          </Report>
          <ReportGroup id="SubReportGroup" name="Sub Logistics Group"
  displayname="SubReport Group" >
             <Report id="Report3" name="Logistics" displayname="Report 3"
  filepath="LogisticsTest.rdl" visible="true" >
                <Image>background.jpg</Image>
             </Report>
             <Report id="Report4" name="Logistics2" displayname="Report 4"
  filepath="LogisticsTest2.rdl" visible="true" roles="Finance;Accountant">
                <Image>background2.jpg</Image>
             </Report>
             <Report id="Report5" name="LogisticsSub2" displayname="Report 5"
  filepath="SubLogisticsTest2.rdl" visible="true" >
                <Image>background2.jpg</Image>
             </Report>
          </ReportGroup>
       </ReportGroup>
       <ReportGroup id="ReportGroup2" name="Finance Group"
  displayname="ReportGroup 2" >
          <!-- Report or sub-report -->
          <Report id="Report6" name="Finance" displayname="Report 6"
  filepath="FinanceTest.rdl" visible="true" roles="Finance" >
             <Image>background.jpg</Image>
          </Report>
           <Report id="Report7" name="Finance root" displayname="Report 7"
  filepath="FinanceTest.rdl" visible="true" roles="Clerk" >
          <Description>Bla bla...</Description>
          <Image>background.jpg</Image>
       </Report>
       </ReportGroup>
       <Report id="Report7" name="Finance root" displayname="Report 7"
  filepath="FinanceTest.rdl" visible="true" roles="Finance" >
          <Description>Bla bla...</Description>
          <Image>background.jpg</Image>
       </Report>
        </ReportGroup>
    </Reports>
 </ReportPackage>
```

How to authenticate to report server to use deployment of reports

It should be possible to do deployment both with and without VUD. You need to have a connection and authenticate a user on the report server. The code shown here is also found in the solution found here inTFS: *$/Visma.Services/Visma.Services.Reporting/MainBranch/Samples/ReportingWinClientIntegrations* .

Using VUD and an existing loginContext

Note: Code under is not a class, it only shows what assemblies that should be needed (not checked) and the methods that needs to be implemented on integrating product side.

```
Deployment - NOTE: not a real class, only methods needed
using Visma.Services.Reporting.Installer.Common;

 //You will need to have a loginContext
public DeploymentHelper(ILoginContext loginContext)
        {
            m_loginContext = loginContext;
        }

public StatusFeedback DeployReportPackage(byte[] zipFolder, string
 reportServerUri)
        {
            try
            {
                ApplicationInstance appInst = new
 ApplicationInstance(reportServerUri);
                IServerAuthentication communicationHelper = new
 ServerAuthenticationLoginContext(m_loginContext);
                var agent = communicationHelper.GetManagementAgent(appInst);
                List<ReportCatalogItem> reportCatalog =
 agent.DeployReportPackage(null, zipFolder);
                if (reportCatalog.Count > 0)
                {
                    // Empty reporting server cache - can take quite a lot of
 time, but is required to make new report visible
                    agent.RefreshServerCache();
                }
            }
            catch (Exception e)
            {
                string message = "Error deploying reports from common library.
 Exception: " + e.Message;
                return new StatusFeedback(StatusValue.Error, message);
            }
            return new StatusFeedback(StatusValue.Ok, "Deployment of report
 package is successful.");
        }
```

The deploymentHelper class is found inside the Visma.Services.Reporting.Installer.Common assembly.

Chapter 10

Visma Business Custom Functions

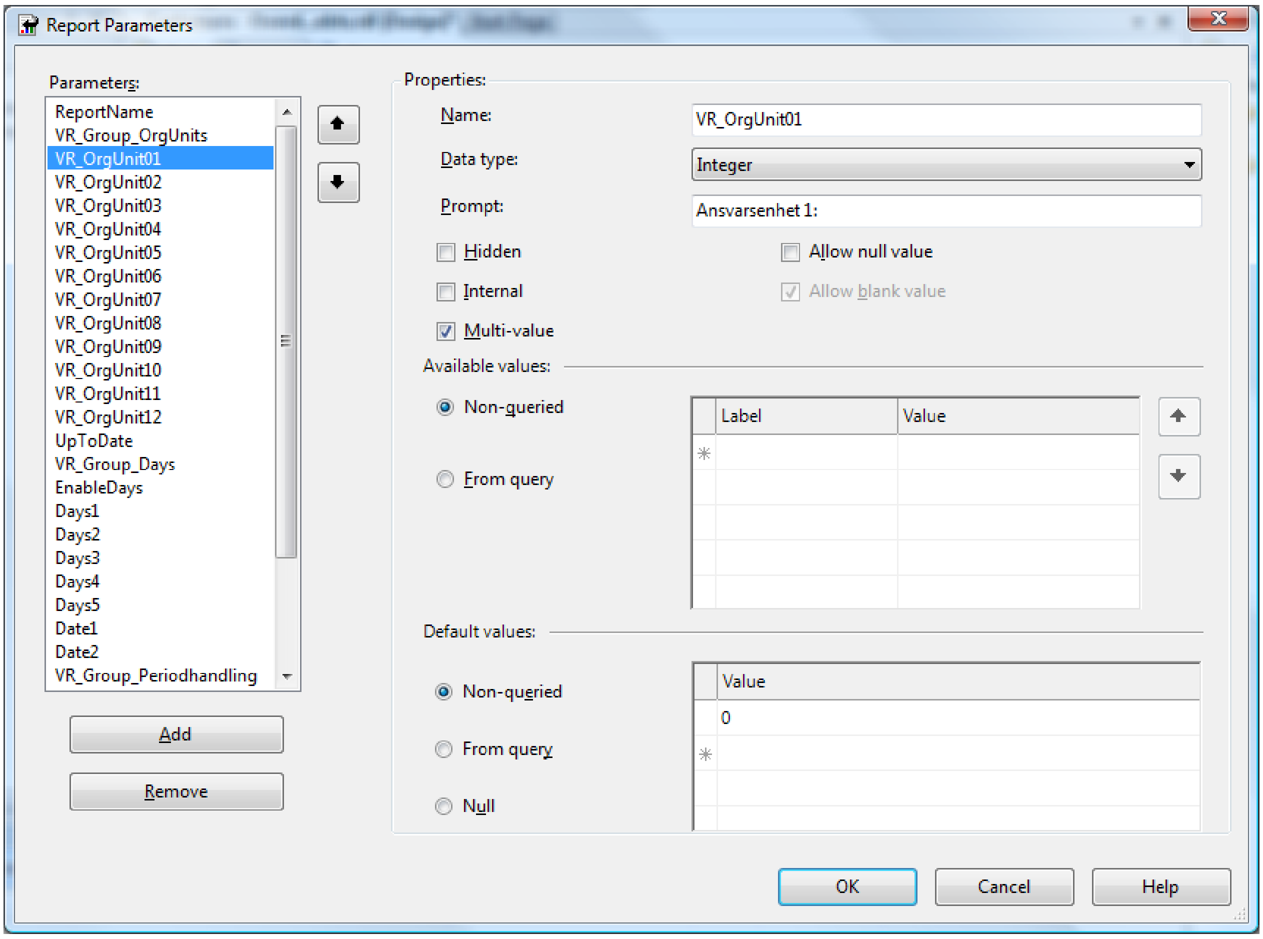
Topics:

- *Organizational Unit Parameters*
- *Organizational Unit in Dynamic SQL*
- *Date Handling, Visualize Company*

This chapter describes functions that are only useful together with Visma Business. To use the describe functions Application Type in the Management Console must be named “Visma Business".

Organizational Unit Parameters

For Visma Business reports it is a typical need to filter based on certain organizational unit values, e.g. department or project. To ease the design of such reports, we have predefined parameters for org.units 1-12 in Visma Business. Call the parameters for VR_OrgUnit01…VR_OrgUnit12. When running, Visma Reporting will automatically check against the Visma Business database whether the OrgUnitNN is in use. If not, it hides the parameter, else it shows it with the name registered for that OrgUnit, e.g. "Project" or "Employee". Set the OrgUnit parameter to right data type (Integer for 1-6, String for 7-12). In addition set the multi-value property to true (checked). When running the report, the user can select multiple OrgUnits values by either typing the values directly or by focusing on the parameter field opening a combobox showing all the alternative values. An example of an OrgUnit parameter definition is shown below:



In addition, it is important to add a tag to the SQL of the report to enforce the user filtering on OrgUnit01 – 12. At the end of the WHERE clause(s) where the OrgUnit filtering is wanted, add the tag /*OrgUnitValuesT*/. NB! The 'T' must be replaced with the table name alias that you prefix the fields in the WHERE clause.

For example if "WHERE M.AgJNo = @JNo AND M.AgEntNo = @EntNo AND M.PayedDt <= @UpToDate" then add tag /*OrgUnitValuesM*/.

Result: "WHERE M.AgJNo = @JNo AND M.AgEntNo = @EntNo AND M.PayedDt <= @UpToDate /*OrgUnitValuesM*/".

Before running the query, Visma Reporting will substitute the tag with a filter according to user request.

Note: See also *Organizational Unit in Dynamic SQL* (see page 67).

Organizational unit limitations on user

Client may have set up limitations on user usage of OrgUnits01 – 12. By adding /*OrgUnitLimitsT*/ to the WHERE clause of the SQL statement these user limitations are taken into account when running the report. Same rules for ‘T’ here as for /*OrgUnitValuesT*/.

Note: See also *Organizational Unit in Dynamic SQL* (see page 67).

Organizational Unit in Dynamic SQL

If you need organizational units to be part of dynamic SQLs, you need to use another tag so that the application knows that the values should be inside a dynamic SQL. This is the issue both for organizational unit parameters and organizational unit limitations. The tags to be used in these cases are:

/*EncodedOrgUnitValuesT*/
/*EncodedOrgUnitLimitsT*/

Voucher Search is an example of a report that uses dynamic SQL:

set @WhereSQL = 'WHERE /*EncodedOrgUnitValuesT*//*EncodedOrgUnitLimitsT*/';

Here the where clause is build up while running the report and the values of the organizational units should then be put inside this string. Since some of the organizational unit values are strings (R7-R12) we will need to have double quotes around these values and this is the reason for these specific tags in dynamic SQLs.

Date Handling, Visualize Company

Date values are stored in the Visma Business database as integers in the format yyyyMMdd. But we advice designers of Visma Business reports to define date parameters with data type DateTime. This enables the calendar control facilities in Visma Reporting GUI, but the date is still converted to integer when being used by the SQL. Some SQL examples for parameter UpToDate:

- Comparing to a date field: T.ValDt <= @UpToDate
- Comparing manipulated parameter to a date field: T.DueDt < CONVERT (CHAR (8), DATEADD (DAY, -30, CONVERT (DATETIME, CONVERT(char(8) , @UpToDate)))

This example converts UpToDate parameter to a DateTime, subtracts 30 days, and converts it back before comparing UpToDate to the database field.

Visma Reporting expects dates from Visma Business to be of type Integer. If dates used are of type Date (for instance when used in external tables), prefix the parameter name with VR_Date_ to avoid any special formatting. To summarize:

- Parameters of type Date in Visma Business are expected to be Integers
- Parameters of type Date and name date starts with VR_Date_ in Visma Business are expected to be Dates

Visualize company number and name

Often it is desired to show company name and/or company no. on report. Use Parameters VR_CompanyNo and VR_CompanyName with data type string and set to hidden. When these parameters are detected, company no and name is set on the parameter for use in the report.

Chapter 11

Visma OnDemand Reports

Topics:

- *How to Activate the Data Extension in MS Report Designer*
- *Data Source Definition*
- *Dataset Query Definition*
- *Passing Date Time Parameters to a Stored Procedure*
- *Send a Report as PDF to Email Recipients*
- *Export a Report for Sending to Visma Print Service (Maventa)*
- *Export a Report for Sending to Itella Print Service*
- *Run-time Translations in a Report*
- *'GlobalizationString' field*

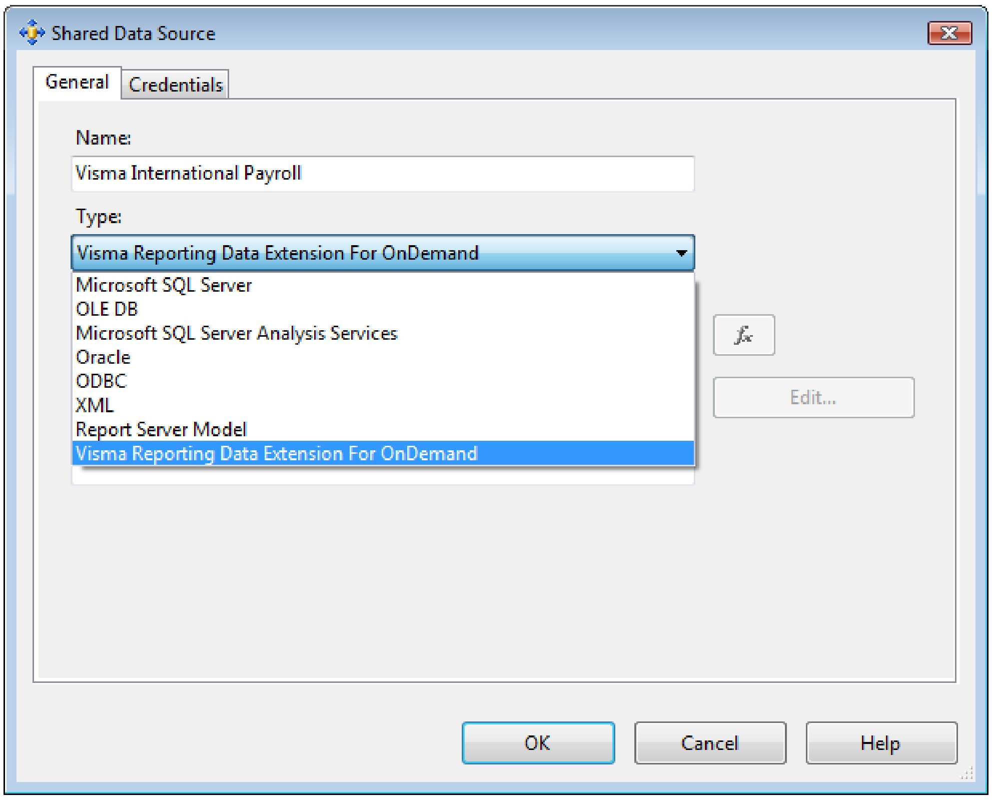
This chapter describes functions that is specific to report development for Visma OnDemand applications.

How to Activate the Data Extension in MS Report Designer

Be sure that all instances of MS Report Designer are closed. Copy the file "Visma.Services.Reporting.DataExtension.OnDemand.dll" to IDE folder for binaries, i.e. "C:\Program Files (x86)\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies". In the same folder, modify "RSReportDesigner.config" to include the data extension:

```
<Data>
...
<Extension Name="VROD"
 Type="Visma.Services.Reporting.DataExtension.OnDemand.DataSetConnection,
 Visma.Services.Reporting.DataExtension.OnDemand" />
...
</Data>

<Designer>
...
<Extension Name="VROD"
 Type="Microsoft.ReportingServices.QueryDesigners.GenericQueryDesigner,
 Microsoft.ReportingServices.QueryDesigners"/>
...
</Designer>
```

Open an instance of MS Reporting Designer. When creating or modifying a data source, you should see the data source type "Visma Reporting Data Extension For OnDemand" listed in the dropdown.



Name:
Visma International Payroll

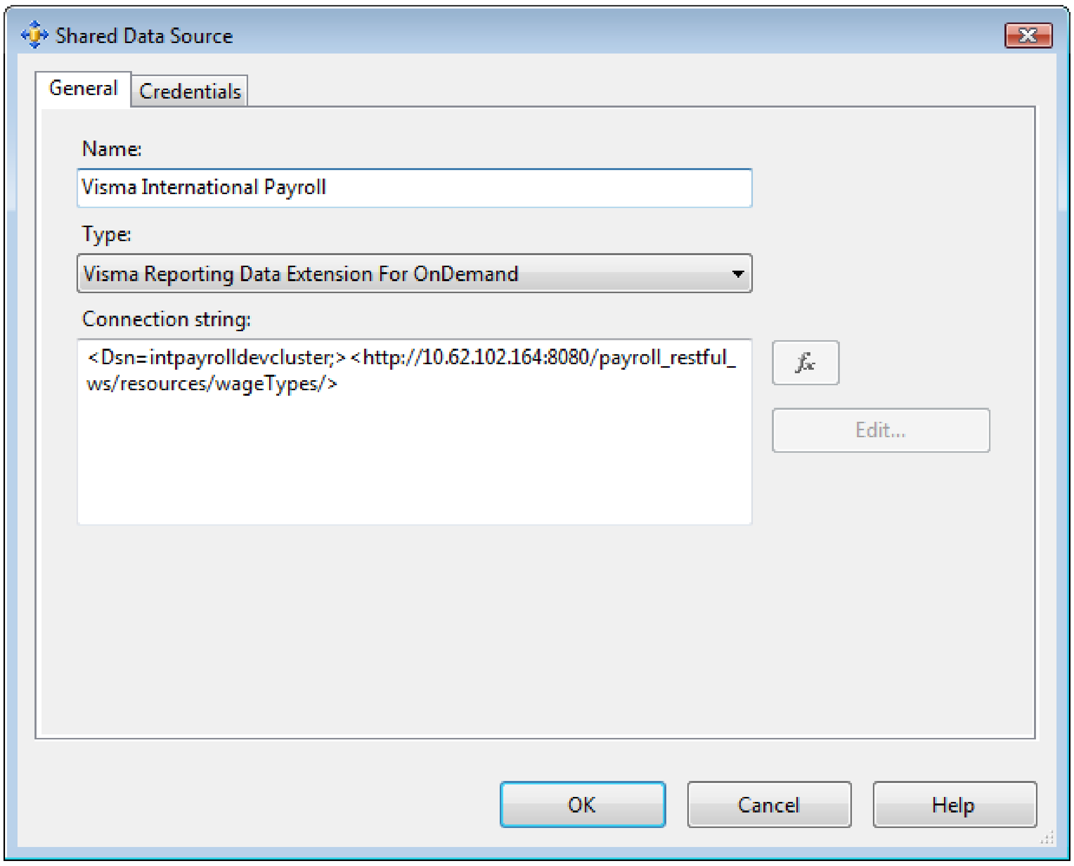
Type:
Visma Reporting Data Extension For OnDemand

Microsoft SQL Server
OLE DB
Microsoft SQL Server Analysis Services
Oracle
ODBC
XML
Report Server Model
Visma Reporting Data Extension For OnDemand

Data Source Definition

All datasets must choose a data source that uses the data source type "Visma Reporting Data Extension For OnDemand". The connection string must contain two sections: One section for the ODBC definition for the application's database (stored procedure queries), and one section for the web service connection to the application's RESTful web services (web service queries).

Format of the connection string: <#ODBC data source#><#Web service URI#>.



Name:
Visma International Payroll

Type:
Visma Reporting Data Extension For OnDemand

Connection string:
<Dsn=intpayrolldevcluster;><http://10.62.102.164:8080/payroll_restful_ws/resources/wageTypes/>

The name of the data source must correspond to the application type name of the OnDemand application you are creating the report for.

Dataset Query Definition

In Visma OnDemand reports, you are allowed to do queries against stored procedures or RESTful web services. No direct SQL are allowed due to security reasons. If you use keywords like "select", "update", "insert", "delete" etc, you will get an error message when you try to run the report within Visma Reporting. Every stored procedure or web service method must include a parameter for company id and language code. The report must include the mandatory report parameters "VR_CompanyId" and "VR_Language", and these parameters must be linked to the corresponding parameters for the stored procedure/web service method.

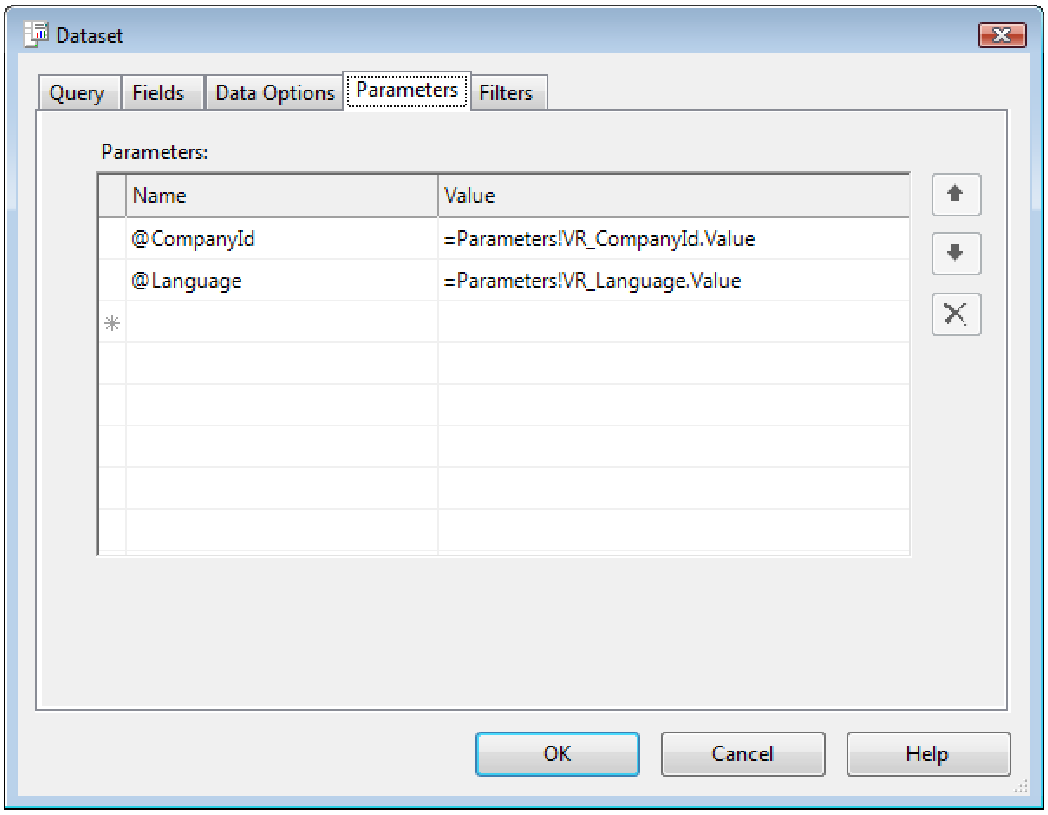
Syntax of a stored procedure query will look like this:

```
<Method Name="sp_vipWageTypeList_Example" Type="StoredProcedure">
   <Parameters>
      <Parameter Name="CompanyId">@CompanyId</Parameter>
      <Parameter Name="Language">@Language</Parameter>
   </Parameters>
</Method>
```

Syntax of a web service query will look like this:

```
<Method Name="postVipCompanyIdWageType" Type="WebService">
   <Parameters>
      <Parameter Name="CompanyId">@CompanyId</Parameter>
      <Parameter Name="Language">@Language</Parameter>
   </Parameters>
</Method>
```

The named parameters are linked in an ordinary way:



Parameters:

Name	Value
@CompanyId	=Parameters!VR_CompanyId.Value
@Language	=Parameters!VR_Language.Value

Passing Date Time Parameters to a Stored Procedure

To pass a DateTime parameter value to a stored procedure, set the parameter Type attribute to DateTime. In this way the parameter value is formatted as an Integer value on the format yyyyMMdd. The stored procedure parameter must be declared as an Integer parameter, and conversion to SQL DateTime must be done in the stored procedure. See below for an example.

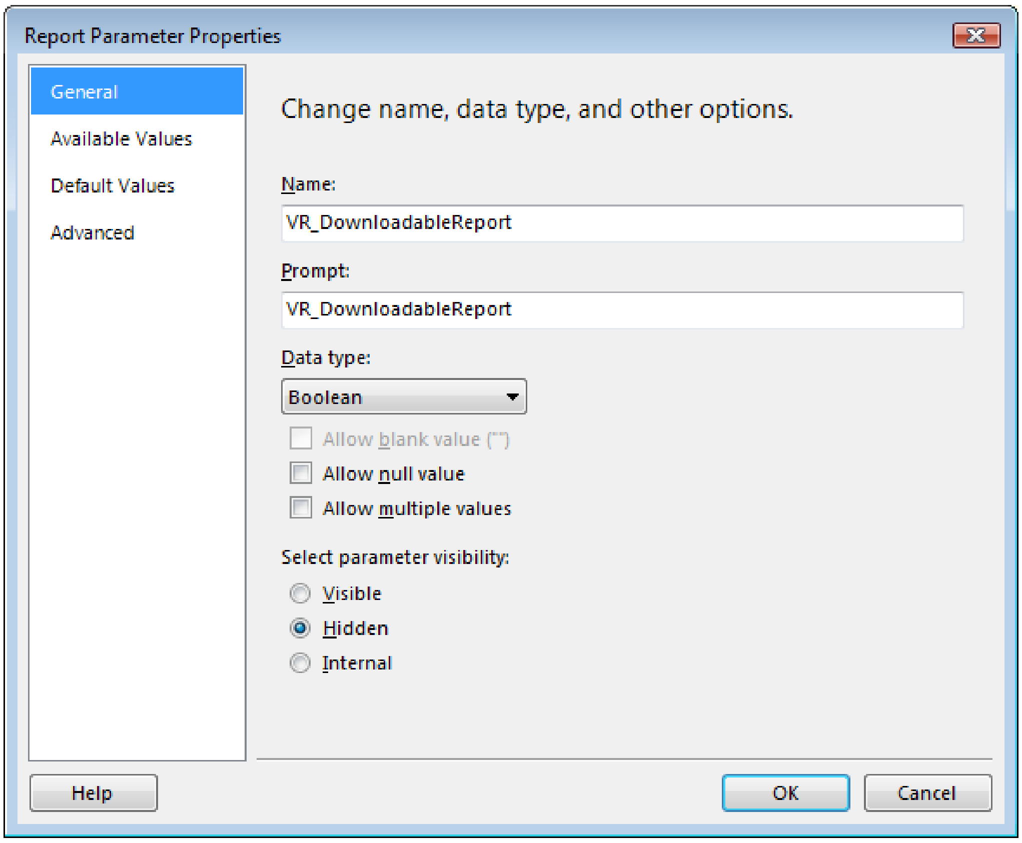
```
<Method Name="sp_vipPensionDeclarationTyEL" Type="StoredProcedure">
   <Parameters>
   <Parameter Name="P_StartDate" Type="DateTime">@P_StartDate</Parameter>
   <Parameter Name="P_CompanyId">@VR_CompanyId</Parameter>
   <Parameter Name="P_Language">@VR_Language</Parameter>
   </Parameters>
</Method>
```

Send a Report as PDF to Email Recipients

It is possible to send a report as PDF to different e-mail recipients, one for each specified recipient (i.e for sending pay slips to employees or absence reports to department managers). The report for each recipient is containing data only relevant for the recipient. Here is a description of what you have to do.

Parameter VR_DownloadableReport - Mandatory

To be able to send a report by e-mail, you must add a hidden boolean parameter called VR_DownloadableReport to the report. Set its default value to True.


Report Parameter Properties

General
Available Values
Default Values
Advanced

Change name, data type, and other options.

Name: VR_DownloadableReport

Prompt: VR_DownloadableReport

Data type: Boolean

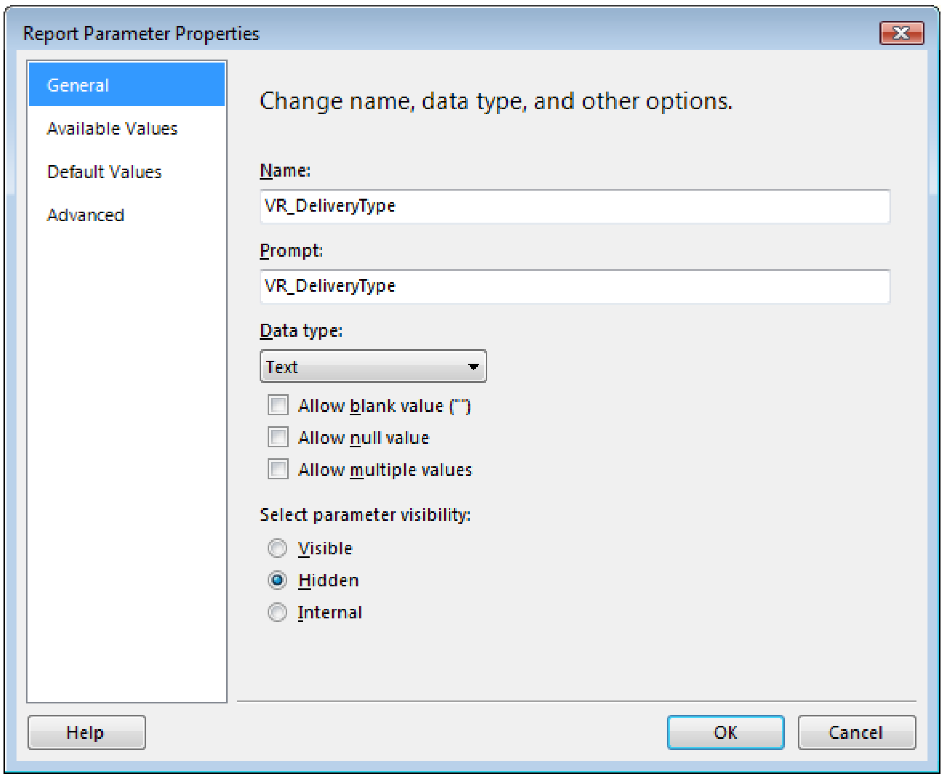
- [ ] Allow blank value ("")
- [ ] Allow null value
- [ ] Allow multiple values

Select parameter visibility: Visible [✓ Hidden] Internal

Help | OK | Cancel

Parameter VR_DeliveryType - Mandatory

To be able to send a report by e-mail, you must add a hidden boolean parameter called VR_DeliveryType to the report. Set its default value to "VR_EMail".


Report Parameter Properties

General
Available Values
Default Values
Advanced

Change name, data type, and other options.

Name: VR_DeliveryType

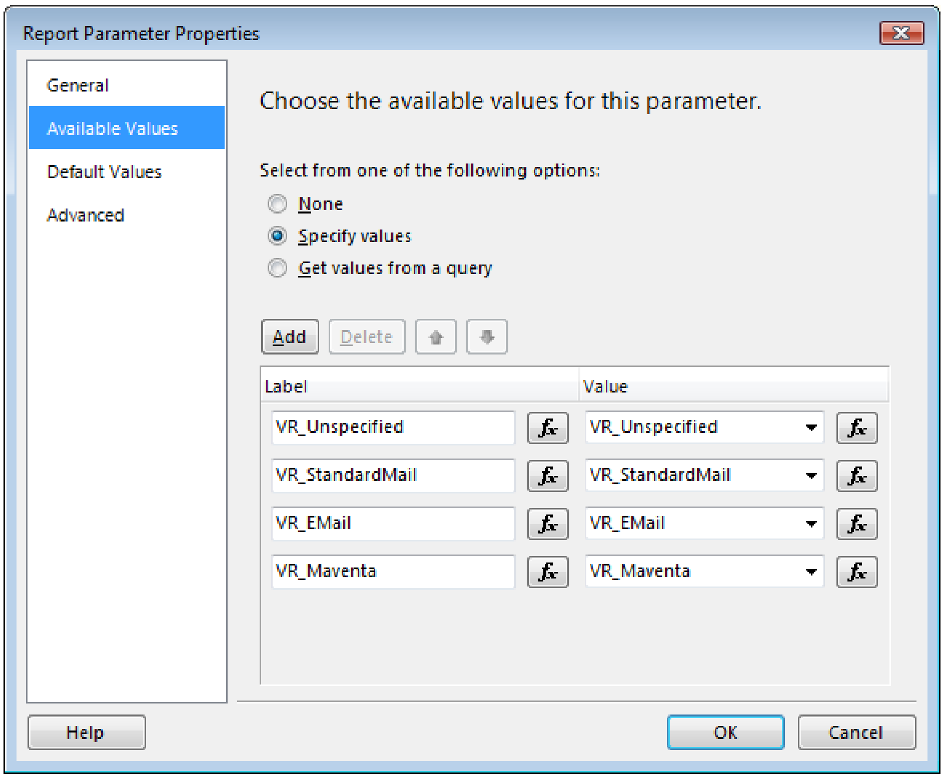
Prompt: VR_DeliveryType

Data type: Text

- [ ] Allow blank value ("")
- [ ] Allow null value
- [ ] Allow multiple values

Select parameter visibility: Visible [✓ Hidden] Internal

Help | OK | Cancel


Report Parameter Properties

Choose the available values for this parameter.

Select from one of the following options:

( ) None
(●) Specify values
( ) Get values from a query

Label	Value
VR_Unspecified	VR_Unspecified
VR_StandardMail	VR_StandardMail
VR_EMail	VR_EMail
VR_Maventa	VR_Maventa

PDFPageInfo Meta Tag

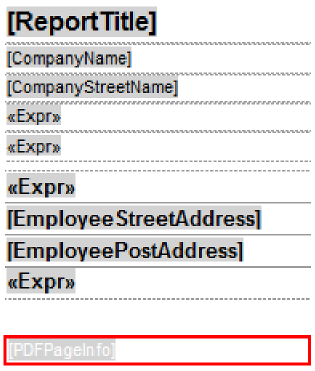
Second, you must define a meta tag for how Visma Reporting will split the report into separate PDFs. The meta tag could be a text box with an expression containing an xml with the id of each mail recipient (i.e. employee number, social security number or department id).

Xml:

```
<PDFPageInfo>
  <Id></Id>
</PDFPageInfo >
```

Expression example:

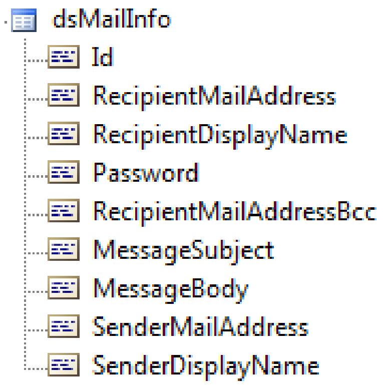
```
=
IIF
(
   Fields!EmployeeEMail.Value <> "",
   "<PDFPageInfo><Id>" + Fields!EmpNo.Value + "</Id></PDFPageInfo>",
   ""
)
```



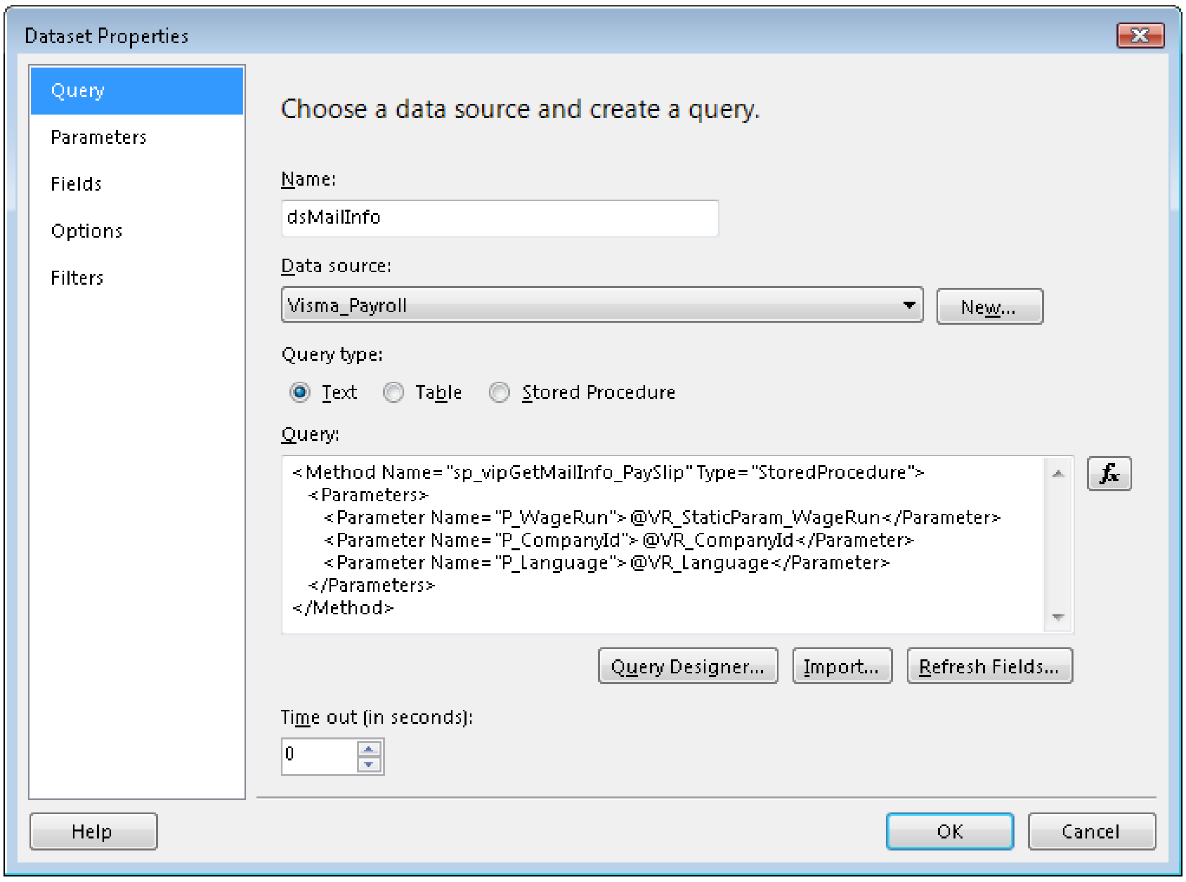
To hide the meta tag from the user, just make sure the fore color of the textbox is equal to the background color of the report, which is usually white. The textbox could be placed anywhere on the report, but the best place is somewhere in the header.

Dataset dsMailInfo - Mandatory

This dataset must provide information about e-mail settings for each id from the PDFPageInfo meta tag. If the id from the meta tag represents the employee code for instance, the dataset must contain e-mail address, encryption password, message subject, message body etc for each employee. See below for mandatory columns in the dataset.



dsMailInfo
- Id
- RecipientMailAddress
- RecipientDisplayName
- Password
- RecipientMailAddressBcc
- MessageSubject
- MessageBody
- SenderMailAddress
- SenderDisplayName



Choose a data source and create a query.

Name: dsMailInfo

Data source: Visma_Payroll

Query type: [✓ Text] Table Stored Procedure

Query:

```
<Method Name="sp_vipGetMailInfo_PaySlip" Type="StoredProcedure">
  <Parameters>
    <Parameter Name="P_WageRun">@VR_StaticParam_WageRun</Parameter>
    <Parameter Name="P_CompanyId">@VR_CompanyId</Parameter>
    <Parameter Name="P_Language">@VR_Language</Parameter>
  </Parameters>
</Method>
```

Time out (in seconds): 0

- **Id**

  This is the Id from the PDFPageInfo meta tag. For example, this could be the employee code if the report should be sent to each employee. If it is a report to be sent to department managers, the id should represent each department manager.
- **RecipientMailAddress**

  This is the mail address of the recipient.

- **RecipientDisplayName**

  This is the display name of the recipient.
- **RecipientMailAddressBcc**

  If you need a copy of all the sent e-mails, you could specify the mail address for the copies in this column. This could be a common mail address for all recipients or an additional mail address for each recipient. Set the column to empty if you don't need a copy.
- **Password**

  This is the password that is being used when encrypting the PDF's. This could be a common password for all recipients or a specified password for each recipient.
- **MessageSubject**

  This column contains the text for the subject field in the mail message.
- **MessageBody**

  This column contains the text for the body field in the mail message.
- **SenderMailAddress**

  This column represents the reply mail address. Notifications of any delivery failures will also be sent to this address. If this value is empty, default value from the server configuration file will be used.
- **SenderDisplayName**

  This column defines how the sender's mail address is displayed in the mail message. If this value is empty, default value from the server configuration file will be used.

Server Configuration File - Mail Server Settings

You must add a configuration node called "MailServer" to the server configuration file. This node contains information about the mail server that is to be used for sending mail.

```
<VismaSettings>
   <node name="Visma">
      <node name="Services">
         <node name="Reporting">
            <node name="MailServer">
               <setting name="Server">mail.datakraftverk.no</setting>
               <setting name="Port">25</setting>
               <setting name="UserName"></setting>
               <setting name="Password"></setting>
               <setting name="UseSSL">false</setting>
               <setting name="SenderMailAddress">noreply@visma.com</setting>
               <setting name="SenderDisplayName">Visma Software International
 AS</setting>
            </node>
         </node>
      </node>
   </node>
</VismaSettings>
```

Note: This feature requires that the reporting server’s user account has access rights to the temporary folder. Temporary folders and files are created.

Export a Report for Sending to Visma Print Service (Maventa)

It is also possible to export a report as a ZIP package with PDFs to send to Visma Print Service for ordinary mail delivery. The PDF for each recipient is containing data only relevant for the recipient. Here is a description of what you have to do.

Parameter VR_DownloadableReport - Mandatory

See previous chapter *Send a Report as PDF to Email Recipients* (see page 71) for details.

Parameter VR_DeliveryType - Mandatory

To be able to send a report by e-mail, you must add a hidden boolean parameter called VR_DeliveryType to the report. Set its default value to "VR_Maventa". See previous chapter *Send a Report as PDF to Email Recipients* (see page 71) for further details.

PDFPageInfo Meta Tag

See previous chapter *Send a Report as PDF to Email Recipients* (see page 71) for details.

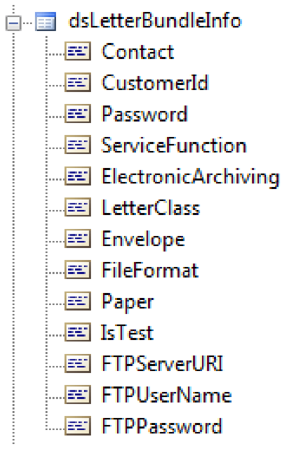
> **Note:** This feature requires that the reporting server's user account has access rights to the temporary folder. Temporary folders and files are created.

Export a Report for Sending to Itella Print Service

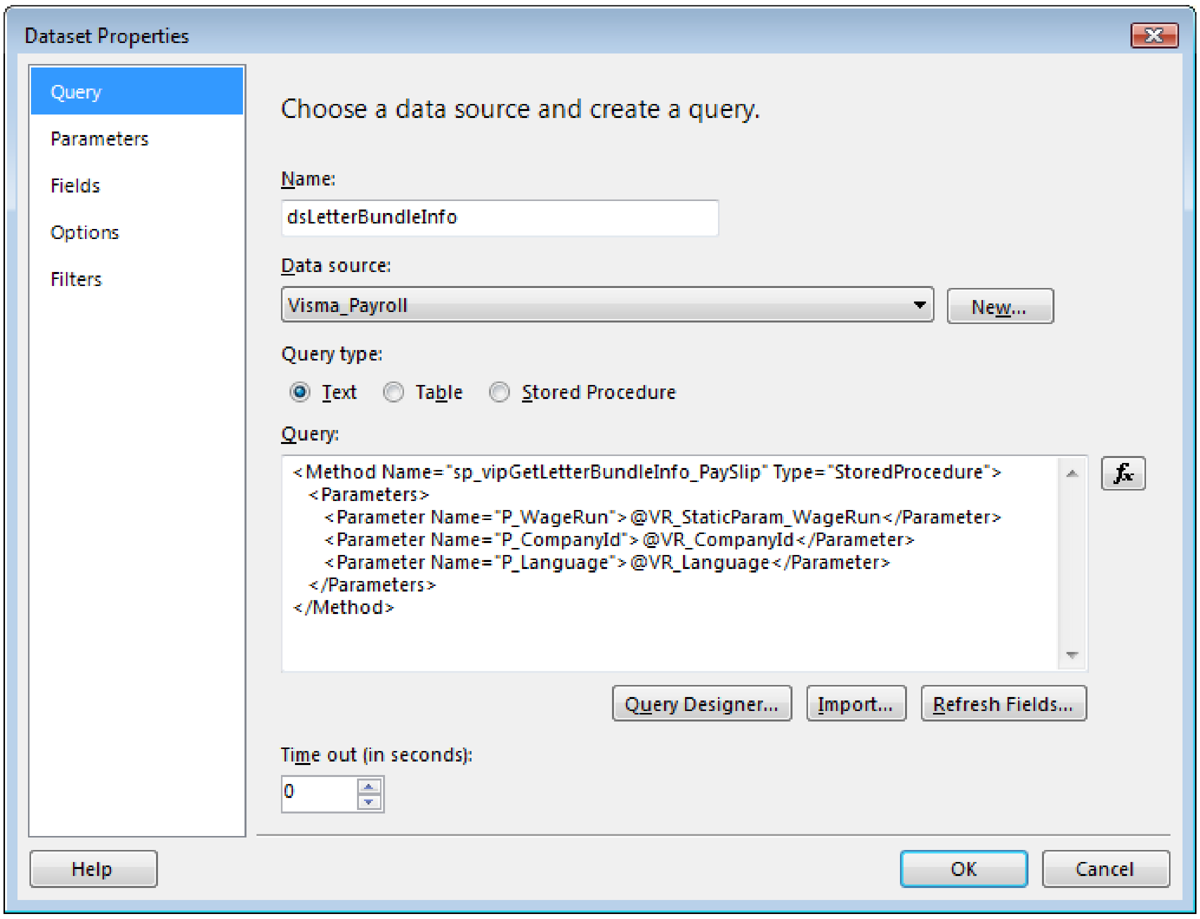
It is also possible to export a report as a ZIP package with a PDF and XML to send to Itella Print Service for delivery by mail or Itella e-mail. The PDF is containing data for all the recipients, and the XML file is describing where to find data for each recipient in the PDF. Here is a description of what you have to do.

Dataset dsLetterBundleInfo - Mandatory

This dataset must provide information about the letter bundle settings for the Itella package. See below for mandatory columns in the dataset.



dsLetterBundleInfo
- Contact
- CustomerId
- Password
- ServiceFunction
- ElectronicArchiving
- LetterClass
- Envelope
- FileFormat
- Paper
- IsTest
- FTPServerURI
- FTPUserName
- FTPPassword



Name: dsLetterBundleInfo

Data source: Visma_Payroll

Query type: [✓ Text] Table Stored Procedure

Query:

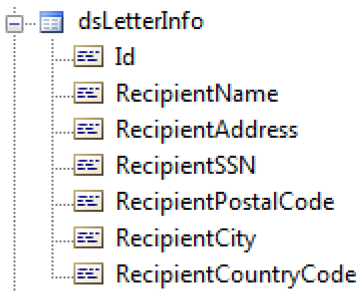
```
<Method Name="sp_vipGetLetterBundleInfo_PaySlip" Type="StoredProcedure">
  <Parameters>
    <Parameter Name="P_WageRun">@VR_StaticParam_WageRun</Parameter>
    <Parameter Name="P_CompanyId">@VR_CompanyId</Parameter>
    <Parameter Name="P_Language">@VR_Language</Parameter>
  </Parameters>
</Method>
```

Time out (in seconds): 0

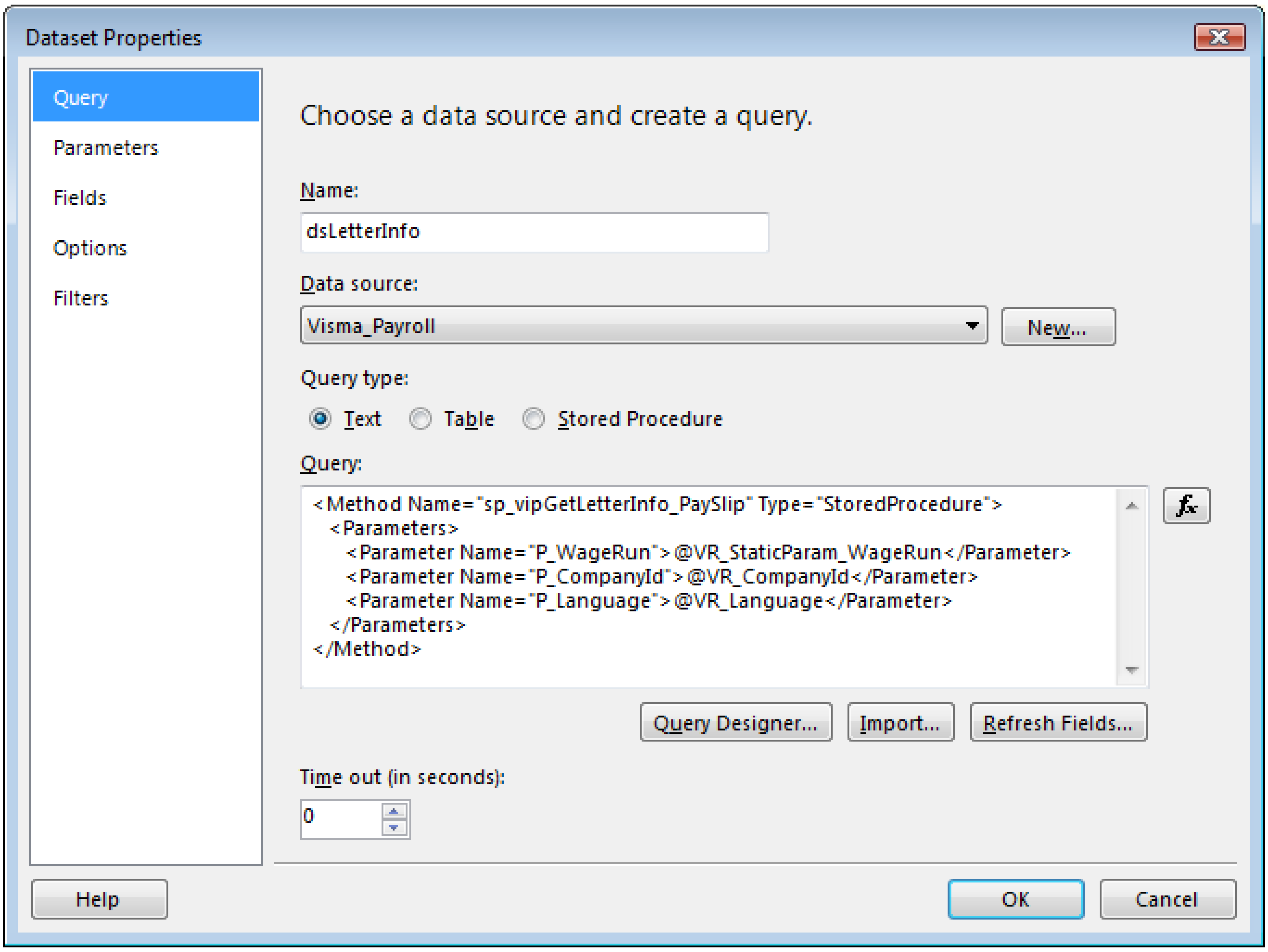
- **Contact:** contact person for the letter bundle
- **CustomerId:** customer id provided by Itella
- **Password:** password provided by Itella
- **ServiceFunction:** 0=normal production material
- **ElectronicArchiving:** T=printing, no archiving
- **LetterClass:** 1=priority, 2=economy
- **Envelope:** S=iPost standard envelope, Q=printing abroad
- **FileFormat:** 0=standard format
- **Paper:** 0=standard paper for iPost letters, W=PDF color printing
- **IsTest:** false=used for production, true=used for testing against Itella Print Service
- **FTPServerURI:** enter URI for Itella FTP Server (including folder) for automatic delivery to Itella FTP Server. If left blank, Itella ZIP package is downloaded to the client for manual uploading to Itella FTP Server
- **FTPUserName:** Itella FTP Server User Name
- **FTPPassword:** Itella FTP Server Password

Dataset dsLetterInfo - Mandatory

This dataset must provide information about letter info settings for each id from the PDFPageInfo meta tag. If the id from the meta tag represents the employee code for instance, the dataset must contain address, name, city etc. for each employee. See below for mandatory columns in the dataset.



dsLetterInfo
- Id
- RecipientName
- RecipientAddress
- RecipientSSN
- RecipientPostalCode
- RecipientCity
- RecipientCountryCode



Choose a data source and create a query.

Name: dsLetterInfo

Data source: Visma_Payroll

Query type: [✓ Text] Table Stored Procedure

Query:

```
<Method Name="sp_vipGetLetterInfo_PaySlip" Type="StoredProcedure">
  <Parameters>
    <Parameter Name="P_WageRun">@VR_StaticParam_WageRun</Parameter>
    <Parameter Name="P_CompanyId">@VR_CompanyId</Parameter>
    <Parameter Name="P_Language">@VR_Language</Parameter>
  </Parameters>
</Method>
```

Time out (in seconds): 0

- **Id:** the Id from the PDFPageInfo meta tag. For example, this could be the employee code if the report should be sent to each employee. If it is a report to be sent to department managers, the id should represent each department manager
- **RecipientName:** the name of the recipient
- **RecipientAddress:** the address of the recipient
- **RecipientSSN:** the social security number of the recipient
- **RecipientPostalCode:** the postal code of the recipient
- **RecipientCity:** the city of the recipient
- **RecipientCountryCode:** the country code of the recipient

Parameter VR_DownloadableReport - Mandatory

See previous chapter *Send a Report as PDF to Email Recipients* (see page 71) for details.

Parameter VR_DeliveryType - Mandatory

To be able to send a report by e-mail, you must add a hidden boolean parameter called VR_DeliveryType to the report. Set its default value to "VR_Itella". See previous chapter *Send a Report as PDF to Email Recipients* (see page 71) for further details.

PDFPageInfo Meta Tag

See previous chapter *Send a Report as PDF to Email Recipients* (see page 71) for details.

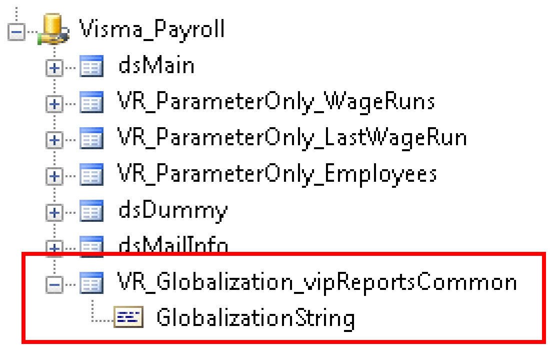
Note: This feature requires that the reporting server's user account has access rights to the temporary folder. Temporary folders and files are created.

Run-time Translations in a Report

Sometimes it is required that report labels must be translated on the fly based on language codes in the report data. As an example this could be a pay slip report that must be translated to different languages based on the language code defined on the employee. Here is a description of how to implement this functionality in your reports.

Dataset VR_Globalization_<domain> - Mandatory

First you will have to define one or more datasets to hold the globalization strings. The datasets should contain one field only, 'GlobalizationString'. This field will be populated automatically by Visma Reporting with the translations for the given domain. For example, the dataset VR_Globalization_vipReportsCommon will be populated with values from Visma Reporting resource files belonging to 'vipReportsCommon' domain.


Visma_Payroll
dsMain
VR_ParameterOnly_WageRuns
VR_ParameterOnly_LastWageRun
VR_ParameterOnly_Employees
dsDummy
dsMailInfo
VR_Globalization_vipReportsCommon
GlobalizationString

However, you will need to provide a dummy query for the dataset(s). In that way you are able to run the report in the report designer (BIDS). A dummy query could look like this:

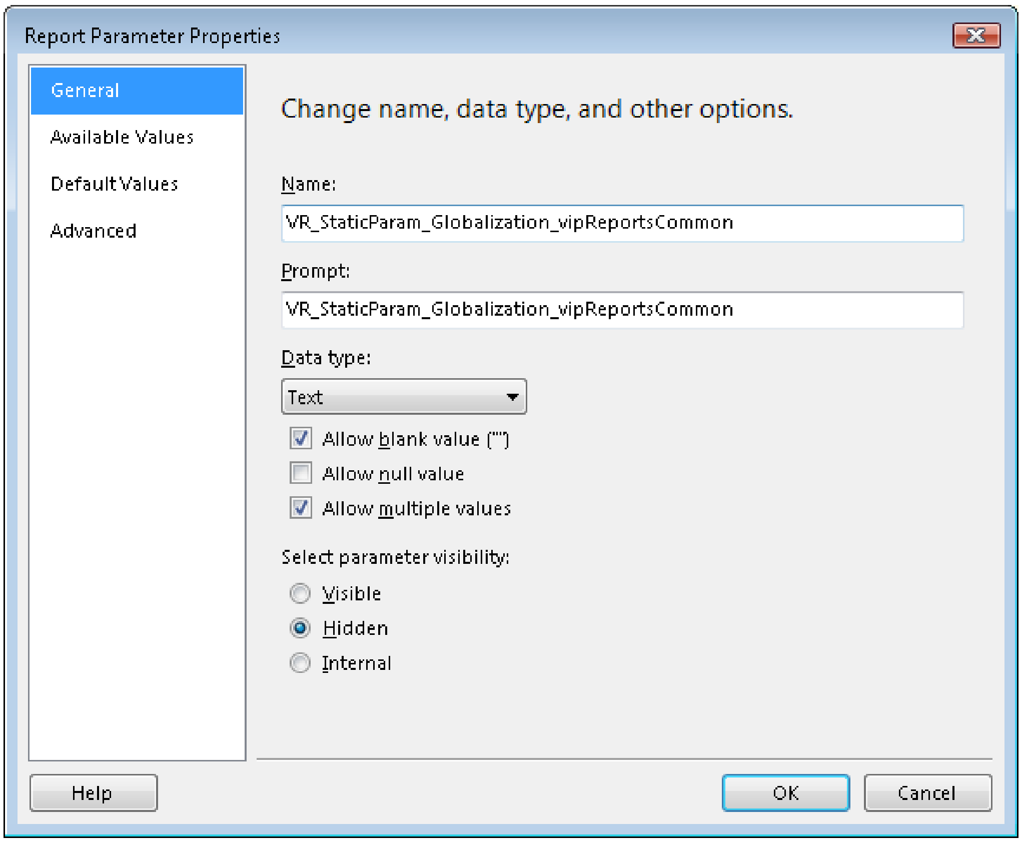
```
<Method Name="sp_vipGetDummyTranslations" Type="StoredProcedure">
   <Parameters>
      <Parameter Name="P_CompanyId">@VR_CompanyId</Parameter>
      <Parameter Name="P_Language">@VR_Language</Parameter>
   </Parameters>
</Method>
```

Then the stored procedure could contain the following sql:

```
select 'en-GB|TXT_Text=Text' as GlobalizationString
union all
select 'fi-FI|TXT_Text=Teksti' as GlobalizationString
union all
select 'nb-NO|TXT_Text=Tekst' as GlobalizationString;
```

Parameter VR_StaticParam_Globalization_<domain> - Mandatory

In the current version of the report viewer, we are not able to do lookups in the datasets using an expression or a custom code block (the lookup functions will be introduced in the report viewer that are compatible with SSRS 2008 R2). Therefore, we'll need a hidden multi-value parameter linked to the corresponding globalization dataset.



Change name, data type, and other options.

Name: VR_StaticParam_Globalization_vipReportsCommon

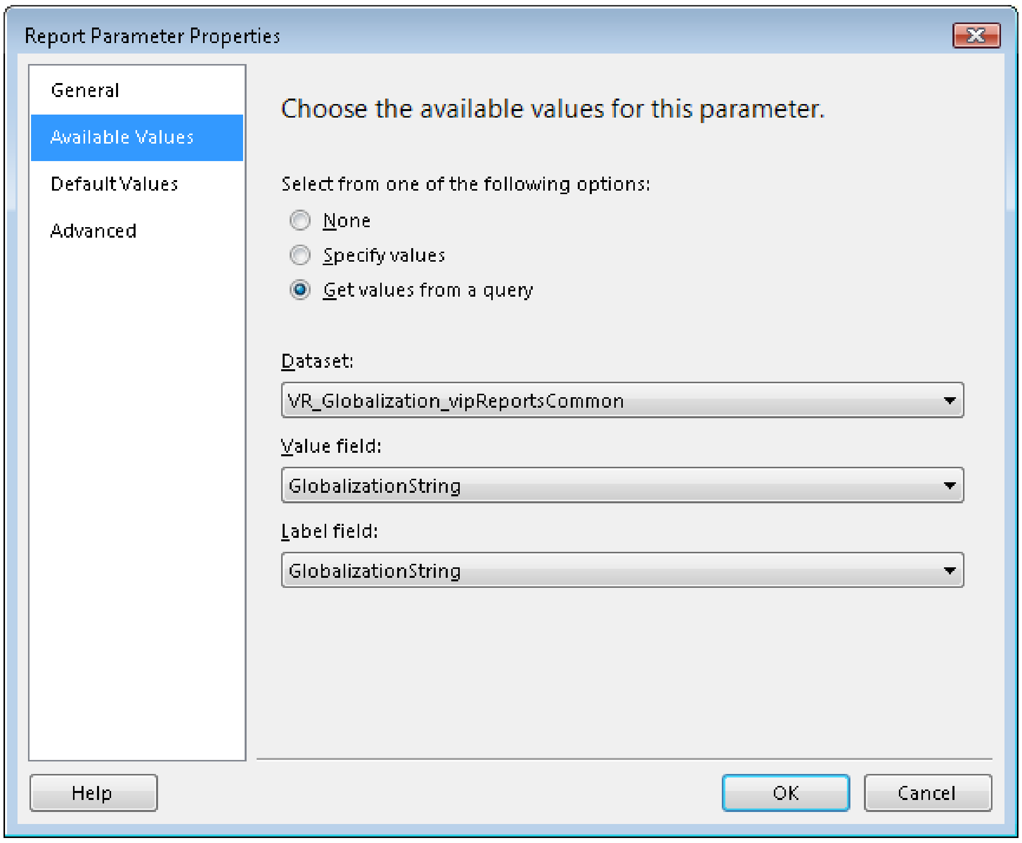
Prompt: VR_StaticParam_Globalization_vipReportsCommon

Data type: Text

[✓] Allow blank value ("")
[ ] Allow null value
[✓] Allow multiple values

Select parameter visibility:
Visible [✓ Hidden] Internal

In the available values section, set both 'Label' and 'Value' to the 'GlobalizationString' field.



Choose the available values for this parameter.

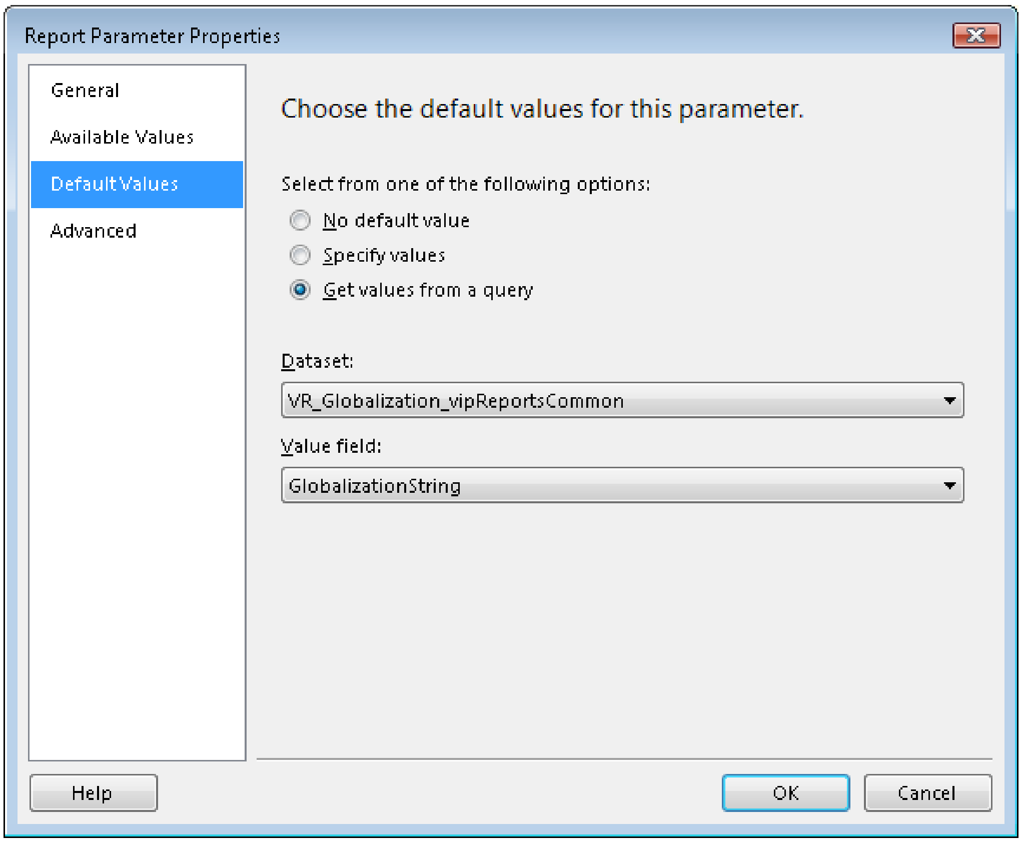
Select from one of the following options:
None Specify values [✓ Get values from a query]

Dataset: VR_Globalization_vipReportsCommon

Value field: GlobalizationString

Label field: GlobalizationString

In the default values section, set ‘Value’ to the ‘GlobalizationString’ field.



Choose the default values for this parameter.

Select from one of the following options:

No default value
Specify values
[✓ Get values from a query]

Dataset: VR_Globalization_vipReportsCommon

Value field: GlobalizationString

By using a parameter, we are able to use a custom code block to iterate through its values (see code example in the next section *’GlobalizationString’ field* (see page 81)).

’GlobalizationString’ field

’GlobalizationString’ field

The ‘GlobalizationString’ field will be on format <languageCode>|<textCode>=<translatedText>, so you will need to parse this in a custom code block within the report. Here is an example:

```
Public Function GetTranslatedText(parameterTranslations As Parameter,
 languageCode As String, textCode As String) as String
            Dim i As Integer
            Dim pos As Integer
            Dim key As String
            Dim parameterValue As String
            Dim translatedText As String

            key = languageCode & "|" & textCode & "="

            translatedText = textCode

            For i = 0 to Ubound(parameterTranslations.Value)
                      parameterValue = parameterTranslations.Value(i)
                      If (InStr(1, LCase(parameterValue), LCase(key)) > 0)
 Then
                               pos = InStr(1, parameterValue, "=")

                               translatedText = Right(parameterValue,
 Len(parameterValue) - pos)
                               Exit For
```

```
                    End If
          Next i

          GetTranslatedText = translatedText
End Function
```

This function could then be called from an expression in the report in this way:

```
=Code.GetTranslatedText(Parameters!
VR_StaticParam_Globalization_vipReportsCommon, "fi-FI", "TXT_Quantity")
```

Chapter 12

SQL Server Analysis Services Reports

Topics:

- *Working With Parent-Child Hierarchies*

This chapter describes guidelines that are applicable to reports created on a SSAS (Sql Server Analysis Services) data source.

Working With Parent-Child Hierarchies

If you need to display the members of a parent-child hierarchy recursively you need to add the following property to your connection string from VUD: **DbpropMsmdFlattened2=true**. This is the description of this property taken from MSDN:

```
Usage
      Optional, read/write Boolean property
Description
      Outputs all members of a parent-child hierarchy in a single table column
 in the flattened result, unless the parent-child hierarchy is requested on Axis
 0. The Level template for output columns is not used.
      The default value for this property is FALSE.
      This property can be used with the Discover and Execute methods.
```

Chapter 13

Subscriptions

Topics:

- *Dynamic auto-update date parameters*

This chapter describes best practices and functionalities available in the subscriptions module.

Dynamic auto-update date parameters

Usually when selecting a report to run as a subscription, one will want this report to update its date parameters as the time passes accordingly.

Because a report can have multiple date parameters and Visma Reporting does not know the business logic of the product, it is impossible for us to choose which parameters to update and how to calculate the next date or the next period.

Hence we recommend a set of best practices when designing the report:

- Define an expression for the parameter default value.

  Example for last period *:=Year(Today())&"" &Month(Today())-1*
- When subscribing to the report use the default values of this parameter. Reset all values and then change only the ones that should not automatically update values.

Chapter 14

Data Providers

Topics:

- *Available Data Providers*

Available Data Providers

Visma Reporting has a set of available data providers that you can use for defining your application type. You define the needed data provider in the server application configuration file under the “Provider” xml tag.

Example:

```
<node name="Provider">
      <setting name="PersonalizationFilePath">C:\Program Files\Visma\Reporting
\Personalization\</setting>
      <setting name="ReportParametersDefaultValuesFolder"> Visma
\Visma.Services.Reporting.Server\Personalization
      </setting>
      <!--Overrides the built-in default providers. If not set built-in
 providers are used-->
      <setting name="DefaultNodeName">Default</setting>
      <!--When true the default providers are used when the requested
 provider fails to load. The default value is true. Set to false to get
 provider configuration error messages if the requested provider doesn't work
 correctly.-->
      <setting name="UseDefaultForMissingProvider">true</setting>
      <node name="Default">
            <setting
 name="DataProvider">Visma.Services.Reporting.Provider.DataProvider,
            Visma.Services.Reporting.Provider</setting>
      </node>
      <node name="Visma Analysis">
            <setting
 name="DataProvider">Visma.Services.Reporting.Provider.DataProviderAdoMd,
            Visma.Services.Reporting.Provider</setting>
      </node>
      <node name="Visma Business">
            <setting
 name="DataProvider">Visma.Services.Reporting.Business.BusinessDataProvider,
            Visma.Services.Reporting.Business</setting>
      </node>
</ node >
```

The default data provider from Visma Reporting is called “DataProvider”. It uses a connection of type OleDbConnection and the parameter values are going to be embedded inside the query. All of our providers are evaluating the query for expressions if necessary. Hence if a query is defined in the report like an expression, then the expression is evaluated first and after that the query is executed.

Oracle example: `="SELECT Id, Name, CreationDate from Employees where to_char(CreationDate,'dd-mm-yy')='" + Format(CDate(Parameters!Date.Value),"dd-MM-yy")  +"' and Name in ('"+Join(Parameters!Name.Value,"','") +"')"`

In the above example the “Format” and the “Join” functions are going to be evaluated before the query is executed.

Note: If the Join function is used, then that parameter values are going to be embedded inside the query, no matter which provider is going to be used.

The default types of providers available are as follows:

1. ODBC Data Provider

This data provider uses a connection of type OdbcConnection.
For this provider the parameters should be defined with “?” and their values are not going to be embedded inside the query at runtime (see above Note). Make sure you define each parameter used by the query in the DataSet parameters menu. If one parameter is used more that once you will need to define it for each use. Keep in mind that the order of the parameters is also important.
Multi-value parameters are going to be sent to the server as a comma-separated list of strings.

2. MySql Data Provider

This data provider uses a connection of type MySqlConnection.
For this provider the parameters should be defined with “?Name” and their values are not going to be embedded inside the query at runtime (see above Note).
Multi-value parameters are going to be sent to the server as a comma-separated list of strings. So you are able to use functions like “FIND_IN_SET” in order to check the multi-value parameter for a given value. The “IN” function is not going to work.
Example: `SELECT * FROM table WHERE FIND_IN_SET(id, ?parameter)!=0`

3. AdoMd Data Provider

This data provider uses a connection of type AdomdConnection.
For this provider the parameter values are not going to be embedded inside the query at runtime (see above Note).
If all values have been selected for a parameter the AllMembers function is going to be used. In this case `CONTRAINED` flags have to be removed (see below Note).
Multi-value parameters are going to be sent to the server as a comma-separated list of strings enclosed in `{}`.
Because the AdomdCommand supports only parameters of type string, when replacing the parameter values inside the query it will enclose all values into single quotes. Because of this we need to modify the query at runtime by adding STRTOSET function around all parameter definitions, if not already present.

4. Oracle Data Provider

This data provider uses a connection of type OracleConnection.
For this provider the parameters should be defined with “:” and their values are not going to be embedded inside the query at runtime (see above Note).
Multi-value parameters are going to be sent to the server as a comma-separated list of strings.

Note: If Allmember function is used, the `CONSTRAINED` flags have to be removed from the `STARTOSET` function.

Example: `FROM ( SELECT ( STRTOSET(@TidTid, CONSTRAINED ) )`

Product specific providers available:

- Business Data Provider
- OnDemand Data Provider
- Proceedo Data Provider
- Global Data Provider

Chapter
15

Tips and Tricks

Topics:

- *Report Design Tips and Tricks*

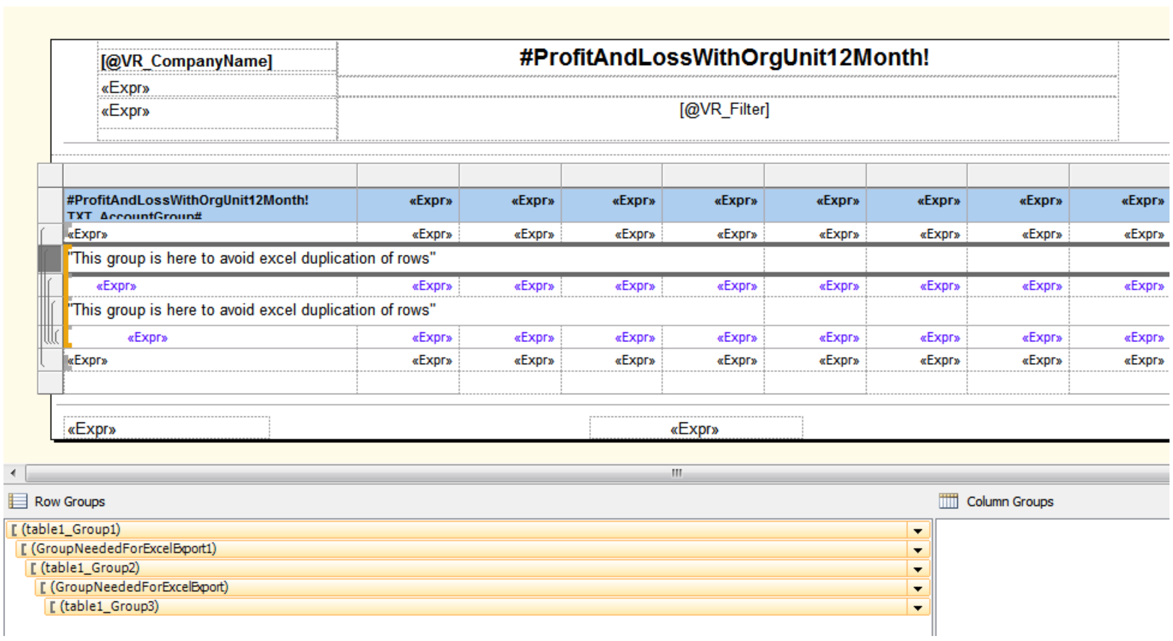
Report Design Tips and Tricks

Export to Excel duplicates some rows

The problem appears in reports that have multiple nested groups that can be toggled to hide or show details. When the report is exported to MS Excel, the first detail row is duplicated for each group. The extra row is a generated auto subtotal row.

Solution: add a "fake" group (group by ="" formula) as last level (the most deep) group before the details row or the new subgroup row, and set its visibility to hidden.

Example from a Visma Business report:

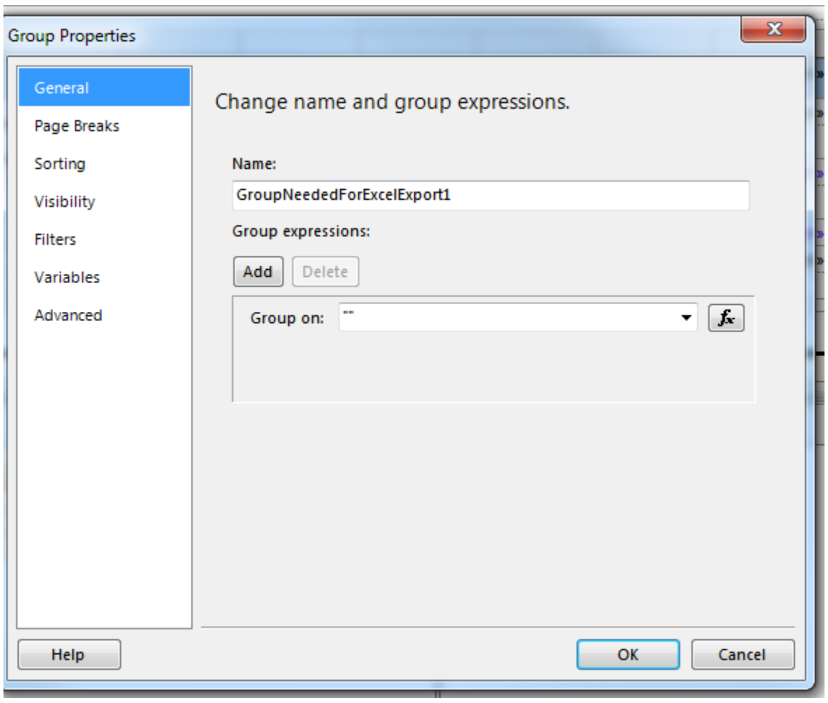


```
[@VR_CompanyName]      #ProfitAndLossWithOrgUnit12Month!
«Expr»
«Expr»                 [@VR_Filter]

#ProfitAndLossWithOrgUnit12Month!  «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr»
TXT_AccountGroup#
«Expr»                             «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr»
"This group is here to avoid excel duplication of rows"
  «Expr»                           «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr»
"This group is here to avoid excel duplication of rows"
    «Expr»                         «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr»
«Expr»                             «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr» «Expr»

«Expr»                  «Expr»
```

Row Groups

- (table1_Group1)
  - (GroupNeededForExcelExport1)
    - (table1_Group2)
      - (GroupNeededForExcelExport)
        - (table1_Group3)

Column Groups



Group Properties

- General
- Page Breaks
- Sorting
- Visibility
- Filters
- Variables
- Advanced

Change name and group expressions.

Name: GroupNeededForExcelExport1

Group expressions:

Add | Delete

Group on: ""

Help | OK | Cancel

Running values and global variables

If you use a public or a private Visual Basic global variable in custom code for storing a running total value, you need to consider the fact that the variable's value is going to be reset after each page.

When using a shared global variable in custom code, the variable value is not going to be reset on each page. Also when exporting the report the shared variable value is still going to be remembered; the running total is continued in the exported report.

Note: We have encountered different behavior depending on the rendering engine in some cases when global variables and groups were used inside a report. As a best practice it is recommended that you reset the global variables values in group/report footer.

Summary

The "VR_" prefix is only needed in the following cases:

Common:

- Properties for dynamically setting of available parameter values

  - VR_Query
  - VR_QueryType
  - VR_QueryValueField
  - VR_QueryLabelField
- Grouping of parameters – parameter name must start with "VR_Group_"
- VR_Group_Collapsed – set parameter group initially to collapsed
- VR_ParamDependency – Property to state child parameters or indicate dependency between parameters
- VR_Filter – shows the selected report parameter values
- VR_CompanyName – active company name
- VR_User – username of logged on user
- VR_Language – language and culture of logged on user
- VR_ParentParam – Property to state parent parameter
- VR_Visible –Property to state when parameter is visible (dependent on parent parameter value)
- VR_VisibleExtended –Property to state when parameter is visible (dependent on parent parameter value)

Visma Business:

- VR_OrgUnit01-12 parameters
- VR_CompanyNo - active company no