Creating BIS Reports

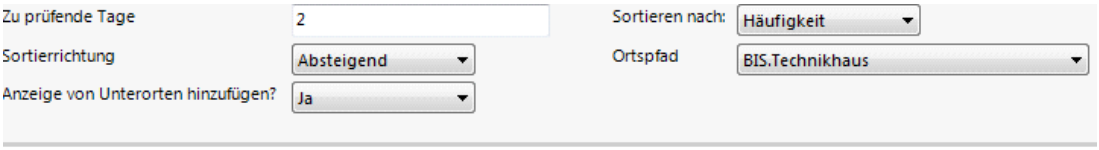
BOSCH

en Technical note

Table of contents

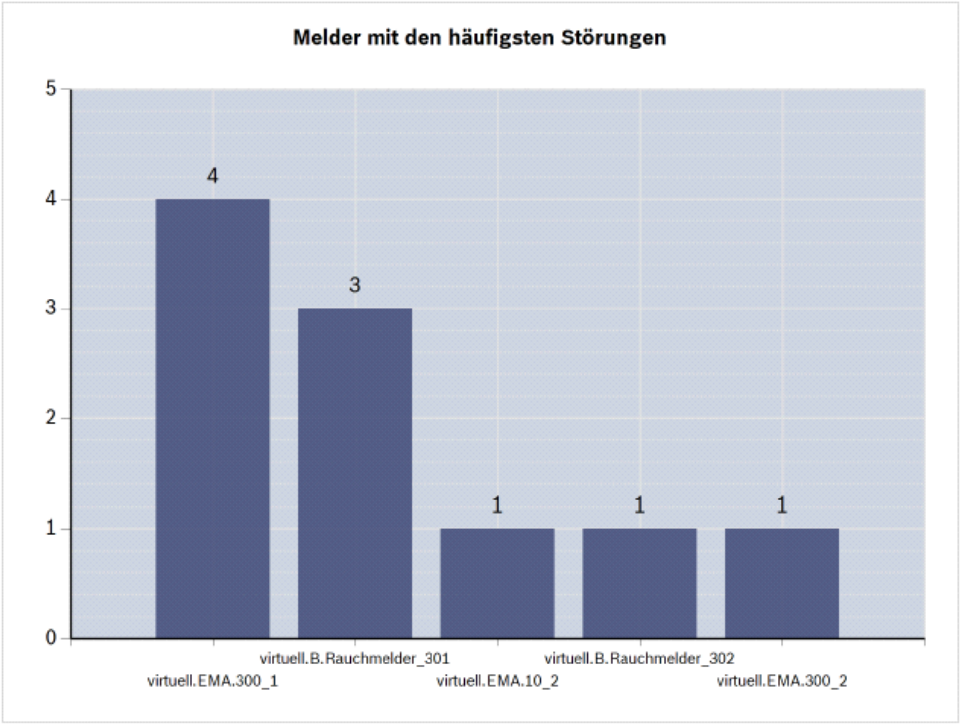
1 What is a report?

A report is generally used to present data.
In our case, reports are used to evaluate and present the large quantity of data recorded by the BIS logbook. The reports make it possible to quickly access the information that is really relevant via the BIS.
The example report "Faults per detector" is shown below. This shows all detectors that have reported a fault most frequently in the last few days. For this purpose, the report uses information from several tables of the logbook produced by the BIS to create a bar chart and a table.
The results are displayed in the BIS and can be exported as a PDF or Excel file by a single click. The files can then be forwarded and saved for documentation purposes.



Störungen pro Melder

BOSCH
Technik fürs Leben



Art des Melders	Adresse	Ort	Häufigkeit
Störmeldekontakt	virtuell.EMA.300_1	BIS.Technikhaus.Tresor	4
optischer Rauchmelder	virtuell.B.Rauchmelder_301	BIS.Technikhaus	3
Bewegungsmelder	virtuell.EMA.10_2	BIS.Technikhaus.Tresor	1
optischer Rauchmelder	virtuell.B.Rauchmelder_302	BIS.Technikhaus	1
Störmeldekontakt	virtuell.EMA.300_2	BIS.Technikhaus.Tresor	1

2 The 4 steps to a finished report

The basis for every report is an SQL procedure. This procedure is written in Transact SQL and provides the data from the logbook to the report editor (Report Builder) in compressed form. A procedure can therefore be used, for example, to output all detectors that currently have a fault or are deactivated with just one click. It is also possible to easily show which detectors are most frequently affected by faults.
The following manual shows you which steps you must perform, starting from the idea of creating a report through implementation of the idea up to display of the finished report in the BIS.
Below you can see which steps are necessary and also a brief summary of the content in the respective section.

2.1 Creating a requirements catalog

Like most projects, creation of a report starts with development of a requirements catalog. This chapter shows you that aspects to which you should pay particular attention.

2.2 Obtaining logbook data using SQL

In order to obtain the logbook data, we need a procedure in SQL which reads this data from the logbook. We will write this using the Microsoft SQL Server Management Studio.
In this chapter, you will be able to create a finished SQL procedure with the following steps:

1. First install the basis for programming, the SQL Management Studio
2. In this step, you will learn about how the BISEventLog, i.e. the BIS logbook, is structured.
3. This is followed by an explanation of how you can write general table outputs with SQL. Above all, you will get to know the commands that are most useful for our task.
4. Building on this basic knowledge, you will then receive answers to further BIS-specific questions. This includes answering questions on how to output the location path, address or detector type of a desired detector with SQL, for example.
5. You then see the annotated source text of finished reports.

At the end of this step you will have created a procedure that outputs all your required information with just one click.

2.3 Presenting data using reports

You can now output all the information in a table. However, the table can be viewed only in a black-and-white design.
Since this is not our goal, you will learn in this step how you can present your output information clearly, professionally and in a visually attractive way.
The Microsoft Report Builder in particular will help you here. What you can expect from this program and how to use it will be explained in this chapter in illustrated step-by-step instructions.
At the end of this chapter your report will have its final design and will be available via the BIS.

2.4 Simple report distribution by an Exe file

In the previous section, you successfully managed to display the report on your BIS as desired. However, the goal is for the report to be accessed by the customer.

To make sure that setup on the customer's premises functions as easily as possible, I will show you how to write an Exe file which integrates the report in the BIS with just a double-click by the technician on-the-spot. This also means that the technician does not have to install any other programs on the customer's computer. Installation of the BIS and the SQL Server is completely sufficient.
The report is then successfully integrated on the customer's system and can be accessed by him in the BIS.

3 Creating a requirements catalog

Since you now have a general overview of the steps that are necessary to create a report, the manual will now start with the first step.
It is necessary for you to produce a requirements catalog.
The document should clarify the following questions, among others:

- What is the purpose of the output report?
- Which parameters should the customer enter himself and which should be predefined? For example, how many days of data should be read out by the procedure?
- If parameters are predefined, what value should be assigned to them?
- Which columns should be displayed? Address, detector type, location path,...
- How should the results be presented? Table, diagram, graph,...
- What design should the report have? Bosch or customer design?
- What is the maximum number of lines that the report should output?

4 Obtaining the logbook data

As soon as the requirements catalog has been produced, the next step is to read out the logbook data.
For this purpose, you must first know that all the data we need is stored in a large logbook produced by the BIS. This logbook records all events such as triggering of a detector and stores these events in a database.
This database is located on an SQL server under the name **BISEventLog**.
Using a procedure, we search through this logbook for the information we want in order to then output the desired results in a table.
You require the program **Microsoft SQL Server Management Studio** both to view the logbook and to write the procedure.
This chapter will take you through the following 4 steps so that you have a functional SQL procedure which can then be visually prepared.

1. Installation of Management Studio
2. Familiarization with the logbook database
3. Writing the procedure in T-SQL
4. Permitting Report Builder access

4.1 Installation of Microsoft SQL Server Management Studio 2012

The Microsoft SQL Server Management Studio is an essential program for developing and implementing new reports. Before installing it, you should first make sure that an SQL Server is installed on your computer. This should have already taken place together with installation of the BIS. When you have installed this basis, you can continue with the following instructions:

Step 1:
Obtain the installation file of the SQL Management Studio
This is located, for example:

- on the BIS installation medium under: `\3rd_Party\SQL2012SP1\1033\32\1033_ENU_LP\x86\Setup\`
  - Select the file **SQL_SSMS_LOC.MSI**
- or on the download page of the Microsoft SQL Server 2012 Express at: www.microsoft.com/en-us/download/details.aspx?id=29062
  - To download, click on **Download** and select the file **ENU\x64\SQLManagementStudio_x64_ENU.exe**.

Note: Installation and the following screenshots are in English. If you want to have your SQL Management Studio in German, please download the German installation files and transfer my screenshots of the installation process to the German version.

Step 2:
Right-click on the installation file and run the setup **as Administrator** .

Step 3:
Wait until the installation files have been extracted and the "SQL Server Installation Center" window has opened. Choose the option **New SQL Server stand-alone installation or add features to an existing installation** and click past the next window in which the program searches for updates.

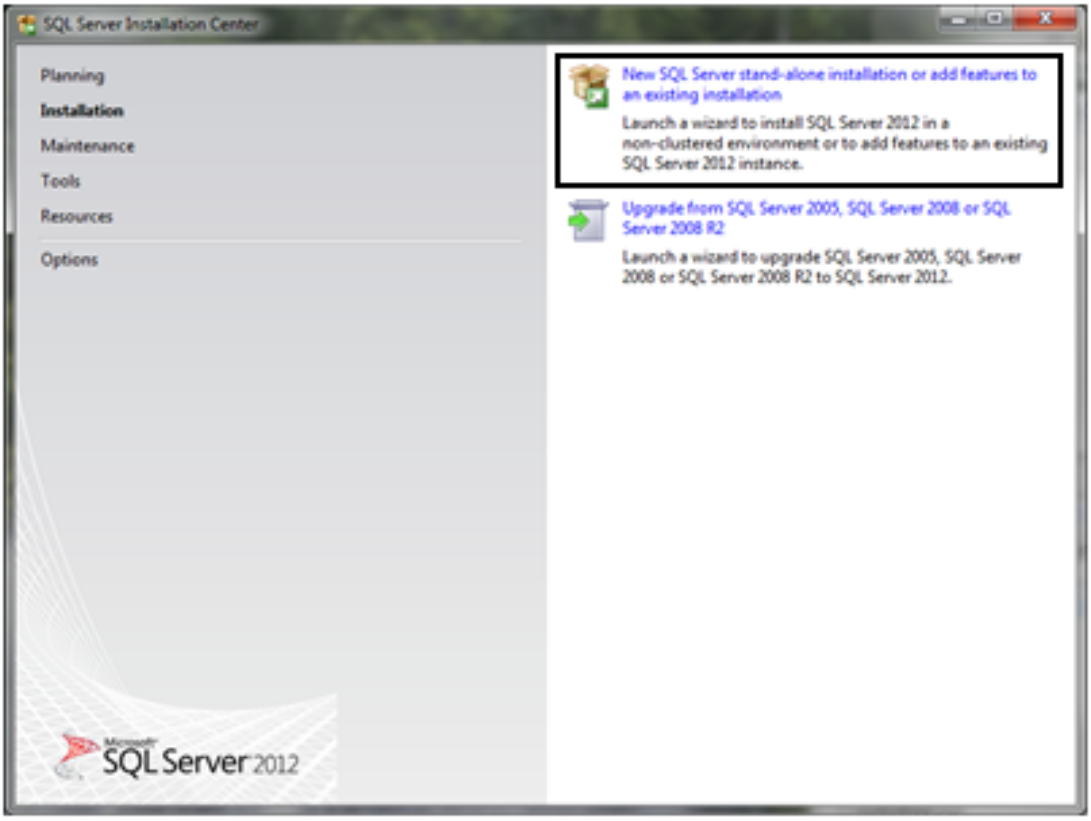


Figure 4.1: Fig. 3 SQL Server Installation Center

Step 4:

After a few files have been installed, you must select the installation type.

In the "Installation Type" window, you have the choice between "**Perform a new Installation [...]"** and "**Add features to an existing instance [...]"**.

Important:

Here, you must select the item **Perform a new installation.**

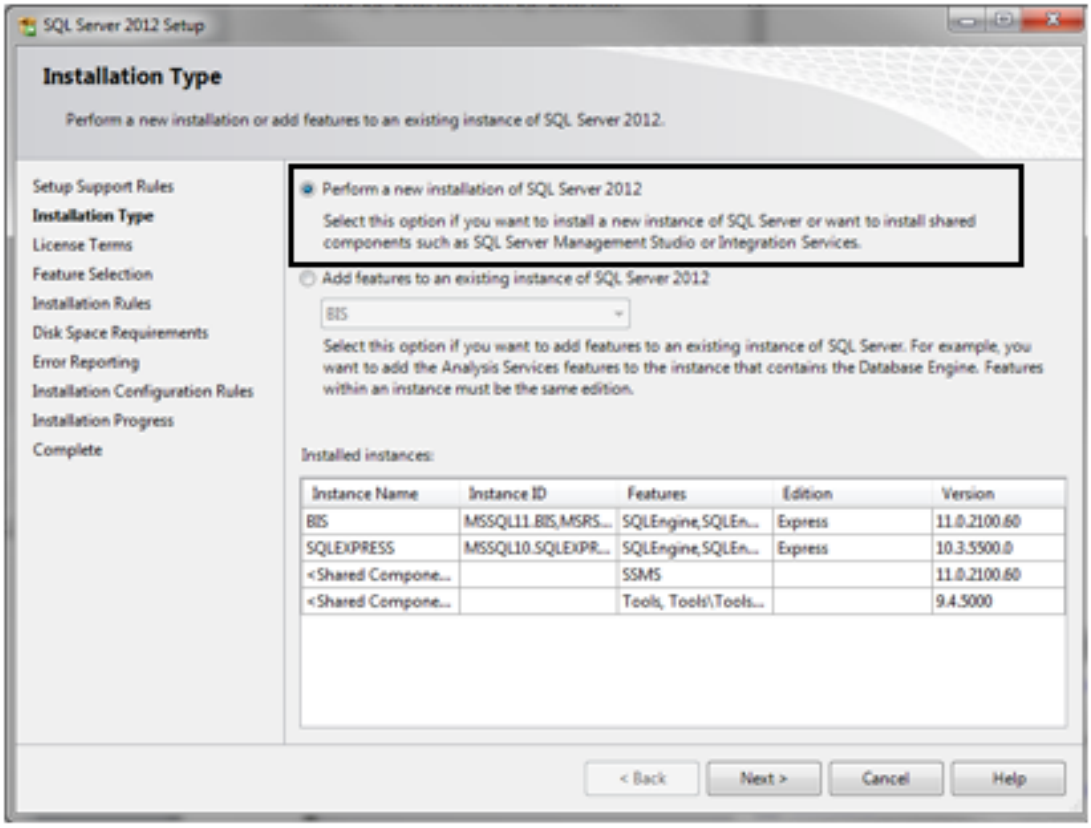


Figure 4.2: Fig. 4 Installation Type

Step 5:

After you have confirmed the License Terms, you will reach the window in which you can select the feature to be installed. Check the box **Management Tools – Basic** and click on Next.

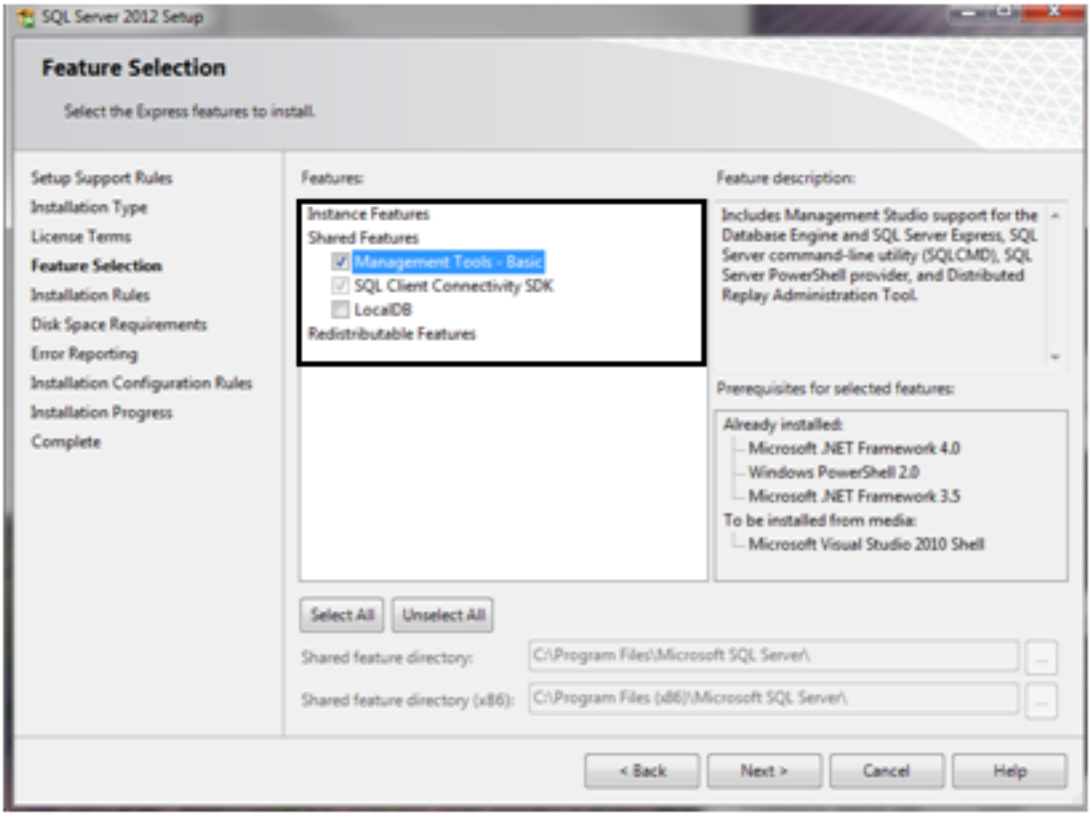


Figure 4.3: Fig. 5 Management Tools selection window

Step 6:

You have completed installation after you have gone past the window in which you can authorize Microsoft to send error reports.

Always launch the SQL Management Studio as administrator in order to obtain full access to all functions. You should find it in your programs under Microsoft SQL Server 2012.

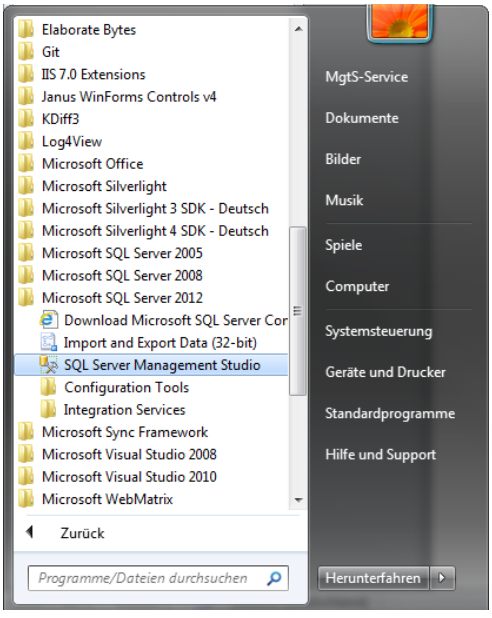


Figure 4.4: Fig. 6 Management Studio file path

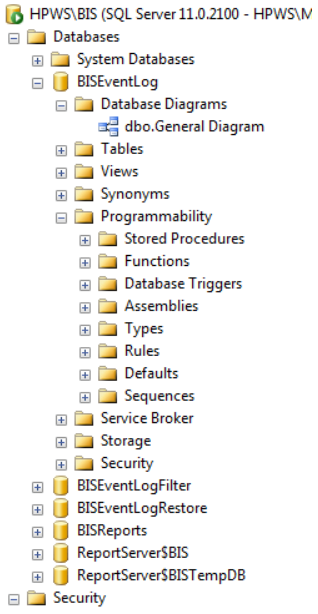
Step 7:
Now start Management Studio in order to implement your report and connect with your local BIS server via the Windows authentication.
If this does not work, you can also log in via the SQL authentication with the following details:
Username: **logbuch_query**
Password: **pw_logbuch_query**

4.2 Structure of the BIS SQL server

When you have logged into the SQL server, you will probably notice that it contains a large quantity of information.
To ensure that you always keep an overview with this abundance of data, the next few lines will explain the structure of the SQL server linked to the BIS.
This includes the structure of the navigation bar as well as the structure of the most important databases. Two databases in particular are especially important for us: **BISEventLog** (actual logbook) and **BISReports**.

4.2.1 Structure of the navigation bar

As soon as you have logged into the SQL Server Management Studio, you will see the navigation bar of your SQL server on the left of the screen.



The login data is hidden in the item **Security** . The item **Databases** leads you to the system and BIS databases. Here you can see the databases relevant for us, namely **BISEventLog** and **BISReports**.
Under **Database Diagrams** you can create a diagram like that shown in 4.2.2.1 which shows all tables with the corresponding attributes and links.
The tables in which the BIS messages are stored are saved in **Tables** .
Under the navigation option **Programmability**, you can see the programming options offered by SQL. We will consider above all the **Stored Procedures** .
Important:
You will create and save all your source codes under Stored Procedures.
Difference between procedure and function
When you get to know SQL a little, you will see that procedures and functions are very similar. They can both be used to produce table outputs. The difference between them is that, unlike

procedures, functions are characterized by a return value. This means that a table or other output must be produced each time with them.
The advantage of procedures is therefore that a table is only displayed. External programs, such as the Report Builder, can then access this displayed table. However, an internal procedure cannot access a table from a different procedure!

4.2.2 Importance and content of the relevant databases

Below you will create a procedure and to do this you must read information out of a database. However, these memories can contain several gigabytes of information. If you try to understand the structure and content of such a database without help, this can be a laborious and not always successful process.
For this reason, I will briefly describe the structure, relationships and meaning of the attributes of the relevant databases in the following paragraphs.

BISEventLog

BISEventLog is the most important and content-rich data source for us. It represents the logbook of the BIS and thus saves all messages and events recorded by the BIS. It therefore contains almost all the information that could possibly be output with a report.
The following paragraphs are intended to provide a quick insight into the structure of the EventLog. The following diagram was created by me under Database Diagrams and shows the most important tables with their links.

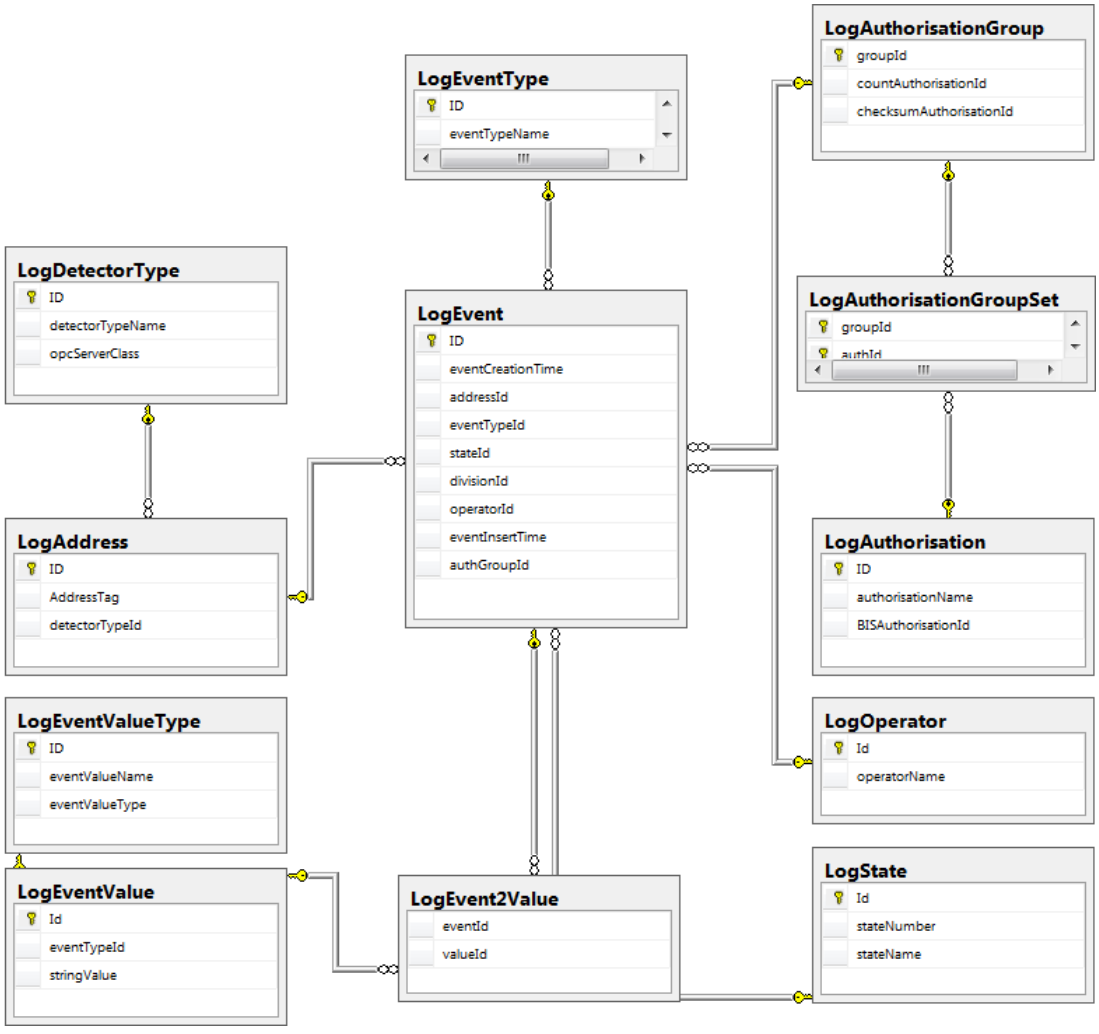


Figure 4.5: Fig. 9 Tables diagram BISEventLog

LogEvent

This is the **main table of the logbook**. Each event recorded in the BIS is stored here. This table is linked to every other relevant table via the individual attributes.

If you look at the table (by SELECT * FROM LogEvent), you will notice that many events have been entered with practically the same time. This is because not only one event is reported in the event of a fire alarm, for example.

Often, up to six events are recorded solely due to triggering of the detector. This is necessary to ensure that all relevant information such as New message, Message distributed, Location path, Action plan, Operator, Status change, etc. is stored.

This multiple entry of events must be taken into account in particular if it is wished to count how often a detector has been triggered. It is important that really only one event is counted per triggering. The solution to this is to consider the eventTypeID with the associated eventTypeName in LogEventType. Here, only the events which are linked with the eventTypeName 'state change' are counted. This is only a single event for each fire alarm.

ID	Consecutive number, starting at 1 with the oldest event
eventCreationTime	Time of event creation

addressID	Link to the log address. States the address.
eventTypeID	Link to LogEventType. Indicates whether the event is of the type 'state change', 'control' or 'message'
stateID	Link to LogState. Indicates the detector state: Standby, Fire,...

LogState

All possible detector states are entered in the table.
Each state has a corresponding stateNumber. This stateNumber is normally the same for every BIS but can be parameterized. For example, Standby has the number 5 and Intrusion the number 21 as default. StateNumber is not comparable with the stateID!
The stateID has resulted from the continuous entry of states in LogState and is different in every logbook.

Important: How do I check the status:

To do this, first check for the name and then take the stateNumber if this is not present. This procedure is necessary due to the fact that the name is frequently changed from BIS to BIS. In contrast, the number usually stays the same. The source code for this query can be found below under the BIS-specific queries.

LogEventValue

eventTypeID	Link to LogEventValueType.ID
stringValue	Contains a very large amount of information. This includes location path, action plan, processing status, command which triggered the detector, etc.

LogEvent2Value

This table is a link table between LogEvent and LogEventValue.

eventID	Link to LogEvent.ID. NOT to LogEvent.eventTypeID
valueID	Link to LogEventValue.ID

LogEventValueType

eventValueName	Indicates the type of the corresponding information in LogEventValue.stringValue. For example, the information is stored here that 'BIS.Technikhaus' corresponds to a location path. This information is particularly relevant when adding location paths for table output. The procedure here is to first link the event with the LogEventValue and LogEventValueType. You then have all stringValues output where the corresponding eventValueName = 'Location path'. This query can also be found under the BIS-specific SQL queries.

LogAddress

This table stores the address information of an event. As a result, it is clear which detector was responsible for triggering for each event.

ID	Link to LogEvent.addressID
AddressTag	States the address as text. e.g.: 'virtual.B.smoke_detector_201‘
detectorTypeID	Link to LogDetectortype. Specifies the type of detector.

LogDetectorType
This table specifies the type of detector and the associated OPC server class.

ID	Link to LogAddress.detectorTypeID
detectorTypeName	Specifies the type of detector. e.g. 'optical smoke detector'
opcServerClass	Specifies the OPC server class. e.g. 'Virtual'

LogEventType
As already mentioned above, this table content is very useful for counting state changes. It is necessary to check whether the eventTypeName of the event corresponds to the type 'state change'.

ID	Link to LogEvent.eventTypeId
eventTypeName	Specifies the type of event. e.g. 'state change', 'message', ...

BISReports
This database becomes relevant only when you make the report accessible to the Microsoft Report Builder. In other words, when the procedure is finished and you want to display it graphically.
When you have reached this point, you must create an additional procedure in BISReports which accesses the content already created and executes this. In addition, you must also adapt the permissions for the database. Further information on this is provided in the section "Adapting the procedure for use with the Report Builder".

4.3 Introduction to the principle of Transact-SQL

Special features of Transact-SQL
The goal of practically every Transact-SQL script is to display a table in which information from several larger tables can be combined and sensibly linked.
The Transact-SQL programming language requires a nested approach which differs considerably from the classic procedural programming style.
This is based on the core command SELECT. You search for and link tables, reduce these in size and define conditions until the desired result is displayed This is all done together with **one** SELECT command.
When you have become familiar with efficient programming with SELECT, it is possible to create complex reports with just a few lines of code and a runtime of a few milliseconds.
A little knowledge and a few helpful tricks are necessary to achieve this result. On the following pages you will therefore get to know and understand the most important functions and commands in SQL.

Further information
This document cannot cover all the possibilities offered by Transact-SQL. ´
If you have questions about one of the commands explained below, enter the command name in an internet search engine followed by **Transact-SQL**. The most helpful results can almost always be found on the msdn homepage.

4.4 Important SQL commands with questions and solutions

If you have now discovered a taste for the SELECT command, you can look forward to the next few pages. That is because you will learn here how you can exploit the unimagined possibilities of SELECT.
For this purpose, I will present the most important SQL commands in combination with a question and an example from BIS in each case. The order of the commands is based on their intended location within the Select statement.

4.4.1 Structure of the SELECT command

You now know what you can achieve with Select. In order to implement this, this chapter will first introduce you to the structure of **SELECT**.
A Select has the task of selecting information from one or more tables and then displaying this. A relatively branched structure can be recognized here. However, this always follows a fixed order, which will be explained briefly below by means of an example.

```
SELECT TOP 10 Tabelle1.Spalte1 AS Erste, MAX(Tabelle2.Spalte2) AS Zweite
FROM Tabelle1
INNER JOIN Tabelle2 ON Tabelle1.Spalte2 = Tabelle2.Spalte1
WHERE Tabelle1.Spalte1 > 20
GROUP BY Erste
ORDER BY Spalte2 DESC
```

- The Select logically starts with the word **SELECT**.
  All the following commands are optional and thus serve only for refinement of the results.
  The main command is followed by the expression **TOP 10**, which results in only the ten topmost rows being returned.
  This is followed by the columns to be selected, which are assigned to their original tables with a point. All columns are selected with a *.
  Using **AS**, you can define a new name for the respective column.
  The function **MAX()** allows you to select only the largest element in the column. This will be explained in more detail below.
- The core table to be read out follows the **FROM** and additional tables the **INNER JOIN**.
- The **ON** is followed by the condition of how the other tables are linked with the core table.
- The **WHERE** must be positioned following this. This is used to define conditions which apply to all tables, columns, etc.. All possible comparison operators such as **>, <, <>** (not equal to), **=, LIKE, BETWEEN, IS NULL**, ... can be used for this.
- When this has been done, the obtained results can be grouped with **GROUP BY**. For example, grouping by the first column can be performed. As a result, there are no duplicate entries in the first column. This inevitably means that entries in the other columns will be eliminated.
- An aggregate function is needed before the second column name in order to know here what will be selected and what not. This can be e.g. **MAX(), MIN(), SUM(), AVG(), COUNT()**.
- Finally, the table output can be sorted in ascending or descending order with **ORDER BY ASC** / **DESC**.

4.4.2 Main commands in the Select statement

The most important commands belonging to the Select statement are now described below. These are explained in each case with a question, an explanation, an example from the BIS and the output.
The examples can be transferred to the Management Studio by copy and paste and applied to the BISEventLog database. For this purpose, it may be necessary to change the database from master to BISEventLog and replace the quotation marks by others.
I wish you much success in getting to know and understanding the main commands in the Select statement.

TOP 10 -> How can I output only the top results?

The TOP N command is used to output only the top N rows. N stands for the number of rows that you wish to output. It is also possible to output only the top N percent. For this, PERCENT is written after the number.
Example: Output the last four events:

```
SELECT TOP 4 ID, eventCreationTime
FROM LogEvent
ORDER BY LogEvent.eventCreationTime DESC

ID eventCreationTime
32535 2013-07-30 09:38:21.043
32534 2013-07-30 09:38:14.850
32533 2013-07-30 09:38:14.850
32532 2013-07-30 09:38:14.007
```

AS -> How do I assign a (new) name to my columns?

A name can be assigned to a column with the **AS** command.
Example:

```
SELECT 'Heidenei‘, 'Huch‘ AS ‘Ausdruck der Überraschung'

Ausdruck der Überraschung
Heidenei
Huch
```

JOIN -> How can I read out data from several tables?

The **JOIN** command is used to merge data from more than one table. Using Select, it is possible to display the columns of all joined tables after the join operation.
Example:
Displays all events linked with the state name from LogState.

```
Select LogEvent.ID, LogEvent.eventCreationTime, LogState.stateName
FROM LogEvent
INNER JOIN LogState on LogState.Id = LogEvent.stateId

ID eventCreationTime stateName
8320 2013-04-25 11:47:34.257 Einbruch Extern
8321 2013-04-25 11:47:49.507 Relais geschlossen
8322 2013-04-25 11:47:49.507 Relais geschlossen
```

```
8323 2013-04-25 11:47:49.523 Licht an
…
```

WHERE -> How do I define conditions?
Conditions are defined with the **WHERE** command.
Further conditions are added with AND or OR. The AND is processed first and then the OR.
Example: ...AND...OR...AND... corresponds to (...AND...)OR(...AND...)
If the OR is to be processed first, the brackets must be placed correspondingly: ...AND(... OR...)AND...
Example:
Selects all events that do not refer to either Fire or Standby.

```
SELECT LogEvent.ID, eventCreationTime, LogState.stateName
FROM LogEvent
INNER JOIN LogState ON LogState.Id = LogEvent.stateId
WHERE LogState.stateName <> 'Feuer-Ext.alarm' AND LogState.stateName <>
'Ruhe'

ID eventCreationTime stateName
25999 2013-07-19 14:03:38.983 Dongle nicht verfügbar
26101 2013-07-19 14:03:42.117 Videosignal Störung
26107 2013-07-19 14:03:41.743 P4 - Störung
26110 2013-07-19 14:03:42.117 Nachricht
…
```

DISTINCT -> How do I filter redundant rows so that only one is displayed?
The **DISTINCT** command allows all completely duplicate entries to be eliminated from a SELECT. This means that all rows are eliminated which are identical to another row in every column.
Example:

```
SELECT DISTINCT * FROM LogState --> sortiert alle doppelten Einträge aus
LogState aus
```

GROUP BY -> How can I output the largest/smallest event of another column for a column?

The **GROUP BY** command is used to eliminate all redundant entries in a column and to combine the remaining entries with the corresponding value of another column.
This command can therefore be used, for example, to find out which detector had which state **last**.
It is important here that all columns occur either after the **GROUP BY** or in an aggregate function. Otherwise it is not possible for SQL to find the element for the remaining row and an error would be returned. l:
Important aggregate functions: **AVG()** -> Average, **COUNT(), MAX(), MIN(), SUM()**
Example:
Outputs how many events are related to the respective state. For this purpose, the date on which the state was entered the **last time** is output.

```
SELECT
MAX(LogEvent.eventCreationTime) AS eventCreationTime,
```

```
LogState.stateName AS state,
COUNT(LogEvent.stateId) AS frequency --> counts how many events can be
included in a line
FROM LogEvent
INNER JOIN LogState on LogState.Id = LogEvent.stateId
GROUP BY LogState.stateName --> Groups according to state name, so that it
does not occur multiple times in the result.
ORDER BY frequency DESC

eventCreationTime state frequency
2013-07-26 08:38:30.857 Ruhe 6912
2013-07-17 10:35:51.993 Feuer-Ext.alarm 849
2013-07-26 08:38:30.917 Licht aus 408
2013-07-26 11:05:52.413 Abmeldung 361
2013-07-26 11:05:52.413 Abgemeldet 351
2013-07-26 08:38:30.167 P4 - Störung 257
…
```

ORDER BY -> How can I sort my outputs?

Sorting takes place in SQL with the command **ORDER BY**. ORDER BY is followed by the column according to which sorting should take place and then by specification of whether sorting should be in ascending (ASC) or descending (DESC) order.

Example:

Sorts the LogEvent table in descending order according to time.

```
SELECT ID, eventCreationTime FROM LogEvent
ORDER BY LogEvent.eventCreationTime DESC

ID eventCreationTime
30549 2013-07-26 11:07:23.293
30548 2013-07-26 11:07:23.187
30547 2013-07-26 11:07:02.637
…
```

UNION -> How can I supplement my table vertically by rows which have nothing to do with the table except for the column name?

Several tables can be vertically linked with each other using the **UNION** command. This function can be very helpful later on in the Report Builder.

The function offers you additional options for adding values in the selection window of a variable parameter. The majority of the values available for selection are specified in a previously defined procedure. With UNION, you can now add an arbitrary value to the actually determined output of the procedure.

Important: the column names of the tables must be the same!

Example:

The **UNION** adds the selection 'Alle' ('All') to the location paths (Ortspfad) of the BIS.

```
SELECT stringValue AS Ortspfad
FROM LogEventValue
INNER JOIN LogEventValueType ON LogEventValueType.ID =
LogEventValue.eventTypeId
```

```
WHERE eventValueName = 'Ortspfad'
UNION
SELECT 'Alle' AS Ortspfad
```

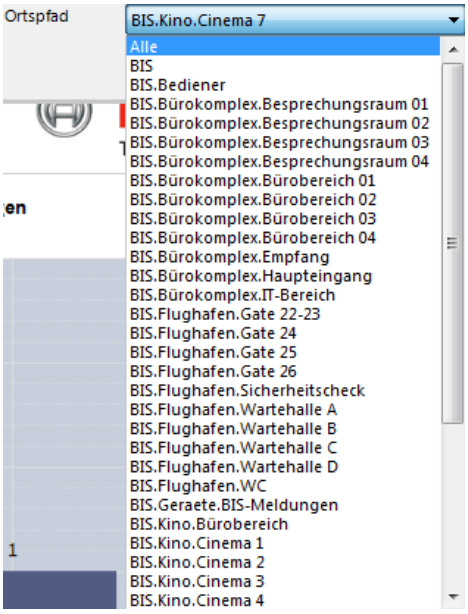


Figure 4.6: Fig. 10 Location path (Ortspfad) selection window

4.4.3 General SQL commands

IIF() -> How do I create an IF query within a Select statement?

The command **IIF()** is very useful for creating an if...then... query.

This command has a special syntax. Here, three parts are written in the bracket after IIF. First the condition, then the output if the condition is true, followed by the output if the condition is false. The condition is a comparison or other statement that returns either true or false.

IIF(condition, following condition true, following condition false)

Example:

Checks whether the variable @Ortspfad (location path) is equal to 'Alle' ('All').

If yes, all events with the corresponding location paths are selected.

If not, all events are selected where the location path starts with @Ortspfad.

```
DECLARE @Ortspfad AS nvarchar(30) = 'BIS.Kino'

SELECT LogEvent.ID, eventCreationTime, stringValue AS Ortspfad FROM
LogEvent
INNER JOIN LogEvent2Value ON LogEvent.ID = LogEvent2Value.eventId
INNER JOIN LogEventValue ON LogEvent2Value.valueId = LogEventValue.Id
INNER JOIN LogEventValueType ON LogEventValue.eventTypeId =
LogEventValueType.ID
WHERE LogEventValueType.eventValueName = 'Ortspfad'
and LogEventValue.stringValue LIKE IIF(@Ortspfad = 'Alle', '%', @Ortspfad
+'%')

ID eventCreationTime Ortspfad
```

```
2079 2013-04-11 11:00:16.330 BIS.Kino.Cinema 8
2080 2013-04-11 11:00:16.330 BIS.Kino.Cinema 9
2081 2013-04-11 11:00:16.330 BIS.Kino.WC
2082 2013-04-11 11:00:16.330 BIS.Kino.WC
2083 2013-04-11 11:00:16.330 BIS.Kino.Bürobereich
2084 2013-04-11 11:00:16.330 BIS.Kino.Hauptbereich
```

LIKE -> How can I compare texts?

Just like numbers, texts can also be compared with the equals symbol **'='**. It addition, it is also possible to check whether parts of the text are the same. This is done with the command **LIKE**. The LIKE compares a text with a text combined with placeholders '%'.

Example: Checks whether the location path begins with 'BIS' and contains 'Kino'.

```
LogEventValue.stringValue LIKE 'BIS%Kino%'
```

IS NULL -> How do I check whether a variable has the value NULL?

The comparison function IS / IS NOT NULL is used for this.

Example:

Does not return anything since the variable @notNull has a value 0 and is therefore not null.

```
DECLARE @notNull AS int = 0
SELECT IIF(@notNull IS NULL,,Kann doch nicht wahr sein!',,Hab ich mir schon
gedacht.')

Hab ich mir schon gedacht.
```

BETWEEN -> How can I display a range between two elements?

The command **BETWEEN** is needed for this.

Syntax: Expression [**NOT**] **BETWEEN** start AND end

Example:

Selects all elements of the LogEvent table where the ID is between 4 and 8.

```
SELECT ID, eventCreationTime FROM LogEvent WHERE LogEvent.ID BETWEEN 4 and
8

ID eventCreationTime
4 2013-04-03 15:37:22.277
5 2013-04-03 15:37:22.277
6 2013-04-03 15:37:22.277
7 2013-04-03 15:37:22.277
8 2013-04-03 15:37:18.533
```

CONVERT() / CAST() -> How can I change the data type of a variable?

In order to convert one data type into another, it is possible to use either **CONVERT** or **CAST** in SQL. Both perform an equivalent job, but have different syntax.

Example: Changes the data type from nvarchar to integer:

CONVERT (target data type, variable)

```
DECLARE @zahl AS nvarchar = '3'
SELECT CONVERT (int, @zahl) AS 'konvertierte Zahl'
```

CAST (Variable AS Target data type)

```
DECLARE @zahl AS nvarchar = '3'
SELECT CAST(@zahl AS int) AS 'konvertierte Zahl'
```

DATEADD() -> How can I add time data?

Using **DATEADD**(), a certain number of time units (minute, hour, day,...) can be added to/ subtracted from a date.
Syntax: DateAdd (type of time unit, number of time units, date)
Example follows for Getdate().

GETDATE() -> How can I find out the current time?

The current time can be determined with GetDate and saved as smalldatetime. Syntax of smalldatetime: YYYYMMDD hh:mm:ss
Example:
Returns the time of two hours ago:

```
SELECT DateAdd(hour, -2, Getdate())
```

CREATE PROC –> How do I create a new procedure?

A procedure is created with the **CREATE PROC** command:
Example:
Creates the procedure EventsMitZustaenden (EventsWithStates) which returns all events with the passed state name.

```
CREATE PROC EventsMitZustaenden @stateName AS VARCHAR(30)
AS
BEGIN
SELECT LogEvent.ID, LogEvent.eventCreationTime, LogState.stateName
FROM LogEvent
INNER JOIN LogState ON LogState.Id = LogEvent.stateId
WHERE LogState.stateName = @stateName
ORDER BY LogEvent.ID DESC
END
```

EXEC -> How can I execute a procedure?

A procedure is executed with **EXEC**. The procedure name is followed by the parameters to be passed:

```
EXEC EventsMitZustaenden 'ASP Sperre'
```

ALTER PROC -> How can I change a procedure?

A procedure can be modified with the **ALTER PROC** command:
Example:
Overwrites the existing procedure EventsMitZustaenden.
It now returns all events that do not have the passed state name.

```
ALTER PROC EventsMitZustaenden @stateName AS VARCHAR(30)
```

```
AS
BEGIN
SELECT LogEvent.ID, LogEvent.eventCreationTime, LogState.stateName
FROM LogEvent
INNER JOIN LogState ON LogState.Id = LogEvent.stateId
WHERE LogState.stateName <> @stateName
ORDER BY LogEvent.ID desc
END
```

4.4.4 Explained example queries from the BIS

Now that you have mastered the most important commands for SQL queries, I will deal with a number of examples specifically from the BIS.
Many parts of the following source texts are the previous copy and paste templates and can be included in your procedure unchanged. For example, the INNER JOINs when adding the location path are always the same.
Finally, there are a few commented source codes of reports that have already been completed.

How do I add the location path of a detector to my output table?

The individual location path is stored under LogEventValue.stringValue.
Important:
Not just location paths are stored under stringValue but also processing states, action plans, etc. The class to which the respective entry under stringValue belongs is stored under LogEventValueType.eventvalueName.
To ensure that the returned location path really is this and not an action plan, the query whether eventValueName = 'Ortspfad' ('Location path') is essential.
When linking the tables, it must be noted that LogEvent is linked with LogEvent2Value via the LogEvent.ID and NOT via LogEvent.eventID.

```
SELECT LogEvent.ID, LogEvent.eventCreationTime, LogEventValue.stringValue
AS Ortspfad
FROM LogEvent
INNER JOIN LogEvent2Value ON LogEvent.ID = LogEvent2Value.eventId
INNER JOIN LogEventValue ON LogEvent2Value.valueId = LogEventValue.Id
INNER JOIN LogEventValueType ON LogEventValue.eventTypeId =
LogEventValueType.ID
WHERE LogEventValueType.eventValueName = 'Ortspfad'
/* --> sorgt dafür, dass die Ortsanzeige auch wirklich dem Ortspfad
entspricht und nicht Bediener, Meldung angenommen,..., denn diese Werte
sind ebenfalls als eventValueName gespeichert. */

ID eventCreationTime Ortspfad
1060 2013-04-03 16:38:29.983 BIS.Flughafen.WC
1061 2013-04-03 16:38:29.983 BIS.Kino.Bürobereich
1062 2013-04-03 16:38:29.983 BIS.Kino.Hauptbereich
1063 2013-04-03 16:38:29.983 BIS.Kino.Cinema1
...
```

How do I add the address of a detector to my output table?

```
Die Adresse ist in LogAddress.addressTag gespeichert. LogAddress ist über
die LogAddress.ID mit der LogEvent.addressID verknüpft.
SELECT LogEvent.ID, LogEvent.eventCreationTime, LogAddress.AddressTag as
Adresse
FROM LogEvent
INNER JOIN LogAddress ON LogEvent.addressId = LogAddress.ID

30509 2013-07-26 08:38:30.857 virtuell.Licht.Zaun_L8
30510 2013-07-26 08:38:30.857 virtuell.EMA.100_1
30511 2013-07-26 08:38:30.857 virtuell.EMA.100_2
...
```

How do I add the detector type to my output table?

The detector type is stored in LogDetectorType.detectorTypeName. This is linked with LogEvent via LogAddress.

```
SELECT LogEvent.ID, LogEvent.eventCreationTime,
LogDetectorType.detectorTypeName AS Melderart
FROM LogEvent
INNER JOIN LogAddress ON LogEvent.addressId = LogAddress.ID
INNER JOIN LogDetectorType ON LogAddress.detectorTypeId =
LogDetectorType.ID

ID eventCreationTime Melderart
19143 2013-07-10 11:03:48.873 Bewegungsmelder
19144 2013-07-10 11:03:48.873 Überfallmelder
19145 2013-07-10 11:03:48.873 Magnetkontakt
19146 2013-07-10 11:03:48.873 Riegelkontakt
...
```

How can I find out the stateID of a state?

The topic of stateID is tricky. That is because the BIS does not have standardized names for the states. It therefore often occurs that a state name is changed slightly or translated into English.
For this reason, stateNumber, which can also be parameterized in the BIS, is usually better suited for checking a state than the name. This is due to the fact that this almost never changes, unlike the stateName. For example, Standby has No. 5 as default and Intrusion No. 21.
In addition, it is possible that the state to be checked has never occurred. This means that it also cannot be found in the logbook. It is therefore important to know the default stateNumber of the state and to output this in an emergency.
In order to prevent one of the two attributes being changed, it is helpful to use an IF query which ensures that the corresponding stateID is still output even if one of the two attributes is modified:

```
DECLARE @ruheID AS INT = (SELECT
```

```
(IIF (
(EXISTS(SELECT stateNumber FROM LogState WHERE stateName = 'Ruhe')),
/* -- Abfrage ob der Zustandsname Ruhe vorhanden ist.*/
(SELECT TOP 1 stateNumber FROM LogState WHERE stateName = 'Ruhe'),
/* -- Wenn ja, gehe nach Namen */
(SELECT 5)))) /* -- wenn nein, gebe die ID aus, die standardmäßig Ruhe
ausgibt */
```

How can I find out how often a detector has changed to a state in the last few days?

The first task is to filter out all state changes to the selected state (e.g. ASP-Sperre (ASP block)) from the LogEvent. Here, the address of each state change must be noted. All multiple addresses are then grouped. When grouping takes place, the number of grouped addresses is counted. Since each state change has generated an address, the number of grouped addresses corresponds to the number of state changes.

```
SELECT
LogAddress.AddressTag AS Adresse,
COUNT(LogEvent.addressId)AS Häufigkeit /* -->zählt wie viele Einträge
gruppiert wurden */
FROM LogEvent
    INNER JOIN LogAddress ON LogAddress.ID = LogEvent.addressId
    INNER JOIN LogState ON LogState.Id = LogEvent.stateId
WHERE
    LogState.stateNumber = 31     /* --> ASP-Sperre */
    AND LogEvent.eventTypeId = 1 /* --> ^= state Change => Ohne diese
Abfrage wäre die Häufigkeit unkorrekt, da viel zu groß */
    AND LogEvent.eventCreationTime > DATEADD(DAY, -5, GETDATE()) /* -->
letzte 5 Tage */
GROUP BY LogAddress.AddressTag
ORDER BY Häufigkeit DESC, AddressTag

Adresse             Häufigkeit
virtuell.EMA.300_1 4
virtuell.B.Rauchmelder_302 2
virtuell.B.Rauchmelder_300 1
virtuell.EMA.6_1 1
```

How can I output all detectors that now have a specific state (e.g.: ASP Sperre (ASP block)?

The principle for solving this problem is to create an interim table with the last state changes of all detectors that were NOT is ASP Sperre (ASP block) state. You save the time of these state changes.
You now search for all state changes to ASP Sperre (ASP block) which occurred AFTER this time. From these, you choose the one with the earliest time since a detector can also be switched off twice in succession without being switched on in the meantime. If this is the case, it has been deactivated since it was first switched off. The earliest time after the last state change is therefore selected.
Example:

Which detectors must be switched on because they were switched off in the last shift?

```
DECLARE @shiftDuration AS INT = 8 /* -- Schichtdauer = Acht Stunden */
DECLARE @stateOff AS INT = /* --speichert die standardmäßige stateNumber
von ASP Sperre */
(SELECT
(IIF (
(EXISTS (SELECT stateNumber FROM LogState WHERE stateName = 'ASP
Sperre')), /* -- Abfrage ob der Zustandsname Ruhe vorhanden ist.*/
(SELECT TOP 1 stateNumber FROM LogState WHERE stateName = 'ASP Sperre'), /*
-- Wenn ja, gehe nach Namen */
(SELECT 31)))) /* -- wenn nein, gebe die ID aus, die standardmäßig Ruhe
ausgibt */

select
LogAddress.AddressTag as Adresse,
Min(logevent.eventcreationtime) as Ausschaltzeitpunkt /* --> Früheste Zeit
bei 2mal Ausschalten */

from LogEvent
    inner join LogState on LogEvent.stateId = LogState.Id
    inner join LogAddress on LogEvent.addressId = LogAddress.ID
    inner join ( /* -- Alle Melder, die NICHT in ASP Sperre waren:*/
        select
        MAX(eventcreationtime) as onTime, /* --> der LETZTE Zustandswechsel
*/
        LogEvent.addressId
        from LogEvent
        inner join LogState on LogState.Id = LogEvent.stateId
    where
        LogState.StateNumber<>@stateOff /*-->Zustandswechsel NICHT
ASPSperre */
        and LogEvent.eventCreationTime > DATEADD(hour, -@shiftDuration,
Getdate())/* --> in den letzten 8 Stunden */
group by LogEvent.addressId
) as notASP
on notASP.addressId = LogEvent.addressId
where
    LogEvent.eventCreationTime > notASP.onTime
/* --> alle Zustandswechsel, die NACH der notASP Zeit geschehen sind */
    and LogState.stateNumber = @stateoff /* --> Zustandswechsel = ASP
Sperre */
group by LogAddress.AddressTag, notASP.onTime
```

How is it possible to output which detectors have the desired state at a given time?

This query is very similar to the above-described query, which returns all detectors that are currently deactivated – i.e. in ASP block state.
The difference is that it is now possible to check for two arbitrary states and it is also possible to define the time differently.

A procedure is created by the following source text which requires input of two parameters. These are @stateNumber and @zeitpunkt (@time).
When this procedure is executed, all detectors will be returned that have the entered state at the entered time.

```
CREATE PROC [dbo].[selectZumZeitpunktTAusgeloesteMelder] @stateNumber INT,
@zeitpunkt DATETIME
AS
BEGIN

DECLARE @eventtypeid AS INT =
(SELECT (IIF (
    (SELECT ID FROM LogEventType WHERE eventTypeName = 'state change') <>
NULL,
    (SELECT TOP 1 ID FROM LogEventType WHERE eventTypeName = 'state
change'),
    (SELECT 1)))) /* --besagt Zustandswechsel */

SELECT LogAddress.AddressTag AS Adresse ,
    MAX(LogEventValue.stringValue) AS Ort,
    LogState.stateName AS Zustand,
    MAX(logevent.eventcreationtime) AS Auslösezeitpunkt ,
    MIN(RuheDanach.ruhezeit) AS Ruhezeit /* -- Ruhezeit ist die Zeit zu der
der ausgelöste Melder wieder zurück in Ruhe gegangen ist.*/
FROM LogEvent
INNER JOIN (
/* -- Es wird eine Tabelle verknüpft, die pro Melder, die Zeit des letzten
Zustandswechsels, der NICHT nach @stateNumber war, ausgibt. */
    SELECT MAX(eventcreationtime) AS ruhezeit, /* --Zeit des letzten
Zustandwechsels */
addressID
    FROM LogEvent
    WHERE logevent.eventCreationTime < @zeitpunkt /* -- vor dem gesuchten
Zeitpunkt */
        and eventTypeId = @eventtypeid /* -- stateID 1 = state change */
        and stateId <> @stateNumber /* -- Zustandswechsel NICHT in
@stateNumber */
    GROUP BY addressId
) AS ruhe
    ON logevent.addressId = ruhe.addressId
INNER JOIN LogEvent2Value
    ON LogEvent2Value.eventId = LogEvent.ID
INNER JOIN LogEventValue
    ON logeventvalue.Id = LogEvent2Value.valueId
INNER JOIN LogEventValueType
    ON LogEventValueType.ID = logeventvalue.eventtypeid
INNER JOIN LogState
    ON logstate.Id = logevent.stateId
INNER JOIN LogAddress
```

```
    ON LogAddress.ID = LogEvent.addressId
/* -- Verknüpft die Ruhezeit. Full outer join bewirkt, dass der Melder
trotzdem als ausgelöst angezeigt wird, obwohl er noch nicht in Ruhe
versetzt wurde. Bei normalem inner join werden nur Melder, die nicht in
Ruhe gesetzt wurden aussortiert, da eine mit inner join verknüpfte Tabelle
nichts ausgibt, wenn der Melder nicht in Ruhe gesetzt wurde.
Somit würde die Zeile dann automatisch gelöscht werden.
*/
FULL OUTER JOIN (
SELECT logevent.eventCreationTime AS ruhezeit, logevent.addressId
FROM LogEvent
    WHERE logevent.eventTypeId = @eventtypeid
        AND logevent.stateId <> @stateNumber
        AND logevent.eventCreationTime > @zeitpunkt
) AS RuheDanach
    ON ruhedanach.addressId = logevent.addressId
WHERE logevent.eventCreationTime BETWEEN ruhe.ruhezeit AND @zeitpunkt
/* -- Auslösezeitpunkt liegt zwischen der letzten Ruhemeldungszeit und @t
*/
    AND logevent.eventTypeId = @eventtypeid /* -- entspricht 'state change'
*/
    AND (LogEventValueType.eventValueName = 'Ortspfad')
    AND logstate.stateNumber = @stateNumber /* -- Der Zustand in den
gewechselt wurde ist @stateID */

/* Es wird nach der Adresse gruppiert. Somit wird es möglich, dass nur die
maximale eventcreationtime ausgewählt wird. Also das erste vor dem
Zeitpunkt erzeugte Event, dass zwischen Erzeugungszeitpunkt und @zeitpunkt
den Zustand nicht wieder in Ruhe gewechselt hat.
*/
GROUP BY LogAddress.AddressTag, logstate.stateName
ORDER BY AuslöseZeitpunkt
```

4.5 Adapting the procedure for graphical processing

Since your procedure can now display the desired results, the steps now follow which you must perform in order to present your results in a visually attractive way with the Report Builder. For this purpose, it is first necessary for you to carry out a small step in Management Studio

4.5.1 Creating the procedure in BISReports

As I already mentioned above, it is necessary to create an additional procedure in BISReports after completing the procedure in BISEventLog.
This procedure calls the main procedure, checks for a possible timeout and makes it available to the Report Builder.
The source code is standardized and is as follows, for example, for the procedure for output of the detectors to be switched on:

```
CREATE PROC [dbo].[report_EinzuschaltendeMelder]
@shiftDuration AS INT – Parameterübergaben; von Report Builder zu übergeben
```

```
AS
BEGIN

DECLARE @rc INT
SET @rc = 0

EXEC @rc = [BISEVENTLOG]...[report_EinzuschaltendeMelder] @shiftDuration /*
-- Aufruf der Prozedur und Übergabe der Parameter
@rc will be NULL, if timeout occured during calling Linked Server. No
exception will be thrown by default. We must do it by ourselves */
IF @rc IS NULL AND @@ERROR = 0
RAISERROR('Timeout occured! Query timeout expired. Please, try to open
report again or contact your administrator.', 16, 1);
RETURN @@error

END
```

4.5.2 Adapting the permissions

A user is needed so that you can later access your procedure with the report editing tool. This user is called **logbuch_query**. This user is not permitted as default to access the procedure created in BISReports. The user must therefore be manually authorized.
For this reason, execute the following source text once.

```
GRANT EXECUTE ON BISReports.dbo.<report_BeispielStoerungen> -- Name muss
individuell editiert werden
TO logbuch_query
```

The result can be checked by right-clicking on your procedure -> Properties -> Permissions. The user logbuch_query with execution permission must now be there.

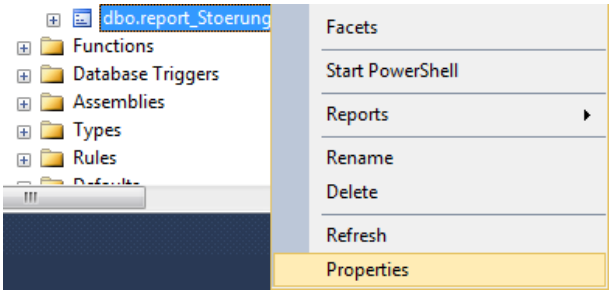


Figure 4.7: Fig. 11 Properties of logbuch_query

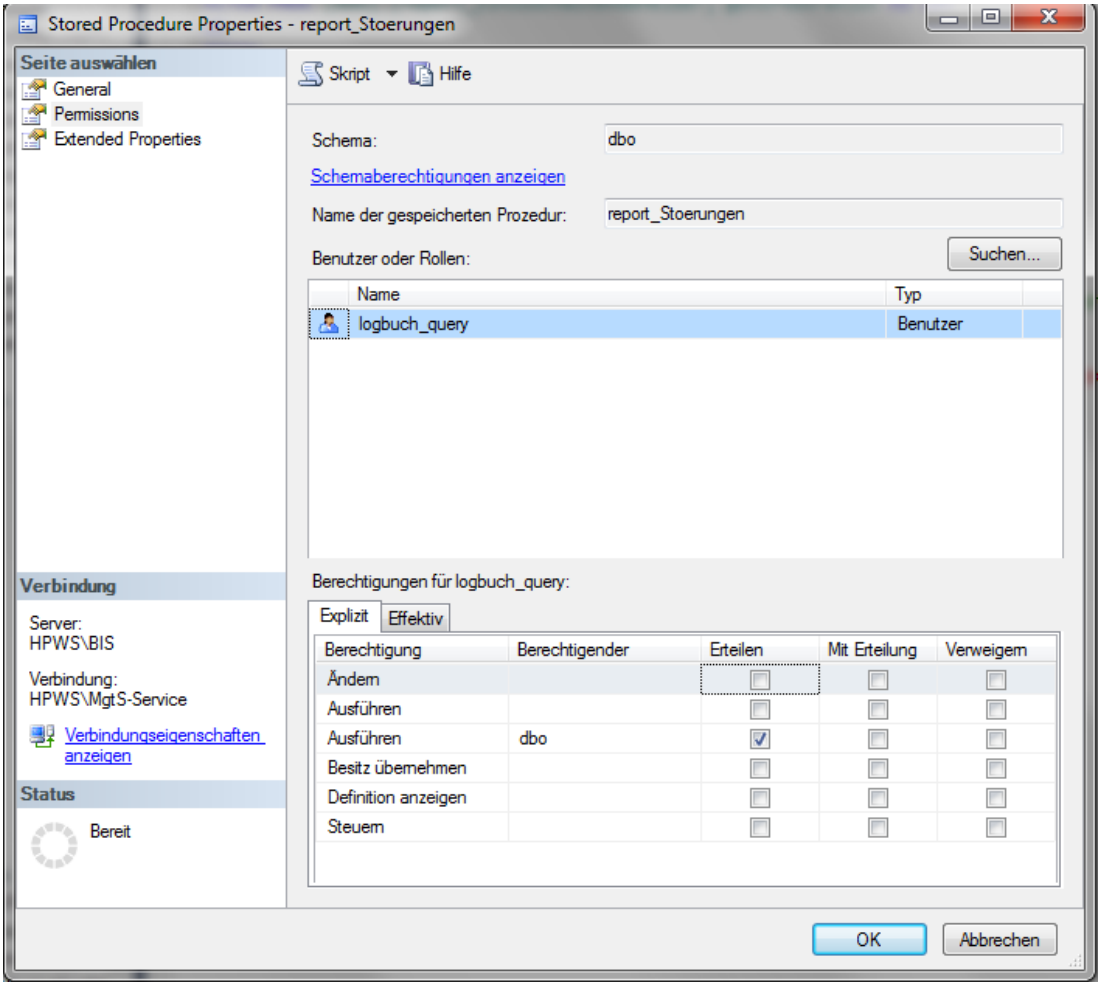


Figure 4.8: Fig. 12 Properties window

5 Presentation of the data

You can not output all desired information within SQL with your procedure. From now on we will work with the Microsoft Report Builder so that these results can be retrieved via the BIS. This program allows you to present your table output in a clear, professional and visually attractive way via the BIS.
What you can expect from this program and how to use it will be explained in this chapter in illustrated step-by-step instructions.
You will learn how you can design the report individually and integrate it in the BIS so that the customer can call up your report quickly and easily with the current data.
At the end of this chapter your report will have its final design and will be available via the BIS.

5.1 Installation of the Microsoft Report Builder

First, we must establish the basis for creating a report. This includes installation of Microsoft Report Builder 2.0 or 3.0.
For this purpose, you must make sure first of all that the Microsoft Reporting Service has been installed. This has generally already taken place during installation of the BIS. You can check this by opening the page **http://localhost/reports_bis** or by searching for the program.
It is important to know which version (2.0 or 3.0) of the Report Builder you are using since only these are compatible with subsequent functions.
If you accidentally open a 2.0-version report with the Report Builder 3.0, for example, a backup will be automatically created with the name <reportName> - Backup.rdl. This backup contains the version of the report created with 2.0.
Now please run the setup file.
This can be found on the BIS installation medium under the following path:
BIS\3RD_party\SQL2008\ReportBuilder<Version>**\ReportBuilder.msi**
whereby <Version> can be **2.0** or **3.0 .**
Setup opens as soon as you launch the **.MSI** file.
During installation, you will be requested to define the "**Default Target Server**". This corresponds to an SQL server entity with the name Reportserver. This report server was created during automatic installation of the Microsoft Reporting Service.
You should fill in the field in the following way:
http://localhost/Reportserver_<SQL Server Entity Name>
or for a specific computer (Port is optional):
http://<BIS Computer Name>:<Port>/Reportserver_<SQL Server Entity Name>

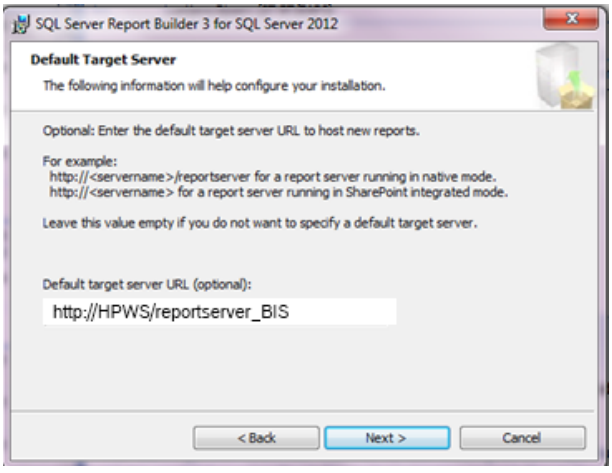


Figure 5.1: Fig. 13 Input of Default Target Server

When you click on Next, Setup should be completed and you can launch the RB. It can usually be found under the path
C:\Program Files (x86)\Microsoft SQL Server\Report Builder <Version> **\MSReportBuilder.exe.**

5.2 Creating and integrating your report

Have you installed the Report Builder and taken other precautions?
Then you can finally start to create your own report on the basis of the following step-by-step instructions. At the end, you should have completed a report in Bosch design that can be accessed via the BIS.
Step 1: Launch Report Builder
Launch the Report Builder.

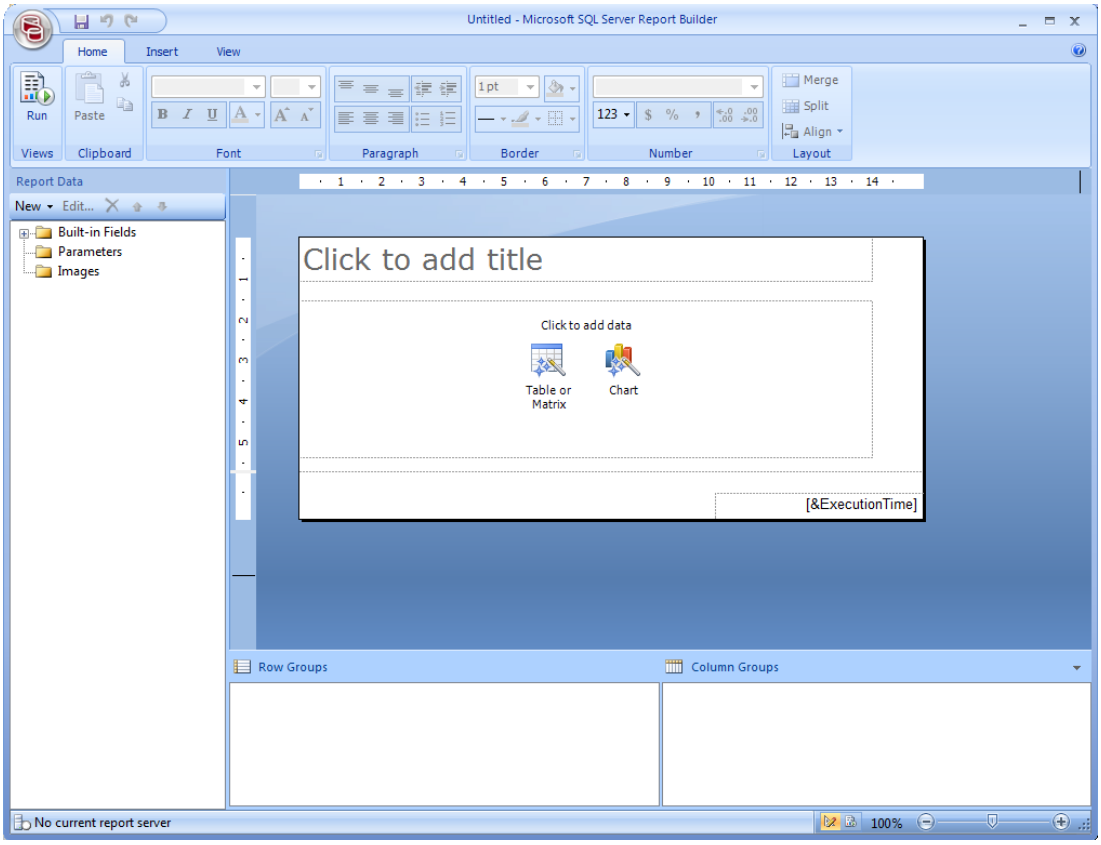


Figure 5.2: Fig. 14 Start view of RB

Step 2: Open report template
Click on the button in the top left corner. Select Open and open the supplied Bosch report template. This can be found under the following path in the files for publishing reports:
BIS_Berichte\Berichte\BoschBerichtTemplate.rdl
If the RB cannot connect with the report server, check the report server path defined during installation. This can be set under the **Report Builder button -> Options-> "Use this report server or SharePoint site by default:"**. You may not be logged on at the BIS computer and must therefore replace localhost with the BIS computer name and its port.

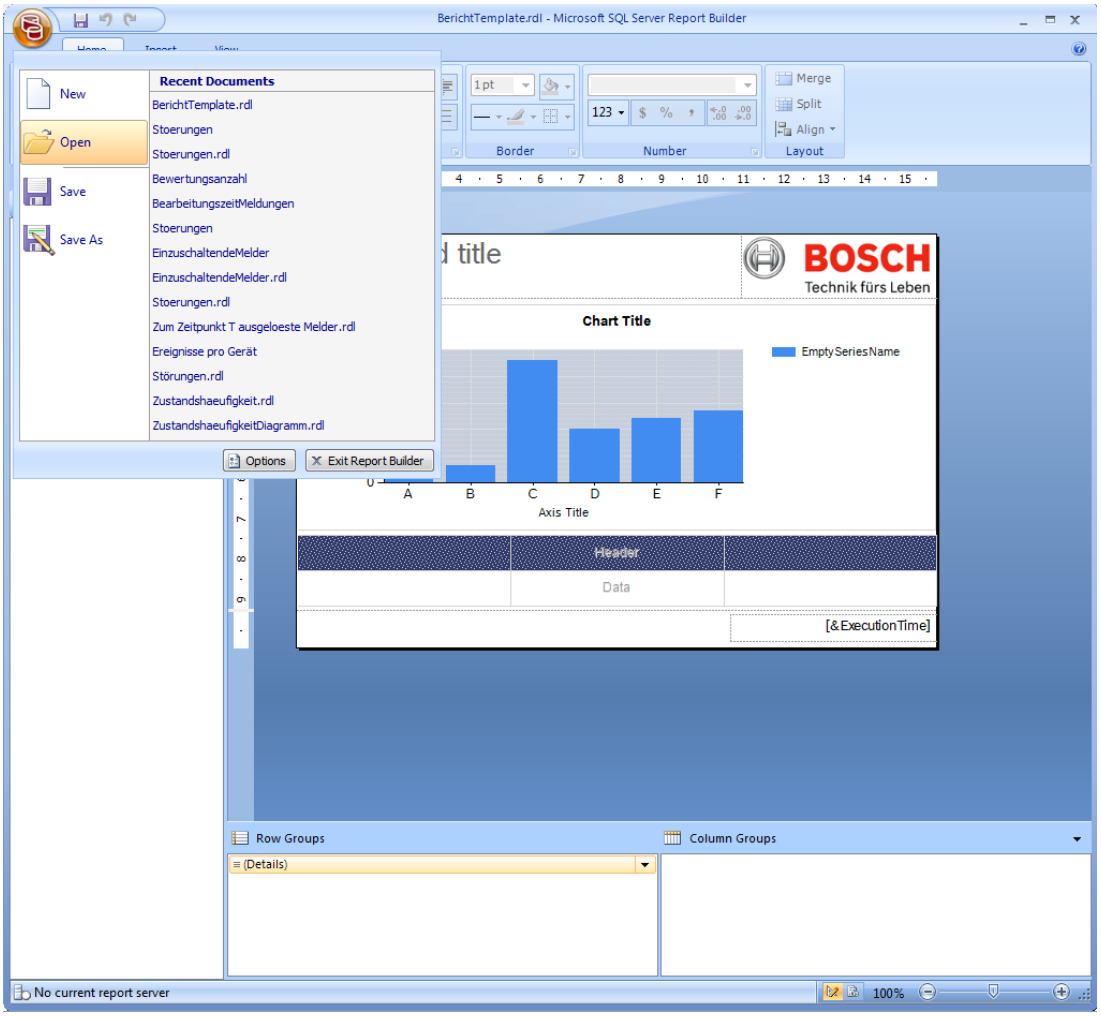


Figure 5.3: Fig. 15 BoschBerichtTemplate (Bosch report template)

Step 3: Open dataset window

Next, define the dataset to be used. To do this, click on New -> **Dataset...** The Dataset Properties dialog now opens.

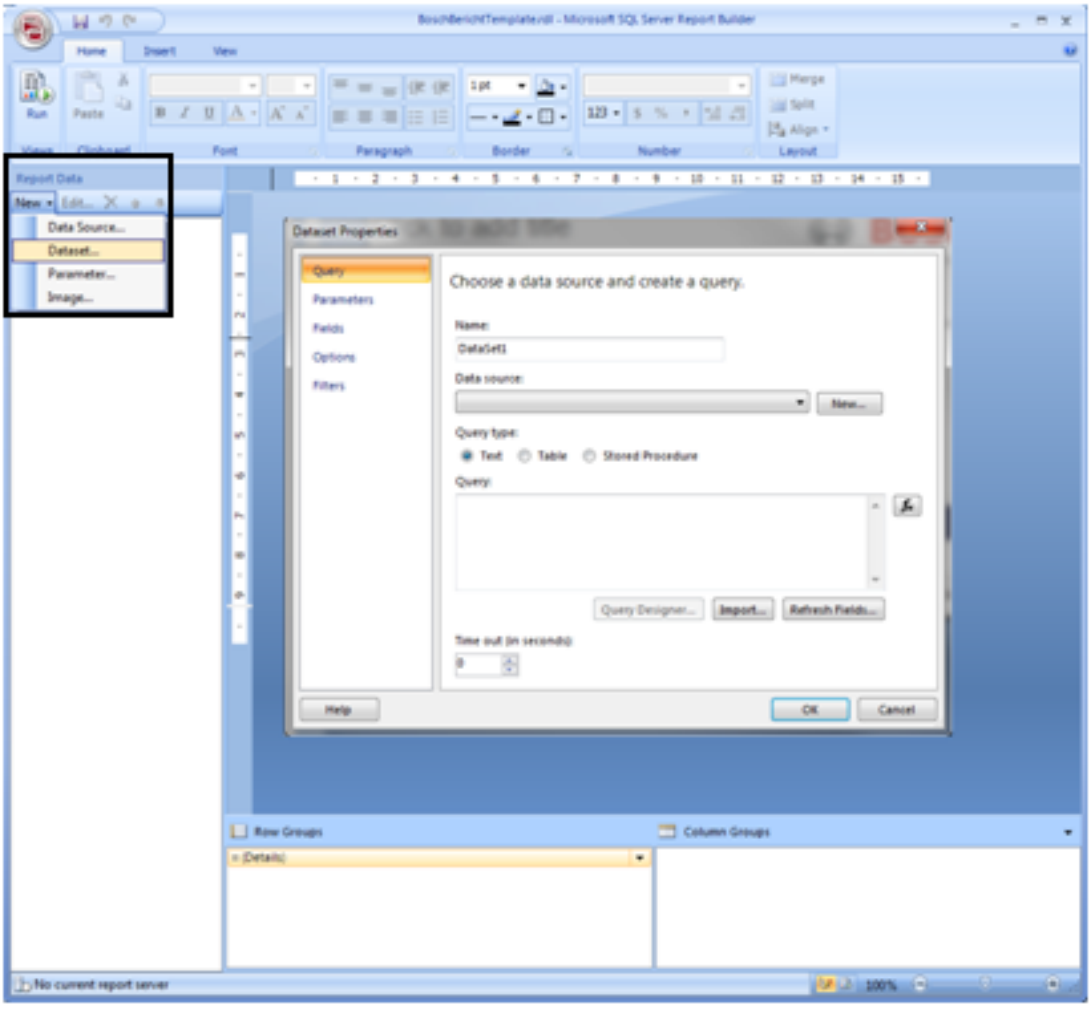


Figure 5.4: Fig. 16 Defining the dataset

Step 4: Define data source credentials
Click on **New...** to open the Data Source Properties window.
In order to change the permissions, click on **Use a connection embedded in my report** and select the tab Credentials on the left. Select **Do not use credentials** here.
This option is often reset by the Report Builder. As a result, you will be unable to connect with the report server when testing your report. If the error message "Failed to preview report" is displayed when testing, you should therefore check this option.

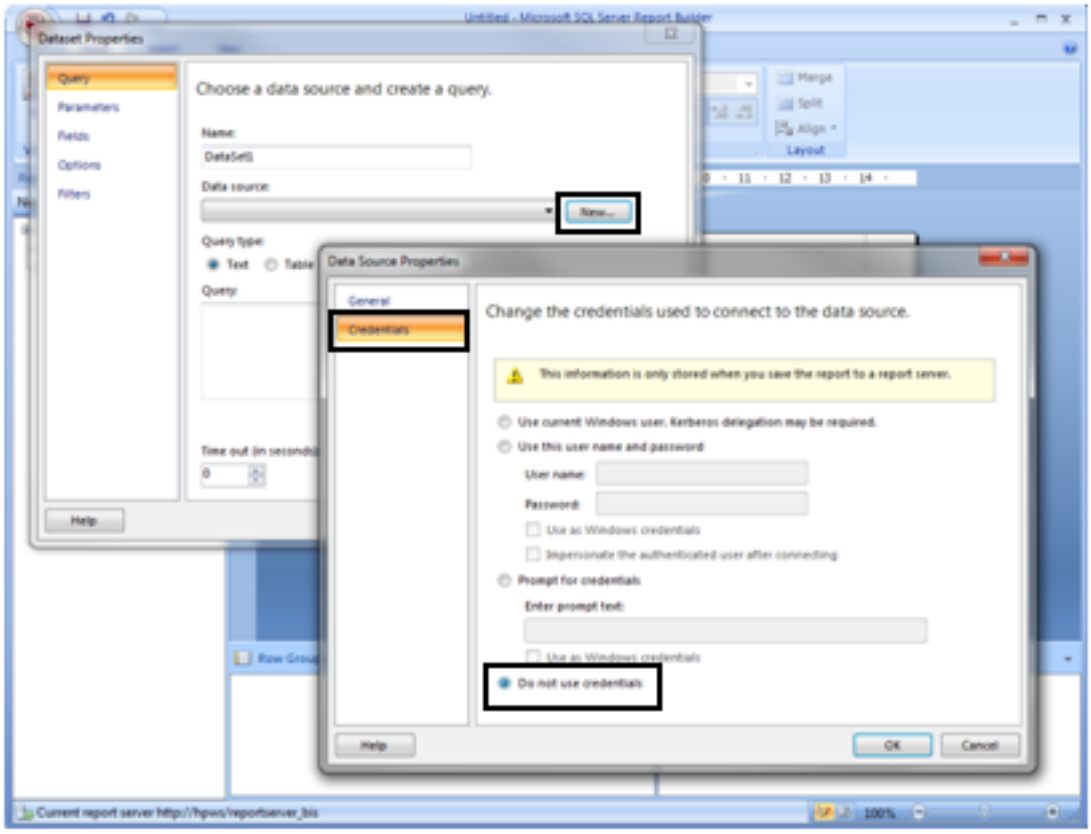


Figure 5.5: Fig. 17 Data Source Credentials

Step 5: Select data source
Go back to the General tab and select the function **Use a shared connection or report model**. Click on **Browse...**. The Report Builder now receives the files and links created on the report server. These should contain the pre-installed database **EventLog Datasource**. Select this and give your data source the name BISReports, for example. My data source is erroneously called BISEventLog. Now click on OK to integrate the procedures.

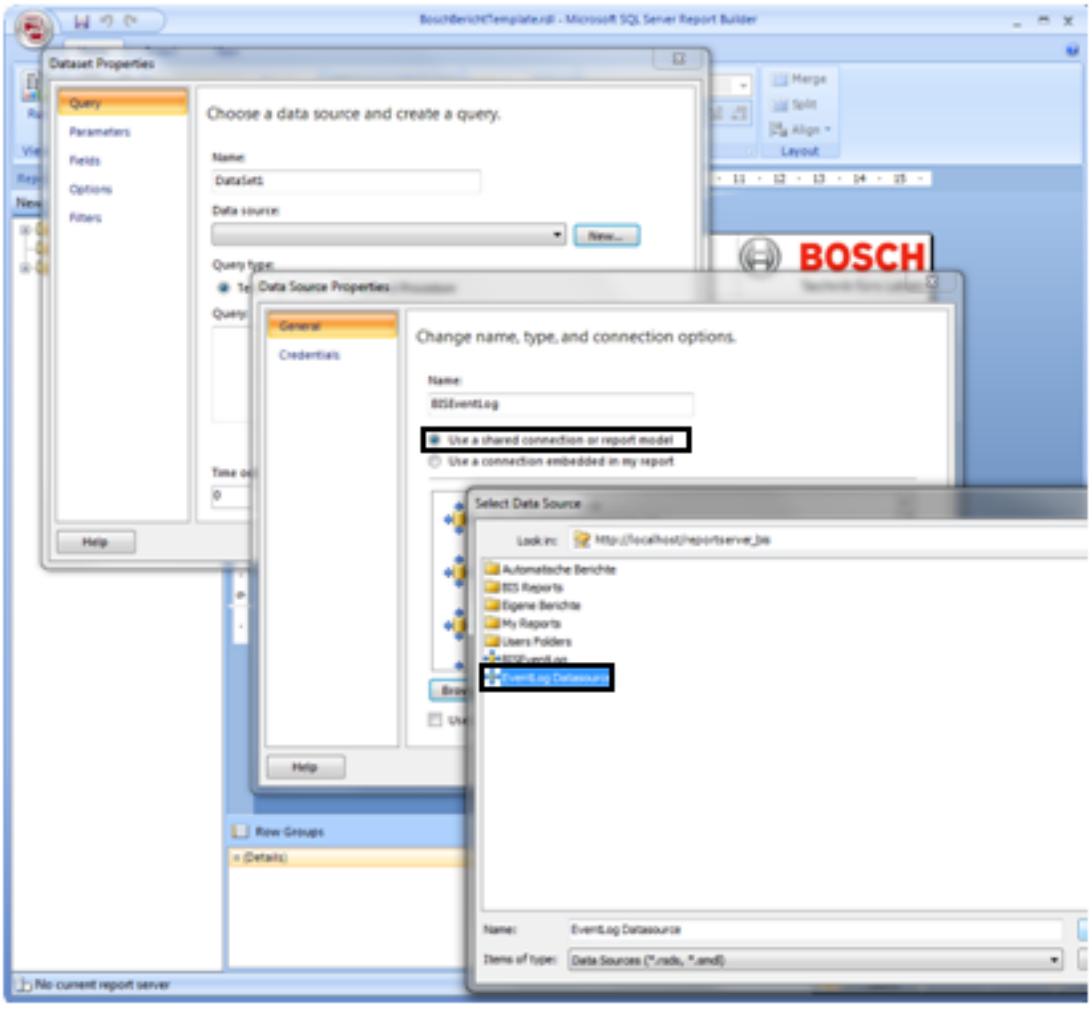


Figure 5.6: Fig. 18 Defining the data source

Step 6: Select dataset / procedure

In the Dataset Properties window, now select the Query type **Stored Procedure** in order to integrate the procedure you previously created in SQL.

A window opens in which you must enter the access data for access to the EventLog data source. Use the following user data for this:

Username: **logbuch_query**

Password: **pw_logbuch_query**

Then select the desired SQL procedure from the database **BISReports**. Here I select the procedure **report_Stoerungen**(report_malfunctions), for example.

It is important that you keep the name **DataSet1**, otherwise it will be necessary to create a new table and bar chart.

Confirm your inputs with **OK.**

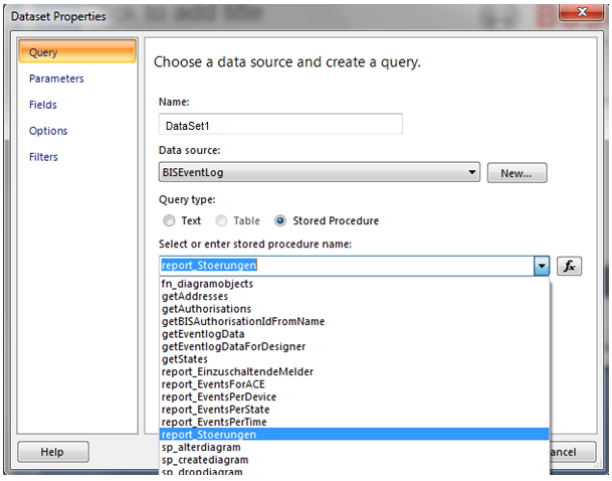


Figure 5.7: Fig. 19 Selecting the procedure

Step 7: Add parameter userPermission

Before displaying a report, the BIS checks the parameter userPermission as a security measure. For this reason, we must add this parameter for every report to be displayed. To do this, right-click on the tab **Parameters** and select **Add Parameter...**. The Report Parameter Properties window now opens.

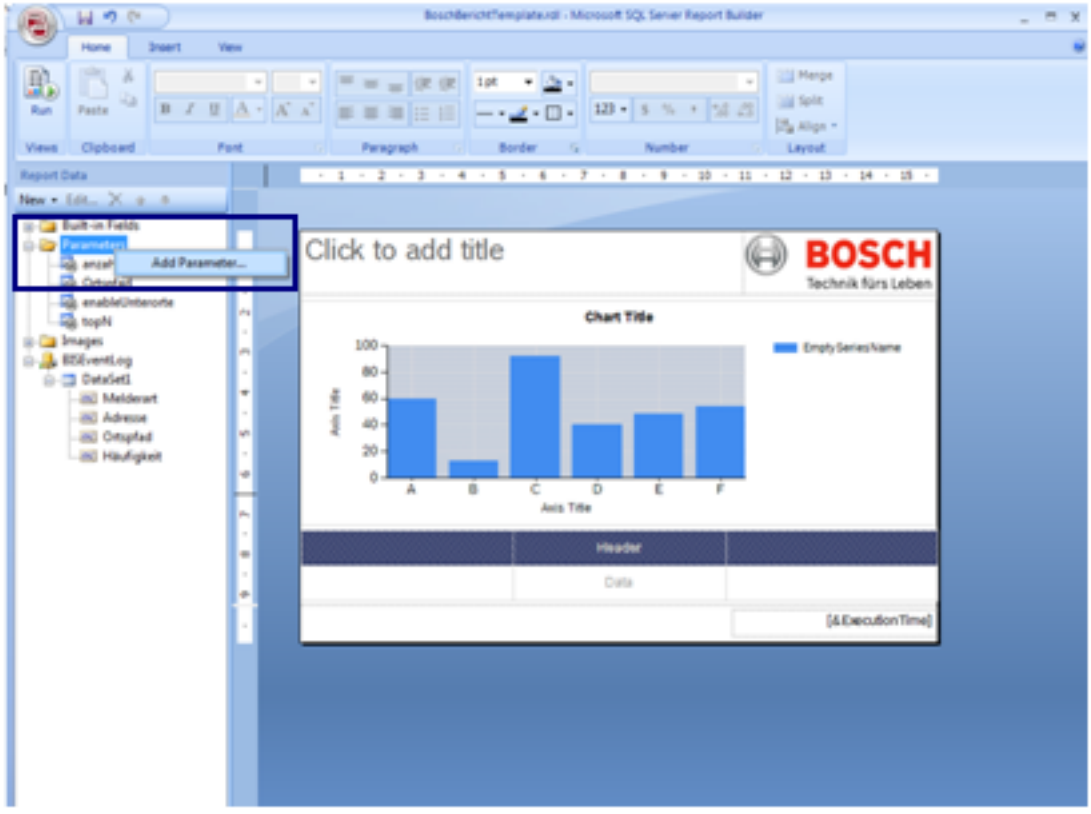


Figure 5.8: Fig. 20 Adding parameter

Step 8: Declare parameter userPermission

Now change Name and Prompt to **userPermission**.

Also allow inputs of **blank** and **null values**. In addition, you should set the **visibility** to **Hidden** . As a result, the parameter will not be displayed each time a report is called up.

Create the parameter by clicking on **OK.**

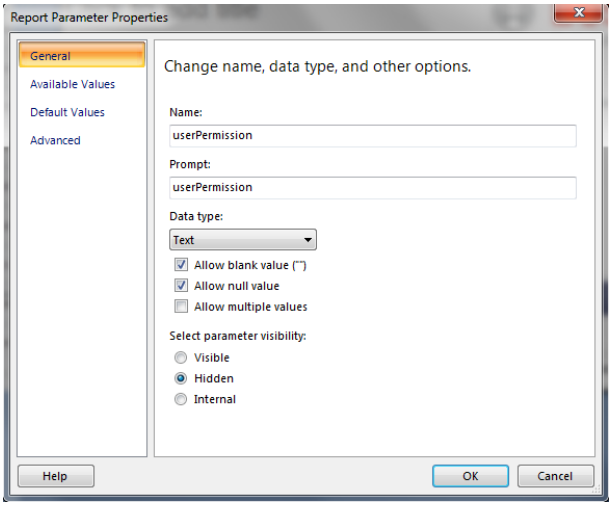


Figure 5.9: Fig. 21 userPermission

Step 9: Testing the report

Since you have now spent some time designing the report, you will no doubt be eager to test your report. To do this, click either on **F5** or the Button **Run** at the top left. Now enter the parameters to be passed and click on **View Report** to see a completed report. You can return to report editing mode by pressing **F8** or **Design**.

Step 10: Designing the report

To design the report, first adapt the table by moving the columns from the dataset to the empty table columns.

It may not be possible to see the columns under the dataset since your procedure has not yet been called up with the Report Builder. If this should be the case, right-click on your DataSet1 and open the Query Designer. Click here on the exclamation mark and enter the values for your parameters to run the report for the first time.

Important:

When you insert columns in the table, inclusion of more than 3 columns will result in the table and thus the page becoming larger.

Make sure that you adjust the page back to its original width so that the report can be output as a clear PDF document. If you do not reduce the report to the original width, this will automatically result in annoying page breaks which make reading the report much more difficult.

To add further elements, right-click on a free position in the report and click **Insert** --> ....

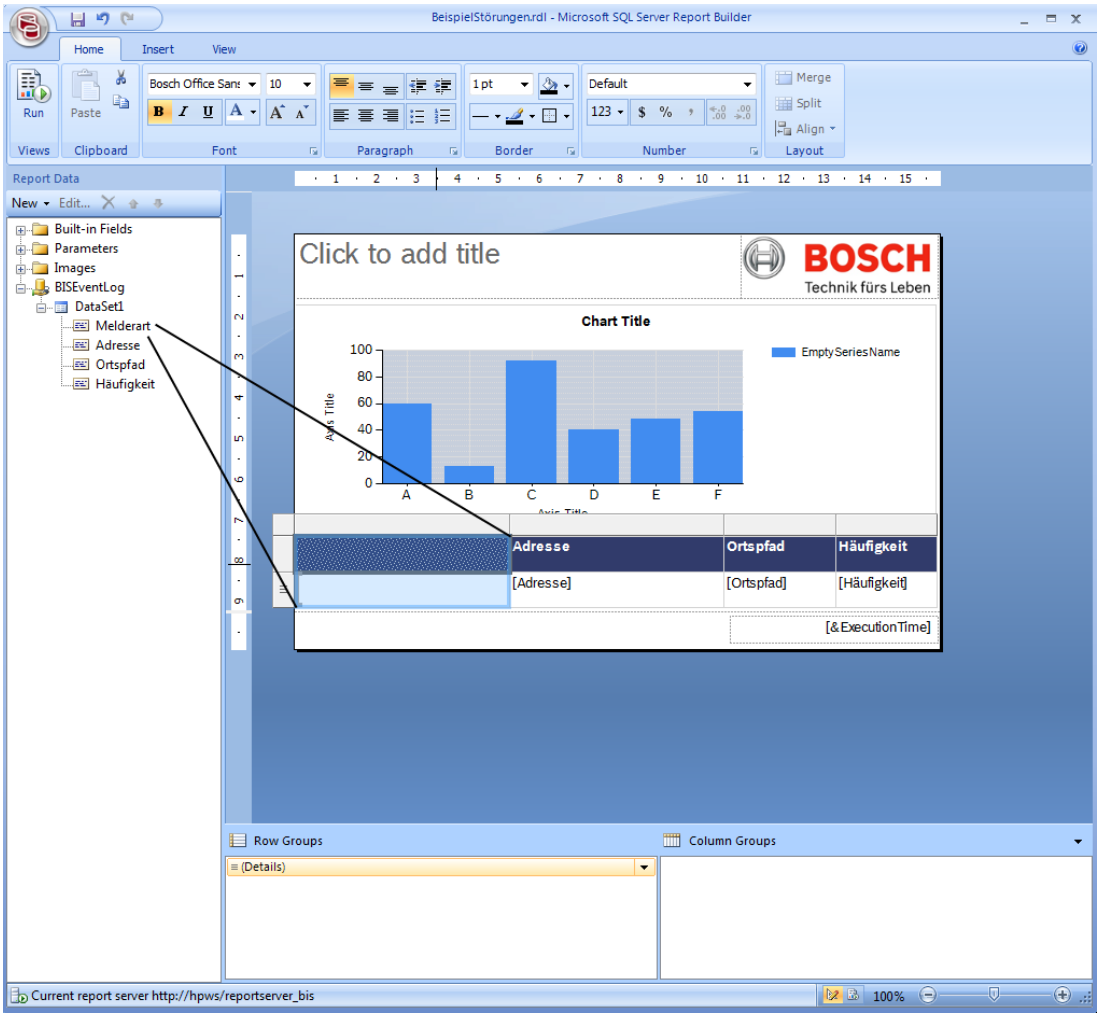


Figure 5.10: Fig. 22 Inserting table columns

Editing the bar chart

If you want to use the bar chart in addition to the table, drag the desired entries into the fields provided.

Then adapt the chart as required.

In the example, I have deleted most labels.

In order to show the heights of the columns directly above them, right-click on them and select **Show Data Labels** .

In order to obtain a standard Bosch design, it is now necessary to adapt the color of the columns. To do this, you must first right-click on the columns and open the **Series Properties** window. Now select the tab **Fill** and click on the **Fx** button to open the Color Expression window.

To give the report a Bosch design, change the color code here to **#c8313b6b**. The color corresponds to a semi-transparent dark Bosch Blue.

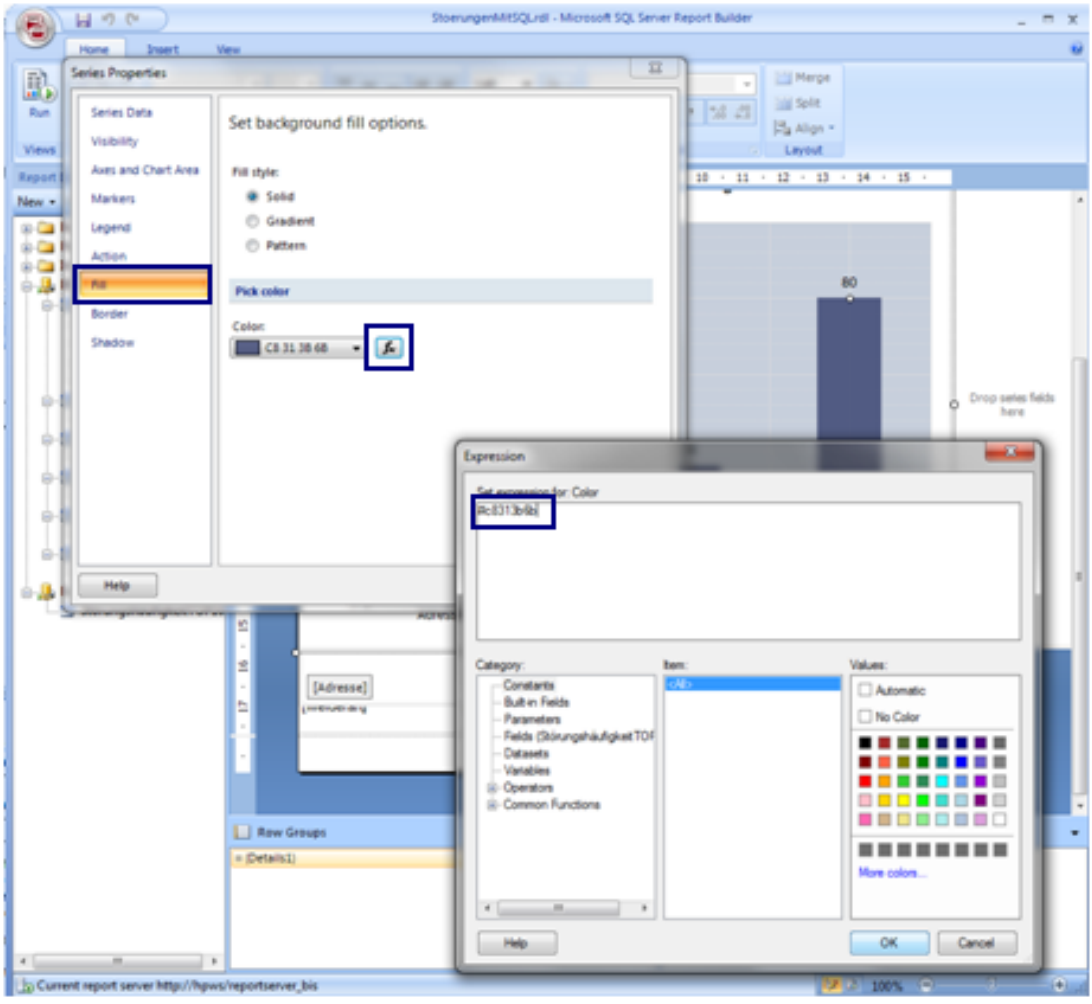


Figure 5.11: Fig. 23 Changing the color

Bosch Corporate Identity Guidelines

The example report is based on the Bosch Corporate Identity Guidelines. You can find these in the BGN under the following link:
https://inside-ws.bosch.com/FIRSTspiritWeb/permlink/wcms_rgap_-09_rbei_xxx_guidelines_and_standards_134-EN
If you would like to create your report in a Bosch design, make sure that the fonts in the given elements have already been set to Bosch Office Sans. If you add new elements, make sure that you change the font correspondingly.
Use the colors from the Bosch Style Guide so that the newly created elements also look like Bosch. The following color codes are a small extract of the colors:
Dark Bosch Blue: #313b6b
Light Bosch Blue: #92a0b9
Very light Bosch Blue: #c7cfdc

Creating alternating row colors

I have provided the table elements in the BoschBerichtTemplate (Bosch report template) which show the column content with alternating colors. This is done using the command:
=IIF(RowNumber(Nothing) Mod 2, "#92a0b9", "#c7cfdc")
#... represents the respective color code here.

Defining default values for variables

After pressing Run, you very probably first saw an empty report page. You were able to access the relevant results only after entering the parameters and pressing View Report.

So that this process can take place more quickly, the Report Builder has an integrated **Default Values** function for parameters. This allows a certain value to be assigned as a default value to all parameters. Report Builder then already generates a standard report with the set default values after clicking on Run.

So that this also happens in your case, right-click on one of your parameters and open the **Parameter Properties** window. Now go to the tab **Default Values.** You can now define fixed values or find out your desired value by means of a procedure. In the example, I choose the default value 2 for the parameter anzahlTage (number of days).

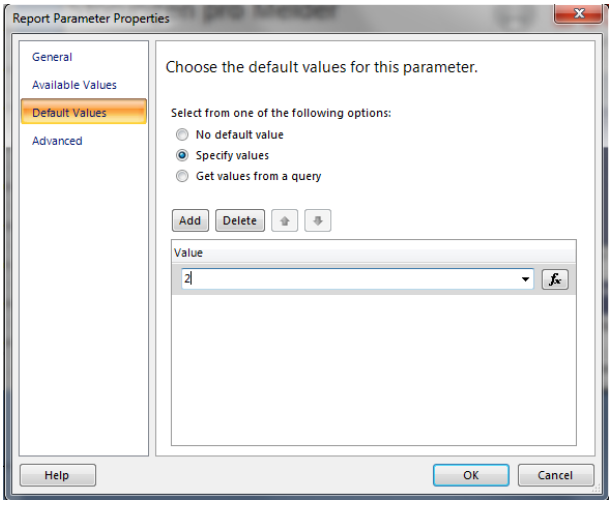


Figure 5.12: Fig. 24 Default values of variables

Changing the displayed name of the parameters

So that the operator knows what he has to enter in the case of unclear parameter names, it is also possible to change the displayed parameter name.

To do this, change the **Prompt** on the tab **General** in the **Report Parameter Properties** window.

Adding selection window for parameter input

You may have noticed that input of the parameters is not particularly user-friendly in its original form. To change this, you can offer the operator a selection of predefined values, for example.

To do this, select the tab **Available Values** in the **Report Parameter Properties** window.

In the example I create a selection window which displays a selection of all location paths. To do this, I use the option **Get Values from a query** to select my previously created dataset/ procedure getOrtspfade (get location paths). Further adaptations are possible by means of **Value** and **Label field**. Different selections are made here, for example, if you want to output the StatusID as a value and the StatusName as a displayed name.

Incidentally, the source text of the procedure getOrtspfade (get location paths) can be found earlier in the manual under the explained command UNION. This returns all location paths entered in the BIS and adds the selection 'Alle' ('All') to the output.

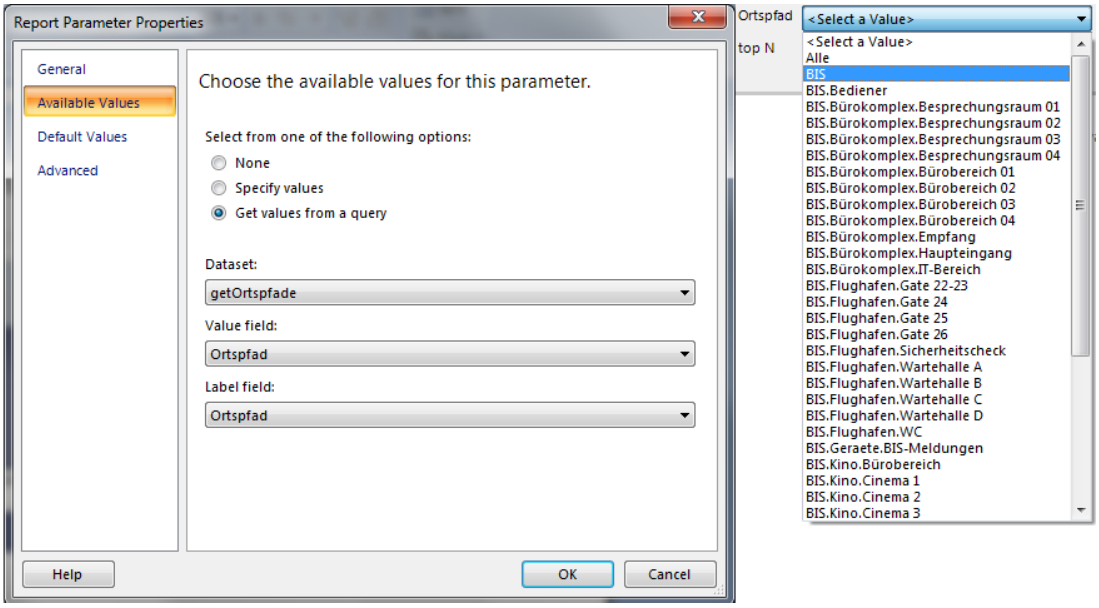


Figure 5.13: Fig. 25 Adding parameter selection

Inserting selection window "Sort by"

The displayed tables tend to become unclear if the outputs are too large. The Report Builder offers a sorting function so that you can find a specific detector more quickly or arrange the table in a clearer way. This overrides the sorting function of Select and offers the advantage that the customer can choose the criterion for sorting himself.

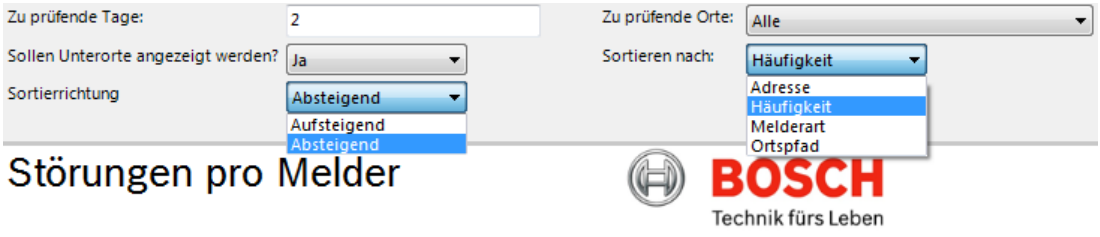


Figure 5.14: Fig. 26 Sorting function selection window

To set up a sorting function for your table, click on the table and then right-click on **gray square** that appears at the top left.

Now select **Tablix Properties...** and click on the **Sorting** tab in this window.

To allow the customer to choose whether to sort the table in ascending or descending order, add two new sorting options here by means of **Add** . The first option sorts the table in ascending order and the second in descending order. The option used depends on the parameter **direction**.

You should therefore set the **Order** of the first sorting option to **A to Z**. To do this, open the corresponding function description using **fx** and enter the following command:

=IIF(Parameters!direction.Value="Ascending“, Fields(Parameters!SortBy.Value).Value, 0)

The second option specifies the order **Z to A** and is programmed with the following command:

=IIF(Parameters!direction.Value="Descending",Fields(Parameters!SortBy.Value).Value, 0)

The close the sorting options by clicking on OK.

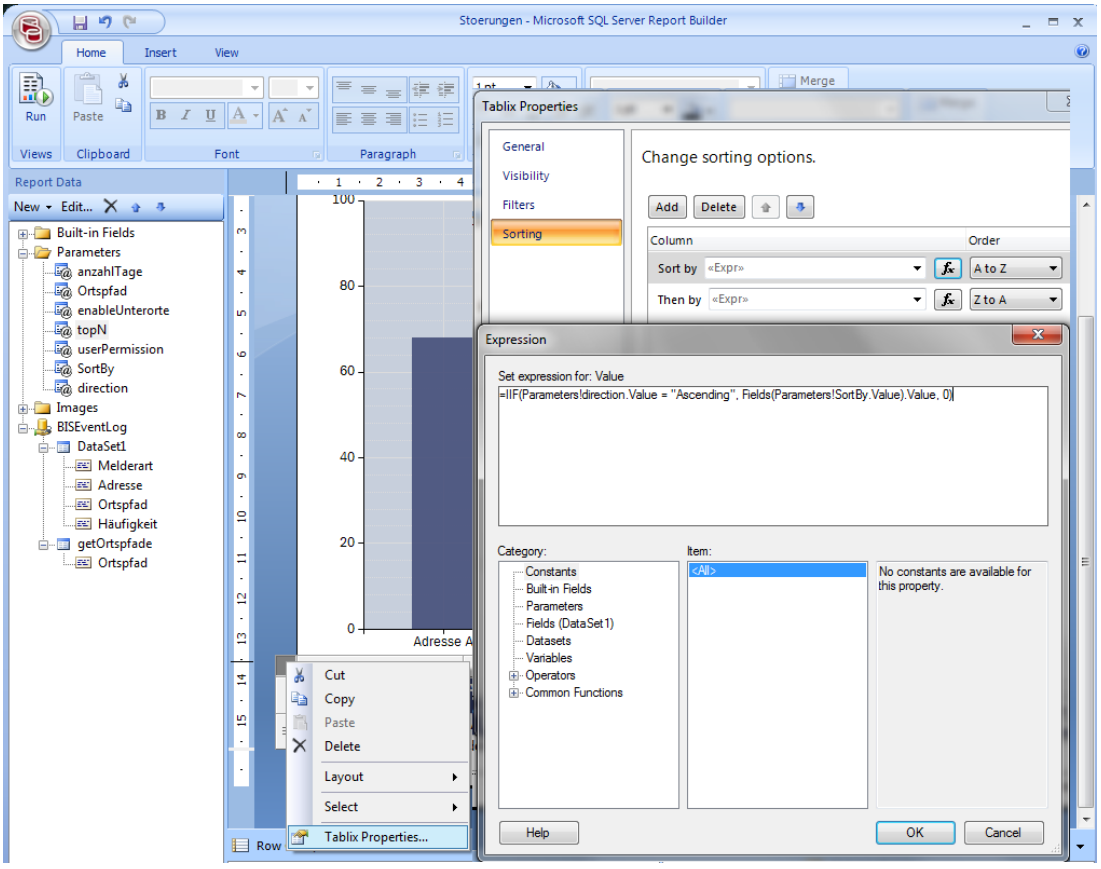


Figure 5.15: Fig. 27 Tablix Properties

The last item for setting up a sorting function is to add two new parameters by means of **Add Parameter** .

The first with the name **SortBy** specifies the column according to which the table should be sorted.

The second with the name **direction** specifies whether the tables are to be sorted in ascending or descending order.

Rename the first parameter to **SortBy** , change the **Prompt** of the parameter and open the tab **Available Values.**

Under **Specify Values**, add all your column names that should be available for sorting. Finally, define a **Default Value** so that the report is displayed immediately when it is called up.

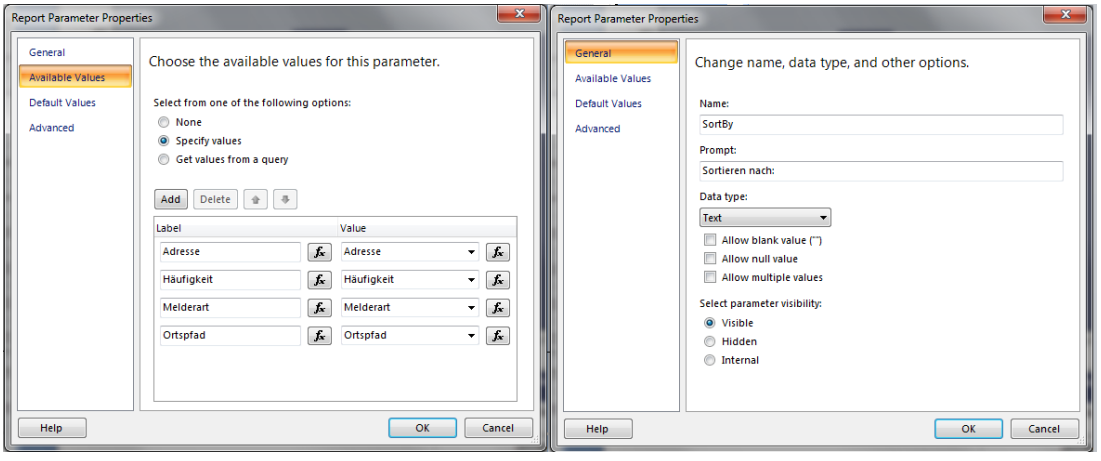


Figure 5.16: Fig. 28 Parameter SortBy

Name the second parameter **direction**, change its **Prompt** and open the tab **Available Values.** Add two new values here. You can choose any label, but the value must be **Ascending** in one case and **Descending** in the other.

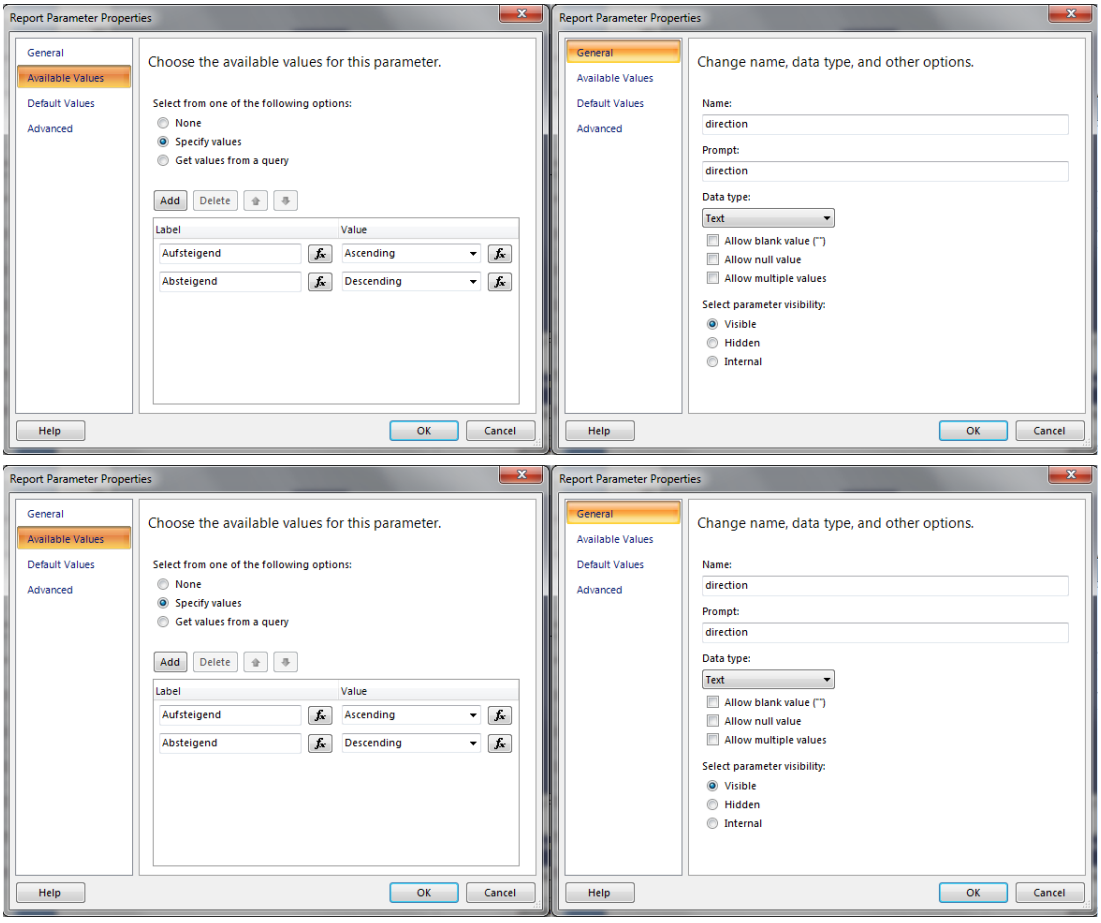


Figure 5.17: Fig. 29 Parameter direction

Step 11: Saving and integrating the report in BIS

When you have finished your report, you can save it so that it is displayed in the BIS.

To do this, press **Save** and select your report server under **Recent Sites and Servers** . In the example this is http://HPWS/Reports_BIS

Select the pre-installed folder **BIS Reports** here.

In the example, I have previously created a folder with the name **Eigene Berichte** (Own reports) via the page http://localhost/Reports_BIS and saved the report there. In Section 6 "Report distribution", you will learn how the folder can also be generated by means of a batch file.

I have saved my example report as **BeispielStoerungen.rdl** . The report can now be found in the BIS under this name.

Important:

Do not save the report under MyReports as the BIS does not have access to this folder.

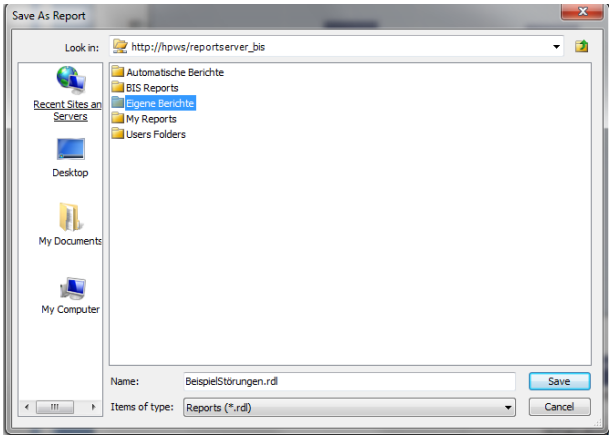
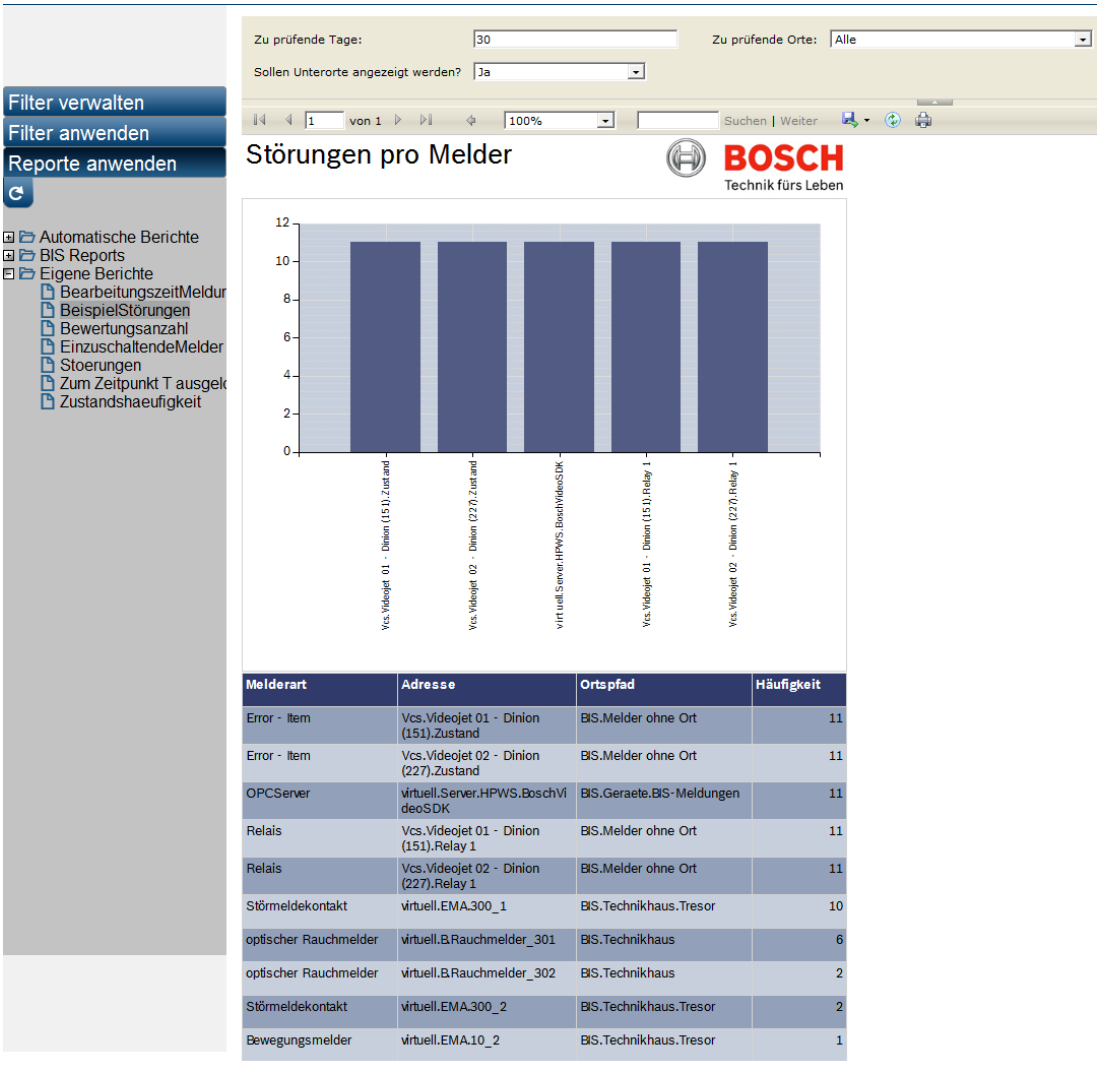


Figure 5.18: Fig. 30 Save location for report

Step 12: Calling up the report

Now start the BIS client and open the **Logbook**. Click there on the tab **Reporte anwenden** (Apply reports). Now select your report. In the example, the report can be found under the name BeispielStoerungen (ExampleMalfunctions).



Melderart	Adresse	Ortspfad	Häufigkeit
Error - Item	Vcs.Videojet 01 - Dinion (151).Zustand	BIS.Melder ohne Ort	11
Error - Item	Vcs.Videojet 02 - Dinion (227).Zustand	BIS.Melder ohne Ort	11
OPCServer	virtuell.Server.HPWS.BoschVideoSDK	BIS.Geraete.BIS-Meldungen	11
Relais	Vcs.Videojet 01 - Dinion (151).Relay 1	BIS.Melder ohne Ort	11
Relais	Vcs.Videojet 02 - Dinion (227).Relay 1	BIS.Melder ohne Ort	11
Störmeldekontakt	virtuell.EMA.300_1	BIS.Technikhaus.Tresor	10
optischer Rauchmelder	virtuell.B.Rauchmelder_301	BIS.Technikhaus	6
optischer Rauchmelder	virtuell.B.Rauchmelder_302	BIS.Technikhaus	2
Störmeldekontakt	virtuell.EMA.300_2	BIS.Technikhaus.Tresor	2
Bewegungsmelder	virtuell.EMA.10_2	BIS.Technikhaus.Tresor	1

Figure 5.19: Fig. 31 Example report in BIS

5.3 Common error messages when creating reports:

On the following pages you will find the most common errors which occur when working with the Report Builder and corresponding solutions.
The individual errors can be identified by means of the respective error messages.

In the Report Builder:

- „This report cannot be run in Report Builder because it contains one or more embedded data sources with credential options that are not supported. Instead of embedded data sources use shared data sources or save and view the report on the server”
  - Report no longer functions in the Report Builder and on the server. However, it still mostly functions in the BIS.
  - It is therefore not absolutely necessary to rectify the error.
- Possible solution for rectifying the error message:
  - Data source properties -> in Credentials, set to "do not use credentials" -> in General, select "Use a shared connection or report model”. -> select created database BISEventLog.
  - Confirm with OK. Report functions in the Report Builder, on the server and in the BIS.
- Possible solution for rectifying the error message:
  - Right-click on DataSource -> Data source properties -> Credentials -> change from “use current Windows user. Kerberos delegation may be required.” to “use this user name and password”. Enter username logbuch_query and password pw_logbuch_query. Do not check box. Click on OK and test again. -> Report should be displayed in test.

- „The EXECUTE permission was denied on the object […]-Dataset […]”:
  - The SQL procedure in the BISReports database does not have the permission for the user logbuch_query to access it.
  - If this error message occurs, please take another look at the section *Adapting the permissions, page 30*.
  - **Remedy:** Execute the code "`GRANT EXECUTE ON BISReports.dbo.<Prozedurname> TO logbuch_query`" or add it in the procedure report_Stoerungen

- The Report Builder cannot access the database when opening the Report Query Designer.
  - **Remedy:** Report Builder "Open as administrator"

- „… Parameter is missing a value“
  - The procedure that is to output the parameter does not have any output. In other words, when searching for a stateNumber in the BIS, no detector had this state yet.
  - **Remedy:** You can find the solution to this problem in the example in the section *Explained example queries from the BIS, page 24* under the question "How can I find out the stateID of a state"

After pressing “Run” in the Report Builder : "Failed to preview report":

This message can have a large number of different causes. For example, it is caused by an incorrectly/undefined data source. Display the details to obtain a more informative error message.

„This report cannot be run in Report Builder because it contains one or more embedded data sources with credential options that are not supported. Instead of embedded data sources use shared data sources or save and view the report on the server”

- 1. Possible solution for rectifying the error message:
  - Right-click on the data source -> Data source properties
  - In Credentials, set to "do not use credentials" -> In General, select "Use a shared connection or report model”.
  - Select database Eventlog Datasource.
  - Confirm with OK.
- 2. Possible solution for rectifying the error message:
  - Right-click on DataSource -> Data source properties -> Credentials -> change from “use current Windows user. Kerberos delegation may be required.”
  - to “use this user name and password”.
  - Enter username logbuch_query and password pw_logbuch_query.
  - Do not check box.
  - Click on OK and test again.

In the browser (http://localhost/reports_bis**):**

- "Im userPermission-Parameter fehlt ein Wert" ("A value is missing in the userPermission parameter")
  - This error message as such is not a problem. It is caused by the fact that the Visibility is set to "Hidden" in the Parameter Properties. It can be ignored because precisely the fact that userPermission is set to Hidden often means that the report is displayed only in the BIS.

In the BIS

- Report is not displayed:
  Possible causes:
  - The parameter "userPermission" is missing
    Remedy: Create the parameter again: Name and Prompt= "userPermission" Data type=Text; check "Allow blank value" and „Allow null value“, set Visibility to Hidden
  - The report was saved in the folder "My Reports". This folder is protected and is not displayed by the BIS.
    Remedy: Save the report in a different folder.

6 Report distribution

You can significantly reduce the amount of work involved in distributing the reports to the technicians and then installing them at the customer by following the steps described below. This is because these steps allow you to create an exe file which the technician simply then has to run on the customer's premises. The program will create the SQL procedure on its own and integrate the report so that everything is done with just a double-click.
For this purpose, place all the following files and source texts in one folder.

- Finished report as .rdl
- Source code as .sql for creating the main procedure in BISEventLog
- Source code for creating the link procedure in BISReports
- Source code which allows logbuch_query to access procedure
- .rss file, which saves the report on the report server
- .exe file, which executes the codes and calls the .rss file

You will notice that you have already created practically all the files mentioned. The only files that you are still missing should be the .rss and .bat files.

6.1 Creating the .RSS file

The .rss file is used to save the finished report on your SQL report server.
Since this involves a relatively large amount of source text, I have enclosed an example with this manual. It is saved in the following folder:
`BIS_Berichte\Berichte\Bericht_Stoerungen_pro_Melder \Stoerungen_pro_Melder.rss`
Open the document **Stoerungen_pro_Melder.rss** with the Notepad/Editor.
All you need to do now is edit the variables **parentFolder** and **reportName**.
ParentFolder is the name of the report server folder in which the report should be placed. e.g. **Eigene Berichte** (Own reports)
ReportName is the name of your report under which you saved it.
In the example this would be **BeispielStoerungen.**
Save the document with a name without spaces and continue with creation of the .bat file

6.2 Creating the .BAT file

Create a new text document and write down the following edited command lines there.
Note:

- The texts in pointed brackets (<>) are variables which you must substitute with actual values. The variables are as follows:
  - `Localhost\BIS` is the server name where you have also logged in in the Management Studio.
  - `<OrdnernameDerSQLQueries>` stands for the file path which leads from the storage location of the Exe file to the source text.
  - `<BISEventLog_Prozedurcode.sql>` stands for your SQL file which creates a procedure in the BISEventLog database.
  - `<grantExecutePermission.sql>` refers to the SQL file which grants the user logbuch_query access to the procedure in BIS Reports. It contains the following code:
    `GRANT EXECUTE ON BISReports.dbo.<reportName> TO logbuch_query`

– `Localhost` is a variable which is automatically replaced by the computer name from which the batch file is called. Please note:
The .EXE file must be executed on the server computer!
– The commands have been written in two lines due to the page margins. However, you must write the commands in the text editor **without a line break**.

```
sqlcmd -S localhost\BIS -E -i <OrdnerpfadDerSQLQueries>
\<BISEventLog_Prozedurcode.sql>
sqlcmd -S localhost\BIS -E -i <OrdnerpfadDerSQLQueries>
\<BISReports_Prozedurcode.sql>
sqlcmd -S localhost\BIS -E -i <OrdnerpfadDerSQLQueries>
\<grantExecutePermission.sql>
rs -i <publishBeispielStoerungen.rss> -s http://localhost/reportserver_BIS
```

Save the text document with the file extension **.BAT** .

Additional notes
rs –i,... in the .BAT file will lead to your report being loaded to the report server with the name `reportserver_BIS`. 'Reportserver_BIS' is the default name for the report server and you can therefore adopt this also.
The batch file is not yet suitable for testing without conversion to EXE format since the file paths are not adapted for execution as .BAT.
If you do not want to convert the file for each test, you must add a new first line **cd /d "%~dp0"**. In addition, the five characters **%~dp0** must be added before the folder path of SQL Queries and the file must be "Run as administrator“.

6.3 Converting the batch file to EXE

Output of the batch file involves two difficulties.
The first is that the technician must manually execute the file with "Run as administrator". The second is that the complete file path of the batch file must not contain any spaces.
Since the latter is often not the case and the former is often forgotten, we will now convert the batch file to an .EXE.
To do this, first open the supplied program **Bat To Exe Converter v1.6**, which can be found under the following link:
`BIS_Berichte\Ext._Programme_und_Setups\Bat_To_Exe_Converter`

You can also alternatively download it from http://www.f2ko.de/downloads/Bat_To_Exe_Converter.zip.
Now select your previously created batch file and the location where you want to save the .EXE.
Now confirm the field **Add administrator manifest** so that the Exe is automatically run as administrator and confirm this by clicking on **Compile**.

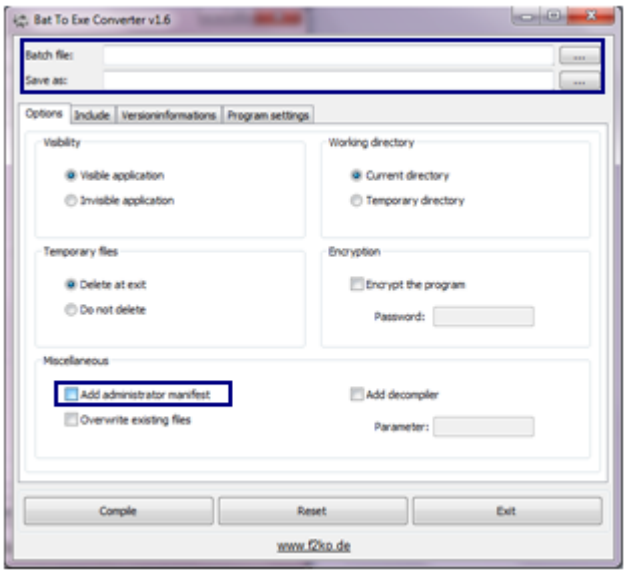


Figure 6.1: Fig. 32 Bat to Exe Converter

You are finished when you have completed all the above steps correctly and you can provide the technician with the .EXE file together with the SQL source texts and reports.

Bosch Sicherheitssysteme GmbH
Robert-Bosch-Ring 5
85630 Grasbrunn
Germany
www.boschsecurity.com