Erstellung von BIS-Berichten

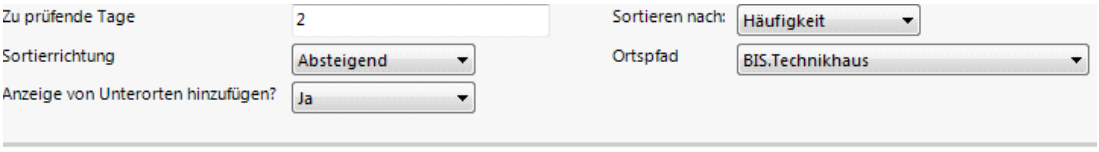
BOSCH

de Technical note

Inhaltsverzeichnis

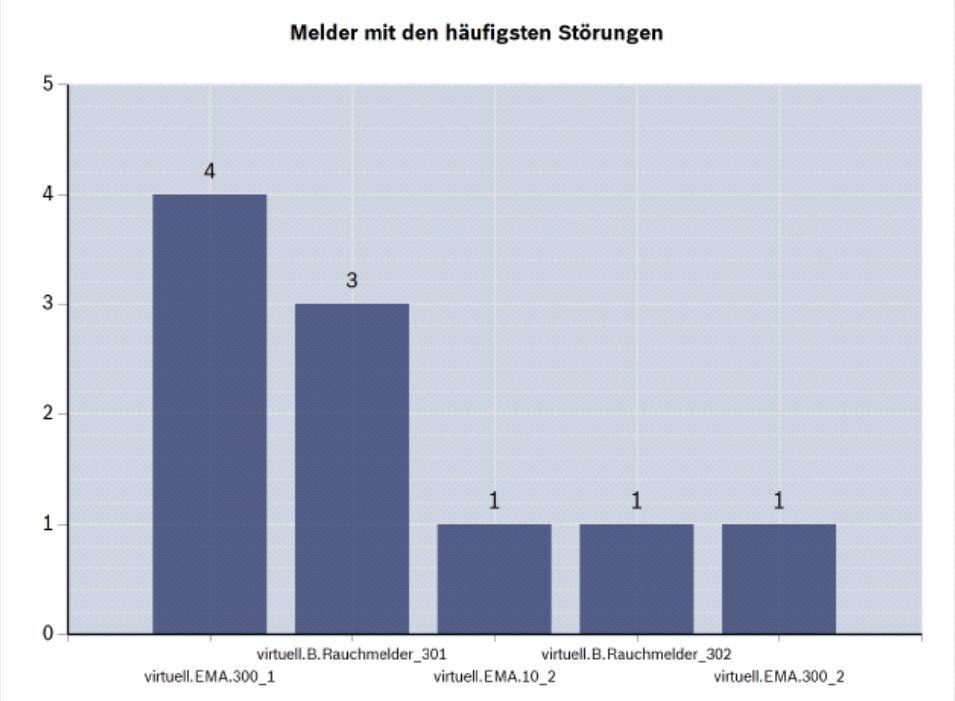
1 Was ist ein Bericht?

Ein Bericht wird generell zur Präsentation von Daten verwendet.
In unserem Fall dient er dazu die Fülle an Daten, welches das Logbuch der BIS aufzeichnet, auszuwerten und zu präsentieren. So können wirklich relevante Informationen mit Hilfe von Berichten schnell über die BIS abgerufen werden.
Im Folgenden ist der Beispielbericht „Störungen pro Melder“ zu sehen. Dieser zeigt, alle Melder, die in den letzten Tagen am häufigsten eine Störung gemeldet haben. Dazu erlangt er Informationen aus mehreren Tabellen des von der BIS erstellten Logbuchs um ein Säulendiagramm und eine Tabelle zu erstellen.
Die Ergebnisse werden in der BIS angezeigt und können durch einen einfachen Klick als PDF- oder Excel Datei exportiert werden. Somit lassen sie sich weiter verschicken und zur Dokumentation speichern.



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 Die 4 Schritte zum fertigen Bericht

Der Kern eines jeden Berichts ist eine SQL Prozedur. Diese Prozedur wird in Transact SQL geschrieben und stellt die Daten aus dem Logbuch dem Berichtseditor (Report Builder) in komprimierter Form zur Verfügung.
So kann man sich mit Hilfe einer Prozedur beispielsweise alle Melder, die zum jetzigen Zeitpunkt gestört oder abgeschaltet sind mit nur einem Klick ausgeben lassen. Auch die Darstellung, welche Melder am öftesten durch Störungen beeinflusst werden, ist problemlos möglich.
Das folgende Handbuch zeigt Ihnen welches Schritte Sie von der Idee einen Bericht zu kreieren, über das Umsetzen der Idee bis zum Anzeigen des fertigen Berichts in der BIS unternehmen müssen.
Nachfolgend sehen Sie welche Schritte dafür notwendig sind und den kurz umrissenen Inhalt des jeweiligen Kapitels.

2.1 Anforderungskatalog erstellen

Das Erstellen eines Berichts fängt wie die meisten Projekte mit dem Entwickeln eines Anforderungskatalogs an. Dieses Kapitel zeigt Ihnen, auf welche Sie Aspekte Sie dabei besonders Acht geben sollten.

2.2 Erlangen der Logbuchdaten mit Hilfe von SQL

Zum Erlangen der Logbuchdaten benötigen wir eine Prozedur in SQL welche diese aus dem Logbuch ausliest. Diese werden wir mit Hilfe des Microsoft SQL Server Management Studios schreiben.
In diesem Kapitel werden Sie mit den folgenden Schritten eine fertige SQL Prozedur erstellen können:

1. Installieren Sie zuerst die Grundlage des Programmierens; das SQL Management Studio
2. In diesem Schritt werden Sie lernen, wie die Datenbank BISEventLog, also das Logbuch der BIS aufgebaut ist.
3. Es folgt die Erläuterung, wie Sie allgemeine Tabellenausgaben mit SQL verfassen können. Dabei werden Sie vor allem die Befehle kennen lernen, die für unsere Aufgabenstellung am hilfreichsten sind.
4. Anschließend werden Sie, aufbauend auf diesem Basiswissen, weitere BIS spezifische Fragen beantwortet bekommen. Dies beinhaltet die Beantwortung von Fragen wie man sich z.B. den Ortspfad, die Adresse oder den Meldertyp eines gewünschten Melders mit SQL ausgeben lassen kann.
5. Darauf folgend sehen Sie noch den erklärten Quelltext von fertigen Berichten.

Zum Schluss dieses Schrittes, haben Sie eine Prozedur erstellt, die all Ihre benötigten Informationen mit nur einem Klick ausgibt.

2.3 Präsentation der Daten mit Hilfe von Berichten

Sie können sich nun alle Informationen in einer Tabelle ausgeben lassen. Allerdings ist die Tabelle nur in einem Schwarz-Weiß Design zu betrachten.
Da dies nicht unser Ziel ist, werden Sie in diesem Schritt lernen, wie Sie Ihre Informationsausgabe übersichtlich, professionell und optisch ansprechend darstellen können. Dabei hilft Ihnen vor allem der der Microsoft Report Builder weiter. Was genau Sie mit diesem Programm erwartet und wie man damit umzugehen hat, finden Sie in diesem Kapitel in einer mit Bildern verfeinerten Schritt für Schritt Anleitung.

Am Ende dieses Kapitels ist Ihr Bericht fertig designt und über die BIS abrufbar.

2.4 Einfache Verteilung der Berichte durch eine Exe Datei

In dem vorherigen Abschnitt haben Sie erreicht, dass der Bericht auf Ihrer BIS wie gewünscht angezeigt wird. Das Ziel ist allerdings, dass der Bericht beim Kunden abgerufen werden kann. Damit die Einrichtung beim Kunden möglichst einfach funktioniert, zeige ich Ihnen, wie Sie eine Exe Datei schreiben können, welche allein durch einen Doppelklick vom Techniker vor Ort den Bericht in die BIS einbindet. Dies beinhaltet ebenso, dass der Techniker keine weiteren Programme beim Kundenrechner installieren muss. Die Installation der BIS und des SQL Servers ist vollkommen ausreichend.
Schlussendlich ist der Bericht erfolgreich beim Kunden eingebunden und durch diesen in der BIS aufrufbar.

3 Anforderungskatalog erstellen

Da Sie nun grob über die Schritte, die notwendig sind um einen Bericht zu erstellen Bescheid wissen, beginnt nun das Handbuch mit dem ersten Schritt.
Es ist notwendig, dass Sie ein Handbuch erstellen.
Im Dokument sollten unter anderem folgende Fragen geklärt werden:

- Was ist der Zweck des ausgegebenen Berichts?
- Welche Parameter soll der Kunde selbst eingeben und welche sollen vordefiniert werden? z.B. die Daten von wie vielen Tagen soll die Prozedur auslesen?
- Falls Parameter vordefiniert werden, mit welchem Wert sollen Sie belegt werden?
- Welche Spalten sollen angezeigt werden? Adresse, Meldertyp, Ortspfad,...
- Wie sollen die Ergebnisse präsentiert werden? Tabelle, Diagramm, Graph,...
- In welchem Design soll der Bericht verfasst werden? Bosch- o. Kundendesign?
- Wie viele Zeilen soll der Bericht maximal ausgeben?

4 Erlangen der Logbuchdaten

Sobald der Anforderungskatalog erstellt wurde, ist der nächste Schritt die Logbuchdaten auszulesen.
Dazu müssen Sie zuerst wissen, dass alle Daten, die wir benötigen in einem großen von der BIS erstellten Logbuch hinterlegt sind. Dieses Logbuch zeichnet alle Ereignisse, wie das Auslösen eines Melders, auf und speichert es in einer Datenbank.
Diese Datenbank befindet sich auf einem SQL Server unter dem Namen **BISEventLog**.
Dieses Logbuch wird von uns mit einer Prozedur nach den gesuchten Informationen durchforstet, um anschließend die gewünschten Ergebnisse in einer Tabelle ausgeben zu können.
Sowohl um einen Einblick in das Logbuch zu bekommen, als auch um die Prozedur zu schreiben, benötigen Sie das Programm **Microsoft SQL Server Management Studio**.
Dieses Kapitel führt Sie durch die folgenden 4 Schritte damit Sie eine funktionstüchtige SQL Prozedur haben, welche anschließend optisch aufgearbeitet werden kann.

1. Installation des Management Studios
2. Logbuchdatenbank kennenlernen
3. Schreiben der Prozedur in T-SQL
4. Zugriff des Report Builders ermöglichen

4.1 Installation des Microsoft SQL Server Management Studio 2012

Das Microsoft SQL Server Management Studio ist ein unerlässliches Programm um neue Berichte entwickeln und implementieren zu können. Um es installieren zu können, sollten Sie zuallererst sicherstellen, dass ein SQL Server auf Ihrem Rechner installiert ist. Dies sollte schon mit der Installation der BIS einhergegangen sein. Wenn Sie die Grundlage installiert haben, können Sie mit folgender Anleitung fortfahren:

Schritt 1:
Beschaffen Sie sich die Installationsdatei des SQL Management Studios
Diese befindet sich, zum Beispiel:

- auf dem BIS Installationsmedium unter: `\3rd_Party \SQL2012SP1\1033\32\1033_ENU_LP\x86\Setup\`
  - Wählen Sie die Datei **SQL_SSMS_LOC.MSI** aus
- oder auf der Downloadseite des Microsoft SQL Server 2012 Express unter: www.microsoft.com/en-us/download/details.aspx?id=29062
  - Klicken Sie dazu auf **Download** und wählen Sie die Datei **ENU\x64\SQLManagementStudio_x64_ENU.exe** aus.

Hinweis: Die Installation und die folgenden Bilder sind auf Englisch. Falls Sie Ihr SQL Management Studio auf Deutsch haben wollen, laden Sie sich bitte die deutschen Installationsdateien herunter und denken Sie sich meine Screenshots der Installation auf Deutsch um.

Schritt 2:
Klicken Sie mit einem Rechtsklick auf die Installationsdatei und führen Sie das Setup **als Administrator** aus.

Schritt 3:
Warten Sie bis die Installationsdateien entpackt worden sind und sich das „SQL Server Installation Center“-Fenster geöffnet hat. Wählen Sie die Option **New SQL Server stand-alone installation or add features to an existing installation aus** und klicken Sie über das nächste Fenster in dem nach Updates gesucht wird hinweg.

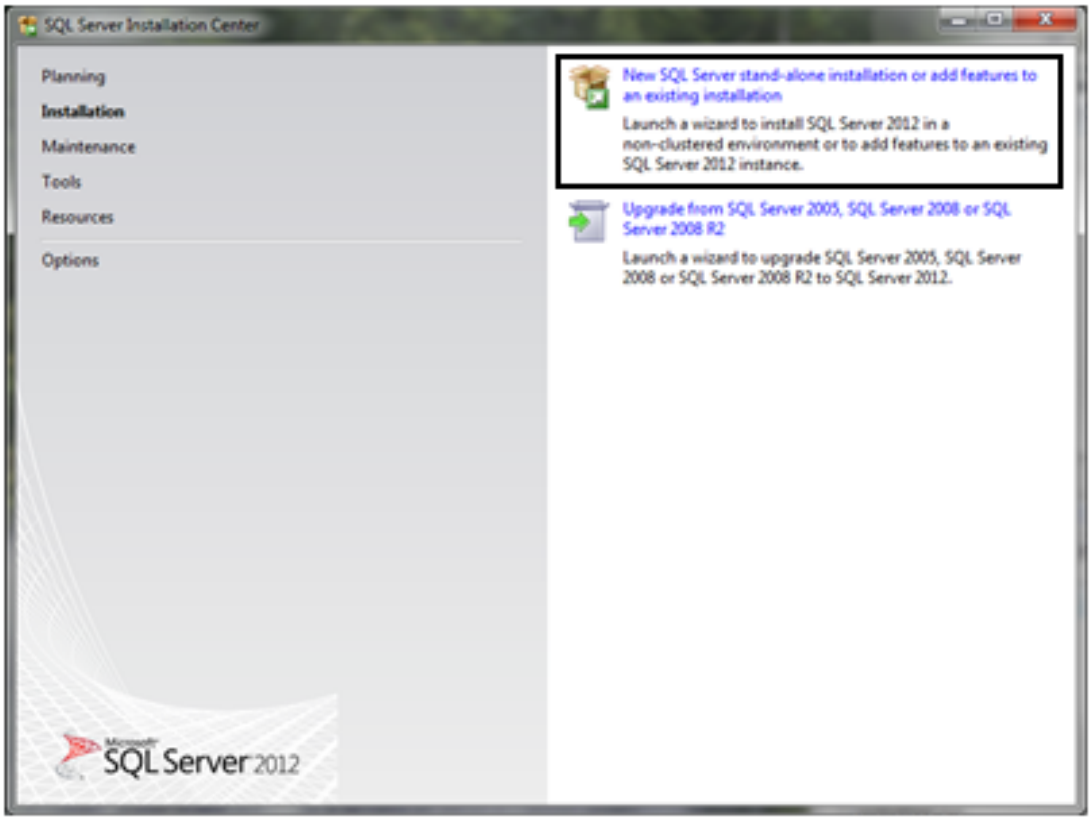


Bild 4.1: Abb. 3 SQL Server Installation Center

Schritt 4:
Nachdem ein paar Dateien installiert worden sind, müssen Sie den Installationstyp auswählen. Auf dem „Installation Type“ Fenster, haben Sie die Auswahl zwischen „**Perform a new Installation [...]“** und „**Add features to an existing instance [...]“**.
Achtung!
Wählen Sie hier den Punkt **Perform a new installation aus.**

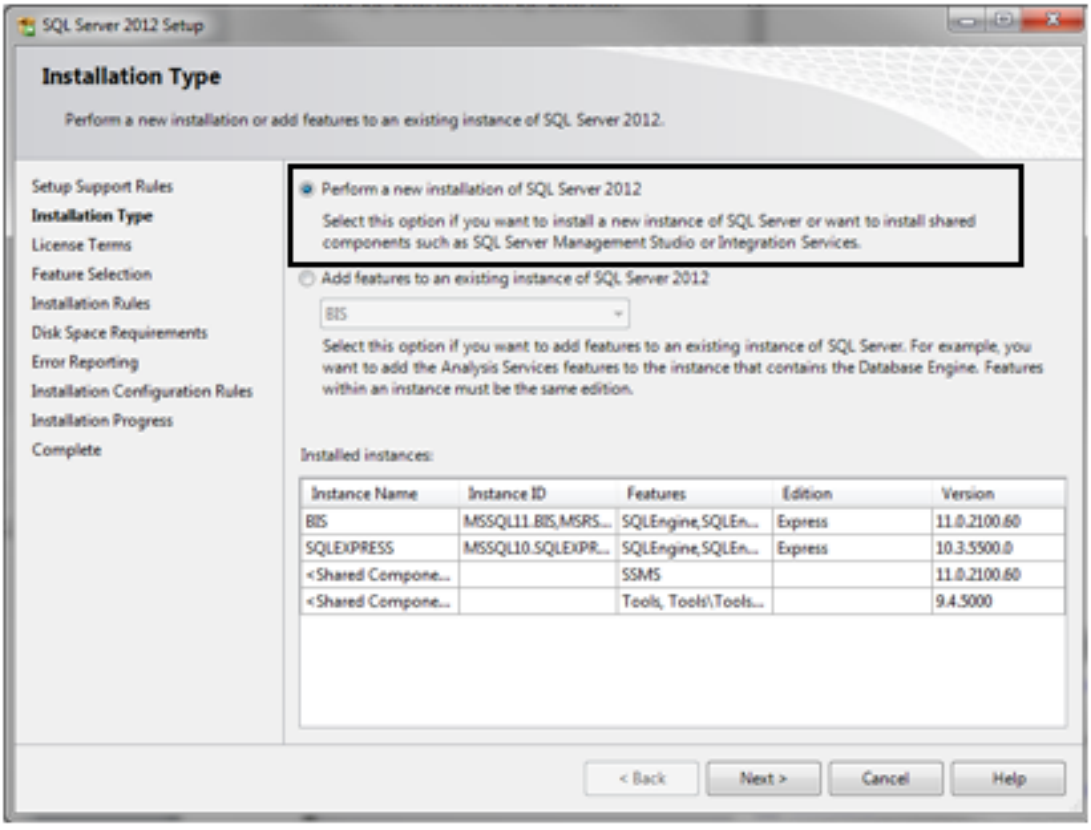


Bild 4.2: Abb. 4 Installation Type

Schritt 5:

Nachdem Sie die AGBs bestätigt haben, kommen Sie zum dem Fenster, bei dem Sie das zu installierende Feature auswählen können. Setzten Sie hier einen Haken bei **Management Tools – Basic** aus und klicken Sie auf Next.

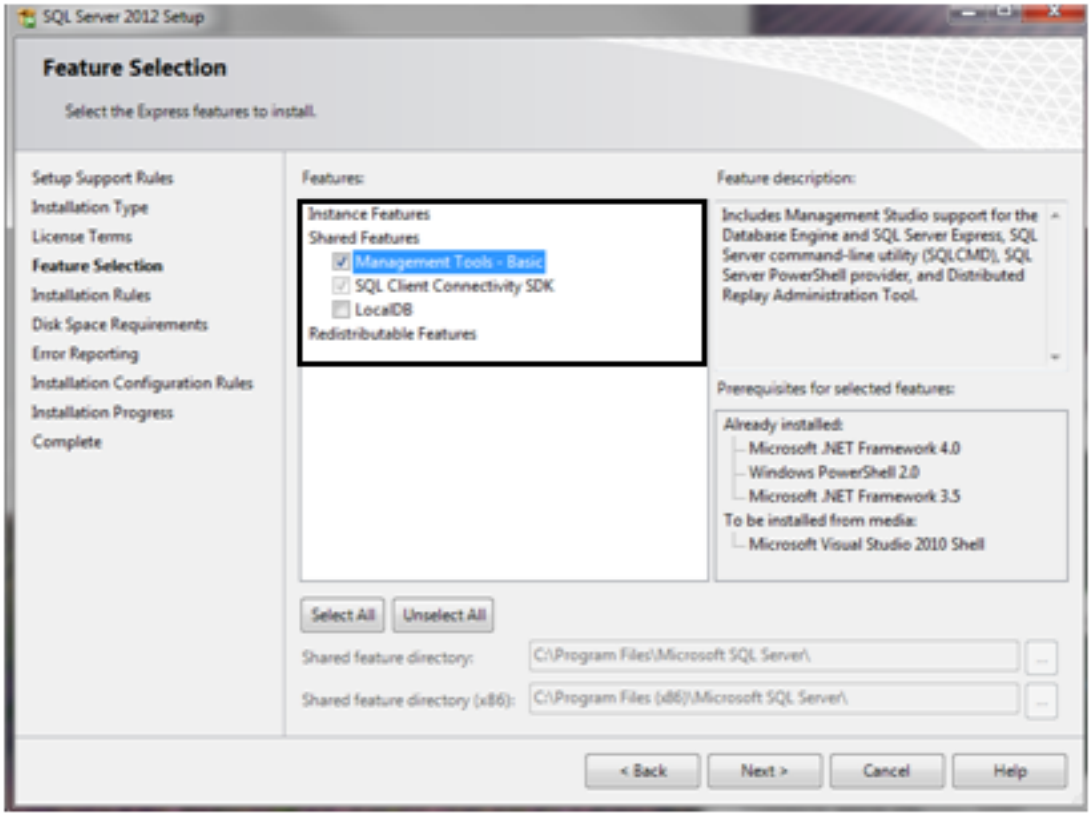


Bild 4.3: Abb. 5 Auswahlfenster Management Tools

Schritt 6:

Nachdem Sie über das Fenster, bei dem Sie Microsoft zu dem Senden von Error Reports berechtigen können hinweg sind, haben Sie die Installation abgeschlossen.

Starten Sie das SQL Management Studio stets als Administrator um vollständigen Zugriff auf alle Funktionen erhalten zu können. Sie sollten es in Ihren Programmen unter Microsoft SQL Server 2012 wiederfinden.

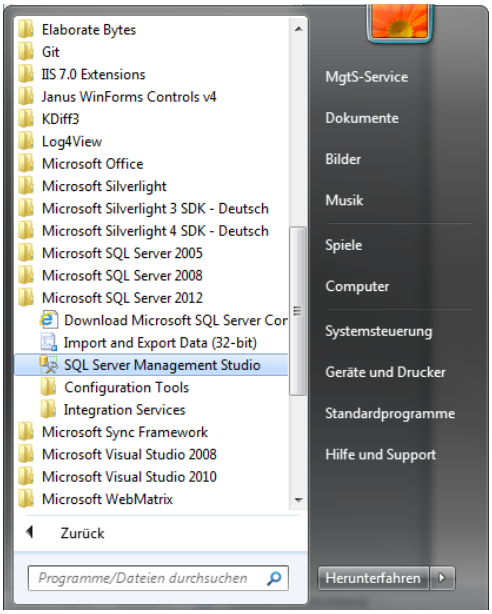


Bild 4.4: Abb. 6 Dateipfad Management Studio

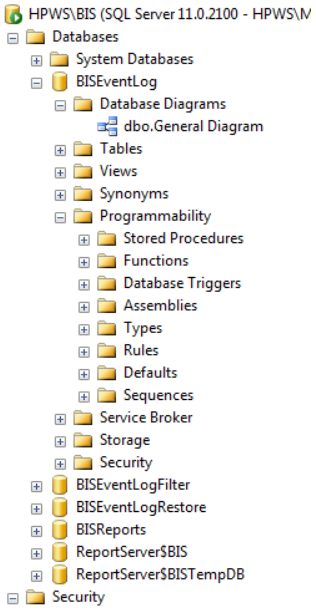
Schritt 7:
Starten Sie nun das Management Studio um Ihren Bericht zu implementieren und verbinden Sie sich mit Ihrem lokalen BIS Server über die Windows-Authentifizierung.
Falls diese nicht funktionieren sollte, können Sie sich auch über die SQL Authentifizierung mit folgenden Daten einloggen:
Benutzername: **logbuch_query**
Passwort: **pw_logbuch_query**

4.2 Aufbau des SQL Servers der BIS

Sobald Sie sich auf dem SQL Server angemeldet haben, ist Ihnen wahrscheinlich aufgefallen, dass dieser eine Fülle an Informationen beinhaltet.
Damit Sie bei diesem Überfluss an Daten stets den Überblick behalten können, erklären Ihnen die nächsten Zeilen den Aufbau des mit der BIS verknüpften SQL Servers.
Dazu gehören der Aufbau Navigationsleiste und die Struktur der wichtigsten Datenbanken. Für uns sind speziell zwei Datenbanken besonders wichtig: **BISEventLog** (eigentliches Logbuch) und **BISReports**.

4.2.1 Aufbau der Navigationsleiste

Sobald Sie sich im SQL Server Management Studio angemeldet haben, ist am linken Bildschirmrand die Navigationsleiste Ihres SQL Servers zu sehen.



Unter dem Punkt **Security** verbergen sich die Logindaten. Der Punkt **Databases** führt Sie zu den System- und zu den BIS-Datenbanken. Hier sehen Sie die für uns relevanten Datenbanken **BISEventLog** und **BISReports**.
Unter **Database Diagrams** können Sie ein Diagramm erstellen, wie es unter 4.2.2.1 zu sehen ist, welches alle Tabellen mit den dazugehörigen Attributen und Verknüpfungen anzeigt.
In **Tables** sind die Tabellen, in denen die Meldungen der BIS gespeichert wurden hinterlegt.
Im Navigationspunkt **Programmability** sehen Sie die Möglichkeiten zum Programmieren, die uns SQL liefert. Wir werden dabei vor allem die **Stored Procedures** betrachten.
Wichtig!
Alle Ihre Quellcodes werden Sie unter Stored Procedures erstellen und speichern.

Unterschied zwischen Prozedur und Funktion
Arbeitet man sich ein bisschen in SQL ein, erkennt man das Prozeduren und Funktionen sich sehr ähnlich sind. Mit beiden lassen sich Tabellenausgaben produzieren. Der Unterschied zwischen beiden liegt darin, dass Funktionen im Vergleich zu Prozeduren durch einen Rückgabewert ausgezeichnet sind. Bei Ihnen muss folglich jedesmal eine Tabelle oder sonstiges erstellt werden.
Somit ist der Vorteil von Prozeduren, dass bei Ihnen eine Tabelle lediglich angezeigt wird. Auf diese angezeigte Tabelle können nun externe Programme, wie der Report Builder, zugreifen. Eine interne Prozedur kann allerdings nicht auf eine Tabelle einer anderen Prozedur zugreifen!

4.2.2 Bedeutung und Inhalt der relevanten Datenbanken

Im Folgenden werden Sie eine Prozedur erstellen und dazu müssen Sie Informationen aus einer Datenbank auslesen. Diese Speicher können allerdings mehrere Gigabyte an Informationen enthalten. Falls man nun versucht ohne Hilfe den Aufbau und Inhalt einer solchen Datenbank zu verstehen, ist dies ein langwieriger nicht immer erfolgreicher Prozess. Deshalb umreiße ich in den folgenden Absätzen den Aufbau, die Zusammenhänge und die Bedeutung der Attribute der relevanten Datenbanken.

BISEventLog

BISEventLog ist für uns die wichtigste und inhaltsreichste Datenquelle. Sie stellt das Logbuch der BIS dar und vermerkt somit alle Meldungen und Ereignisse, die die BIS registriert. Deshalb sind in Ihr fast alle Informationen gespeichert, die man mit Hilfe eines Berichts ausgeben lassen könnte.
Die folgenden Absätze dienen dazu schnellen Einblick in die Struktur des EventLogs zu gewinnen. Das folgende Diagramm wurde von mir unter Database Diagrams erstellt und zeigt die wichtigsten Tabellen mit ihren Verknüpfungen.

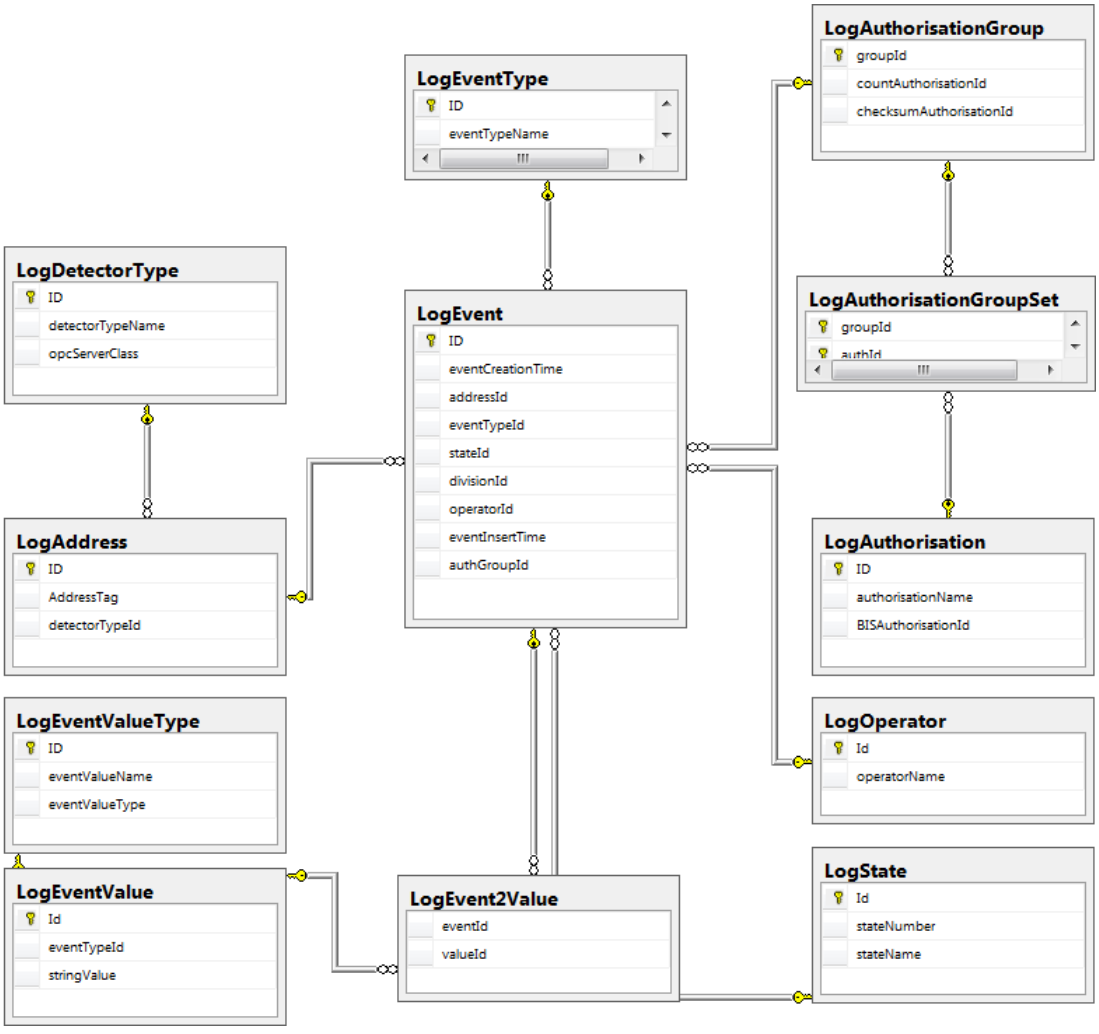


Bild 4.5: Abb. 9 Tabellen Diagramm BISEventLog

LogEvent

Dies ist die **Haupttabelle des Logbuchs**. Jedes Ereignis in der BIS registrierte Ereignis wird hier gespeichert. Über die einzelnen Attribute ist diese Tabelle mit jeder anderen relevanten verbunden.

Beim Betrachten der Tabelle (durch SELECT * FROM LogEvent) wird Ihnen Auffallen, dass viele Ereignisse mit nahezu der gleichen Zeit eingetragen worden sind. Dies liegt daran, dass z.B. bei einem Feueralarm nicht nur ein Ereignis gemeldet wird.

So werden oft bis zu sechs Ereignisse allein beim Auslösen des Melders registriert. Dies ist notwendig um sicherzugehen, dass alle relevanten Informationen wie Neue Meldung, Meldung verteilt, Ortspfad, Aktionsplan, Bediener, Zustandsänderung,... zu speichern.

Dieses vermehrte Eintragen von Ereignissen ist besonders zu beachten, falls man zählen will, wie oft ein Melder ausgelöst hat. Es ist nun wichtig, dass pro Auslösung wirklich nur ein Ereignis gezählt wird. Die Lösung dieser Aufgabe besteht darin, die eventTypeID mit dem dazugehörigen eventTypeName in LogEventType zu betrachten. Nun zählt man lediglich die Ereignisse, welche mit dem eventTypeName ‚state change' verknüpft sind. Dies ist pro Feueralarm nur ein einziges Ereignis.

ID	Fortlaufende Nummer, beginnt bei 1 mit dem ältestes Ereignis

eventCreationTime	Zeit des Erstellens des Events
addressID	Verknüpfung zu Logaddress. Gibt die Adresse an.
eventTypeID	Verknüpfung zu LogEventType. Gibt an ob Ereignis vom Type ‚state change', ‚control' oder ‚message' ist
stateID	Verknüpfung zu LogState. Gibt den Zustand des Melders an: Ruhe, Feuer,...

LogState

In dieser Tabelle sind alle möglichen Zustände der Melder eingetragen.
Jeder Zustand besitzt eine zugehörige stateNumber. Diese stateNumber ist normalerweise bei jeder BIS gleich, ist aber parametrierbar. So besitzt Ruhe standartmäßig die Nummer 5 und Einbruch die Nummer 21. StateNumber ist nicht mit der stateID zu vergleichen!
Die stateID ist durch das fortlaufende Eintragen der Zustände in LogState entstanden und ist bei jedem Logbuch unterschiedlich.

Wichtig! Wie prüfe ich auf den Status:

Dazu prüft man zuerst auf den Namen und nimmt, falls dieser nicht vorhanden ist, die stateNumber. Dieses Vorgehen entsteht dadurch, dass der Name von BIS zu BIS häufig abgeändert wird. Die Nummer dagegen bleibt meistens gleich. Der Quellcode zu dieser Abfrage ist weiter unten unter den BIS spezifischen Abfragen zu finden.

LogEventValue

eventTypeID	Verknüpfung zu LogEventValueType.ID
stringValue	Beinhaltet sehr viele Informationen. Dazu gehören Ortspfad, Aktionsplan, Bearbeitungszustand, durch welches Kommando der Melder ausgelöst wurde, etc.

LogEvent2Value

Diese Tabelle ist eine Verknüpfungstabelle zwischen LogEvent und LogEventValue.

eventID	Verknüpfung zu LogEvent.ID. NICHT zu LogEvent.eventTypeID
valueID	Verknüpfung zu LogEventValue.ID

LogEventValueType

eventValueName	Gibt an von welchem Typ die zugehörige Information in LogEventValue.stringValue ist. So ist hier zum Beispiel vermerkt, dass ‚BIS.Technikhaus' einem Ortspfad entspricht. Beim Hinzufügen von Ortspfaden zur Tabellenausgabe ist diese Information besonders relevant. Das Vorgehen ist dabei, dass man das Ereignis zuerst mit dem LogEventValue und LogEventValueType verknüpft. Anschließend lässt man alle stringValues ausgeben, bei denen der zugehörige eventValueName = ‚Ortspfad' ist. Auch diese Abfrage ist unter den BIS spezifischen SQL Abfragen zu finden.

LogAddress

Diese Tabelle speichert die Adressinformation eines Ereignisses. So weiß man bei jedem Ereignis genau welcher Melder der Auslösende gewesen ist.

ID	Verknüpfung zu LogEvent.addressID
AddressTag	Gibt die Adresse als Text an. Bsp.: ‚virtuell.B.Rauchmelder_201‘
detectorTypeID	Verknüfung zu LogDetectortype. Gibt Art des Melders an.

LogDetectorType

Diese Tabelle gibt die Art des Melders und die dazugehörige OPC Serverklasse aus.

ID	Verknüpfung zu LogAddress.detectorTypeID
detectorTypeName	Gibt die Art des Melders an. Z.B. ‚optischer Rauchmelder‘
opcServerClass	Gibt die OPC Serverklasse an. Z.B. ‚Virtuell‘

LogEventType

Dieser Tabelleninhalt ist wie zuvor erwähnt, beim Zählen von Zustandswechseln sehr hilfreich. Dabei muss überprüft werden, ob der eventTypeName des Ereignisses dem Typ ‚state change‘ entspricht.

ID	Verknüpfung zu LogEvent.eventTypeId
eventTypeName	Gibt Typ des Ereignisses an. Z.B. ‚state change‘, ‚message‘, ...

BISReports

Diese Datenbank wird erst relevant, sobald Sie den Bericht für den Microsoft Report Builder zugreifbar machen. Mit anderen Worten, wenn die Prozedur fertig gestellt ist und sie nun grafisch dargestellt werden soll.
Sobald Sie soweit sind, müssen Sie eine zusätzliche Prozedur in BISReports erstellen, die auf die bereits Erstellte zugreift und diese ausführt. Außerdem müssen Sie noch die Berechtigungen auf die Datenbank anpassen. Weiteres dazu finden Sie unter dem Punkt „Anpassung der Prozedur zur Verwendung mit dem Report Builder“.

4.3 Einführung in das Prinzip von Transact-SQL

Besonderheiten von Transact-SQL

Das Ziel von fast jedem Transact-SQL-Skript, ist es, eine Tabelle anzuzeigen, in der Informationen aus mehreren größeren Tabellen zusammengefasst und sinnvoll verknüpft werden.
Die Programmiersprache Transact-SQL erfordert dazu eine verschachtelte Denkweise, die sich immens von der des klassischen prozeduralen Programmierstils unterscheidet.
Diese ist auf den Kernbefehl SELECT ausgelegt. Man sucht und verknüpft Tabellen, verkleinert diese und stellt so lange Bedingungen, bis das gewünschte Ergebnis angezeigt wird. Dies geschieht alles zusammengefasst unter **einem** SELECT Befehl.
Wenn man sich in das effiziente Programmieren mit SELECT eingearbeitet hat, kann man komplexe Berichte mit wenigen Zeilen Code und wenigen Millisekunden Laufzeit erstellen.
Um dieses Ergebnis zu erreichen, sind einige Kenntnisse und Kniffe von Nöten. Auf den folgenden Seiten, werden sie deshalb die wichtigsten Funktionen und Befehle in SQL kennenlernen und verstehen können.

Weitere Informationen

Dieses Dokument kann keineswegs alle Möglichkeiten von Transact-SQL abdecken. ´

Wenn Sie Fragen zu einem unten erläuterten Befehl haben, tippen Sie den Befehlsnamen gefolgt von **Transact-SQL** in eine Internet-Suchmaschine ein. Die hilfreichsten Ergebnisse befinden sich fast immer unter dem msdn Homepage.

4.4 Wichtige SQL Befehle mit Fragestellungen und Lösungen

Wenn Sie nun auf den Geschmack des SELECT Befehls gekommen sind, können Sie sich über die nächsten Seiten freuen. Denn hier werden Sie lernen, wie Sie die ungeahnten Möglichkeiten des SELECTs ausschöpfen können.
Dazu stelle ich Ihnen die wichtigsten SQL Befehle jeweils in Kombination mit einer Fragestellung und einem Beispiel aus der BIS vor. Die Reihenfolge der Befehle resultiert aus dem für sie vorgesehenen Ort innerhalb des Select Statements.

4.4.1 Aufbau des SELECT-Befehls

Sie wissen nun, was Sie mit dem Select erreichen können. Um dies auch umzusetzen, führt Sie dieses Kapitel zunächst in den Aufbau des **SELECTS** ein.
Die Aufgabe eines Selects ist es Informationen von einer oder mehreren Tabellen auszuwählen und diese anschließend anzuzeigen. Dabei ist eine relativ verzweigte Struktur zu erkennen. Diese folgt jedoch stets einer feststehenden Reihenfolge, welche im Folgenden anhand eines Beispiels kurz erläutert wird.

```
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
```

- Das Select beginnt logischerweise mit dem Wort **SELECT**.
  Alle darauf folgenden Befehle sind optional und dienen somit zur Verfeinerung.
  Auf den Hauptbefehl folgt der Ausdruck **TOP 10**, welcher bewirkt, dass nur die zehn obersten Zeilen ausgegeben werden.
  Darauf folgen die auszuwählenden Spalten, welche mit einem Punkt Ihren ursprünglichen Tabellen zugeordnet werden. Alle Spalten wählen Sie mit einem * aus.
  Mit Hilfe des **AS** setzen Sie einen neuen Namen für die jeweilige Spalte fest.
  Durch die Funktion **MAX()** wählen nur das größte Element der Spalte aus. Näheres dazu folgt gleich.
- Die auszulesende Kerntabelle folgt auf das **FROM** und weitere Tabellen auf das **INNER JOIN**.
- Hinter dem **ON** folgt die Bedingung wie die weiteren Tabellen mit der Kerntabelle verknüpft sind.
- Darauf folgend ist das **WHERE** zu platzieren. Es dient dazu Bedingungen zu stellen, welche sich auf alle Tabellen, Spalten oder sonstiges beziehen. Hierzu können alle möglichen Vergleichsoperator wie **>, <, <>** (ungleich), **=, LIKE, BETWEEN, IS NULL**, ... verwendet werden.
- Ist dies erledigt, können die erreichten Ergebnisse mit **GROUP BY** gruppiert werden. So kann zum Beispiel nach der ersten Spalte gruppiert werden. Dies führt dazu, dass kein Eintrag der ersten Spalte doppelt vorkommt. Dabei werden zwangsweise auch Einträge der anderen Spalten aussortiert.

– Um hier zu wissen, was ausgewählt wird und was nicht, ist eine Aggregatfunktion vor dem zweiten Spaltennamen nötig. Diese kann z.B. **MAX(), MIN(), SUM(), AVG(), COUNT()**, etc. sein.
– Zu allerletzt, kann die Tabellenausgabe mit **ORDER BY ASC** / **DESC** noch auf- oder absteigend sortiert werden.

4.4.2 Hauptbefehle im Select-Statement

Es folgen nun die wichtigsten Befehle, welche zum Select Statement gehören. Diese sind jeweils mit einer Fragestellung, einer Erklärung, einem Beispiel aus der BIS und deren Ausgabe erklärt.
Die Beispiele können per Copy und Paste ins Management Studio übernommen werden und auf die Datenbank BISEventLog angewendet werden. Dazu muss eventuell die Datenbank von master auf BISEventLog geändert werden und die Anführungszeichen durch andere ersetzt werden.
Ich wünsche Ihnen viel Erfolg beim Kennenlernen und Verstehen der Hauptbefehle im Select-Statement.

TOP 10 -> Wie lass ich mich nur die obersten Ergebnisse ausgeben?
Man lässt sich durch den Befehl TOP N nur die obersten N Zeilen ausgeben. N steht dabei für die Anzahl der Zeilen, die man ausgegeben haben will. Es ist auch möglich sich nur die obersten N Prozent ausgeben zu lassen. Dazu schreibt man hinter die Zahl PERCENT.
Beispiel: Gibt die letzten vier Events aus:

```
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 -> Wie gebe ich meinen Spalten einen (neuen) Namen?
Man ordnet einer Spalte mit dem Befehl **AS** einen Namen zu.
Beispiel:

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

Ausdruck der Überraschung
Heidenei
Huch
```

JOIN -> Wie kann ich Daten aus mehreren Tabellen auslesen?

Um Daten aus mehr als einer Tabelle in eine zusammenzufassen, benutzt man den **JOIN** Befehl. Über das Select kann man sich, nach dem Einbinden, die Spalten aller eingebundenen Tabellen anzeigen lassen.
Beispiel:
Zeigt alle Events verknüpft mit dem Zustandsnamen aus LogState an.

```
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 -> Wie stelle ich Bedingungen?
Bedingungen werden mit dem Befehl **WHERE** gestellt.
Weitere Bedingungen werden mit AND oder OR hinzugefügt. Dabei wird zuerst das AND und anschließend das OR verarbeitet.
Bsp.: ...AND...OR...AND... entspricht (...AND...)OR(...AND...)
Soll das OR zuerst bearbeitet werden, müssen die Klammern entsprechend gesetzt werden: ... AND(...OR...)AND...
Beispiel:
Wählt alle Events aus, die sich weder auf Feuer, noch auf Ruhe beziehen.

```
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 -> Wie filtere ich redundante Zeilen, so dass nur eine angezeigt wird?
Mit dem Befehl **DISTINCT** können alle komplett doppelte Einträge aus einem SELECT aussortiert werden. Das heißt, dass somit alle Zeilen aussortiert werden, die in jeder Spalte mit einer anderen übereinstimmen.
Beispiel:

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

GROUP BY -> Wie lass ich mir zu einer Spalte jeweils das größte/kleinste Ereignis einer anderen ausgeben?

Der Befehl **GROUP BY** dient dazu in einer Spalte alle redundanten Einträge auszusortieren und die übrigen Einträge mit dem passenden Wert einer anderen Spalte zu kombinieren.
So lässt sich mit diesen Befehl beispielsweise herausfinden, welcher Melder, welchen Zustand **als letztes** besaß.

Dabei ist es wichtig, dass alle Spalten entweder hinter dem **GROUP BY** oder in einer Aggregatfunktion vorkommen. Ansonsten ist es SQL nicht möglich das zu der übrig gebliebenen Zeile passende Element zu finden und würde somit einen Fehler melden. l:
Wichtige Aggregatfunktionen: **AVG()** -> Average, **COUNT(), MAX(), MIN(), SUM()**
Beispiel:
Gibt aus, wie viele Events mit dem jeweiligen Zustand zusammenhängen. Dazu wird das jeweilige Datum ausgegeben, an dem der Zustand das **letzte Mal** eingetragen wurde.

```
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 -> Wie kann ich meine Ausgaben sortieren?
Sortiert wird in SQL mit dem Befehl **ORDER BY**. Nach order by folgt die Spalte nach der sortiert wird und dann eine Angabe ob ansteigend(ASC) oder abfallend (DESC) sortiert werden soll.
Beispiel:
Sortiert die Tabelle LogEvent absteigend nach der Zeit.

```
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 -> Wie kann ich meine Tabelle vertikal um Zeilen ergänzen, die außer dem Spaltennamen nichts mit der Tabelle zu tun haben?
Mit dem Befehl **UNION** können mehrere Tabellen vertikal miteinander verknüpft werden. Diese Funktion kann Ihnen später im Report Builder sehr hilfreich werden.
Dies geschieht, indem Sie Ihnen bei einem Auswahlfenster eines variablen Parameters

zusätzliche Möglichkeiten zum Hinzufügen bietet. Der Hauptteil der zur Wahl stehenden Werte werden in einer zuvor definierten Prozedur festgelegt. Mit UNION können Sie nun der eigentlich determinierten Ausgabe der Prozedur einen beliebigen Wert hinzufügen.
Wichtig: Die Spaltennamen der Tabellen müssen übereinstimmen!
Beispiel:
Das **UNION** fügt den Ortspfaden der BIS die Auswahl ‚Alle' hinzu.

```
SELECT stringValue AS Ortspfad
FROM LogEventValue
INNER JOIN LogEventValueType ON LogEventValueType.ID =
LogEventValue.eventTypeId
WHERE eventValueName = 'Ortspfad'
UNION
SELECT 'Alle' AS Ortspfad
```

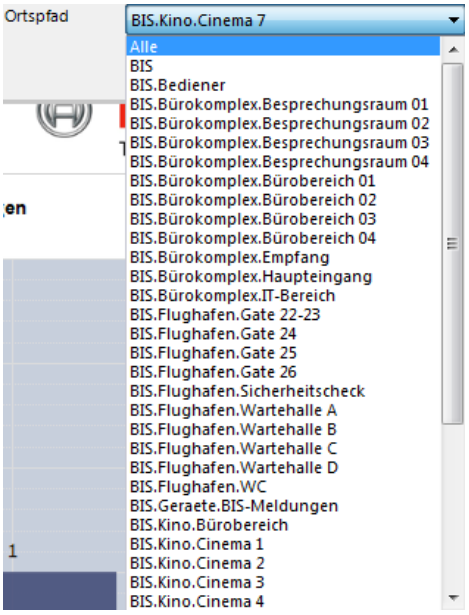


Bild 4.6: Abb. 10 Auswahlfenster Ortspfad

4.4.3 Allgemeine SQL Befehle

IIF() -> Wie erstelle ich eine IF-Abfrage innerhalb eines selects?
Um eine wenn...dann... Abfrage zu erstellen ist der Befehl **IIF()** sehr hilfreich.
Dieser hat eine spezielle Syntax. Sie besteht darin, dass drei Teile in die Klammer hinter IIF geschrieben werden.
Zuerst die Bedingung, dann die Ausgabe bei wahrer Bedingung und darauf folgend die bei falscher. Die Bedingung ist ein Vergleich oder sonstiges, welcher entweder true oder false ausgibt.
IIF(Bedingung, Folge Bedingung wahr, Folge Bedingung falsch)
Beispiel:
Prüft ob die Variable @Ortspfad = ‚Alle' entspricht.
Falls ja, werden alle Events mit den dazugehörigen Ortspfaden ausgewählt.
Falls nicht, werden alle Events ausgewählt, bei denen der Ortspfad mit @Ortspfad anfängt.

```
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 -> Wie kann ich Texte vergleichen?
Texte können genauso wie zahlen mit dem Gleichheitssymbol **‚=‘** verglichen werden. Zusätzlich dazu kann man auch prüfen, ob Teile des Textes übereinstimmen. Dies geschieht mit dem Befehl **LIKE**. Das LIKE vergleicht einen Text mit einem Text kombiniert mit Platzhaltern ‚%‘.
Beispiel: Prüft ob der Ortspfad mit ‚BIS‘ beginnt, und ‚Kino‘ beinhaltet.

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

IS NULL -> Wie überprüfe ich ob eine Variable den Wert NULL besitzt?
Hierzu dient die Vergleichsfunktion IS / IS NOT NULL.
Beispiel:
Gibt nichts aus, da die Variable @notNull einen Wert 0 besitzt und somit nicht null ist.

```
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 -> Wie kann ich einen Bereich zwischen zwei Elementen anzeigen?
Der Befehl **BETWEEN** ist hierzu von Nöten.
Syntax: Ausdruck [**NOT**] **BETWEEN** Anfang AND Ende
Beispiel:
Wählt alle Elemente der LogEvent Tabelle aus, bei der die ID zwischen 4 und 8 liegt.

```
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() -> Wie kann ich den Datentyp einer Variable verändern?
Um einen Datentyp in einen anderen zu konvertieren, kann man in SQL entweder **CONVERT** oder **CAST** verwenden. Beide erledigen einen gleichwertigen Job, haben aber eine abweichende Syntax.
Beispiel: Ändert den Datentyp von nvarchar nach integer:
CONVERT (Zieldatentyp, Variable)

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

CAST (Variable AS Zieldatentyp)

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

DATEADD() -> Wie kann ich zeitliche Daten addieren?
Mit **DATEADD**() werden zu einem Daten eine gewisse Anzahl an Zeiteinheiten (minute, hour, day,...) dazu addiert/subtrahiert.
Syntax: DateAdd (Art der Zeiteinheit, Anzahl der Zeiteinheiten, Datum)
Beispiel folgt bei Getdate().

GETDATE() -> Wie finde ich den jetzigen Zeitpunkt heraus?
Der aktuelle Zeitpunkt wird mit GetDate herausgefunden und als smalldatetime gespeichert.
Syntax von smalldatetime: YYYYMMDD hh:mm:ss
Beispiel:
Gibt die Zeit von vor zwei Stunden aus:

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

CREATE PROC –> Wie erstelle ich eine neue Prozedur?
Eine Prozedur wird mit dem Befehlt **CREATE PROC** erstellt:
Beispiel:
Erstellt die Prozedur EventsMitZustaenden, die alle Events ausgibt, welche den übergebenen Zustandsnamen besitzen.

```
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 -> Wie kann ich eine Prozedur ausführen?

Eine Prozedur wird mit **EXEC** ausgeführt. Hinter dem Prozedurnamen folgen die zu übergebenden Parameter:

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

ALTER PROC -> Wie kann ich eine Prozedur verändern?

Eine Prozedur wird mit dem Befehlt **ALTER PROC** modifiziert:
Beispiel:
Überschreibt die vorhandene Prozedur EventsMitZustaenden.
Sie gibt nun alle Events aus, welche den übergebenen Zustandsnamen nicht besitzen.

```
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 Erläuterte Beispielabfragen aus der BIS

Nachdem Sie nun die wichtigsten Befehle für SQL Abfragen beherrschen, werde ich noch einige Beispiele speziell aus der BIS behandeln.
Viele Teilbereiche der folgenden Quelltexte sind die vorigen Copy und Paste Vorlagen und können 1 zu 1 in Ihre Prozedur aufgenommen werden. So sind zum Beispiel die INNER JOINs beim Hinzufügen des Ortspfades stets die Gleichen.
Zum Schluss folgen noch ein paar kommentierte Quellcodes von Berichten, die bereits fertig gestellt worden sind.

Wie füge ich den Ortspfad eines Melders zu meiner Ausgabetabelle hinzu?

Der individuelle Ortspfad ist unter LogEventValue.stringValue gespeichert.
Wichtig!
Unter stringValue sind nicht nur Ortspfade, sondern auch Bearbeitungszustände, Aktionspläne, etc. gespeichert. Zu welcher Klasse der jeweilige Eintrag unter stringValue gehört, ist unter LogEventValueType.eventvalueName abgespeichert.
Damit der ausgegebene Ortspfad auch wirklich dieser ist und kein Aktionsplan, ist also die Abfrage, ob der eventValueName = ‚Ortspfad' ist essentiell.
Bei der Verknüpfung der Tabellen ist zu beachten, dass LogEvent mit LogEvent2Value über die LogEvent.ID verknüpft ist und NICHT über 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
...
```

Wie füge ich die Adresse eines Melders zu meiner Ausgabetabelle hinzu?

```
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
...
```

Wie füge ich die Art des Melders zu meiner Ausgabetabelle hinzu?

Die Art des Melders ist in LogDetectorType.detectorTypeName gespeichert. Diese ist über LogAddress mit LogEvent verknüpft.

```
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
...
```

Wie finde ich die stateID eines Zustands heraus?

Das Thema mit der stateID ist tückisch. Dies kommt daher, dass die BIS keine standardisierten Namen für die Zustände besitzt. Es kommt somit des Öfteren vor, dass ein Zustandsname leicht abgeändert oder ins Englische übersetzt wird.
Daher ist stateNumber, welche ebenfalls in der BIS parametriert werden kann, zum Überprüfen eines Zustands meistens besser geeignet als der Name. Dies liegt daran, dass sie sich im Vergleich zum stateNamen fast nie ändert. So besitzt Ruhe standardmäßig die Nr. 5 und Einbruch die Nr. 21.
Zusätzlich dazu ist es möglich, dass der zu prüfende Zustand noch nie vorgekommen ist. Dies bedeutet, dass er auch im Logbuch nicht zu finden ist. Deshalb ist es wichtig die standardmäßige stateNumber des Zustands zu kennen und sie im Notfall auszugeben.
Um der Veränderung eines der beiden Attribute vorzubeugen hilft eine IF Abfrage, welche sicherstellt, dass auch wenn eines der beiden Attribute verändert wird, immer noch die passende stateID herausgegeben wird:

```
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 */
```

Wie finde ich heraus, wie oft ein Melder in den letzten Tagen in einen Zustand gewechselt ist?

Zuerst gilt es alle Zustandswechsel in den letzten Tagen in den gewählten Zustand (Bsp.: ASP-Sperre) aus dem LogEvent herauszufiltern. Dabei merkt man sich von jedem Zustandswechsel die Adresse. Anschließend werden alle mehrfach vorkommenden Adressen gruppiert. Bei der Gruppierung wird gezählt, wie viele Adressen gruppiert wurden. Da jeder Zustandswechsel eine Adresse erzeugt hat, entspricht die Anzahl gruppierter Adressen, der Anzahl der Zustandswechsel.

```
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
```

Wie kann ich alle Melder ausgeben, die jetzt einen bestimmten Zustand (Z.B.: ASP Sperre) besitzen?

Das Prinzip zur Lösung dieses Problems ist es, eine Zwischentabelle mit den letzten Zustandswechseln aller Melder, die NICHT in ASP Sperre waren, zu erstellen. Von diesen Zustandswechseln speichert man die Zeit.
Nun sucht man alle Zustandswechsel nach ASP Sperre, die NACH dieser Zeit geschehen sind. Von diesen wählt man die mit der frühesten Zeit aus, da ein Melder auch zweimal hintereinander ausgeschaltet werden kann, ohne dass er dazwischen eingeschaltet wurde. Ist dies der Fall, ist er seit dem ersten Ausschalten außer Betrieb. Also wird die früheste Zeit nach dem letzten Zustandswechsel ausgewählt.
Beispiel:
Welche Melder sind einzuschalten, da sie innerhalb der letzten Schicht ausgeschaltet wurden?

```
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
```

Wie kann man ausgeben, welche Melder zu einem Zeitpunkt den gewünschten Zustand besitzen?

Diese Abfrage ähnelt sehr stark der oben beschriebenen Abfrage, die alle Melder ausgibt, die Im Moment abgeschaltet – also in ASP Sperre – sind.
Der Unterschied liegt darin, dass nun auf zwei x-beliebige Zustände geprüft werden kann und auch der Zeitpunkt anders festgelegt wird.
Durch den folgenden Quelltext wird eine Prozedur erstellt, die die Eingabe von zwei Parametern verlangt. Diese sind @stateNumber und @zeitpunkt.
Bei Ausführung der Prozedur werden nun alle Melder ausgegeben, die zum eingegebenen Zeitpunkt den eingegebenen Zustand besitzen.

```
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 Anpassen der Prozedur zur graphischen Verarbeitung

Da Ihre Prozedur nun die gewünschten Ergebnisse anzeigen kann, folgen ab jetzt die Schritte, die Sie unternehmen müssen um Ihre Ausgabe optisch ansprechend mit dem Report Builder darstellen zu können. Dazu folgt zunächst ein kleiner Schritt, den Sie im Management Studio ausführen müssen.

4.5.1 Erstellen der Prozedur in BISReports

Wie ich bereits weiter oben erwähnt habe, muss nach dem Abschluss der Prozedur in BISEventLog noch eine weitere Prozedur in BISReports erstellt werden.
Diese Prozedur ruft die Hauptprozedur auf, prüft auf einen möglichen Timeout und stellt sie dem Report Builder zur Verfügung.
Der Quellcode ist standardisiert und lautet bspw. für die Prozedur zur Ausgabe der einzuschaltenden Melder wie folgt:

```
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 Anpassen der Berechtigungen

Damit Sie später mit dem Berichtbearbeitungstool auf Ihre Prozedur zugreifen können ist ein Benutzer notwendig. Dieser Benutzer heißt **logbuch_query**. Standardmäßig ist es diesem Benutzer nicht gestattet auf die in BISReports erstellte Prozedur zuzugreifen. Folglich muss er manuell dazu berechtigt werden.
Führen Sie daher einmalig folgenden Quelltext aus.

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

Das Ergebnis ist unter Rechtsklick auf Ihre Prozedur -> Properties -> Permissions zu überprüfen. Dort müsste nun der Benutzer logbuch_query mit dem Recht zum Ausführen erstellt worden sein.

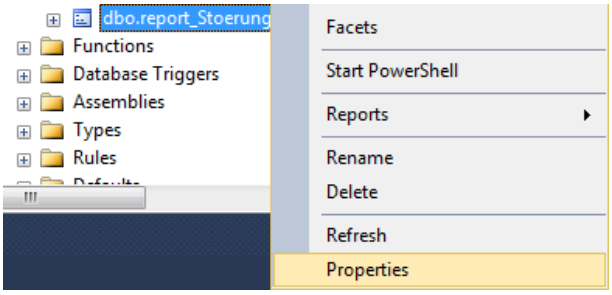


Bild 4.7: Abb. 11 Properties logbuch_query

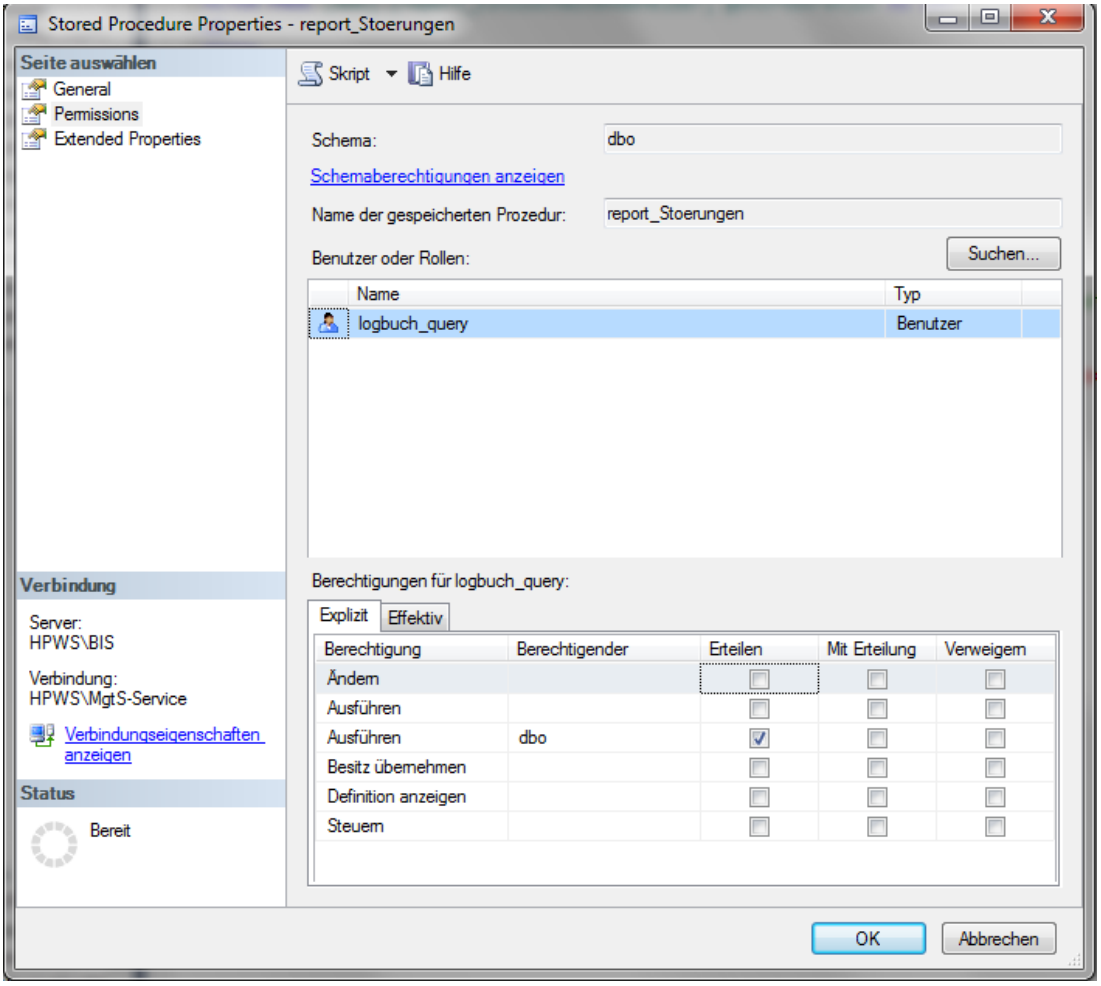


Bild 4.8: Abb. 12 Properties Fenster

5 Präsentation der Daten

Sie können sich nun alle gewünschten Informationen innerhalb von SQL mit Ihrer Prozedur ausgeben lassen. Damit diese Ergebnisse, über die BIS abgerufen werden können, arbeiten wir von nun an mit dem Microsoft Report Builder.
Dieses Programm ermöglicht es Ihnen Ihre Tabellenausgabe übersichtlich, professionell und optisch ansprechend über die BIS darstellen zu können.
Was genau Sie mit diesem Programm erwartet und wie man damit umzugehen hat, finden Sie in diesem Kapitel in einer mit Bildern verfeinerten Schritt für Schritt Anleitung.
Dabei lernen Sie, wie Sie den Bericht individuell gestalten und so in die BIS einbinden können, dass der Kunde schnell und einfach Ihren Bericht mit den aktuellen Daten aufrufen kann.
Am Ende dieses Kapitels ist Ihr Bericht fertig designt und über die BIS abrufbar.

5.1 Installation des Microsoft Report Builders

Zuerst müssen wir die Basis für das Erstellen eines Berichts erschaffen. Dies inkludiert die Installation des Microsoft Report Builders 2.0 oder 3.0.
Dazu müssen Sie zuallererst sicherstellen, dass der Microsoft Reporting Service installiert worden ist. Im Allgemeinen ist dies bereits während der Installation der BIS geschehen. Sie können dies durch das Aufrufen der Seite **http://localhost/reports_bis** oder durch das Suchen des Programmes überprüfen.
Es ist wichtig zu wissen, welche Version (2.0 oder 3.0) des Report Builders Sie verwenden, da nur diese mit anschließenden Funktionen kompatibel sind.
Sollten Sie zum Beispiel einen 2.0er Bericht aus Versehen mit dem Report Builder 3.0 öffnen, wird automatisch ein Backup mit dem Namen <reportName> - Backup.rdl erstellt. Dieses Backup beinhaltet die mit 2.0 erstellte Version des Berichts.
Führen Sie bitte nun die Setup Datei aus.
Diese ist auf dem BIS Installationsmedium unter folgendem Pfad zu finden:
BIS\3RD_party\SQL2008\ReportBuilder<Version>**\ReportBuilder.msi**
wo <Version> **2.0** oder **3.0** sein kann**.**
Sobald Sie die **.MSI** Datei gestartet haben, öffnet sich das Setup.
Während der Installation werden Sie dazu aufgefordert, den „**Default Target Server**“ zu bestimmen. Dieser entspricht einer SQL Server Instanz namens Reportserver. Dieser Reportserver wurde bei der automatischen Installation des Microsoft Reporting Service erstellt.
Sie sollten das Feld in folgender Form ausfüllen:
http://localhost/Reportserver_<SQL Server Instanzname>
oder für einen speziellen Rechner (Port ist optional):
http://<BIS Rechnername>:<Port>/Reportserver_<SQL Server Instanzname>

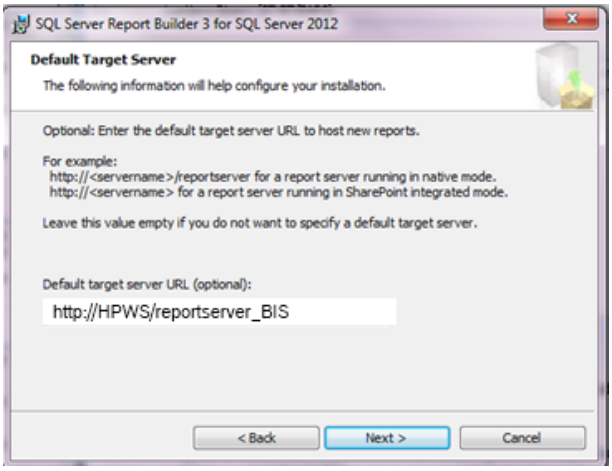


Bild 5.1: Abb. 13 Eingabe Default Target Server

Beim Klick auf Next sollte das Setup fertig gestellt werden und Sie den RB starten können. Er ist üblicherweise unter dem Pfad

C:\Program Files (x86)\Microsoft SQL Server\Report Builder <Version> **\MSReportBuilder.exe**

zu finden.

5.2 Erstellen und Einbinden Ihres Berichts

Sie haben den Report Builder installiert und sonstige Vorkehrungen getroffen?
Dann können Sie nun endlich damit anfangen Ihren eigenen Bericht anhand folgender Schritt-für-Schritt Anleitung zu erstellen. Zum Schluss sollten Sie einen nach Bosch designten, über die BIS abrufbaren Bericht fertig gestellt haben.

Schritt 1: Report Builder starten

Starten Sie den Report Builder.

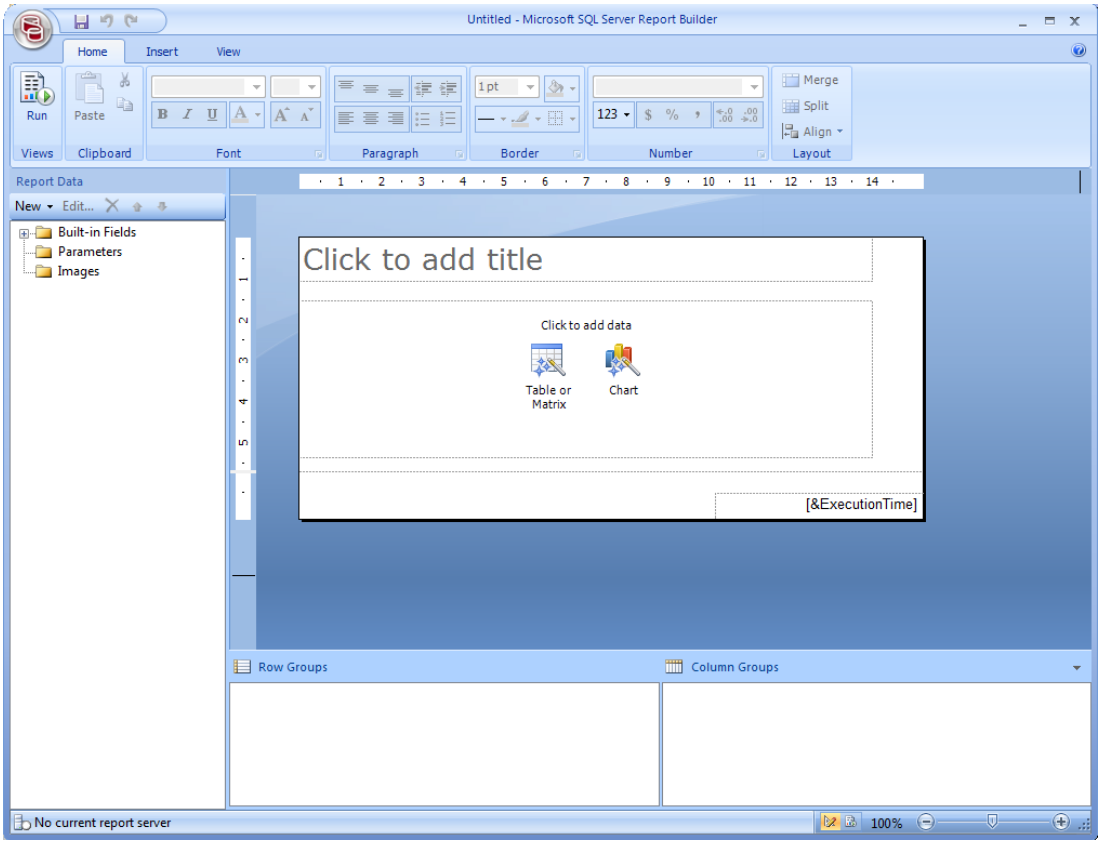


Bild 5.2: Abb. 14 Startansicht RB

Schritt 2: Berichttemplate öffnen

Klicken Sie auf den Button oben links in der Ecke. Wählen Sie Open aus und öffnen Sie das mitgelieferten Bosch Berichte Template. Dieses ist in den Dateien zum Berichte veröffentlichen unter folgendem Pfad zu finden:

BIS_Berichte\Berichte\BoschBerichtTemplate.rdl

Falls sich der RB nicht mit dem Reportserver verbinden kann, überprüfen Sie den bei der Installation festgesetzten Reportserverpfad. Dieser lässt sich unter dem **Report Builder Button -> Optionen-> „Use this report server or SharePoint site by default:“** einstellen.

Eventuell sind Sie nicht an dem BIS Rechner angemeldet und müssen daher localhost mit dem BIS Rechnernamen und dessen Port ersetzen.

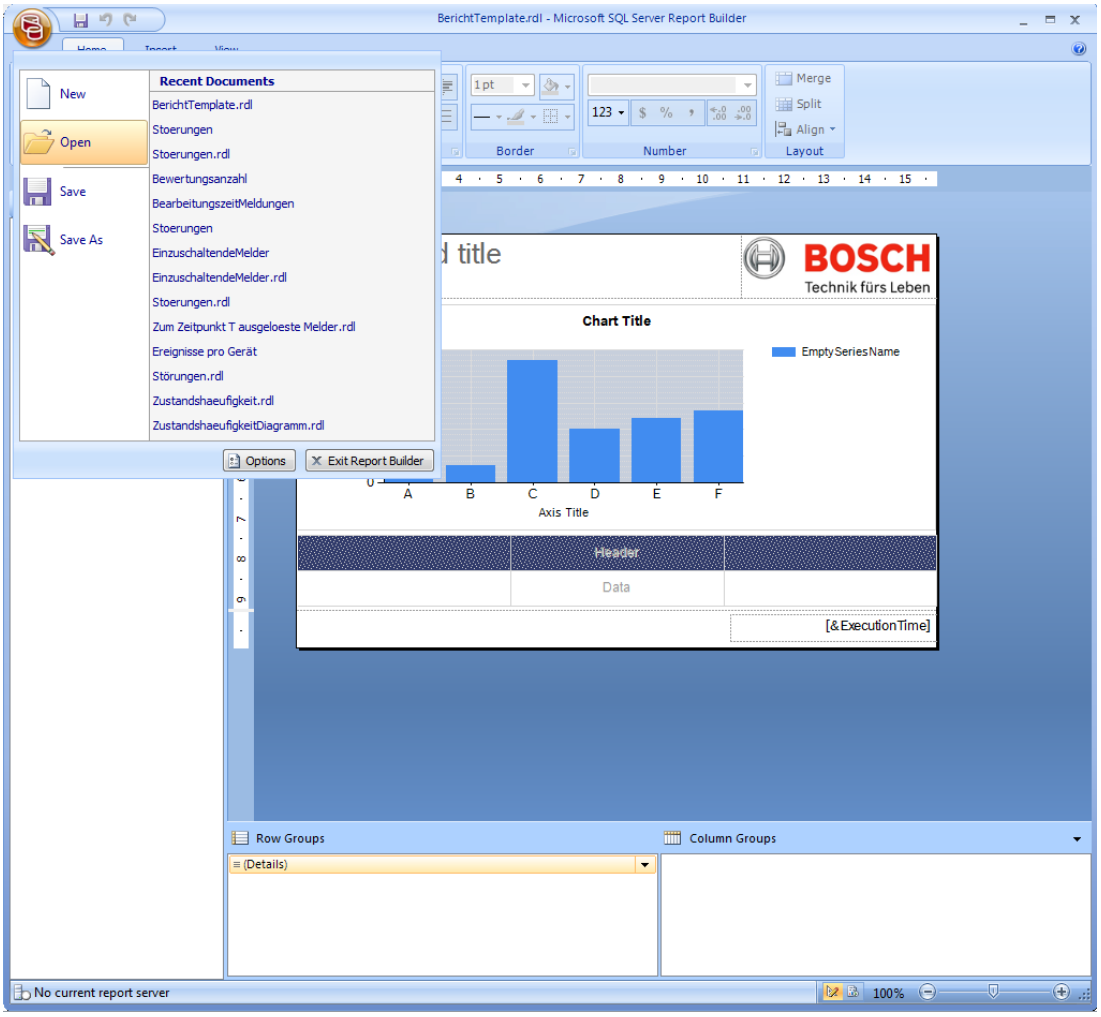


Bild 5.3: Abb. 15 BoschBerichtTemplate

Schritt 3: Dataset Fenster öffnen

Legen Sie als nächstes das zu verwendende Dataset fest. Klicken Sie dazu auf New -> **Dataset...** Der Dataset Properties Dialog öffnet sich.

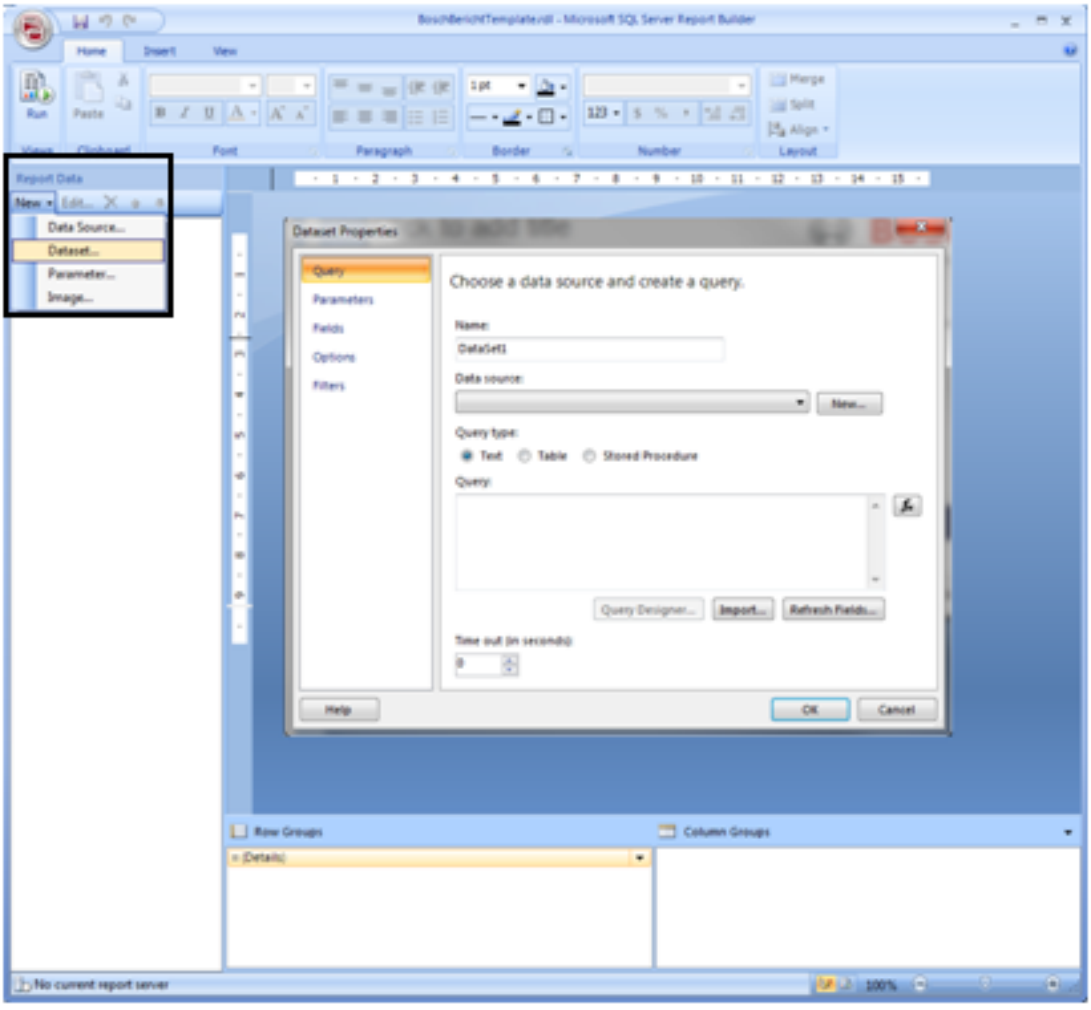


Bild 5.4: Abb. 16 Dataset bestimmen

Schritt 4: Data Source Credentials bestimmen
Klicken Sie auf **New...** um das Data Source Properties Fenster zu öffnen.
Um die Berechtigungen anzupassen, klicken Sie auf **Use a connection embedded in my report** und wählen Sie links den Reiter Credentials aus. Hier wählen Sie **Do not use credentials** aus. Diese Option wird öfters vom Report Builder zurückgesetzt. Das führt dazu, dass Sie sich beim Testen Ihres Berichts nicht mit dem Reportserver verbinden können. Falls Ihnen also beim Testen die Fehlermeldung „Failed to preview report“ angezeigt wird, überprüfen Sie diese Option.

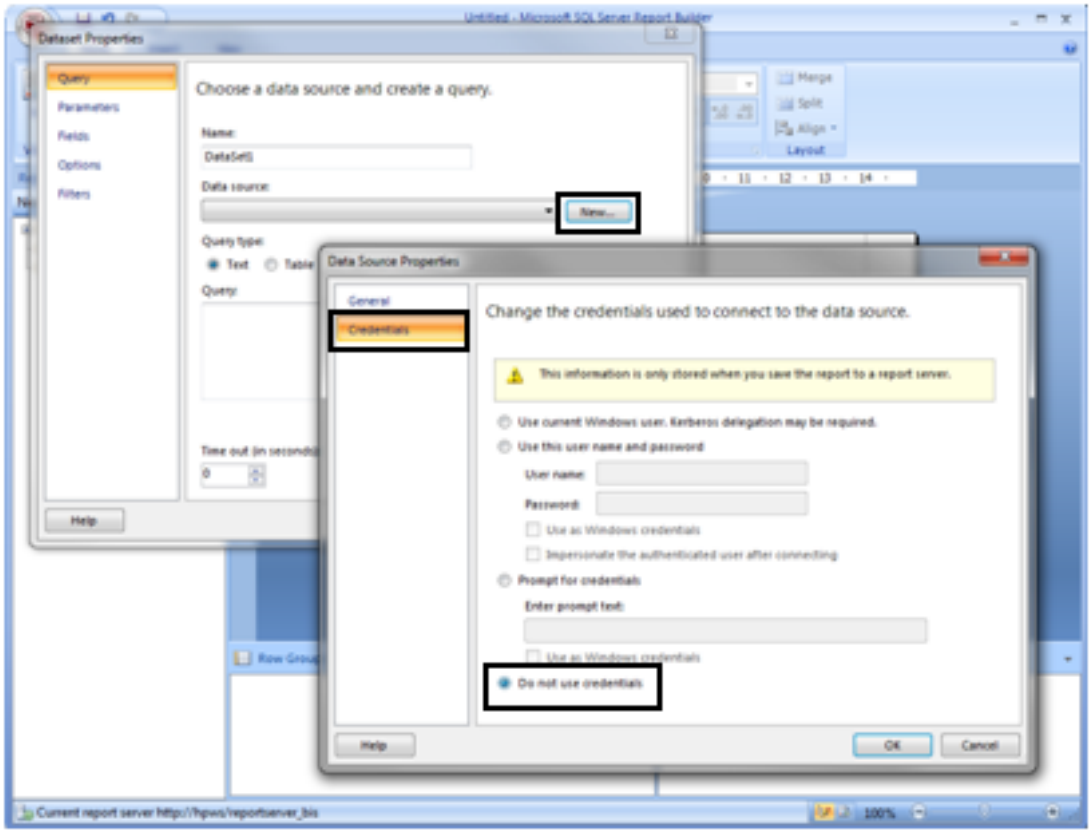


Bild 5.5: Abb. 17 Data Source Credentials

Schritt 5: Data Source auswählen
Gehen Sie wieder zurück auf den Reiter General und wählen Sie die Funktion **Use a shared connection or report model** aus. Klicken Sie auf **Browse...** Nun empfängt der Report Builder, die auf dem Reportserver erstellten Dateien und Verknüpfungen. Diese sollten die vorinstallierte Datenbank **EventLog Datasource** enthalten. Wählen Sie diese aus und geben Sie Ihrer Data Source z.B. den Namen BISReports. Meine Datasource heißt irrtümlicherweise BISEventLog. Klicken Sie nun auf OK um die Prozeduren einbinden zu können.

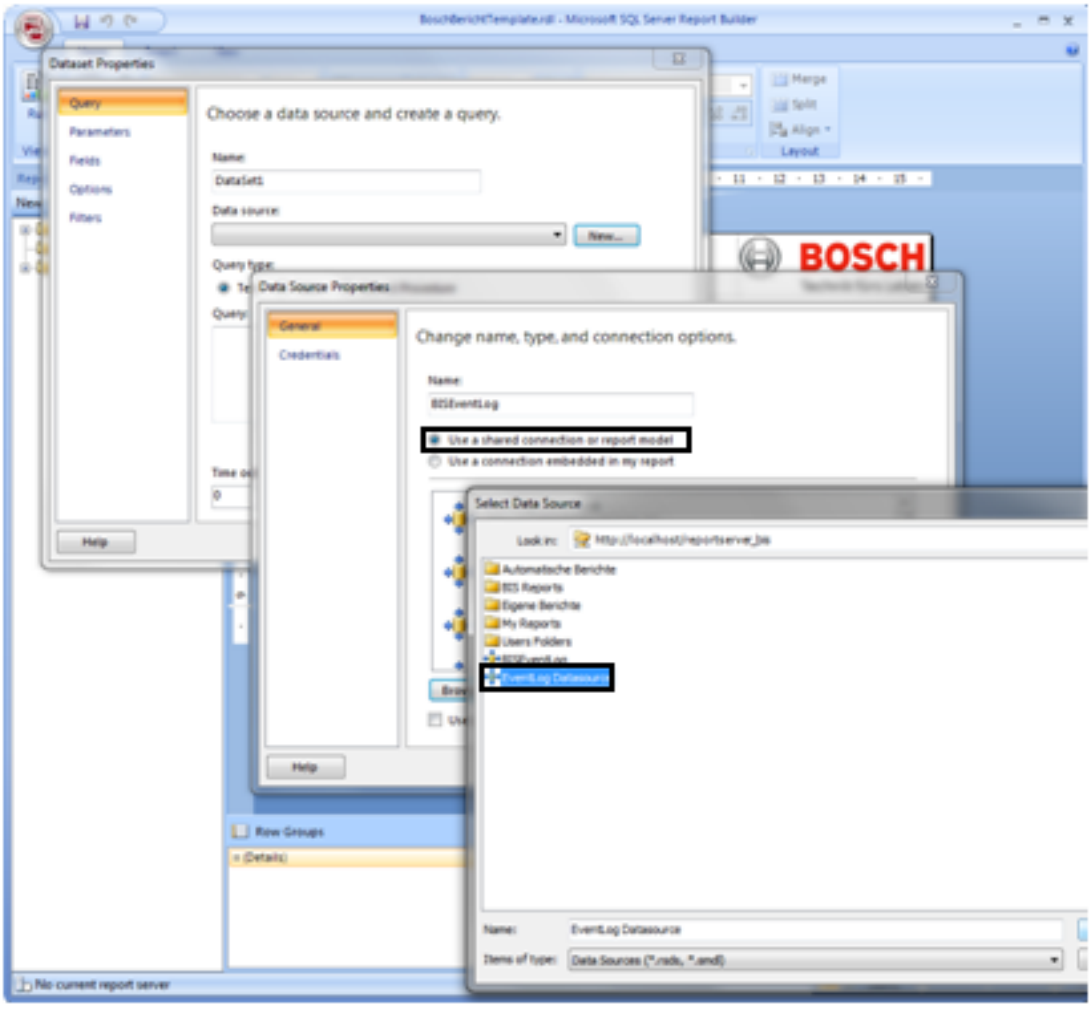


Bild 5.6: Abb. 18 Data Source bestimmen

Schritt 6: Dataset / Prozedur auswählen

Wählen Sie nun im Dataset Properties Fenster den Query type **Stored Procedure** aus, um Ihre zuvor in SQL erstellte Prozedur einzubinden.

Es erscheint ein Fenster, in dem Sie die Zugangsdaten für den Zugriff auf die EventLog Datasource eingeben müssen. Benutzen Sie hierzu folgende Benutzerdaten:

Benutzername: **logbuch_query**

Passwort: **pw_logbuch_query**

Wählen Sie anschließend die gewünschte SQL Prozedur aus der Datenbank **BISReports** aus. Ich wähle hier als Beispiel die Prozedur **report_Stoerungen** aus.

Wichtig ist, dass Sie den Namen **DataSet1** beibehalten, da sonst Tabelle und Säulendiagramm neu erstellt werden müssen.

Bestätigen Sie Ihre Eingaben mit **OK.**

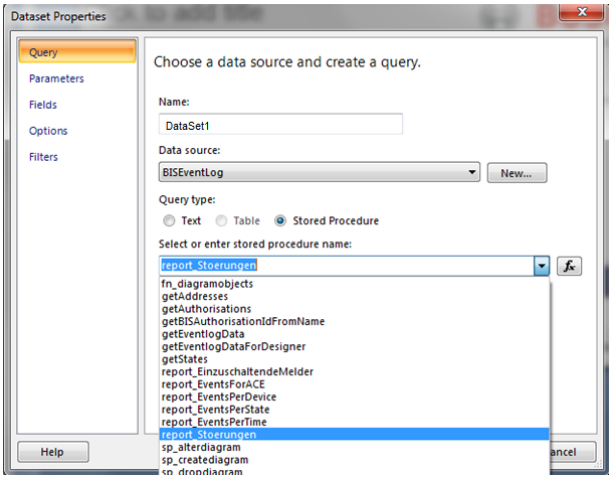


Bild 5.7: Abb. 19 Prozedur auswählen

Schritt 7: Parameter userPermission hinzufügen

Vorm Anzeigen eines Berichts, überprüft die BIS als Sicherheitsmaßnahme den Parameter userPermission. Daher müssen wir diesen Parameter bei jedem anzuzeigenden Bericht hinzufügen.

Klicken Sie dazu mit einem Rechtsklick auf den Reiter **Parameters** und wählen Sie **Add Parameter...** aus. Es erscheint das Report Parameter Properties Fenster.

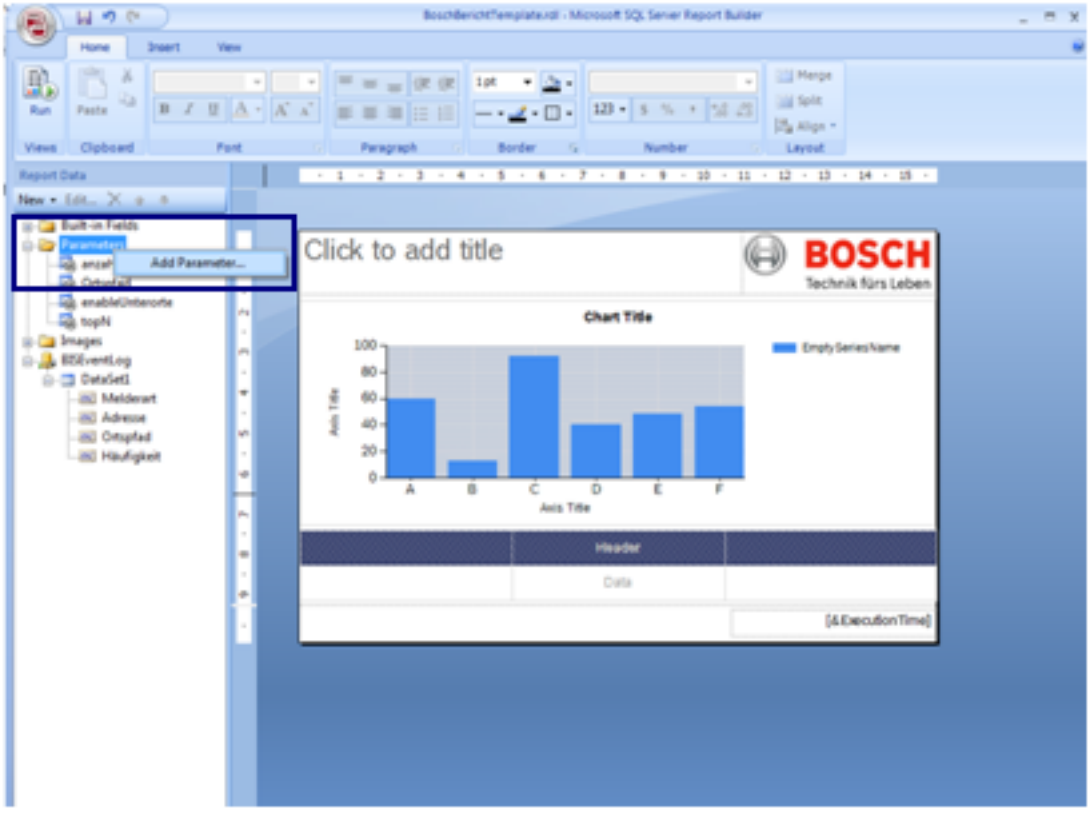


Bild 5.8: Abb. 20 Parameter hinzufügen

Schritt 8: Parameter userPermission deklarieren

Ändern Sie nun Name und Prompt in **userPermission** um.

Erlauben Sie außerdem die Eingaben von **blank** und **null values**. Zusätzlich dazu, sollten Sie die **visibility** auf **Hidden** stellen. Somit wird der Parameter nicht bei jedem Aufrufen eines Berichts angezeigt.

Erstellen Sie den Parameter mit dem Klick auf **OK.**

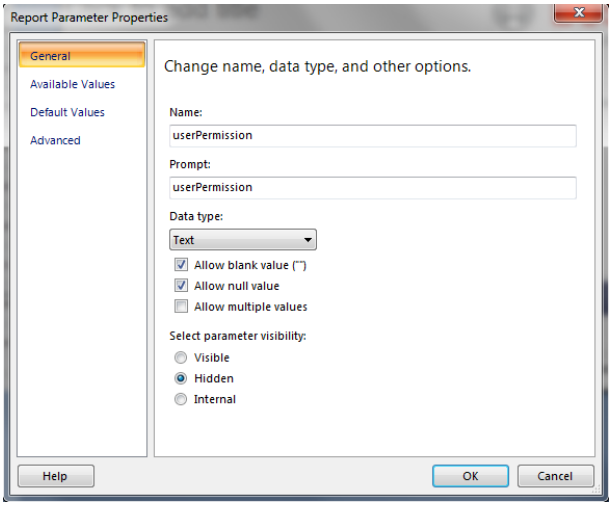


Bild 5.9: Abb. 21 userPermission

Schritt 9: Testen des Berichts

Da Sie nun eine Weile mit dem Designen des Berichts zu tun hatten, sind Sie sicher schon heiß darauf Ihren Bericht zu testen. Klicken Sie dazu wahlweise auf **F5** oder den Button **Run** oben links. Geben Sie nun die zu übergebenden Parameter ein und klicken Sie auf **View Report** um einen ausgefüllten Bericht zu sehen. Mit dem Drücken von **F8**, bzw. **Design** gelangen Sie zurück zum Editieren des Berichts.

Schritt 10: Gestalten des Berichts

Um den Bericht zu gestalten, passen Sie zunächst die Tabelle an, indem Sie die Spalten aus dem Dataset in die leeren Tabellenspalten ziehen.

Eventuell sind die Spalten unter dem Dataset nicht zu sehen, da Ihre Prozedur noch nie mit dem Report Builder aufgerufen wurde. Falls dies der Fall sein sollte, klicken Sie mit Rechtsklick auf Ihr DataSet1 und öffnen Sie den Query Designer. Klicken Sie in diesem auf das Ausrufezeichen und geben Sie Werte für Ihre Parameter ein, um den Bericht das erste Mal auszuführen.

Achtung!

Beim Einfügen der Spalten in die Tabelle, wird es bei mehr als 3 Spalten dazu führen, dass sich die Tabelle und somit die Seite vergrößert.

Achten Sie darauf, dass Sie die Seite wieder zurück in Ursprungsbreite bringen, damit der Bericht als übersichtliches PDF Dokument ausgegeben werden kann. Falls Sie den Bericht nicht auf die ursprüngliche Breite verkleinern, werden automatisch störende Seitenumbrüche erstellt, die das Lesen sehr erschweren.

Um weitere Elemente hinzuzufügen, klicken Sie mit Rechtsklick auf einen freien Platz im Bericht und wählen **Insert** --> ... aus.

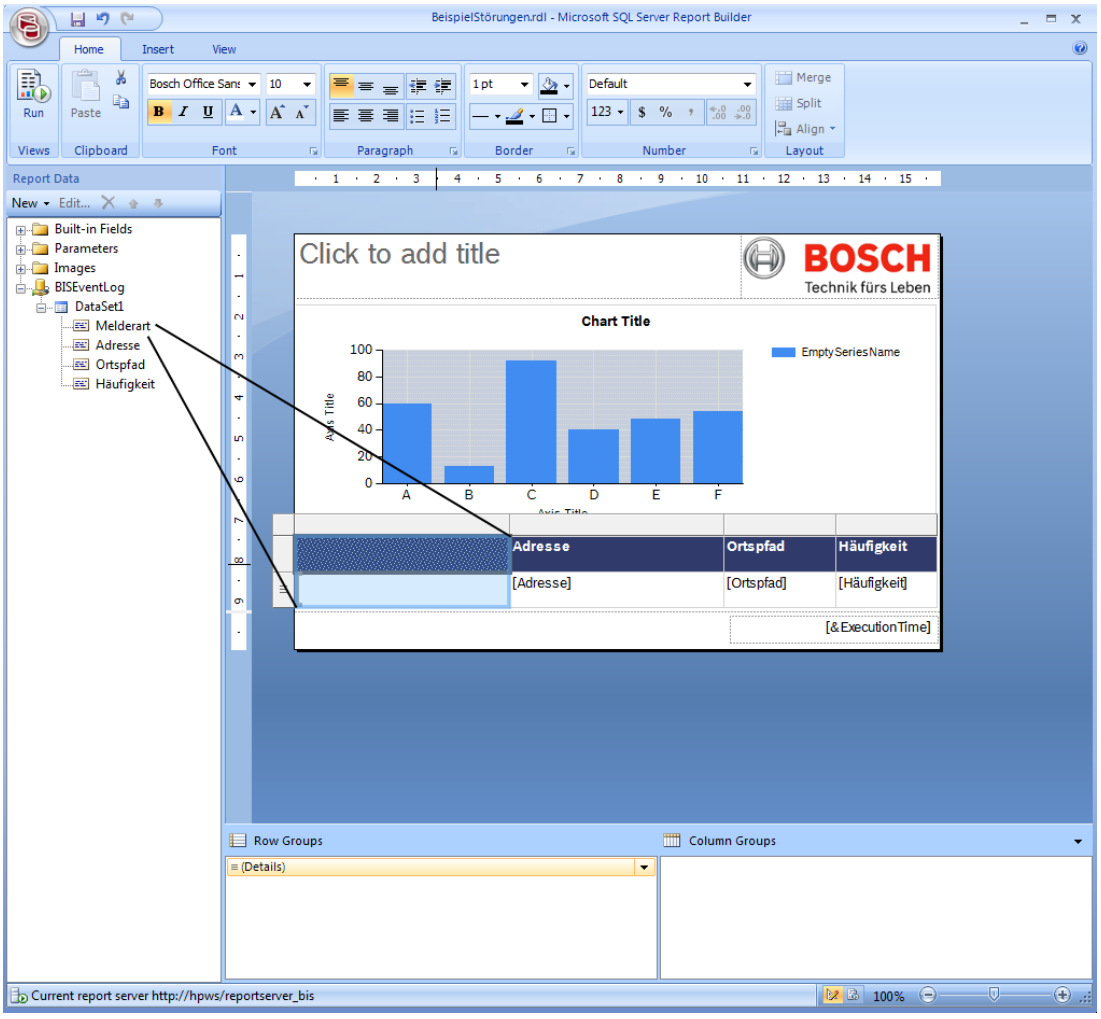


Bild 5.10: Abb. 22 Tabellenspalten einfügen

Bearbeitung des Säulendiagramms

Falls Sie zusätzlich zu der Tabelle noch das gegebene Säulendiagramm verwenden möchten, ziehen Sie nun die gewünschten Einträge in die dafür vorgesehenen Felder.

Passen Sie anschließend das Diagramm nach Ihren Wünschen an.

Im Beispiel habe ich die meisten Beschriftungen gelöscht.

Um die Höhen der Säulen direkt über Ihnen anzeigen zu können, klickt man mit Rechtsklick auf diese und wählt **Show Data Labels** aus.

Um ein einheitlichen Bosch Design zu erhalten, muss auch die Farbe der Säulen angepasst werden. Dazu müssen Sie zunächst mit Rechtsklick auf die Säulen klicken und das **Series Properties** Fenster öffnen. Wählen Sie nun den Reiter **Fill** aus und öffnen mit dem Klick auf den **Fx** Button das Color Expression Fenster.

Um das Ihrem Bericht ein Bosch Design zu geben, ändern Sie hier den Farbcode in **#c8313b6b** ab. Die Farbe entspricht einem halbtransparenten dunklen Bosch Blue.

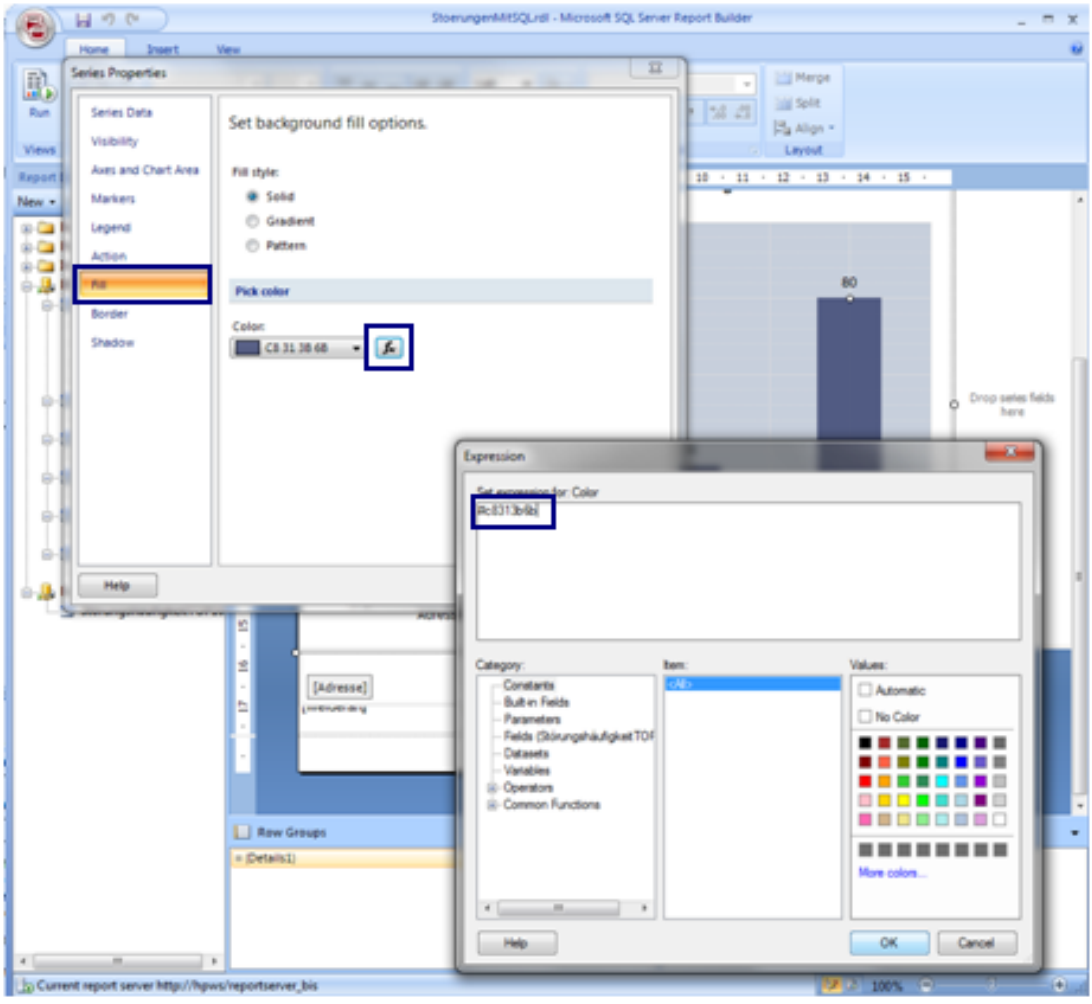


Bild 5.11: Abb. 23 Farbe ändern

Bosch Corporate Identity Guidelines

Der Beispielbericht richtet sich nach dem Bosch Corporate Identity Guidelines. Diese finden Sie im BGN unter folgendem Link:
https://inside-ws.bosch.com/FIRSTspiritWeb/permlink/wcms_rgap_-09_rbei_xxx_guidelines_and_standards_134-EN
Falls Sie Ihren Bericht auch einem Bosch Design erstellen möchten, achten Sie darauf, dass die Schriftarten in den gegebenen Elementen schon auf Bosch Office Sans gestellt worden sind. Falls Sie neue Elemente hinzufügen, achten Sie darauf die Schriftart dementsprechend anzupassen.
Verwenden Sie die Farben aus dem Bosch Style Guide, damit auch die neu erstellten Elemente nach Bosch aussehen. Folgende Farbcodes bieten einen kleinen Auszug:
Dunkles Bosch Blue: #313b6b
Helles Bosch Blue: #92a0b9
Sehr helles Bosch Blue: #c7cfdc

Alternierende Zeilenfarben erstellen

Die Tabellenelemente im BoschBerichtTemplate, welche den Inhalt der Spalten darstellen, sind von mir mit alternierenden Farben ausgestattet worden. Dies geschieht mit Hilfe des Befehls:
=IIF(RowNumber(Nothing) Mod 2, „#92a0b9“, „#c7cfdc“)
#... stellt dabei den jeweiligen Farbcode dar.

Standardwerte für Variablen festlegen

Nachdem Sie auf Run gedrückt hatten, haben Sie höchstwahrscheinlich auf eine leere Berichtsseite geblickt. Erst nachdem Sie die Parameter eingegeben und auf View Report gedrückt hatten, hatten Sie Zugriff auf die relevanten Ergebnisse.
Damit dies schneller von Statten geht, hat Report Builder die Funktion der **Default Values** für Parameter integriert. Damit lassen sich alle Parameter mit einem bestimmten Wert vorbelegen. Report Builder erstellt nun bereits beim Klicken auf Run einen standardmäßigen Bericht mit den gesetzten Default Werten.
Damit dies auch bei Ihnen der Fall sein wird, klicken Sie mit Rechtsklick auf einen Ihrer Parameter und öffnen Sie das **Parameter Properties** Fenster. Gehen Sie nun auf den Reiter **Default Values.** Nun können Sie feste Werte festlegen oder Ihren gewünschten Wert durch eine Prozedur herausfinden. Im Beispiel wähle ich für den Parameter anzahlTage den Standardwert 2.

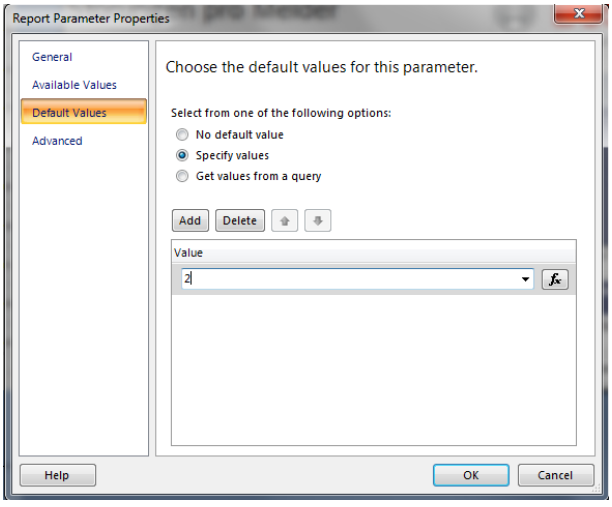


Bild 5.12: Abb. 24 Standartwerter von Variablen

Angezeigten Name der Parameter ändern
Damit der Bediener auch bei unklaren Parameternamen weiß, was er einzugeben hat, kann man den angezeigten Namens des Parameters ändern.
Dazu passen Sie den **Prompt** unter dem Reiter **General** im **Report Parameter Properties** Fenster an.

Auswahlfenster bei der Parametereingabe hinzufügen

Eventuell ist Ihnen aufgefallen, dass die Eingabe der Parameter ursprünglich nicht gerade benutzerfreundlich gestaltet ist. Um dies zu verändern, können Sie dem Bediener beispielsweise eine Auswahl von vorgegebenen Werten anbieten.
Dazu wählen Sie im **Report Parameter Properties** Fenster den Reiter **Available Values** aus. Im Beispiel erzeuge ich ein Auswahlfenster, in dem eine Auswahl aller Ortspfade angezeigt wird. Dazu wähle ich mit Hilfe des Punktes **Get Values from a query** mein/-e zuvor erstelltes Dataset/Prozedur getOrtspfade aus. Mit Hilfe des **Value** und des **Label fields** kann man nun

noch weitere Anpassungen vornehmen. Unterschiedliche Auswahlen trifft man hier zum Beispiel wenn man die StatusID als Wert und den StatusNamen als angezeigten Namen ausgeben will.
Der Quelltext der Prozedur getOrtspfade ist übrigens weiter vorne im Handbuch unter dem erklärten Befehl UNION zu finden. Dieser gibt nämlich alle in der BIS eingetragenen Ortspfade aus und fügt der Ausgabe die Auswahl ‚Alle' hinzu.

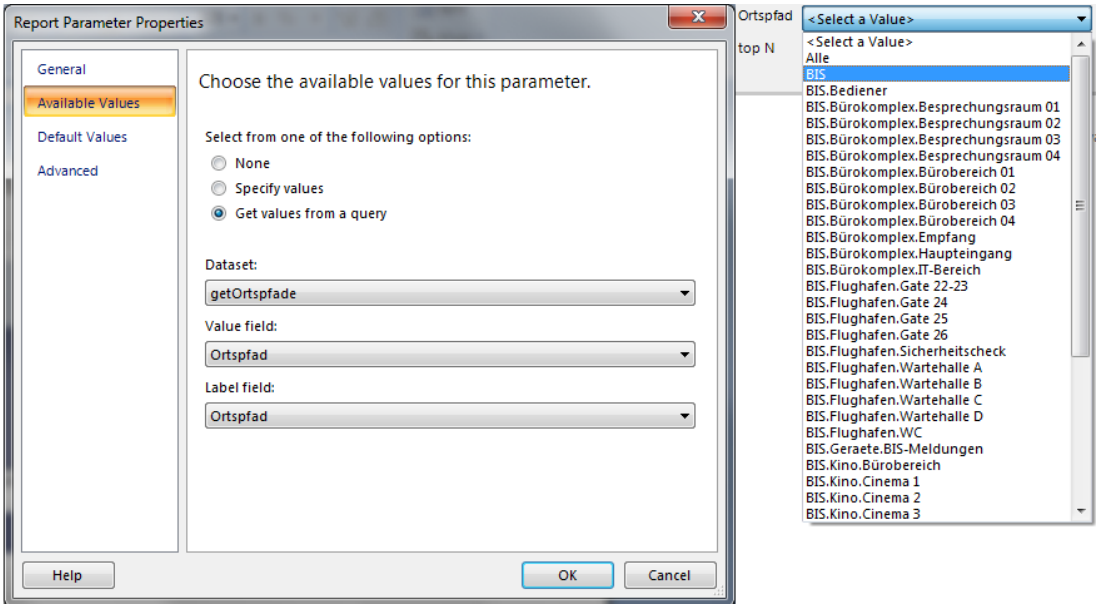


Bild 5.13: Abb. 25 Parameterauswahl hinzufügen

Auswahlfenster „Sortieren nach" einfügen

Die angezeigten Tabellen tendieren dazu bei großen Ausgaben unübersichtlich zu werden. Damit Sie einen bestimmten Melder schneller finden oder die Tabelle geordneter gestalten können, bietet der Report Builder eine Sortierfunktion aus. Diese überdeckt die Sortierfunktion des Selects und bietet den Vorteil, dass der Kunde selbst bestimmen kann, nach welchem Kriterium sortiert werden soll.

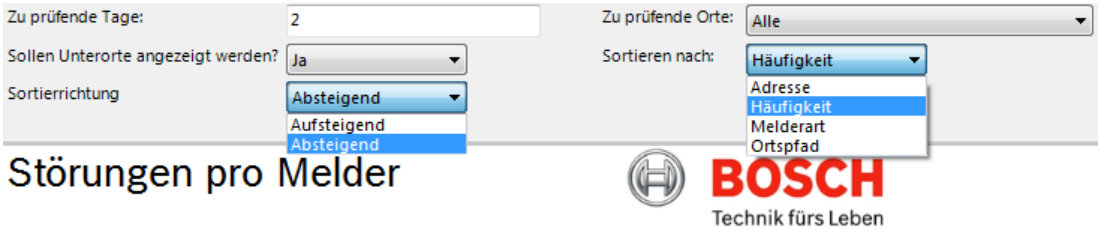


Bild 5.14: Abb. 26 Sortierfunktion Auswahlfenster

Um bei Ihnen auch eine Sortierfunktion einrichten zu können, klicken Sie auf Ihre Tabelle und anschließend mit Rechtsklick auf das oben links erscheinende **graue Quadrat**.
Wählen Sie nun **Tablix Properties...** aus und klicken Sie innerhalb dieses Fensters auf den Reiter **Sorting.**
Damit der Kunde bestimmen kann, ob die Tabelle aufsteigend oder abfallend sortiert werden soll, fügen Sie hier über **Add** zwei neue Optionen zum Sortieren hinzu. Die erste sortiert die Tabelle aufsteigend und die zweite abfallend. Welche Option verwendet wird, hängt von dem Parameter **direction** ab.
Stellen Sie also die **Order** der ersten Sortiermöglichkeit auf **A to Z**, öffnen Sie über **fx** die zugehörige Funktionsbeschreibung und tippen Sie folgenden Befehl ein:
=IIF(Parameters!direction.Value=„Ascending", Fields(Parameters!SortBy.Value).Value, 0)

Die zweite Option gibt die Order **Z to A** an und wird mit jenem Befehl parametriert:
=IIF(Parameters!direction.Value=„Descending“,Fields(Parameters!SortBy.Value).Value, 0)
Schließen Sie anschließend die Sortieroptionen mit dem Klick auf OK.

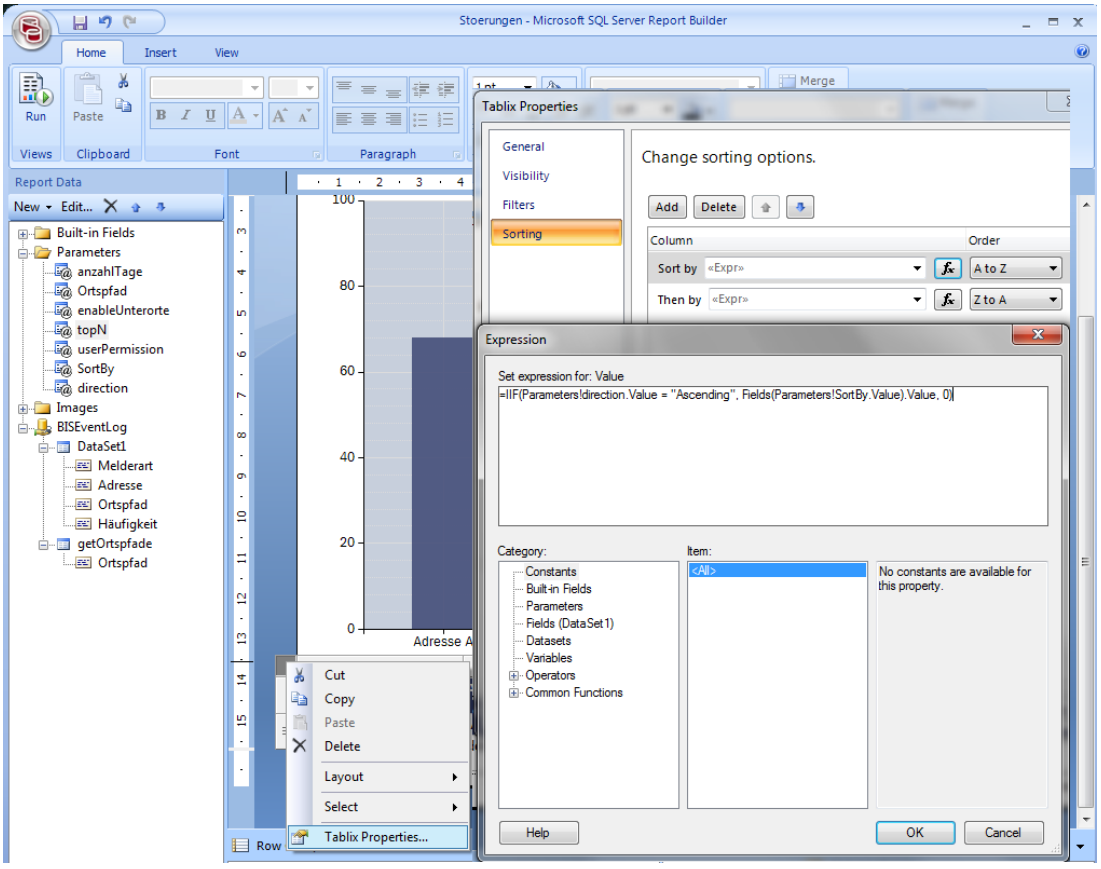


Bild 5.15: Abb. 27 Tablix Properties

Der letzte Punkt zum Errichten einer Sortierfunktion, ist es einen zwei neue Parameter über **Add Parameter** hinzuzufügen.
Der erste namens **SortBy** gibt an, nach welcher Spalte die Tabelle sortiert werden soll.
Der zweite namens **direction** legt fest ob die Tabellen ansteigend oder abfallend sortiert werden soll.
Nennen Sie den ersten Parameter in **SortBy** um, ändern Sie den **Prompt** des Parameters und öffnen Sie den Reiter **Available Values.**
Fügen Sie unter **Specify Values** all Ihre Spaltennamen, die zum Sortieren zur Auswahl stehen sollen hinzu. Setzen Sie schlussendlich noch einen **Default Value** fest, damit der Bericht beim Aufrufen sofort angezeigt wird.

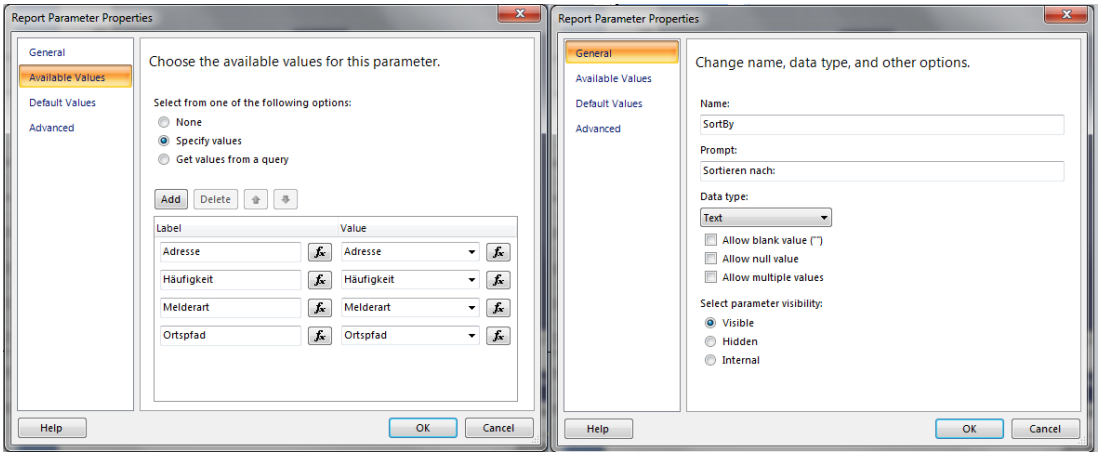


Bild 5.16: Abb. 28 Parameter SortBy

Nennen Sie den zweiten Parameter **direction**, ändern Sie dessen **Prompt** und öffnen Sie den Reiter **Available Values.** Fügen Sie hier zwei neue Werte hinzu. Das Label können Sie beliebig festlegen, der Wert muss allerdings einmal **Ascending** und das andere Mal **Descending** lauten.

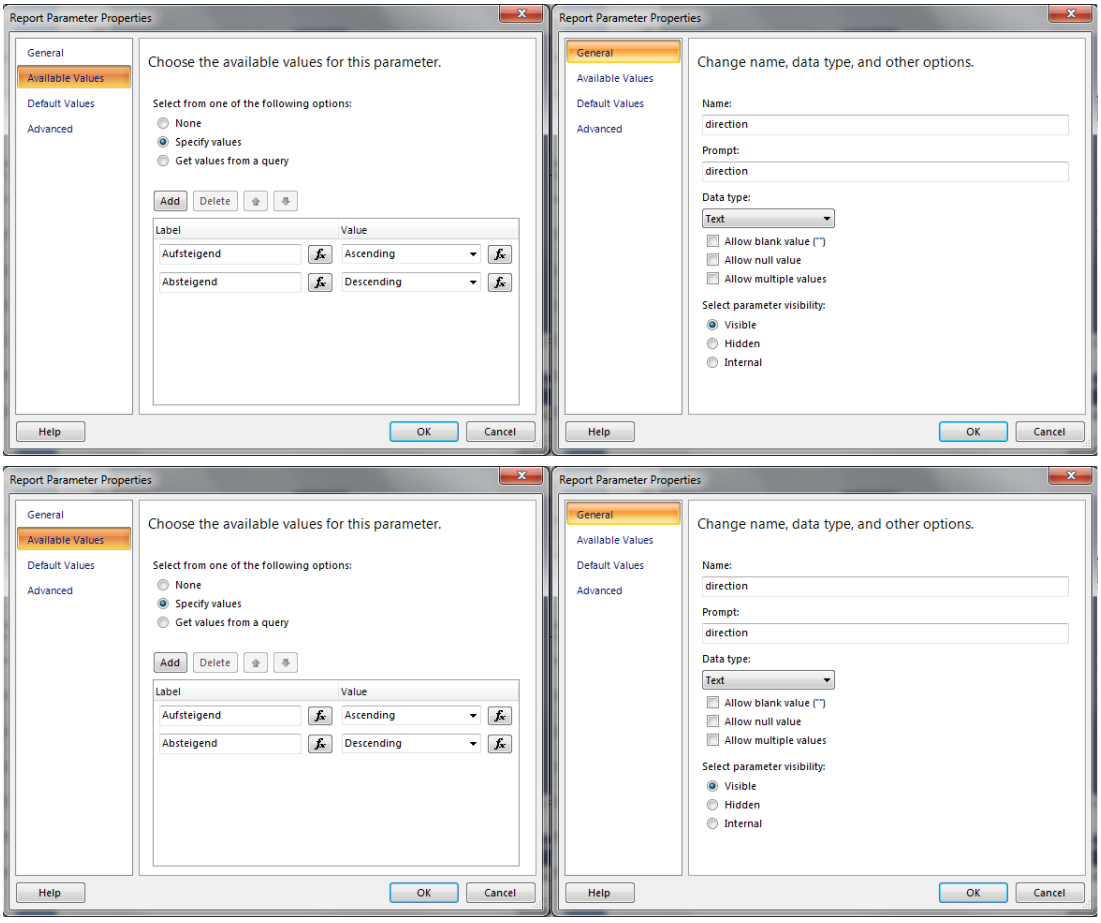


Bild 5.17: Abb. 29 Parameter direction

Schritt 11: Abspeichern u. Einbinden des Berichts in die BIS

Wenn Sie Ihren Bericht fertig gestellt haben, können Sie Ihn nun so abspeichern, dass er in der BIS angezeigt wird.

Drücken Sie dazu auf **Save** und wählen unter **Recent Sites and Servers** Ihren Reportserver aus. Im Beispiel ist dieser http://HPWS/Reports_BIS

Wählen Sie in diesem den vorinstallierten Ordner **BIS Reports** aus.

Im Beispielsfall habe ich zuvor über die Seite http://localhost/Reports_BIS einen neuen Ordner namens **Eigene Berichte** erstellt und den Bericht darin abgespeichert. In Punkt 6 „Verteilung der Berichte“ lernen Sie, wie den Ordner auch über eine Batch Datei erzeugen können. Meinen Beispielbericht habe ich als **BeispielStoerungen.rdl** abgespeichert. Unter diesem Namen ist der Bericht nun in der BIS zu finden.

Wichtig!

Speichern Sie den Bericht nicht unter MyReports ab, da die BIS keinen Zugriff auf diesen Ordner besitzt.

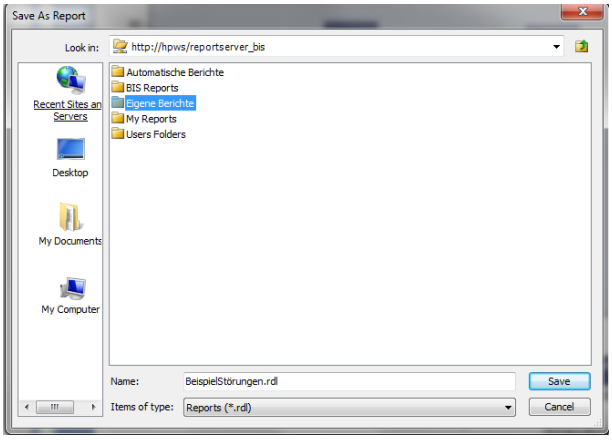
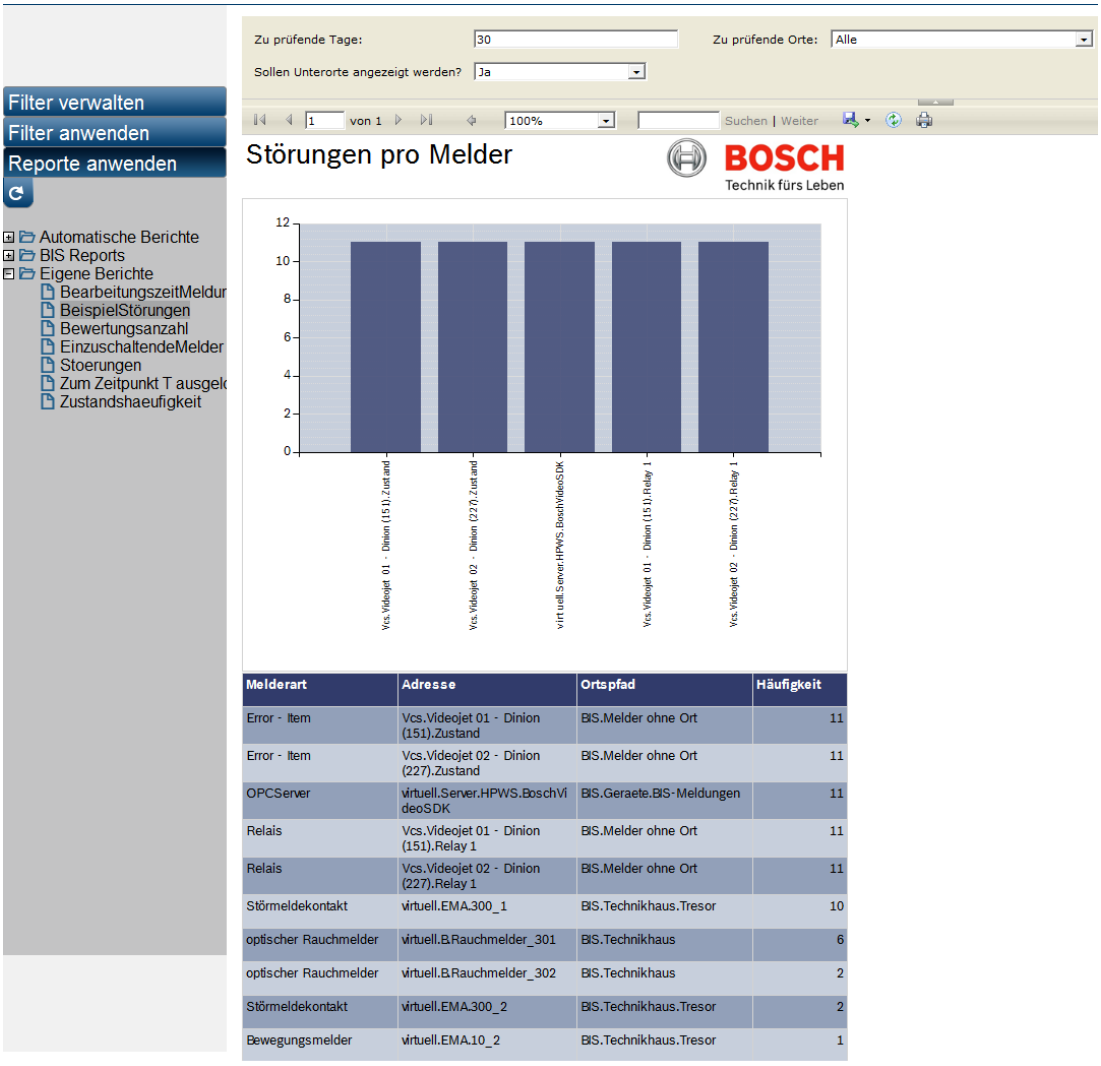


Bild 5.18: Abb. 30 Speicherort Bericht

Schritt 12: Aufrufen des Berichts

Starten Sie nun den BIS Klienten und öffnen Sie das **Logbuch**. Klicken Sie dort auf den Reiter **Reporte anwenden**. Wählen Sie nun Ihren Bericht aus. Im Falle des Beispiels ist der Bericht unter dem Namen BeispielStoerungen zu finden.



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

Bild 5.19: Abb. 31 Beispielbericht in BIS

5.3 Häufige Fehlermeldungen beim Erstellen von Berichten:

Auf den folgenden Seiten finden Sie die häufigsten Fehler und Lösungen, welche beim Arbeiten mit dem Report Builder auftreten.
Dabei sind die einzelnen Fehler über die jeweiligen Fehlermeldungen zu identifizieren.

Im 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”
  - Bericht funktioniert im Report Builder und auf dem Server nicht mehr. In der BIS funktioniert er aber meistens immer noch.
  - Fehler muss daher nicht unbedingt behoben werden.
- Möglichkeit zur Behebung der Fehlermeldung:
  - Data source properties -> in Credentials auf „do not use credentials" stellen -> "in general use a shared connection or report model” auswählen. -> erzeugte Datenbank BISEventLog ausssuchen.

  - Mit OK bestätigen. Bericht funktioniert im Report Builder, auf Server und in der BIS
- Möglichkeit zur Behebung der Fehlermeldung:
  - Rechtsklick DataSource -> Data source properties -> Credentials -> von “use current Windows user. Kerberos delegation may be required.” Auf “use this user name and password” umstellen. Benutzername logbuch_query und Passwort pw_logbuch_query eingeben. Kästchen nicht ankreuzen. Auf OK klicken und erneut testen. -> Bericht sollte beim Testen angezeigt werden.

- „The EXECUTE permission was denied on the object […]-Dataset […]”:
  - Der SQL Prozedur in der Datenbank BISReports fehlt die Berechtigung, dass der Benutzer logbuch_query auf die sie zugreifen darf.
  - Bei dieser Fehlermeldung schauen Sie sich bitte den Punkt *Anpassen der Berechtigungen, Seite 31* erneut an.
  - **Abhilfe:** Führen Sie den Code „`GRANT EXECUTE ON BISReports.dbo.<Prozedurname> TO logbuch_query`“ aus oder ergänzen Sie Ihn in der Prozedur report_Stoerungen

- Beim Öffnen des Report Query Designers kann der Report Builder nicht auf die Datenbank zugreifen.
  - **Abhilfe:** Report Builder „Als Administrator öffnen“

- „… Parameter is missing a value“
  - Die Prozedur, die den Parameter ausgeben soll, hat keine Ausgabe. D.h., dass z.B. bei der Suche nach einer stateNumber in der BIS noch kein Melder diesen Zustand besaß.
  - **Abhilfe:** Die Lösung zu diesem Problem finden Sie im Beispielsfall in Sektion *Erläuterte Beispielabfragen aus der BIS, Seite 25* unter der Frage „Wie finde ich die StateId eines Zustands heraus“

Beim Drücken auf “Run” im Report Builder : „Failed to preview report“:

Diese Meldung kann sehr viele Ursachen haben. Z.B. wird Sie hervorgerufen, durch eine falsch/ nicht definierte Datasource. Lassen Sie sich die Details anzeigen, um eine besser verwertbare Fehlermeldung anzeigen zu lassen.
„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. Möglichkeit zur Behebung der Fehlermeldung:
  - Rechtsklick auf die Datasource -> Data source properties
  - In Credentials auf „do not use credentials stellen“ -> in general “use a shared connection or report model” auswählen.
  - Datenbank Eventlog Datasource auswählen.
  - Mit OK bestätigen.
- 2. Möglichkeit zur Behebung der Fehlermeldung:
  - Rechtsklick auf die DataSource -> Data source properties -> Credentials -> von “use current Windows user. Kerberos delegation may be required.”
  - Auf “use this user name and password” umstellen.
  - Benutzername logbuch_query und Passwort pw_logbuch_query eingeben.

- Kästchen nicht ankreuzen.
- Auf OK klicken und erneut testen.

Im Browser (http://localhost/reports_bis**):**

- „Im userPermission-Parameter fehlt ein Wert“
  - Diese Fehlermeldung bedeutet an sich nichts Schlechtes. Sie wird dadurch verursacht, dass die Visibility in den Parameter Properties auf „Hidden“ gestellt ist. Sie ist zu ignorieren, da genau die Tatsache, dass userPermission auf Hidden gestellt ist, oft dazu führt, dass der Bericht erst in der BIS angezeigt wird.

In der BIS

- Bericht wird nicht angezeigt:
  Mögliche Ursachen:
  - Der Parameter „userPermission“ fehlt
    Abhilfe: Parameter neu erstellen: Name und Prompt= „userPermission“; Data type=Text; „Allow blank value“ und „Allow null value“ ankreuzen, Visibility auf Hidden stellen
  - Der Bericht wurde im Ordner „My Reports“ abgespeichert. Dieser Ordner ist geschützt und wird von der BIS nicht angezeigt.
    Abhilfe: Speichere den Bericht in einem anderen Ordner ab.

6 Verteilung der Berichte

Den Aufwand, welcher damit verbunden ist, die Berichte an die Techniker zu verteilen, und schlussendlich beim Kunden zu installieren, können Sie durch das Befolgen der nächsten Schritte stark reduzieren.
Dies liegt daran dass Sie mit Hilfe dieser Punkte eine Exe Datei erstellen, die vom Techniker beim Kunden nur noch ausgeführt werden muss. Das Programm wird von alleine die SQL Prozedur erstellen und den Bericht so einbinden, dass alles durch einen Doppelklick erledigt ist.
Fügen Sie dazu alle folgende Dateien und Quelltexte in einem Ordner zusammen.

- Fertiger Bericht als .rdl
- Quellcode als .sql zum Erstellen der Hauptprozedur in BISEventLog
- Quellcode zum Erstellen der Verknüpfungsprozedur in BISReports
- Quellcode, der logbuch_query berechtigt auf Prozedur zuzugreifen
- .rss Datei, welche den Bericht auf dem Reportserver speichert
- .exe Datei, welche die Codes ausführt und die .rss Datei aufruft

Sie merken, dass Sie schon so gut wie alle erwähnten Dateien erstellt haben. Das einzige was Ihnen noch fehlt, sollte die .rss und die .bat Datei sein.

6.1 Erstellen der .RSS Datei

Die .rss Datei dient dazu, den fertigen Bericht auf Ihrem SQL Reportserver abzuspeichern. Da dies mit relativ viel Quelltext verbunden ist, habe ich Ihnen ein Beispiel zu diesem Handbuch beigelegt. Es ist unter folgendem Ordner abgelegt:
`BIS_Berichte\Berichte\Bericht_Stoerungen_pro_Melder \Stoerungen_pro_Melder.rss`
Öffnen Sie das Dokument **Stoerungen_pro_Melder.rss** mit Hilfe des Notepads/Editors.
Nun müssen Sie lediglich die Variablen **parentFolder** und **reportName** editieren.
ParentFolder ist der Name des Ordners im Reportserver in den der Bericht eingefügt werden soll. Z.B. **Eigene Berichte**
ReportName ist der Name Ihres Berichts unter dem Sie diesen abgespeichert haben.
Im Beispiel wäre dies **BeispielStoerungen.**
Speichern Sie das Dokument mit einem Namen ohne Leerzeichen ab und machen Sie sich an das Erstellen der .bat Datei

6.2 Erstellen der .BAT Datei

Erstellen Sie ein neues Textdokument und Schreiben Sie dort folgende Befehlszeilen bearbeitet nieder.
Zu beachten:

- Die Texte in spitzen Klammern (<>) sind Variablen, die Sie durch tatsächliche Werte ersetzen müssen. Die Variablen sind:
  - `Localhost\BIS` ist der Servername, in den Sie sich auch im Management Studio eingeloggt haben.
  - `<OrdnernameDerSQLQueries>` steht für den Dateipfad, welcher von dem Speicherpunkt der Exe Datei zum Quelltext führt.
  - `<BISEventLog_Prozedurcode.sql>` steht für Ihre SQL-Datei, welche eine Prozedur innerhalb der Datenbank BISEventLog erstellt.

  - `<grantExecutePermission.sql>` bezieht sich auf die SQL-Datei, welche dem User logbuch_query Zugriff auf die Prozedur in BIS Reports gewährt. Sie enthält folgenden Code:
    `GRANT EXECUTE ON BISReports.dbo.<reportName> TO logbuch_query`
- `Localhost` ist eine Variable, welche automatisch durch den Rechnernamen, von dem die Batch Datei aufgerufen wird, ersetzt wird. Also beachten Sie bitte:
  Die .EXE Datei muss auf dem Serverrechner ausgeführt werden!
- Die Befehle sind durch die Seitenbegrenzung in zwei Zeilen geschrieben worden. Schreiben Sie jedoch die Befehle im Texteditor jeweils **ohne Zeilenumbruch**.

```
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
```

Speichern Sie das Textdokument mit der Dateiendung **.BAT** ab.

Weitere Hinweise
rs –i,... in der .BAT Datei führt dazu, dass Ihr Bericht auf dem Reportserver namens `reportserver_BIS` eingespielt wird. ‚Reportserver_BIS' ist der standardmäßige Name für Reportserver und kann somit auch von Ihnen übernommen werden.
Die Batch Datei ist ohne die Konvertierung ins EXE Format noch nicht testfähig, da die Dateipfade nicht zum Ausführen als .BAT angepasst sind.
Falls Sie nicht für jeden Test die Datei konvertieren wollen, müssen Sie eine neue erste Zeile **cd /d "%~dp0"** hinzufügen. Außerdem müssen vor dem Ordnerpfad der SQL Queries noch die fünf Zeichen **%~dp0** ergänzt werden und die Datei „Als Administrator ausgeführt" werden.

6.3 Konvertieren der Batch Datei als EXE

Das Herausgeben der Batch Datei zieht zwei Schwierigkeiten mit sich.
Die Erste ist, dass der Techniker die Datei manuell „Als Administrator ausführen" muss. Die Zweite ist, dass der komplette Dateipfad der Batch Datei keine Leerzeichen enthalten darf.
Da letzteres oft nicht der Fall ist und vorheriges oft vergessen wird, werden wir nun die Batch Datei in eine .EXE umwandeln.
Dazu öffnen Sie zunächst das mitgelieferte Programm **Bat To Exe Converter v1.6**, das unter folgendem Link zu finden ist:
`BIS_Berichte\Ext._Programme_und_Setups\Bat_To_Exe_Converter`

Sie können es alternativ auch unter http://www.f2ko.de/downloads/Bat_To_Exe_Converter.zip herunterladen.
Wählen Sie nun Ihre zuvor erstellte Batch Datei und den Speicherort der .EXE aus.
Bestätigen Sie nun das Feld **Add administrator manifest**, damit die Exe automatisch als Administrator ausgeführt wird und bestätigen Sie mit dem Klick auf **Compile**.

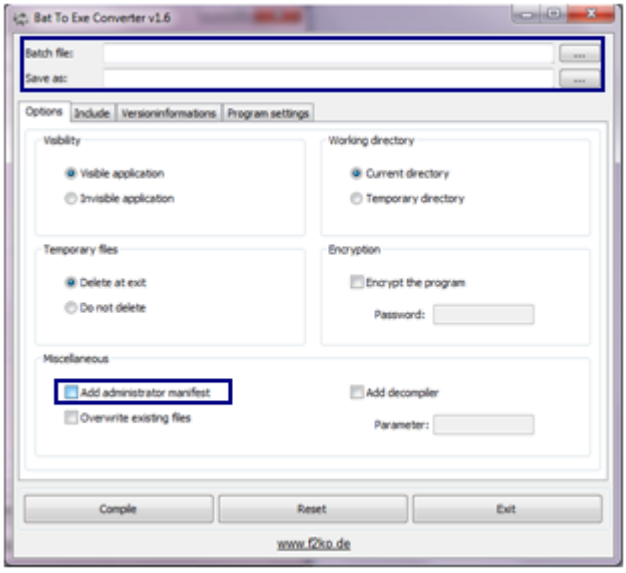


Bild 6.1: Abb. 32 Bat to Exe Converter

Wenn Sie alle obigen Schritte ordnungsgemäß ausgeführt haben, sind Sie nun fertig und können die .EXE-Datei zusammen mit den SQL Quelltexten und den Berichten an die Techniker ausliefern.

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