Creating Custom Data Sources For Custom Reports

THIS ARTICLE IS FOR ADVANCED SQL USERS ONLY. NUMBERCRUNCHER DOES NOT SUPPORT ANY MODIFICATIONS MADE TO THE SQL SERVER DATABASE. THIS GUIDE IS INTENDED FOR INFORMTAIONAL PURPOSES ONLY AND NUMBERCRUNCHER IS NOT RESPONSIBLE FOR ANY DATA ISSUES THAT RESULT FROM FOLLOWING THE STEPS LISTED BELOW.

Intro

The reporting engine in the All Orders user interface gives users an easy way to take existing reports and customize them for the their unique needs. One thing you cannot do through the interface is enter new data fields onto a report if they are not already listed as an available field on the report. This guide will show you where in the SQL database you will find all the data and views which control the reports. With this information you will be able to create new data sources and brand new reports which will run off of these data sources. Be sure to backup your database before running any SQL scripts.

The Tables

There are 4 tables that control the reports in All Orders. They all start with refReports.

refReports: The top level table containing the actual report name, design and data source.
refReports_Data: Contains all the fields listed in the data source which are available on the report.
refReports_Filters: Contains saved filters (‘Where’ clause) for each report.
refReports_Sorts: Contains saved fields to sort by ('Order by' clause) for each report.

Laying The Foundation

The first thing we need to do when creating a custom data source for our new report is to find the existing data source that is used for the one we are trying to copy. In this example we will be creating a new Sales Order form. The form I want to use as my base is named Sales Order in All Orders. By running the following query on the SQL database I can find out the data source for my report:

```
select case when reportsp is null or reportsp='' then reportsource else reportsp end as
DataSource
from refReports where ReportText='Sales Order'
```

Notice I entered the name of the report in my where clause to ensure I am pulling the correct record. Generally if the data source starts with ‘sp’ it is a stored procedure and if it starts with ‘qry’ it is a view. In this case the data source is qryrptSalesOrder so it is a view. My next step is to find the qryrptSalesOrder view and create a copy of it with a new name. I generate a CREATE statement for the view, renamed the view qryrptSalesOrderCustom at the top of the CREATE statement and run the script to create the view. I now have a custom view which no future update to All Orders will overwrite and I can use as a data source for new reports that I am going to create.

The next step is to create entries into all the relevant reporting tables and make a new copy of the Sales Order form which will run against our new data source. I will start with an insert statement for the refReports table which will insert a new record which is an exact copy of the Sales Order record but containing the new data source and report name:

```
Insert Into refReports (ReportText, ReportActive, ReportIsDefault, ReportSource,
ReportDisplayOrder, ReportGroupName, ReportGroup, ReportOriginal, ReportType,
```

```
                    ReportAuthor, ReportDescription, ReportXML, DateFilterField,
ReportLblRepeat, ReportPrinter, ReportTranType, ReportSp, ReportSpFilterField,
                    ReportSpFieldOp, ReportTextReportRPX)

SELECT     'Sales Order Custom', ReportActive, 0, 'qryrptSalesOrderCustom',
ReportDisplayOrder, ReportGroupName, ReportGroup, 'Sales Order Custom', ReportType,
                    ReportAuthor, ReportDescription, ReportXML, DateFilterField,
ReportLblRepeat, ReportPrinter, ReportTranType, ReportSp, ReportSpFilterField,
                    ReportSpFieldOp, ReportTextReportRPX
FROM         dbo.refReports Where ReportText='Sales Order'
```

The ReportText field will get whatever you want the report to be named. In this case I named it Sales Order Custom. The ReportSource field will get the name of the new, custom view data source we created. If it was a stored procedure we would leave the insert to be the ReportSource field without us overriding it and instead we would enter the name of the new, custom stored procedure data source in place of the ReportSp field in the insert. For the ReportOriginal field you can put the name of the report we are using as a base, in this case Sales Order, if you want this custom report to act like any other custom report in the system. If, however, you set the ReportOriginal to be the name of the custom report, as I am doing in the statement above where I specify Sales Order Custom, it will behave like a report that was packaged with the system which means it will have to be copied before any customizations can be made, thus preserving the original in its starting state. Notice the where clause ensures the insert is based on our base report record.

Once we have inserted the record into the refReports table, it is time to run inserts on the other relevant tables. Here is the statement for the refReports_Data table which will insert records based on the records of the report source we based our report on while inserting the name of the new source.

```
insert into refReports_Data ( ReportSource, RptDataFieldName, RptDataDisplaySort,
RptDataDisplayFilter, RptDataDataType, RptDataDisplayOrder, RptDataIsDataField,
                    RptDataDisplayShowEdit, RptDataFieldFormat, RptDataDisplayNull,
FieldChecked, RptDataFormula)
SELECT     'qryrptSalesOrderCustom', RptDataFieldName, RptDataDisplaySort,
RptDataDisplayFilter, RptDataDataType, RptDataDisplayOrder, RptDataIsDataField,
                    RptDataDisplayShowEdit, RptDataFieldFormat, RptDataDisplayNull,
FieldChecked, RptDataFormula
FROM         dbo.refReports_Data where ReportSource='qryrptSalesOrder'
```

Next is the sort table:

```
Insert Into  refReports_Sorts (RptDataReportName, RptDataFieldName, RptDataSortOrder,
RptDataSortMandatory, RptDataSortDefault)
SELECT     'Sales Order Custom', RptDataFieldName, RptDataSortOrder,
RptDataSortMandatory, RptDataSortDefault
FROM         dbo.refReports_Sorts where RptDataReportName='Sales Order'
```

And last the filter table:

```
Insert Into  refReports_Filters (RptDataReportName, RptDataFieldName,
RptDataFilterOperator, RptDataFilterFrom, RptDataFilterTo, RptDataFilterMandatory,
                    RptDataFilterOpNull)
SELECT     'Sales Order Custom', RptDataFieldName, RptDataFilterOperator,
RptDataFilterFrom, RptDataFilterTo, RptDataFilterMandatory,
                    RptDataFilterOpNull
FROM         dbo.refReports_Filters where RptDataReportName='Sales Order'
```

At this point the new report should be showing up in All Orders and it should behave exactly the same as the original report it was based on.

Making The Report You Own

You now have a report in the system which will never be overwritten in future All Orders updates. The refReports, refReport_Sorts and refReport_Filters tables can mostly be ignore moving forward. Most updates that you will want to do to records in these tables can be done in the All Orders user interface, such as making a report the default or adding saved sorts and filter to it.

They key will be in the custom view or stored procedure that was created and the refReports_Data table. You can now modify the view or stored procedure as much as you want. Add and remove fields, create new joins etc... Be sure to keep the refReport_Data table up to date to match the fields returned by your data source as they change. You can always get a list of the field with a select statement such as this one:

```
SELECT     dbo.refReports_Data.*
FROM         dbo.refReports_Data
WHERE     (ReportSource = 'qryrptSalesOrderCustom')
```

If a field is removed from the data source, be sure to remove its associated record from the refReports_Data table as well. If a field is added to the data source a new record must be created for it in the refReports_Data table. Take a look at a record for another field on the report that shares a similar data type (string, double etc...) to get a good understanding of what values will go into the different columns for the new records.

Enjoy Your New Report!