Creating a SalesPad Custom Procedure (SPCP)

Megan De Freitas - 2025-06-18 - Miscellaneous

Overview

This document reviews the steps needed to create a SalesPad Custom Procedure (SPCP) in SQL Server Management Studio (SSMS). Custom procedures add functionality to SalesPad beyond what is provided in the standard stored procedures, and they can be tailored to provide customer specific features. Many of SalesPad's default procedures have built-in support for a custom procedure. For a list of procedures that have a code call to an SPCP, see the documentation here. Unlike SalesPad's default procedures, custom procedures do not get overwritten or refreshed when you run a database update.

Disclaimer: SalesPad is not responsible for the code in the SPCP or any issues that are a result of modified logic used in the SPCP. Testing any changes to stored procedures in a non-production database is highly recommended. If a support issue is found to be related to a customer- or partner-written SPCP, any time spent by SalesPad may be billable.

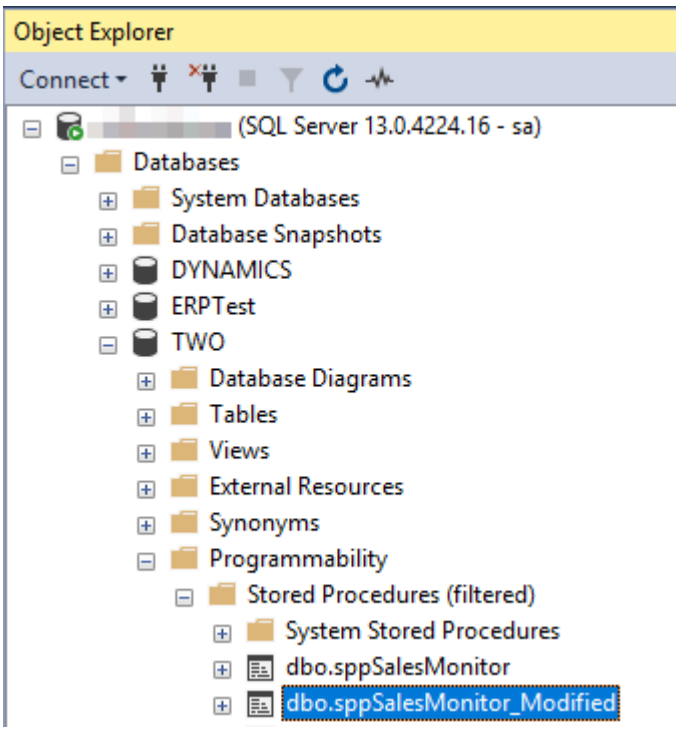
Creating a Custom Procedure

Some default procedures (see the list mentioned above) have a SQL call to a custom procedure. If a custom procedure does not exist for the chosen default procedure, the default procedure runs normally. If a custom procedure does exist, the code call in the default procedure will cause the custom procedure to execute in place of the default.

Using a Predefined Template

Some stored procedures have an included template meant for use as a template to create a custom procedure. In this documentation, we'll be starting with the template procedure sppSalesMonitor_Modified. Once created, certain sections of this template can be un-commented to add functionality that the default procedure does not have.

1. In SSMS, locate the stored procedure that you will be creating a custom procedure for.



2. Right-click on the stored procedure, then click **Modify.** In the query window that appears, change the SQL command from 'ALTER' to 'CREATE,' and rename the procedure from 'sppSalesMonitor_Modified' to 'spcpSalesMonitor.'

```
USE [TWO]
GO
/****** Object:  StoredProcedure [dbo].[sppSalesMonitor_Modified]    Script Date: 10/2/2018 9:56:23 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[spcpSalesMonitor] @docid AS VARCHAR(25) = NULL
    ,@doctypeEnum AS INT = 11 /* 1=Quotes, 2=Orders, 3=Ivoices, 4=Returns, 5=Backorders, 11=ALL*/
    ,@Customer AS VARCHAR(50) = NULL
    ,@Sales_Doc_Num AS VARCHAR(50) = NULL
    ,@Customer_PO_Num AS VARCHAR(50) = NULL
    ,@Sales_Person_ID AS VARCHAR(50) = NULL
    ,@Warehouse_Code AS VARCHAR(4000) = NULL
    ,@Hold AS CHAR(15) = NULL
    ,@Sales_Batch AS VARCHAR(MAX) = NULL
    ,@CustomFilter AS VARCHAR(MAX) = NULL
```

3. Create the desired new functionality. In the case of the procedure sppSalesMonitor_Modified, additional functionality has already been included, and simply needs to be un-commented.

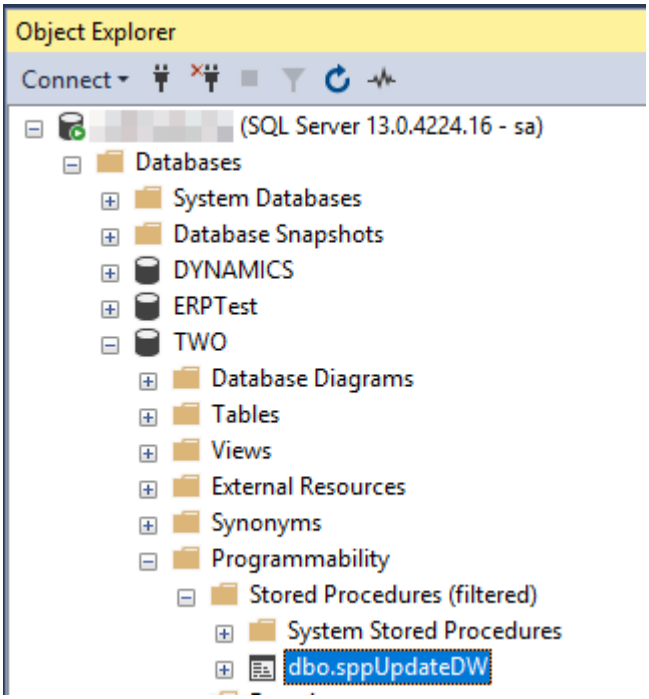
```
--NEW------->
--,[Total_Less_Backorders] = isnull(OREMSUBT -New.totalBackordered, 0)
--,[GM_Percent_Less_Backorders] = CASE (OREMSUBT -New.totalBackordered)
--WHEN 0
--  THEN 0
--ELSE (
--      (sh.OREMSUBT -New.totalBackordered) -(
--          SELECT isnull(sum(sl.ORUNTCST * (sl.QUANTITY -sl.QTYCANCE -sl.QTYTBAOR)), 0)
--          FROM sop10200 AS sl (NOLOCK)
--          WHERE sl.sopnumbe = sh.sopnumbe
--              AND sl.soptype = sh.soptype
--              AND sl.CMPNTSEQ = 0
--          )
--      ) / (sh.OREMSUBT -New.totalBackordered)
--END
```

4. Execute the changes to create the custom procedure. The new SPCP will now be used in place of the default procedure.

Using a Default Procedure as a Template

Most default procedures with support for an SPCP do not have a predefined template with additional functionality. For these procedures, a custom procedure can be created from the default procedure, which can be modified further for new functionality.

1. Similar to step 1 from above, choose a stored procedure with a code call to an SPCP. For this example, we will use the stored procedure sppUpdateDW.



2. Following the 2nd step from above, right-click on the stored procedure, then click **Modify.** In the query window that appears, change the SQL command 'ALTER' to 'CREATE,' and rename the procedure from 'sppUpdateDW' to 'spcpUpdateDW.'

```
USE [TWO]
GO
/****** Object:  StoredProcedure [dbo].[sppUpdateDW]    Script Date: 10/3/2018 11:23:41 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo] [spcpUpdateDW]
AS
IF (object_id('spcpUpdateDW') IS NOT NULL)
BEGIN
    EXEC spcpUpdateDW

    RETURN
END
```

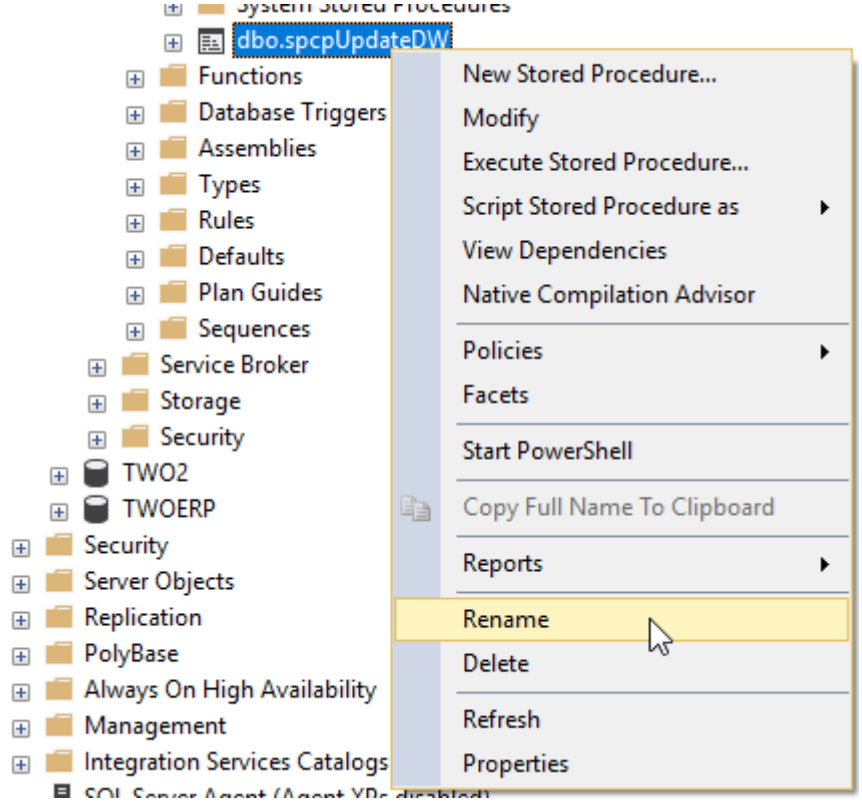
3. To avoid a nesting (infinite loop) error, remove the code call to the SPCP. Failure to remove this will result in the procedure continuously looping back into itself until SSMS recognizes the nesting error and cancels the SQL transaction.

4. Write the desired functionality changes.

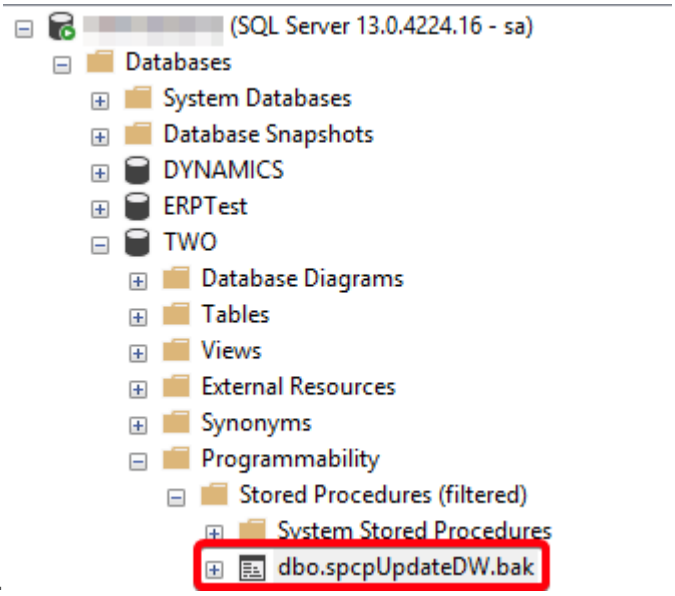
5. Execute the changes. The newly created custom procedure will now be used in place of the default.

Reverting Back to a Default Procedure

To stop using a custom procedure and revert back to SalesPad's default procedure, right-click on the desired custom procedure and click **Rename.**



Add the text ".bak" to the end of the procedure name, indicating that the custom procedure

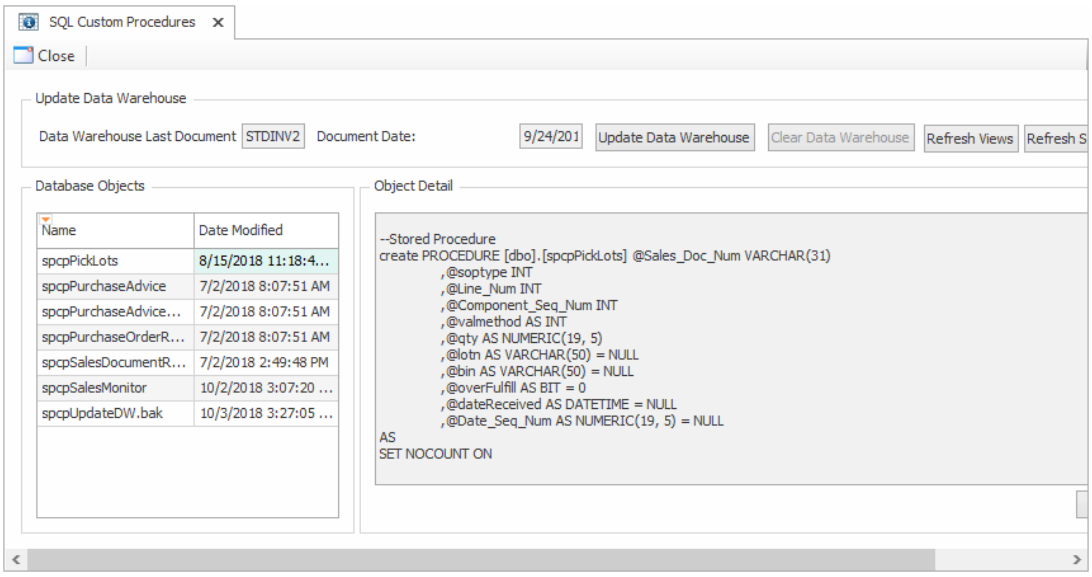


is now a backup.

Renaming the procedure effectively changes its identity, causing the default procedure to no longer "see" the existence of a matching custom procedure. To begin using the custom procedure again, simply rename it, removing the ".bak."

Viewing Custom Procedures in SalesPad

To see a list of all the custom procedures currently in place, open the SQL Custom Procedures module from the main menu. The list of custom procedures appears in the left grid, and the SQL code for the selected procedure appears on the Object Detail window to the right.



Security

SQL Custom Procedures - Grants access to the module to observe existing custom procedures.