**Stored Procedure Template Documentation**

**Overview**

The stored procedure template is designed as a starting spot to create stored procedures. The template uses the built in capabilities of token replacement within SSMS. To activate this feature, you can press CTRL+SHIFT+M and a dialog box will pop up which will allow you to enter the values for things such as the procedure name, variable names, data types, etc.

**Dependencies**

The template depends on four supporting stored procedures, three which are by default in a DBA database. The stored procedures are:

* DBA.s\_AddErrorLog – This procedure stores errors into the DBA.LogSQLError table
* DBA.s\_AddProcExecLog – This procedure stores execution information in the DBA.LogProcExec table
* DBA.s\_DropObject – This procedure drops objects
* master.dbo.sp\_UpsertExtendedProperty – This procedure adds/updates extended properties.

**Description by line**

|  |  |
| --- | --- |
| Lines 1-3 | This is the database that the procedure will be created in. |
| Lines 4-7 | These are the SET options to control the creation of the procedure. Different editor environments set these differently. This makes sure we always have consistency. |
| Lines 8-12 | This will drop the procedure before we create it. This ensures that we’re constantly dropping and creating. This can be an issue if we grant permissions to a specific proc and don’t script out those permissions. It can also create a different object\_id within the metadata, so this may create inconsistencies with execution history if we log such things such as in DBA.LogProcExec. |
| Lines 13-32 | This is xml formatted commenting that can be parsed by SqlSpec, a documenting tool that we use. The most important part is the history log. Each time you make a change, you should put another line in the history log with the date, and increment the version. |
| Lines 33-37 | The template comes initially with two parameter placeholders. If you need more, you will need to add them manually. If you need only one, you can remove the other. If you don’t need any parameters, then just remove all of these lines including the parentheses. |
| Lines 38-39 | This allows the stored procedure to be ran under the execution account of the creator of the procedure. For most development, this is insignificant, but for some types of development this can be handy. (e.g. you can allow a user with limited permissions, perform actions that he/she may not have been able to do otherwise) |
| Line 40 | Disables the messages that would normally be received by an ODBC client (e.g. xxx number of rows processed) |
| Line 41-42 | This forces SQL Server to automatically roll back transactions when errors are raised |
| Lines 43-52 | These are just setup variables for the audit logging later on. You should never need to change these. |
| Lines 53-55 | These are basically a way to not only comment a section, but to allow error handling to know exactly which section within the proc was being executed at the time of error |
| Lines 56-63 | If you want to log what parameters were passed to the stored procedure (It is recommended that you always log them) then you should uncomment this section out. You will need to add more rows if there are more than 2 parameters. There are two sections of parameters. The first section is for character and date parameters, use the first two lines, for numeric parameters, use the bottom two lines. Make sure to replace xxx with a valid data type length also. |
| Lines 64-70 | It is recommend that you always have a default value for your input parameters. This is a section where you can validate that the parameters are valid (e.g. for sql injection, or for business rules) and then just RAISEERROR or however you want to handle invalid parameters (this could also be moved down into the BEGIN TRY section |
| Lines 71-72 | This is the beginning of the TRY section. |
| Lines 73-77 | If the procedure will create temp tables, it’s a good practice to make sure they don’t exist first. This will drop the temp table. Replace #t with the temp table name. You can add as many EXEC DBA.dbo.s\_DropObject rows as needed if you’re creating multiple temp tables. You can delete this section if not creating temp tables within procedure. |
| Lines 78-84 | Placeholder for section to DECLARE local variables. Put all the variables you’ll use within the procedure in this section. |
| Lines 85-91 | Placeholder for the main code of the stored procedure |
| Lines 92-100 | If you have many different TSQL statements within your procedure and you want to audit each one separately (e.g. to measure timing, number of rows, or for some other reason), then this section of code should be put after each section. It will only log the activity for the previous section of code. Line 92 would have a comment describing the section, that way the audit log would be easier to understand. |
| Lines 101-107 | This section of audit logging will be placed after the last bit of TSQL is executed. That way, the entire execution statistics of the procedure will be logged. |
| Lines 108-112 | Repeat of lines 73-77, except trying to basically clean up after ourselves if we’ve created any temp tables. |
| Lines 113-114 | If all goes well, the stored procedure will return an execution result of 0 which denotes success. (e.g. anything not zero is an error) |
| Line 115 | End of the TRY section |
| Lines 116-119 | Beginning of the CATCH error handling section |
| Lines 120-128 | If you’ve used any cursors within your stored procedure, then if there’s an error, this section will destroy that cursor. The code assumes that you initially have scoped the cursor as LOCAL. It also assumes the cursor name is ‘curs’. This needs to be changed if cursor has a different name. If no cursors are present in code, these comments can be deleted. |
| Lines 129-132 | This will rollback any transactions |
| Lines 133-144 | This captures all the error details we can get and logs them to the DBA.LogSQLError table. |
| Lines 145-148 | This will re-raise the error to the calling application. |
| Lines 149-151 | This is the end of the CATCH error handling section and the end of the CREATE PROCEDURE statement. |
| Lines 152-156 | This will generate extended property descriptions for documentation purposes. The first one is for the stored procedure itself. The next two are for the parameters. SqlSpec will read these extended properties and generate documentation using them. Replace ‘Placeholder’ with descriptive text. If the procedure has more or less than 2 parameters, these lines will need to be added or subtracted from. |

**Key Concepts**

There are a few concepts that are key to understanding this template. The template was designed for stored procedures where we wanted to 1) log execution details, and 2) capture execution errors cleanly and consistently. With that understanding, it’s good to understand these concepts:

The @err\_sec should always contain a plain English description of the section of code being executed. It is used for commenting, logging, and error handling purposes. When an error is trapped, the value in @err\_sec will be logged to the error table. When logging a particular section, this parameter will be used to identify the section. You can also use this in place of actual comments.

The @params parameter is used to capture a comma delimited list of variables that were passed into the stored procedure. These are logged and used for troubleshooting. The beginning commented out section is commented out because the template doesn’t know what data types will be used for the parameters and it handles the conversion of parameters into varchar differently depending on the data type.

The template is setup to drop temporary tables in the beginning and end of the procedure as a defensive programming methodology. If temporary tables are not required, both of these sections can be removed.

The procedure can log the execution timing and row count of individual T-SQL statements/sections as well as the entirety of the procedure.