Overview of RAISERROR

RAISERROR allows developers to produce our own error message. Using RAISERROR, we can throw our own error message while running our Query or Stored procedure. RAISERROR is used to return messages back to applications using the same format as a system error or warning message generated by the SQL Server Database Engine. We can also set our own severity for each and every individual message.

To conclude the summary:

* It allows developers to generate their own messages
* It returns the same message format that is generated by SQL Server Database Engine
* We can set our own level of Severity for messages
* It can be associated with Query and stored procedure

General Syntax for using RAISERROR

Below is the general syntax for RAISERROR:

RAISERROR ( { Message ID | Message Text}

{ ,severity ,state }

[ ,argument [ ,...n ] ] )

[ WITH option [ ,...n ] ]

Let's divide the block for better understanding.

Parameters of RAISERROR

RAISERROR ( { Message ID | Message Text}

Generally we can use a specific message id (msg\_id) or provide an error message string.

Message ID

Is a user-defined error message number stored in the sys.messages catalog view. Error numbers for user-defined error messages should be greater than 50000. By default, RAISERROR raises an error message with an error number of 50000.

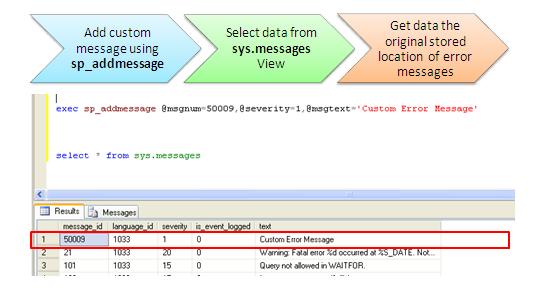
We can add error number using sp\_addmessge in the following way:

exec sp\_addmessage @msgnum=50009,@severity=1,@msgtext='Custom Error Message'

Now, if you want to check what the original location that messages are stored in, you need to run the following query:

select \* from sys.messages

Have a look at the below images, where I have explained the step by step procedures. Right now, forget about @Severity parameter and other optional parameter. I have explained them later.



This is how the data is stored inside SQL Server and returned by the SQL Server Database Engine when we call like this...

RAISERROR ( 50009,1,1)

... which will show us the below output:

Custom Error Message

Msg 50009, Level 1, State 1

Now, I guess you can co-relate things. Let's move to message text parameter.

Message Text

This is the message description, that I have already explained in the last example and you are now also aware where it is stored physically. Error message has certain limitations:

* The error message can have a maximum of 2,047 characters
* If the message has more than 2,047 characters, then will show only 2,044 characters with an ellipsis to indicate that the message has been truncated

As I have already mentioned, if we did not specify any message number, RAISERROR will use 50000 ID by default. This is the only reason we need to specify the error message ID more than 50000. If we add any message with ID 50000, it will throw the following error:

User-defined error messages must have an ID greater than 50000.

There are certain parameters used with message text. For that, I will recommend you read the article that I have mentioned in the **Further Study** section.

Now, just have a look at the other two parameters of RAISERROR:

RAISERROR ( { Message ID| Message Text}

**{ ,severity ,state }**

These stand for set Severity and state for the message. These two are numeric types and relate to how severe the message is.

Severity

We have to mention severity while adding the message using sp\_addmessage. Range of Severity level is 0-25. But for user defined message, we have to set it up to 0-19. 20-25 can only be set by the administrator. Severity levels from 20 through 25 are considered fatal.

Gives you information about the type of error that ocurred

Upto level 10 are informational messages

11-16 are considered errors that can be fixed by the user

17-19 are considered Non-fatal errors

20-25 are considered Fatal Errors which cause Sql Server to shut down the process

States : This is the point the location where the error has coccured in the code. It helps you to check the which error was raised

For any message related to RAISERROR, we have to specify the state also. The default State value is 1. The range of state is from 1 to 127. But for most implementations, we use 1. We can use it to indicate which error was thrown by providing a different state for each RAISERROR function in our stored procedure. This is a required parameter.

WITH Options

Finally, there are options that we can set, these are the WITH options.

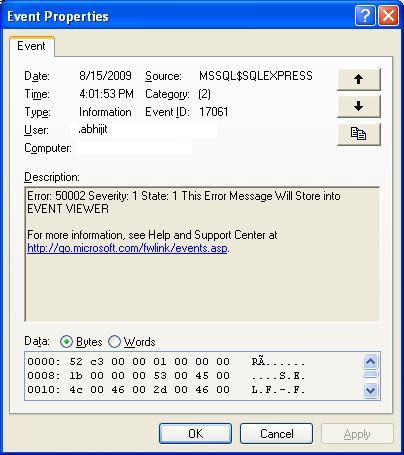
LOG

It will place the error in windows Error log. We have to mention this parameter while adding the message using sp\_addmessage.

exec sp\_addmessage @msgnum=50002,@severity=1,\_

@msgtext='This Error Message Will Store into EVENT VIEWER',@with\_log='true'

Now, if we want to test the result, just execute this command RAISERROR ( 50002,1,1) , we can get an entry in EventViewer.



NOWAIT

Send the error directly to client.

SETERROR

It will replace the error ID with 5000.

Looking Inside Sp\_addmessage

I have already explained it previously. Here I am going to give you a brief overall syntax and the arguments it needs.

sp\_addmessage [ @msgnum= ] msg\_id, [ @severity= ] severity ,

[ @msgtext = ' ] 'message'

[, [ @lang = ] 'Language' ]

[, [ @with\_log = ] 'log' ]

[, [ @replace = ] 'replace' ]

Here is the general overview of those parameters.

@msgnum

Indicates the number of the message. It should be greater than 50000.

@severity

We used 1 in most cases. General range is 1 to 25. I have already covered the details.

@msgtext

Message text, maximum characters limit is 2,047. More than that will be truncated.

@lang

Used if you want to specify any language.

@with\_log

As I have already shown in example, Set 'TRUE' to log the error in EventViewer.

@replace

If the same message number already exists, but you want to replace the string for that ID, you have to use this parameter.

ALL In One Example

Now have a look into a simple example where we can check each and every point that has been discussed above.

Add a User-defined Message

To add an error message, we have to use sp\_addmessgae stored procedure. This message will store inside sys.messages. The error number should be greater than 5000.

exec sp\_addmessage @msgnum=50010,@severity=1,\_

@msgtext='User-Defined Message with ID 50010'

Check The Details Inside

This is not mandatory, you can check the original location and how it is stored by just running the following query:

select \* from sys.messages

This will give you the following output:

message\_id language\_id severity is\_event\_logged text

----------- ----------- -------- --------------- ----------------------------

50010 1033 1 1 User-Defined Message with ID 50010

Call RAISERROR

Now we can retrieve the message using RAISERROR:

RAISERROR ( 50010,1,1)

This will give the following output:

User-Defined Message with ID 50010

Msg 50010, Level 1, State 1

Store Error Inside Event Log

If we need to store the message inside Event Viewer, we have to use @with\_log parameter:

sp\_addmessage @msgnum=50002,@severity=1,\_

@msgtext='This Error Message Will Store into EVENT VIEWER',@with\_log='true'

I have already covered that part how it is stored inside Event Viewer.

Replace the Existing Message

If we have already set the message for some error id and we want to replace the message, it will throw an error as follows:

You must specify 'REPLACE' to overwrite an existing message.

So, for replacing message we have to use

@replace parameter with sp\_addmessge Stored procedure.

Here is the syntax:

exec sp\_addmessage @msgnum=50010,@severity=1,\_

@msgtext='User-Defined Message with ID 50010-Replaced Message',\_

@with\_log='true', **@replace='replace'**

This will replace the message for the id 50010.

You can try it using run RAISERROR again.

Use Try-Catch with RAISE ERROR

The below code is a simple example where I have shown how we can use RAISERROR inside TRY-CATCH Block:

BEGIN TRY

DECLARE @Intval int;

SET @Intval = 1/0; *-- Divide by Zero Error*

END TRY

BEGIN CATCH

RAISERROR (50009, 11, 1); *-- 50009 is our pre-defined error*

*-- that are stored using sp\_addmessage*

END CATCH;