SQL Server RAISERROR

**I**n this tutorial, you will learn how to use the SQL Server RAISERROR statement to generate user-defined error messages.

SQL Server RAISEERROR statement overview

The RAISERROR statement allows you to generate your own error messages and return these messages back to the application using the same format as a system error or warning message generated by SQL Server Database Engine. In addition, the RAISERROR statement allows you to set a specific message id, level of severity, and state for the error messages.

The following illustrates the syntax of the RAISERROR statement:

RAISERROR ( { message\_id | message\_text | @local\_variable }

{ ,severity ,state }

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

[ **WITH** **option** [ ,...n ] ];

Let’s examine the syntax of the RAISERROR for better understanding.

message\_id

The message\_id is a user-defined error message number stored in the sys.messages catalog view.

To add a new user-defined error message number, you use the [stored procedure](https://www.sqlservertutorial.net/sql-server-stored-procedures/) sp\_addmessage. A user-defined error message number should be greater than 50,000. By default, the RAISERROR statement uses the message\_id 50,000 for raising an error.

The following statement adds a custom error message to the sys.messages view:

EXEC sp\_addmessage

@msgnum = 50005,

@severity = 1,

@msgtext = 'A custom error message';

To verify the insert, you use the following query:

**SELECT**

\*

**FROM**

sys.messages

**WHERE**

message\_id = 50005;

To use this message\_id, you execute the RAISEERROR statement as follows:

RAISERROR ( 50005,1,1)

Here is the output:

A custom error message

Msg 50005, Level 1, State 1

To remove a message from the sys.messages, you use the stored procedure sp\_dropmessage. For example, the following statement deletes the message id 50005:

EXEC sp\_dropmessage

@msgnum = 50005;

message\_text

The message\_text is a user-defined message with formatting like the printf function in C standard library. The message\_text can be up to 2,047 characters, 3 last characters are reserved for ellipsis (…). If the message\_text contains 2048 or more, it will be truncated and is padded with an ellipsis.

When you specify the message\_text, the RAISERROR statement uses message\_id 50000 to raise the error message.

The following example uses the RAISERROR statement to raise an error with a message text:

RAISERROR ( 'Whoops, an error occurred.',1,1)

The output will look like this:

Whoops, an error occurred.

Msg 50000, Level 1, State 1

severity

The severity level is an integer between 0 and 25, with each level representing the seriousness of the error.

0–10 Informational messages

11–18 Errors

19–25 Fatal errors

state

The state is an integer from 0 through 255. If you raise the same user-defined error at multiple locations, you can use a unique state number for each location to make it easier to find which section of the code is causing the errors. For most implementations, you can use 1.

WITH option

The option can be LOG, NOWAIT, or SETERROR:

* WITH LOG logs the error in the error log and application log for the instance of the SQL Server Database Engine.
* WITH NOWAIT sends the error message to the client immediately.
* WITH SETERROR sets the ERROR\_NUMBER and @@ERROR values to message\_id or 50000, regardless of the severity level.

SQL Server RAISERROR examples

Let’s take some examples of using the RAISERROR statement to get a better understanding.

A) Using SQL Server RAISERROR with TRY CATCH block example

In this example, we use the RAISERROR inside a [TRY](https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-try-catch/) block to cause execution to jump to the associated CATCH block. Inside the CATCH block, we use the RAISERROR to return the error information that invoked the CATCH block.

**DECLARE**

@ErrorMessage **NVARCHAR**(4000),

@ErrorSeverity INT,

@ErrorState INT;

**BEGIN** TRY

RAISERROR('Error occurred in the TRY block.', 17, 1);

**END** TRY

**BEGIN** CATCH

**SELECT**

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

*-- return the error inside the CATCH block*

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

**END** CATCH;

Here is the output:

Msg 50000, Level 17, State 1, Line 16

Error occurred in the TRY block.

B) Using SQL Server RAISERROR statement with a dynamic message text example

The following example shows how to use a local variable to provide the message text for a RAISERROR statement:

**DECLARE** @MessageText **NVARCHAR**(100);

**SET** @MessageText = N'Cannot delete the sales order %s';

RAISERROR(

@MessageText, *-- Message text*

16, *-- severity*

1, *-- state*

N'2001' *-- first argument to the message text*

);

The output is as follows:

Msg 50000, Level 16, State 1, Line 5

Cannot **delete** the sales **order** 2001

When to use RAISERROR statement

You use the RAISERROR statement in the following scenarios:

* Troubleshoot Transact-SQL code.
* Return messages that contain variable text.
* Examine the values of data.
* Cause the execution to jump from a TRY block to the associated CATCH block.
* Return error information from the CATCH block to the callers, either calling batch or application.