**THROW (Transact-SQL)**

**SQL Server 2012**

Raises an exception and transfers execution to a CATCH block of a TRY…CATCH construct in SQL Server 2012.

[Transact-SQL Syntax Conventions](http://technet.microsoft.com/en-us/library/ms177563.aspx)

[Syntax](javascript:void(0))

THROW [ { error\_number | @local\_variable },

        { message | @local\_variable },

        { state | @local\_variable } ]

[ ; ]

[Arguments](javascript:void(0))

error\_number

Is a constant or variable that represents the exception. error\_number is int and must be greater than or equal to 50000 and less than or equal to 2147483647.

message

Is an string or variable that describes the exception. message is nvarchar(2048).

state

Is a constant or variable between 0 and 255 that indicates the state to associate with the message. state is tinyint.

[Remarks](javascript:void(0))

The statement before the THROW statement must be followed by the semicolon (;) statement terminator.

If a TRY…CATCH construct is not available, the session is ended. The line number and procedure where the exception is raised are set. The severity is set to 16.

If the THROW statement is specified without parameters, it must appear inside a CATCH block. This causes the caught exception to be raised. Any error that occurs in a THROW statement causes the statement batch to be ended.

**Differences Between RAISERROR and THROW**

The following table lists differences between the RAISERROR and THROW statements.

|  |  |
| --- | --- |
| **RAISERROR statement** | **THROW statement** |
| If a msg\_id is passed to RAISERROR, the ID must be defined in sys.messages. | The error\_number parameter does not have to be defined in sys.messages. |
| The msg\_str parameter can contain printf formatting styles. | The message parameter does not accept printf style formatting. |
| The severity parameter specifies the severity of the exception. | There is no severity parameter. The exception severity is always set to 16. |

[Examples](javascript:void(0))

**A. Using THROW to raise an exception**

The following example shows how to use the THROW statement to raise an exception.

Transact-SQL

THROW 51000, 'The record does not exist.', 1;

Here is the result set.

Msg 51000, Level 16, State 1, Line 1

The record does not exist.

**B. Using THROW to raise an exception again**

The following example shows how use the THROW statement to raise the last thrown exception again.

Transact-SQL

USE tempdb;

GO

CREATE TABLE dbo.TestRethrow

(    ID INT PRIMARY KEY

);

BEGIN TRY

INSERT dbo.TestRethrow(ID) VALUES(1);

-- Force error 2627, Violation of PRIMARY KEY constraint to be raised.

INSERT dbo.TestRethrow(ID) VALUES(1);

END TRY

BEGIN CATCH

PRINT 'In catch block.';

THROW;

END CATCH;

Here is the result set.

PRINT 'In catch block.';

Msg 2627, Level 14, State 1, Line 1

Violation of PRIMARY KEY constraint 'PK\_\_TestReth\_\_3214EC272E3BD7D3'. Cannot insert duplicate key in object 'dbo.TestRethrow'.

The statement has been terminated.

**C. Using FORMATMESSAGE with THROW**

The following example shows how to use the FORMATMESSAGE function with THROW to throw a customized error message. The example first creates a user-defined error message by using sp\_addmessage. Because the THROW statement does not allow for substitution parameters in the message parameter in the way that RAISERROR does, the FORMATMESSAGE function is used to pass the three parameter values expected by error message 60000.

Transact-SQL

EXEC sys.sp\_addmessage

@msgnum = 60000

,@severity = 16

,@msgtext = N'This is a test message with one numeric parameter (%d), one string parameter (%s), and another string parameter (%s).'

,@lang = 'us\_english';

GO

DECLARE @msg NVARCHAR(2048) = FORMATMESSAGE(60000, 500, N'First string', N'second string');

THROW 60000, @msg, 1;

Here is the result set.

Msg 60000, Level 16, State 1, Line 2

This is a test message with one numeric parameter (500), one string parameter (First string), and another string parameter (second string).