(T29)介紹 DDLTrigger 搭配 AllServerScope 和 LoginTrigger
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc
==========
(T29)介紹 DDLTrigger 搭配 AllServerScope 和 LoginTrigger
==========

==========

0. Summary

1.
DDL Trigger Reference:
https://docs.microsoft.com/en-us/sql/relational-databases/triggers/ddl-events
Drop DDL trigger(ALL Server scope/Database scope) if it exists
http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/
Syntax
```
--CREATE TRIGGER [TriggerName]
--ON (All Server/Database)
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
--   ...
--END
```

2.
Database Scope DDL Trigger
2.1.
Create Database Scope DDL Trigger
Syntax
```
--CREATE TRIGGER [TriggerName]
--ON Database
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
```

```
--   ...
--END
```
2.1.1.
E.g.
```
--IF EXISTS ( SELECT  *
--          FROM    sys.triggers
--          WHERE   name = 'trgNoCreateAlterDropTable' )
--    BEGIN
--       DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;
--    END;
--GO -- Run the previous command and begins new batch
--CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
--AS
--    BEGIN
--       PRINT 'Create table is prohibited.';
--       ROLLBACK;
--    END;
--GO -- Run the previous command and begins new batch
```
2.1.2.
Create DDL Database Triggers in SSMS
Database Name --> Programmability --> Database Triggers

2.2.
Enable/Sisable Trigger
Syntax
```
--DISABLE TRIGGER trgName ON DATABASE
```
Disable trgName DATABASE scope trigger.
```
--ENABLE TRIGGER trgName ON DATABASE
```
Enable trgName DATABASE scope trigger.
E.g.
```
--DISABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE
--ENABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE
```

2.3.
```
----Drop database scope trigger if it exists
--IF EXISTS ( SELECT  *
--          FROM    sys.triggers
--          WHERE   name = 'trgNoCreateAlterDropTable' )
--    BEGIN
--       DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;
--    END;
--GO -- Run the previous command and begins new batch
```

2.4.
```
--CREATE TRIGGER trgRename ON DATABASE
--    FOR RENAME
--AS
--    BEGIN
--       PRINT 'Rename DDL trigger is fired.';
--    END;
--GO -- Run the previous command and begins new batch
--...
--sp_rename 'TestTable', 'TestTable2';
```
Rename 'TestTable' to 'TestTable2'
```
--sp_rename 'TestTable2.ID' , 'ID2', 'column'
```
Rename TestTable2.ID Column to TestTable2.ID2 Column,
the third parameter means dealing with column.

3.
All Server Scope DDL Trigger
3.1.
Create All Server Scope DDL Trigger
Syntax
```
--CREATE TRIGGER [TriggerName]
```

```
--ON All Server
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
--   ...
--END
```

3.1.1.

E.g.

```
--IF EXISTS ( SELECT  *
--          FROM    sys.server_triggers
--          WHERE   name = 'trgNoCreateAlterDropTable2' )
--    BEGIN
--       DROP TRIGGER trgNoCreateAlterDropTable2 ON DATABASE;
--    END;
--GO -- Run the previous command and begins new batch
--CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
--AS
--    BEGIN
--       PRINT 'Any definition change for table is not prohibited.';
--       ROLLBACK;
--    END;
--GO -- Run the previous command and begins new batch
```

The scope is ALL SERVER.

This is the CREATE_TABLE, ALTER_TABLE, DROP_TABLE event trigger.

3.1.2.

Create DDL Database scope Triggers in SSMS

Database Name --> Programmability --> Database Triggers

Create DDL All Server scope Triggers in SSMS

Server Objects --> Triggers --> ...

3.2.

Enable/Sisable ALL SERVER scope DDL trigger

Syntax

```
--DISABLE TRIGGER trgName ON ALL SERVER
```

Disable trgName All Server scope trigger.

```
--ENABLE TRIGGER trgName ON ALL SERVER
```

Enable trgName All Server scope trigger.

E.g.

```
--DISABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
--ENABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
```

3.3.

Drop ALL SERVER scope trigger if it exists

Reference: http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/

```
--IF EXISTS ( SELECT  *
--          FROM    sys.server_triggers
--          WHERE   name = 'trgNoCreateAlterDropTable2' )
--    BEGIN
--       DROP TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;
--    END;
--GO -- Run the previous command and begins new batch
```

4.

4.1.

Even if we use sp_settriggerorder to change the order,

server scoped triggers will always fire before any of the database scoped triggers.

4.2.

sp_settriggerorder can set the order of server scoped DDL triggers

or database scoped triggers .

4.3.

E.g.

```
--EXEC sp_settriggerorder
--    @triggername = 'trgFirstTrigger',
```

-- @order = 'first',
-- @stmttype = 'CREATE_TABLE',
-- @namespace = 'DATABASE';
Set the database scoped trigger, trgFirstTrigger,
to the first order of CREATE_TABLE event.
4.3.1.
-- @triggername = 'trgFirstTrigger',
1st parameter is the trigger name that you want to set.
4.3.2.
--@order = 'first',
--@order = 'last',
--@order = 'none',
2nd parameter is the running order.
Value can be First, Last or None.
4.3.3.
-- @stmttype = 'CREATE_TABLE',
3rd parameter is the SQL statement that fires the trigger.
You many only put ONE statement type here.
If you want 'ALTER_TABLE' or 'DROP_TABLE' or other statement,
you need to "ECECUTE sp_settriggerorder" several times
for each statement type you want to set.
4.3.4.
-- @namespace = 'DATABASE';
or
-- @namespace = 'SERVER';
4th parameter is the scope of the trigger.
Value can be DATABASE, SERVER, or NULL.

4.4.
If there is a database scoped DDL Triggers and a server scoped DDL triggers
handling the same event.
Here is the execution order.
4.4.1.
The server scope DDL trigger which @order = 'first'
-->
other server scope DDL triggers which @order = 'none'
-->
The server scope DDL trigger which @order = 'Last'
-->
The database scope DDL trigger which @order = 'first'
-->
other database scope DDL triggers which @order = 'none'
-->
The database scope DDL trigger which @order = 'Last'

5.
--EVENTDATA()
5.1.
--EVENTDATA()
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/eventdata-transact-sql
EventData() returns information about server or database events in XML format.
E.g.
--<EVENT_INSTANCE>
-- <EventType>CREATE_TABLE</EventType>
-- <PostTime>2017-10-10T04:42:27.870</PostTime>
-- <SPID>54</SPID>
-- <ServerName>N550JKL\SQL2016</ServerName>
-- <LoginName>MicrosoftAccount\lpmplpmp01@hotmail.com</LoginName>
-- <UserName>dbo</UserName>
-- <DatabaseName>Sample3</DatabaseName>
-- <SchemaName>dbo</SchemaName>
-- <ObjectName>TestTable</ObjectName>
-- <ObjectType>TABLE</ObjectType>
-- <TSQLCommand>

```
--    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON" ENCRYPTED="FALSE" />
--    <CommandText>CREATE TABLE TestTable
--    (
--      ID INT ,
--      [Name] NVARCHAR(100)
--    )</CommandText>
--  </TSQLCommand>
--</EVENT_INSTANCE>
-----------------------------------------------------
5.2.
----Drop Table if it exists
--IF ( EXISTS ( SELECT    *
--            FROM     INFORMATION_SCHEMA.TABLES
--            WHERE    TABLE_NAME = 'AuditTableStructureChanges' ) )
--    BEGIN
--       TRUNCATE TABLE dbo.AuditTableStructureChanges;
--       DROP TABLE AuditTableStructureChanges;
--    END;
--GO -- Run the previous command and begins new batch
--CREATE TABLE AuditTableStructureChanges
--    (
--      EventType NVARCHAR(300) ,
--      PostTime DATETIME ,
--      --Server Process ID
--      SPID NVARCHAR(300) ,
--      ServerName NVARCHAR(300) ,
--      LoginName NVARCHAR(300) ,
--      UserName NVARCHAR(300) ,
--      DatabaseName NVARCHAR(300) ,
--      SchemaName NVARCHAR(300) ,
--      ObjectName NVARCHAR(300) ,
--      ObjectType NVARCHAR(300) ,
--      TSQLCommand NVARCHAR(MAX)
--    );
--GO
-----------------------------------------------------
----Drop ALL SERVER scope trigger if it exists
----Reference: http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/
--IF EXISTS ( SELECT  *
--          FROM    sys.server_triggers
--          WHERE   name = 'trgAuditTableStructureChanges' )
--    BEGIN
--       DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;
--    END;
--GO -- Run the previous command and begins new batch
--CREATE TRIGGER trgAuditTableStructureChanges ON ALL SERVER
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
--AS
--    BEGIN
--       DECLARE @EventData XML;
--       SELECT  @EventData = EVENTDATA();
--       INSERT  INTO Sample3.dbo.AuditTableStructureChanges
--             ( EventType ,
--               PostTime ,
--               SPID ,
--               ServerName ,
--               LoginName ,
--               UserName ,
--               DatabaseName ,
--               SchemaName ,
--               ObjectName ,
--               ObjectType ,
--               TSQLCommand
--             )
```

```
--      VALUES  ( @EventData.value('(/EVENT_INSTANCE/EventType)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/PostTime)[1]', 'DATETIME') ,
--            @EventData.value('(/EVENT_INSTANCE/SPID)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/ServerName)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/LoginName)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/UserName)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/DatabaseName)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/SchemaName)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/ObjectType)[1]',
--                        'NVARCHAR(300)') ,
--            @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
--                        'NVARCHAR(MAX)')
--          );
--   END;
--GO -- Run the previous command and begins new batch
```

==========

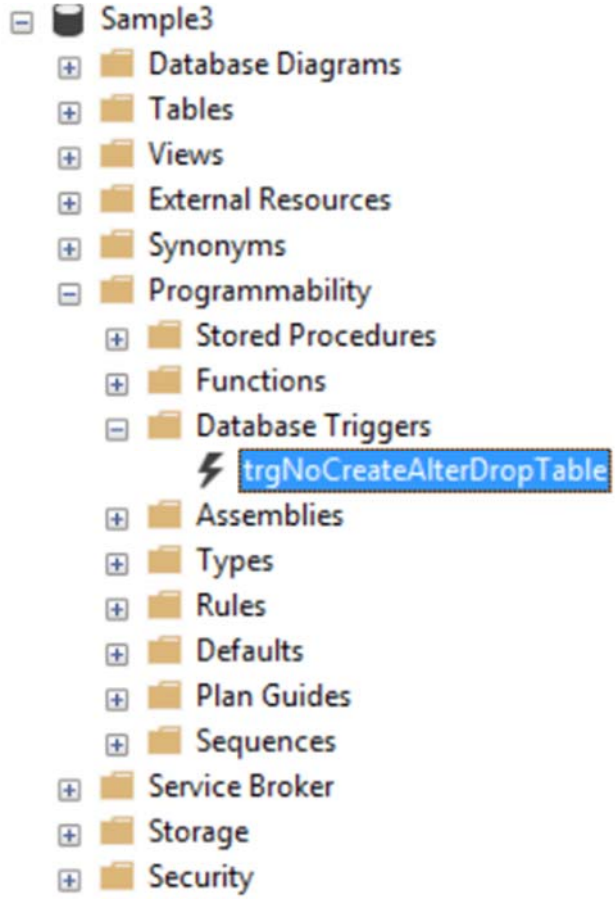
1. Database Scoped Data Definition Language (DDL) Triggers event : CREATE_TABLE, ALTER_TABLE , DROP_TABLE, RENAME

```
--==============================================================================
--T029_01_Database Scoped Data Definition Language (DDL) Triggers event : CREATE_TABLE, ALTER_TABLE ,
DROP_TABLE, RENAME
--==============================================================================
```

1.1. DDL Trigger event: CREATE_TABLE

```
--==============================================================================
--T029_01_01
--DDL Trigger event: CREATE_TABLE
-------------------------------------------------------------------------------
--T029_01_01_01
--Create DDL Trigger : CREATE_TABLE
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgNoCreateAlterDropTable' )
    BEGIN
        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE
    FOR CREATE_TABLE
AS
    BEGIN
        PRINT 'Create table is prohibited.';
        ROLLBACK;
    END;
GO -- Run the previous command and begins new batch
/*
```

```
1.
--CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE
--    FOR CREATE_TABLE
--AS
--    BEGIN
--        PRINT 'Create table is prohibited.';
--        ROLLBACK;
--    END;
prohibit create table in Sample3 database.
1.1.
Syntax
--CREATE TRIGGER [TriggerName]
--ON (All Server/Database)
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
--   ...
--END
1.2.
--CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE
--    FOR CREATE_TABLE
Create a trigger which name as trgNoCreateAlterDropTable
The scope is current database.
This is the CREATE_TABLE event trigger.
1.3.
Create DDL Database Triggers in SSMS
Database Name --> Programmability --> Database Triggers
*/
```




```
-----------------------------------------------------------------------------
--T029_01_01_02
--Test DDL Trigger CREATE_TABLE
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'TestTable' ) )
   BEGIN
       TRUNCATE TABLE dbo.TestTable;
```

```
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE TestTable ( ID INT PRIMARY KEY );
GO -- Run the previous command and begins new batch
/*
Output Error Message
--Create table is prohibited.
--Msg 3609, Level 16, State 2, Line 88
--The transaction ended in the trigger. The batch has been aborted.
*/
```

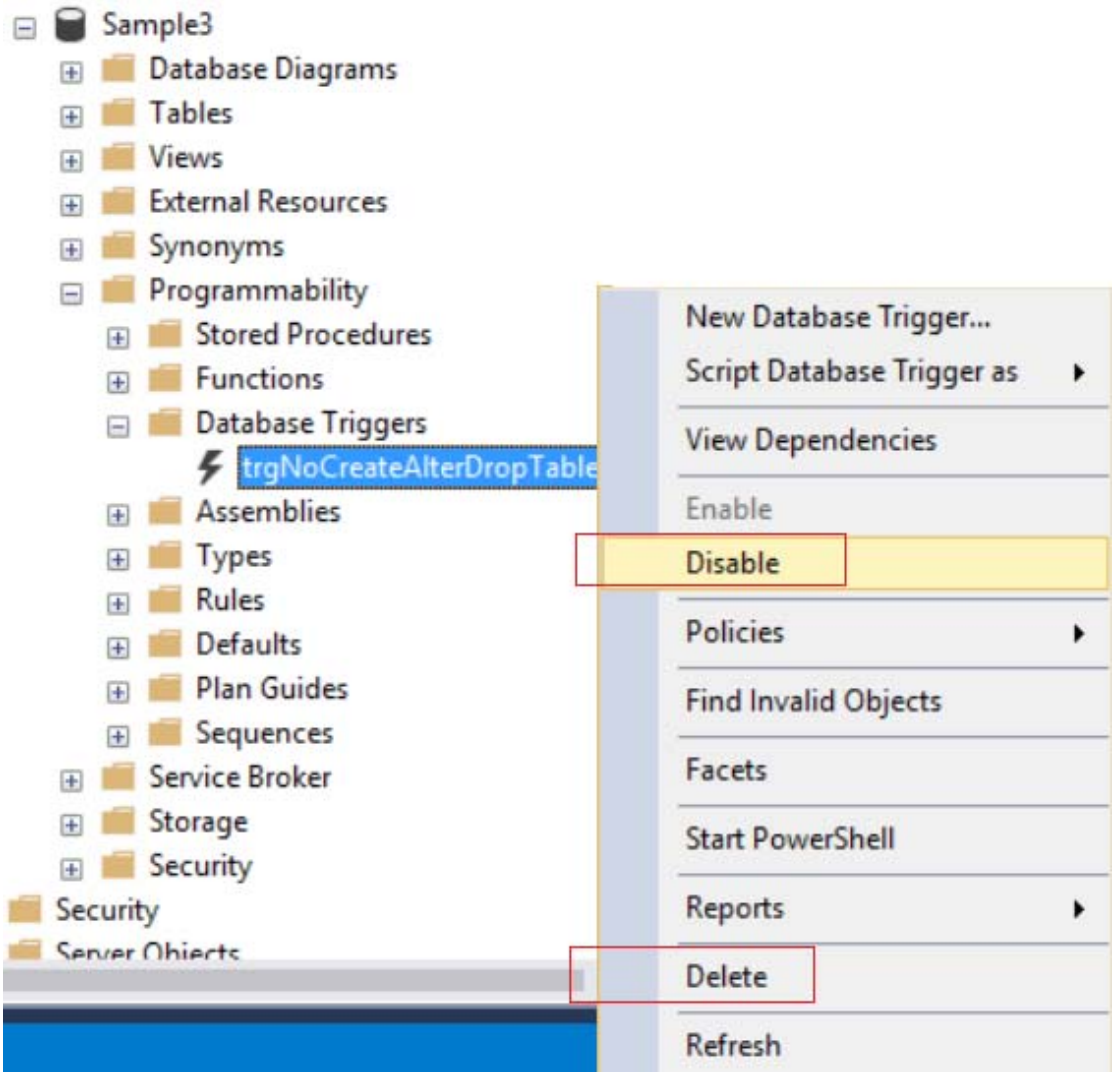
Messages

```
Create table is prohibited.
Msg 3609, Level 16, State 2, Line 387
The transaction ended in the trigger. The batch has been aborted.
```

1.2. Enable/Disable DDL Trigger : CREATE_TABLE, ALTER_TABLE, DROP_TABLE

```
--==============================================================================
--T029_01_02
--Enable/Disable DDL Trigger : CREATE_TABLE, ALTER_TABLE, DROP_TABLE
-------------------------------------------------------------------------
--T029_01_02_01
--Create DDL Trigger : CREATE_TABLE, ALTER_TABLE, DROP_TABLE
ALTER TRIGGER trgNoCreateAlterDropTable ON DATABASE
    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
    BEGIN
        PRINT 'Any change for table is not prohibited.';
        ROLLBACK;
    END;
GO -- Run the previous command and begins new batch
/*
1.
--ALTER TRIGGER trgNoCreateAlterDropTable ON DATABASE
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
--AS
--    BEGIN
--        PRINT 'Any change for table is not prohibited.';
--        ROLLBACK;
--    END;
prohibit create/Alter/Drop table in Sample3 database.
1.1.
Syntax
--CREATE TRIGGER [TriggerName]
--ON (All Server/Database)
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
--   ...
--END
1.2.
--ALTER TRIGGER trgNoCreateAlterDropTable ON DATABASE
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
Alter a trigger which name as trgNoCreateAlterDropTable
The scope is current database.
This is the CREATE_TABLE, ALTER_TABLE, DROP_TABLE event trigger.
1.3.
Create DDL Database Triggers in SSMS
Database Name --> Programmability --> Database Triggers
*/
```

```
-----------------------------------------------------------------------
--T029_01_02_02
--Enable/Disable DDL Trigger : CREATE_TABLE, ALTER_TABLE, DROP_TABLE
/*
Enable/Sisable Trigger
Syntax
--DISABLE TRIGGER trgName ON DATABASE
Disable trgName DATABASE scope trigger.
--ENABLE TRIGGER trgName ON DATABASE
Enable trgName DATABASE scope trigger.
E.g.
--DISABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE
--ENABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE
*/
----------------------------------------
--T029_01_02_02_01
--Disable the Trigger
DISABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE;
GO -- Run the previous command and begins new batch
```




```
----------------------------------------
--T029_01_02_02_02
--Test the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
   BEGIN
       TRUNCATE TABLE dbo.TestTable;
       DROP TABLE TestTable;
```

```
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE TestTable ( ID INT PRIMARY KEY );
GO -- Run the previous command and begins new batch
--Create table successfully..
---------------------------------------
--T029_01_02_02_03
--Test the trigger
DROP TABLE TestTable;
--Drop table successfully..
---------------------------------------
--T029_01_02_02_04
--Enable the Trigger
ENABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE;
GO -- Run the previous command and begins new batch
---------------------------------------
--T029_01_02_02_04
--Test the trigger
CREATE TABLE TestTable ( ID INT PRIMARY KEY );
GO -- Run the previous command and begins new batch
-- Error : Create table un-successfully
```

Messages

```
Any change for table is not prohibited.
Msg 3609, Level 16, State 2, Line 495
The transaction ended in the trigger. The batch has been aborted.
```

```
---------------------------------------
--T029_01_02_02_05
--Test the trigger
DROP TABLE TestTable;
GO -- Run the previous command and begins new batch
--Error : Drop table un-successfully.
```

Messages

```
Msg 3701, Level 11, State 5, Line 502
Cannot drop the table 'TestTable', because it does not exist or you do not have permission.
```

```
---------------------------------------------------------------------
--T029_01_02_03
--Drop DATABASE scoped DDL Trigger : CREATE_TABLE, ALTER_TABLE, DROP_TABLE
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgNoCreateAlterDropTable' )
    BEGIN
        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
```

1.3. DATABASE scoped DDL Trigger : RENAME

```
--==========================================================================
--T029_01_03
--DATABASE scoped DDL Trigger : RENAME
---------------------------------------------------------------------
--T029_01_03_01
--Create DATABASE scoped DDL Trigger : RENAME
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgRename' )
    BEGIN
```

```
        DROP TRIGGER trgRename ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgRename ON DATABASE
    FOR RENAME
AS
    BEGIN
        PRINT 'Rename DDL trigger is fired.';
    END;
GO -- Run the previous command and begins new batch
/*
1.
--CREATE TRIGGER trgRename ON DATABASE
--    FOR RENAME
--AS
--    BEGIN
--        PRINT 'Rename DDL trigger is fired.';
--    END;
Rename DDL trigger will be fired when rename events happen.
1.1.
Syntax
--CREATE TRIGGER [TriggerName]
--ON (All Server/Database)
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
--   ...
--END
1.2.
--CREATE TRIGGER trgRename ON DATABASE
--    FOR RENAME
CREATE a trigger which name as trgRename
The scope is current database.
This is the RENAME event trigger.
1.3.
Create DDL Database Triggers in SSMS
Database Name --> Programmability --> Database Triggers
*/
----------------------------------------------------------------------
--T029_01_03_02
--Test DATABASE scoped DDL Trigger : RENAME
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE TestTable ( ID INT );
GO -- Run the previous command and begins new batch
--Rename 'TestTable' to 'TestTable2'
EXECUTE sp_rename 'TestTable', 'TestTable2';
GO -- Run the previous command and begins new batch
/*
Output message
--Caution: Changing any part of an object name could break scripts and stored procedures.
--Rename DDL trigger is fired.
*/
```

```
Messages
Caution: Changing any part of an object name could break scripts and stored procedures.
Rename DDL trigger is fired.
```

```
--Rename TestTable2.ID Column to TestTable2.ID2 Column,
--the third parameter means dealing with column.
EXECUTE sp_rename 'TestTable2.ID', 'ID2', 'column';
GO -- Run the previous command and begins new batch
/*
Output message
--Caution: Changing any part of an object name could break scripts and stored procedures.
--Rename DDL trigger is fired.
*/
```

```
Messages
Caution: Changing any part of an object name could break scripts and stored procedures.
Rename DDL trigger is fired.
```

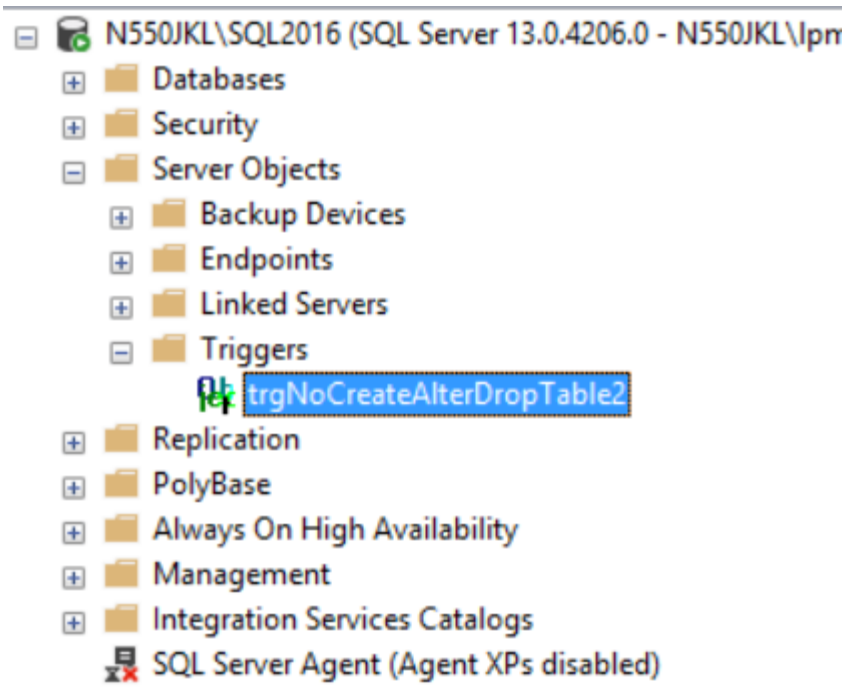
1.4. Clean up

```
--==========================================================================
--T029_01_04
--Clean up
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgNoCreateAlterDropTable' )
    BEGIN
        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgRename' )
    BEGIN
        DROP TRIGGER trgRename ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable2' ) )
    BEGIN
        TRUNCATE TABLE TestTable2;
        DROP TABLE TestTable2;
    END;
GO -- Run the previous command and begins new batch

==================================================
```

2. Server Scoped Data Definition Language (DDL) Triggers event : CREATE_TABLE, ALTER_TABLE , DROP_TABLE

```
--==============================================================================
--T029_02_Server Scoped Data Definition Language (DDL) Triggers event : CREATE_TABLE, ALTER_TABLE ,
DROP_TABLE
--==============================================================================
--==============================================================================
--T029_02_01
--Create Server Scoped DDL Triggers event : CREATE_TABLE, ALTER_TABLE , DROP_TABLE
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgNoCreateAlterDropTable2' )
    BEGIN
        DROP TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
    BEGIN
        PRINT 'Any definition change for table is not prohibited.';
        ROLLBACK;
    END;
GO -- Run the previous command and begins new batch
```




```
/*
1.
--IF EXISTS ( SELECT  *
--            FROM    sys.server_triggers
--            WHERE   name = 'trgNoCreateAlterDropTable2' )
--    BEGIN
--        DROP TRIGGER trgNoCreateAlterDropTable2 ON DATABASE;
--    END;
```

```
--GO -- Run the previous command and begins new batch
--CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
--AS
--    BEGIN
--        PRINT 'Any definition change for table is not prohibited.';
--        ROLLBACK;
--    END;
prohibit create/Alter/Drop table in ALL SERVER.
1.1.
Syntax
--CREATE TRIGGER [TriggerName]
--ON (All Server/Database)
--FOR [EventType1, EventType2, ...],
--AS
--BEGIN
--   ...
--END
1.2.
--CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
--    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
CREATE a trigger which name as trgNoCreateAlterDropTable2
The scope is ALL SERVER.
This is the CREATE_TABLE, ALTER_TABLE, DROP_TABLE event trigger.
1.3.
Create DDL Database scope Triggers in SSMS
Database Name --> Programmability --> Database Triggers
Create DDL All Server scope Triggers in SSMS
Server Objects --> Triggers --> ...
*/
--==============================================================================
--T029_02_02
--Disable/Enable Server Scoped DDL Triggers event : CREATE_TABLE, ALTER_TABLE , DROP_TABLE
--Disable/Enable ALL SERVER scope DDL trigger
DISABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;
GO -- Run the previous command and begins new batch
ENABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;
GO -- Run the previous command and begins new batch
/*
Enable/Sisable ALL SERVER scope DDL trigger
Syntax
--DISABLE TRIGGER trgName ON ALL SERVER
Disable trgName All Server scope trigger.
--ENABLE TRIGGER trgName ON ALL SERVER
Enable trgName All Server scope trigger.
E.g.
--DISABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
--ENABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER
*/
--==============================================================================
--T029_02_03
--Drop Server Scoped DDL Triggers event : CREATE_TABLE, ALTER_TABLE , DROP_TABLE
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgNoCreateAlterDropTable2' )
    BEGIN
        DROP TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
```

3. TriggerExecutionOrder_sp_settriggerorder

```
--==============================================================================
--T029_03_TriggerExecutionOrder_sp_settriggerorder
--==============================================================================
```

3.1. sp_settriggerorder

```
--==============================================================================
--T029_03_01
--sp_settriggerorder
-----------------------------------------------------------------------
--T029_03_01_01
--Create two CREATE_TABLE triggers
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgFirstTrigger' )
    BEGIN
        DROP TRIGGER trgFirstTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgSecondTrigger' )
    BEGIN
        DROP TRIGGER trgSecondTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgSecondTrigger ON DATABASE
    FOR CREATE_TABLE
AS
    PRINT 'This is the second CREATE_TABLE trigger';
GO
CREATE TRIGGER trgFirstTrigger ON DATABASE
    FOR CREATE_TABLE
AS
    PRINT 'This is the first CREATE_TABLE trigger';
GO
-----------------------------------------------------------------------
--T029_03_01_02
--Test the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
Output as the following
--This is the second CREATE_TABLE trigger
--This is the first CREATE_TABLE trigger
*/
--==============================================================================
--T029_03_01
```

```sql
--sp_settriggerorder
-----------------------------------------------------------------------
--T029_03_01_01
--Create two CREATE_TABLE triggers
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgFirstTrigger' )
    BEGIN
        DROP TRIGGER trgFirstTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgSecondTrigger' )
    BEGIN
        DROP TRIGGER trgSecondTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgSecondTrigger ON DATABASE
    FOR CREATE_TABLE
AS
    PRINT 'This is the second CREATE_TABLE trigger';
GO
CREATE TRIGGER trgFirstTrigger ON DATABASE
    FOR CREATE_TABLE
AS
    PRINT 'This is the first CREATE_TABLE trigger';
GO
-----------------------------------------------------------------------
--T029_03_01_02
--Test the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
Output as the following
--This is the second CREATE_TABLE trigger
--This is the first CREATE_TABLE trigger
*/
```

Messages

```
Caution: Changing any part of an object name could break scripts and stored procedures.
Rename DDL trigger is fired.
```

```sql
-----------------------------------------------------------------------
--T029_03_01_03
--sp_settriggerorder
EXEC sp_settriggerorder @triggername = 'trgFirstTrigger', @order = 'first',
       --@order = 'last',
       --@order = 'none',
    @stmttype = 'CREATE_TABLE', @namespace = 'DATABASE';
       --@namespace = 'SERVER';
/*
```

```
1.
--EXEC sp_settriggerorder
--     @triggername = 'trgFirstTrigger',
--     @order = 'first',
--  @stmttype = 'CREATE_TABLE',
--     @namespace = 'DATABASE';
Set the database scoped trigger, trgFirstTrigger,
to the first order of CREATE_TABLE event.
1.1.
--     @triggername = 'trgFirstTrigger',
1st parameter is the trigger name that you want to set.
1.2.
--@order = 'first',
--@order = 'last',
--@order = 'none',
2nd parameter is the running order.
Value can be First, Last or None.
1.3.
--  @stmttype = 'CREATE_TABLE',
3rd parameter is the SQL statement that fires the trigger.
You many only put ONE statement type here.
If you want 'ALTER_TABLE' or 'DROP_TABLE' or other statement,
you need to "ECECUTE sp_settriggerorder" several times
for each statement type you want to set.
1.4.
-- @namespace = 'DATABASE';
or
-- @namespace = 'SERVER';
4th parameter is the scope of the trigger.
Value can be DATABASE, SERVER, or NULL.
*/
----------------------------------------------------------------------
--T029_03_01_04
--Test the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
   BEGIN
       TRUNCATE TABLE dbo.TestTable;
       DROP TABLE TestTable;
   END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
--This is the first CREATE_TABLE trigger
--This is the second CREATE_TABLE trigger
*/
```

Messages

```
This is the first CREATE_TABLE trigger
This is the second CREATE_TABLE trigger
```

```
----------------------------------------------------------------------
--T029_03_01_05
--sp_settriggerorder
--When @order = 'none', trigger is fired in random order
EXEC sp_settriggerorder @triggername = 'trgFirstTrigger',
       --@order = 'first',
       --@order = 'last',
    @order = 'none', -- Clean up
    @stmttype = 'CREATE_TABLE', @namespace = 'DATABASE';
```

```
       --@namespace = 'SERVER';
/*
1.
--EXEC sp_settriggerorder
--     @triggername = 'trgFirstTrigger',
--     @order = 'none',
--  @stmttype = 'CREATE_TABLE',
--     @namespace = 'DATABASE';
Set the database scoped trigger, trgFirstTrigger,
to the none order of CREATE_TABLE event.
When @order = 'none', trigger is fired in random order.
1.1.
--     @triggername = 'trgFirstTrigger',
1st parameter is the trigger name that you want to set.
1.2.
--@order = 'first',
--@order = 'last',
--@order = 'none',
2nd parameter is the running order.
Value can be First, Last or None.
1.3.
--  @stmttype = 'CREATE_TABLE',
3rd parameter is the SQL statement that fires the trigger.
You many only put ONE statement type here.
If you want 'ALTER_TABLE' or 'DROP_TABLE' or other statement,
you need to "ECECUTE sp_settriggerorder" several times
for each statement type you want to set.
1.4.
-- @namespace = 'DATABASE';
or
-- @namespace = 'SERVER';
4th parameter is the scope of the trigger.
Value can be DATABASE, SERVER, or NULL.
*/
----------------------------------------------------------------------
--T029_03_01_06
--Drop the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
   BEGIN
       TRUNCATE TABLE dbo.TestTable;
       DROP TABLE TestTable;
   END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
Output as the following
--This is the second CREATE_TABLE trigger
--This is the first CREATE_TABLE trigger
*/
```

Messages

```
This is the second CREATE_TABLE trigger
This is the first CREATE_TABLE trigger
```

```
----------------------------------------------------------------------
--T029_03_01_07
--Clean up
IF EXISTS ( SELECT  *
            FROM    sys.triggers
```

```
            WHERE   name = 'trgFirstTrigger' )
    BEGIN
        DROP TRIGGER trgFirstTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgSecondTrigger' )
    BEGIN
        DROP TRIGGER trgSecondTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
----------------------------------------------------------------------
--T029_03_01_03
--sp_settriggerorder
EXEC sp_settriggerorder @triggername = 'trgFirstTrigger', @order = 'first',
       --@order = 'last',
       --@order = 'none',
    @stmttype = 'CREATE_TABLE', @namespace = 'DATABASE';
       --@namespace = 'SERVER';
/*
1.
--EXEC sp_settriggerorder
--     @triggername = 'trgFirstTrigger',
--     @order = 'first',
--  @stmttype = 'CREATE_TABLE',
--     @namespace = 'DATABASE';
Set the database scoped trigger, trgFirstTrigger,
to the first order of CREATE_TABLE event.
1.1.
--     @triggername = 'trgFirstTrigger',
1st parameter is the trigger name that you want to set.
1.2.
--@order = 'first',
--@order = 'last',
--@order = 'none',
2nd parameter is the running order.
Value can be First, Last or None.
1.3.
--  @stmttype = 'CREATE_TABLE',
3rd parameter is the SQL statement that fires the trigger.
You many only put ONE statement type here.
If you want 'ALTER_TABLE' or 'DROP_TABLE' or other statement,
you need to "ECECUTE sp_settriggerorder" several times
for each statement type you want to set.
1.4.
-- @namespace = 'DATABASE';
or
-- @namespace = 'SERVER';
4th parameter is the scope of the trigger.
Value can be DATABASE, SERVER, or NULL.
*/
----------------------------------------------------------------------
```

```sql
--T029_03_01_04
--Test the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
--This is the first CREATE_TABLE trigger
--This is the second CREATE_TABLE trigger
*/
----------------------------------------------------------------------
--T029_03_01_05
--sp_settriggerorder
--When @order = 'none', trigger is fired in random order
EXEC sp_settriggerorder @triggername = 'trgFirstTrigger',
       --@order = 'first',
       --@order = 'last',
    @order = 'none', -- Clean up
    @stmttype = 'CREATE_TABLE', @namespace = 'DATABASE';
       --@namespace = 'SERVER';
/*
1.
--EXEC sp_settriggerorder
--     @triggername = 'trgFirstTrigger',
--     @order = 'none',
--  @stmttype = 'CREATE_TABLE',
--     @namespace = 'DATABASE';
Set the database scoped trigger, trgFirstTrigger,
to the none order of CREATE_TABLE event.
When @order = 'none', trigger is fired in random order.
1.1.
--     @triggername = 'trgFirstTrigger',
1st parameter is the trigger name that you want to set.
1.2.
--@order = 'first',
--@order = 'last',
--@order = 'none',
2nd parameter is the running order.
Value can be First, Last or None.
1.3.
--  @stmttype = 'CREATE_TABLE',
3rd parameter is the SQL statement that fires the trigger.
You many only put ONE statement type here.
If you want 'ALTER_TABLE' or 'DROP_TABLE' or other statement,
you need to "ECECUTE sp_settriggerorder" several times
for each statement type you want to set.
1.4.
-- @namespace = 'DATABASE';
or
-- @namespace = 'SERVER';
4th parameter is the scope of the trigger.
Value can be DATABASE, SERVER, or NULL.
*/
----------------------------------------------------------------------
--T029_03_01_06
--Drop the trigger
IF ( EXISTS ( SELECT    *
```

```
            FROM      INFORMATION_SCHEMA.TABLES
            WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
Output as the following
--This is the second CREATE_TABLE trigger
--This is the first CREATE_TABLE trigger
*/
-----------------------------------------------------------------------
--T029_03_01_07
--Clean up
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgFirstTrigger' )
    BEGIN
        DROP TRIGGER trgFirstTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgSecondTrigger' )
    BEGIN
        DROP TRIGGER trgSecondTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
            FROM      INFORMATION_SCHEMA.TABLES
            WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
```

3.2. sp_settriggerorder order

```
--==========================================================================
--T029_03_02
--sp_settriggerorder order
/*
1.
Even if we use sp_settriggerorder to change the order,
server scoped triggers will always fire before any of the database scoped triggers.
2.
sp_settriggerorder can set the order of server scoped DDL triggers
or database scoped triggers.
3.
If there is a database scoped DDL Triggers and a server scoped DDL triggers
handling the same event.
Here is the execution order.
The server scope DDL trigger which @order = 'first'
-->
```

```
other server scope DDL triggers which @order = 'none'
-->
The server scope DDL trigger which @order = 'Last'
-->
The database scope DDL trigger which @order = 'first'
-->
other database scope DDL triggers which @order = 'none'
-->
The database scope DDL trigger which @order = 'Last'
*/
----------------------------------------------------------------------
--T029_03_02_01
--Create Database Scope Create_Table Trigger
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgDatabaseScopeCreateTableTrigger' )
    BEGIN
        DROP TRIGGER trgDatabaseScopeCreateTableTrigger ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgDatabaseScopeCreateTableTrigger ON DATABASE
    FOR CREATE_TABLE
AS
    BEGIN
        PRINT 'Database Scope CREATE_TABLE Trigger';
    END;
GO -- Run the previous command and begins new batch

----------------------------------------------------------------------
--T029_03_02_02
--Create Server Scope Create_Table Trigger
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgServerScopeCreateTableTrigger' )
    BEGIN
        DROP TRIGGER trgServerScopeCreateTableTrigger ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgServerScopeCreateTableTrigger ON ALL SERVER
    FOR CREATE_TABLE
AS
    BEGIN
        PRINT 'Server Scope CREATE_TABLE Trigger';
    END;
GO -- Run the previous command and begins new batch

----------------------------------------------------------------------
--T029_03_02_03
--Test the trigger
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
```

```sql
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO -- Run the previous command and begins new batch
/*
Output as the following
--Server Scope CREATE_TABLE Trigger
--Database Scope CREATE_TABLE Trigger
*/
```

Messages

```
Server Scope CREATE_TABLE Trigger
Database Scope CREATE_TABLE Trigger
```

```sql
---------------------------------------------------------------------
--T029_03_02_04
--sp_settriggerorder order
EXEC sp_settriggerorder @triggername = 'trgDatabaseScopeCreateTableTrigger',
    @order = 'first', @stmttype = 'CREATE_TABLE', @namespace = 'DATABASE';
/*
1.
Even if we use sp_settriggerorder to change the order,
server scoped triggers will always fire before any of the database scoped triggers.
2.
sp_settriggerorder can set the order of server scoped DDL triggers
or database scoped triggers .
*/

---------------------------------------------------------------------
--T029_03_02_05
--Test the trigger
--If Table exists then DROP it
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'TestTable' ) )
   BEGIN
       TRUNCATE TABLE dbo.TestTable;
       DROP TABLE TestTable;
   END;
GO -- Run the previous command and begins new batch
--Create a simple test table
CREATE TABLE TestTable ( ID INT );
GO
/*
Output as the following
--Server Scope CREATE_TABLE Trigger
--Database Scope CREATE_TABLE Trigger
*/
```

Messages

```
Server Scope CREATE_TABLE Trigger
Database Scope CREATE_TABLE Trigger
```

```sql
---------------------------------------------------------------------
--T029_03_02_06
--Clean up
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgDatabaseScopeCreateTableTrigger' )
   BEGIN
       DROP TRIGGER trgDatabaseScopeCreateTableTrigger ON DATABASE;
```

```
    END;
GO -- Run the previous command and begins new batch
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgServerScopeCreateTableTrigger' )
    BEGIN
        DROP TRIGGER trgServerScopeCreateTableTrigger ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
```

4. AuditTableChanges

```
--==========================================================================
--T029_04_AuditTableChanges
--==========================================================================
```

4.1. EVENTDATA

```
--=========================================================================
--T029_04_01
--EVENTDATA()
-----------------------------------------------
--T029_04_01_01
--EVENTDATA()
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgAuditTableStructureChanges' )
    BEGIN
        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgAuditTableStructureChanges ON ALL SERVER
    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
    BEGIN
        SELECT  EVENTDATA();
    END;
GO -- Run the previous command and begins new batch
/*
--EVENTDATA()
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/eventdata-transact-sql
EventData() returns information about server or database events in XML format.
E.g.
--<EVENT_INSTANCE>
--   <EventType>CREATE_TABLE</EventType>
--   <PostTime>2017-10-10T04:42:27.870</PostTime>
```

```
--  <SPID>54</SPID>
--  <ServerName>N550JKL\SQL2016</ServerName>
--  <LoginName>MicrosoftAccount\lpmplpmp01@hotmail.com</LoginName>
--  <UserName>dbo</UserName>
--  <DatabaseName>Sample3</DatabaseName>
--  <SchemaName>dbo</SchemaName>
--  <ObjectName>TestTable</ObjectName>
--  <ObjectType>TABLE</ObjectType>
--  <TSQLCommand>
--    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON"
ENCRYPTED="FALSE" />
--    <CommandText>CREATE TABLE TestTable
--    (
--      ID INT ,
--      [Name] NVARCHAR(100)
--    )</CommandText>
--  </TSQLCommand>
--</EVENT_INSTANCE>
*/
-----------------------------------------------
--T029_04_01_02
--Test EVENTDATA()
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
   BEGIN
       TRUNCATE TABLE dbo.TestTable;
       DROP TABLE TestTable;
   END;
GO -- Run the previous command and begins new batch
CREATE TABLE TestTable
   (
     ID INT ,
     [Name] NVARCHAR(100)
   );
GO -- Run the previous command and begins new batch
```

	(No column name)
1	<EVENT_INSTANCE><EventType>CREATE_TABLE</EventTy...

```
xmlresult2.xml   T029_DDLTriggers_A...550JKL\lpmpl (56))
1  <EVENT_INSTANCE>
2    <EventType>CREATE_TABLE</EventType>
3    <PostTime>2017-11-17T10:43:23.577</PostTime>
4    <SPID>56</SPID>
5    <ServerName>N550JKL\SQL2016</ServerName>
6    <LoginName>MicrosoftAccount\lpmplpmp01@hotmail.com</LoginName>
7    <UserName>dbo</UserName>
8    <DatabaseName>Sample</DatabaseName>
9    <SchemaName>dbo</SchemaName>
10   <ObjectName>TestTable</ObjectName>
11   <ObjectType>TABLE</ObjectType>
12   <TSQLCommand>
13     <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON" ENCRYPTED="FALSE" />
14     <CommandText>CREATE TABLE TestTable
15     (
16       ID INT ,
17       [Name] NVARCHAR(100)
18     )</CommandText>
19   </TSQLCommand>
20 </EVENT_INSTANCE>
```

```
/*
Output
--<EVENT_INSTANCE>
--  <EventType>CREATE_TABLE</EventType>
--  <PostTime>2017-10-10T08:01:28.670</PostTime>
--  <SPID>54</SPID>
--  <ServerName>N550JKL\SQL2016</ServerName>
--  <LoginName>MicrosoftAccount\lpmplpmp01@hotmail.com</LoginName>
```

```
--  <UserName>dbo</UserName>
--  <DatabaseName>Sample3</DatabaseName>
--  <SchemaName>dbo</SchemaName>
--  <ObjectName>TestTable</ObjectName>
--  <ObjectType>TABLE</ObjectType>
--  <TSQLCommand>
--    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON" ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON"
ENCRYPTED="FALSE" />
--    <CommandText>CREATE TABLE TestTable
--    (
--      ID INT ,
--      [Name] NVARCHAR(100)
--    )</CommandText>
--  </TSQLCommand>
--</EVENT_INSTANCE>
*/
```

4.2. AuditTableStructureChanges

```
--==========================================================================
--T029_04_02
--AuditTableStructureChanges
-----------------------------------------------
--T029_04_02_01
--Create AuditTableStructureChanges
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'AuditTableStructureChanges' ) )
   BEGIN
       TRUNCATE TABLE dbo.AuditTableStructureChanges;
       DROP TABLE AuditTableStructureChanges;
   END;
GO -- Run the previous command and begins new batch
CREATE TABLE AuditTableStructureChanges
   (
      EventType NVARCHAR(300) ,
      PostTime DATETIME ,
        --Server Process ID
      SPID NVARCHAR(300) ,
      ServerName NVARCHAR(300) ,
      LoginName NVARCHAR(300) ,
      UserName NVARCHAR(300) ,
      DatabaseName NVARCHAR(300) ,
      SchemaName NVARCHAR(300) ,
      ObjectName NVARCHAR(300) ,
      ObjectType NVARCHAR(300) ,
      TSQLCommand NVARCHAR(MAX)
   );
GO -- Run the previous command and begins new batch
```

	(No column name)
1	<EVENT_INSTANCE><EventType>CREATE TABLE</EventTy...

```
-----------------------------------------------
--T029_04_02_02
--Alter trgAuditTableStructureChanges
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
```

```
            WHERE   name = 'trgAuditTableStructureChanges' )
    BEGIN
        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgAuditTableStructureChanges ON ALL SERVER
    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
    BEGIN
        DECLARE @EventData XML;
        SELECT  @EventData = EVENTDATA();
        INSERT  INTO [Sample].dbo.AuditTableStructureChanges
                ( EventType ,
                  PostTime ,
                  SPID ,
                  ServerName ,
                  LoginName ,
                  UserName ,
                  DatabaseName ,
                  SchemaName ,
                  ObjectName ,
                  ObjectType ,
                  TSQLCommand
                )
        VALUES  ( @EventData.value('(/EVENT_INSTANCE/EventType)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/PostTime)[1]', 'DATETIME') ,
                  @EventData.value('(/EVENT_INSTANCE/SPID)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/ServerName)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/LoginName)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/UserName)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/DatabaseName)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/SchemaName)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/ObjectType)[1]',
                                   'NVARCHAR(300)') ,
                  @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
                                   'NVARCHAR(MAX)')
                );
    END;
GO -- Run the previous command and begins new batch
-----------------------------------------------
--T029_04_02_03
--Create TestTable to test trgAuditTableStructureChanges
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
```

```
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE TestTable
    (
      ID INT ,
      [Name] NVARCHAR(100)
    );
GO -- Run the previous command and begins new batch
SELECT  *
FROM    dbo.AuditTableStructureChanges;
GO -- Run the previous command and begins new batch
```

	EventType	PostTime	SPID	ServerName	LoginName	UserName	DatabaseName	SchemaName	ObjectName	ObjectType	TSQLCommand
1	DROP_TABLE	2017-11-17 10:45:04.230	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	DROP TABLE TestTable;
2	CREATE_TABLE	2017-11-17 10:45:04.290	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	CREATE TABLE TestTable (ID IN...

```
-----------------------------------------------
--T029_04_02_04
--Alter TestTable to test trgAuditTableStructureChanges
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        ALTER TABLE TestTable
        ALTER COLUMN [Name] NVARCHAR(150);
    END;
GO -- Run the previous command and begins new batch
SELECT  *
FROM    dbo.AuditTableStructureChanges;
GO -- Run the previous command and begins new batch
```

	EventType	PostTime	SPID	ServerName	LoginName	UserName	DatabaseName	SchemaName	ObjectName	ObjectType	TSQLCommand
1	DROP_TABLE	2017-11-17 10:45:04.230	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	DROP TABLE TestTable;
2	CREATE_TABLE	2017-11-17 10:45:04.290	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	CREATE TABLE TestTable (ID INT , [Nam...
3	ALTER_TABLE	2017-11-17 10:45:30.940	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	ALTER TABLE TestTable ALTER COLUMN [Nam...

```
-----------------------------------------------
--T029_04_02_05
--Drop TestTable to test trgAuditTableStructureChanges
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
SELECT  *
FROM    dbo.AuditTableStructureChanges;
GO -- Run the previous command and begins new batch
```

	EventType	PostTime	SPID	ServerName	LoginName	UserName	DatabaseName	SchemaName	ObjectName	ObjectType	TSQLCommand
1	DROP_TABLE	2017-11-17 10:45:04.230	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	DROP TABLE TestTable;
2	CREATE_TABLE	2017-11-17 10:45:04.290	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	CREATE TABLE TestTable (ID INT , [Nam...
3	ALTER_TABLE	2017-11-17 10:45:30.940	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	ALTER TABLE TestTable ALTER COLUMN [Nam...
4	DROP_TABLE	2017-11-17 10:45:46.310	56	N550JKL\SQL2016	MicrosoftAccount\lpmplpmp01@hotmail.com	dbo	Sample	dbo	TestTable	TABLE	DROP TABLE TestTable;

4.3. Clean up

```
--==========================================================================
--T029_04_03
--Clean up
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
```

```
            WHERE   name = 'trgAuditTableStructureChanges' )
    BEGIN
        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'TestTable' ) )
    BEGIN
        TRUNCATE TABLE dbo.TestTable;
        DROP TABLE TestTable;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'AuditTableStructureChanges' ) )
    BEGIN
        TRUNCATE TABLE dbo.AuditTableStructureChanges;
        DROP TABLE AuditTableStructureChanges;
    END;
GO -- Run the previous command and begins new batch
```

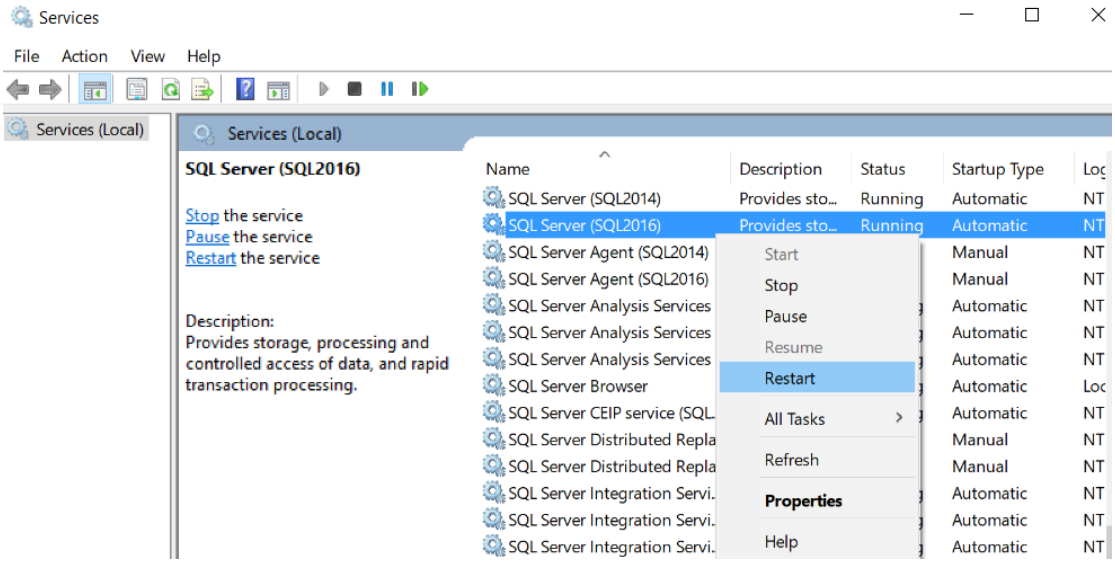
==========

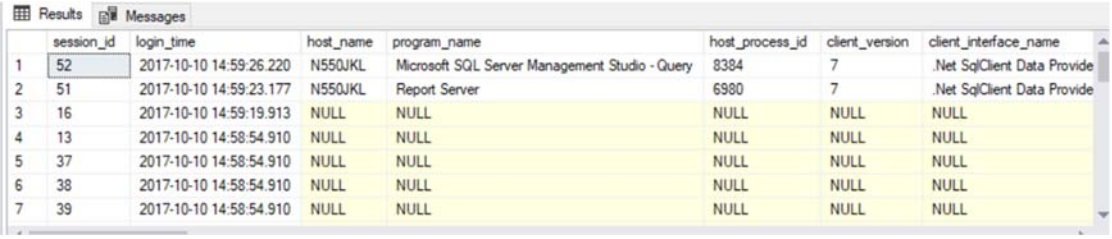
5. LogonTriggers_sys.dm_exec_sessions_ORIGINAL_LOGIN()

```
--==========================================================================
--T029_05_LogonTriggers_sys.dm_exec_sessions_ORIGINAL_LOGIN()
--==========================================================================
--========================================================================
--T029_05_01
--sys.dm_exec_sessions and ORIGINAL_LOGIN()
SELECT  ORIGINAL_LOGIN();
GO -- Run the previous command and begins new batch
```

	(No column name)
1	MicrosoftAccount\pmplpmp01@hotmail.com

```
/*
ORIGINAL_LOGIN()
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/original-login-transact-sql
Returns the name of the login that connected to the instance of SQL Server.
Output
--MicrosoftAccount\XXXXXX@hotmail.com
*/
--========================================================================
--T029_05_02
SELECT  *
FROM    sys.dm_exec_sessions
ORDER BY login_time DESC;
GO -- Run the previous command and begins new batch
```





Results | Messages

	session_id	login_time	host_name	program_name	host_process_id	client_version	client_interface_name
1	52	2017-10-10 14:59:26.220	N550JKL	Microsoft SQL Server Management Studio - Query	8384	7	.Net SqlClient Data Provide
2	51	2017-10-10 14:59:23.177	N550JKL	Report Server	6980	7	.Net SqlClient Data Provide
3	16	2017-10-10 14:59:19.913	NULL	NULL	NULL	NULL	NULL
4	13	2017-10-10 14:58:54.910	NULL	NULL	NULL	NULL	NULL
5	37	2017-10-10 14:58:54.910	NULL	NULL	NULL	NULL	NULL
6	38	2017-10-10 14:58:54.910	NULL	NULL	NULL	NULL	NULL
7	39	2017-10-10 14:58:54.910	NULL	NULL	NULL	NULL	NULL

```
/*
1.
Go to Service Manager and restart the SQL server service
2.
--sys.dm_exec_sessions
Reference:
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-sessions-transact-sql
sys.dm_exec_sessions returns one row
per authenticated session on SQL Server.
*/
--==========================================================================
--T029_05_03
--sys.dm_exec_sessions
SELECT  is_user_process ,
        original_login_name ,
        session_id
FROM    sys.dm_exec_sessions
WHERE   is_user_process = 1
        AND original_login_name = ORIGINAL_LOGIN()
ORDER BY login_time DESC;
GO -- Run the previous command and begins new batch
--Return all the sessions which is using by current user.
SELECT  COUNT(*)
FROM    sys.dm_exec_sessions
WHERE   is_user_process = 1
        AND original_login_name = ORIGINAL_LOGIN();
GO -- Run the previous command and begins new batch
--Return the number of the sessions which is using by current user.
```

Results | Messages

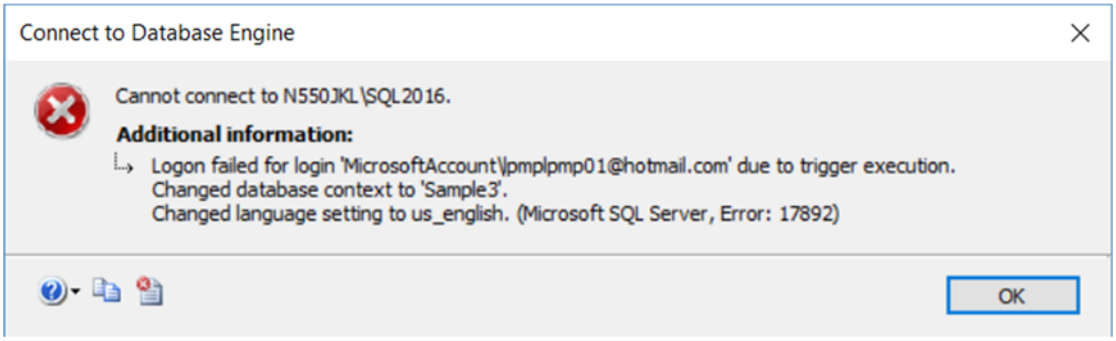
	is_user_process	original_login_name	session_id
1	1	MicrosoftAccount\lpmplpmp01@hotmail.com	52

	(No column name)
1	1

N550JKL\SQL2016 (13.0 SP1) | N550JKL\lpmpl (52) | Sample3 | 00:00:00 | 2 rows

```
/*
Go to Service Manager and restart the SQL server service
1.
ORIGINAL_LOGIN()
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/original-login-transact-sql
Returns the name of the login that connected to the instance of SQL Server.
Output
--MicrosoftAccount\XXXXXX@hotmail.com
2.
--sys.dm_exec_sessions
sys.dm_exec_sessions returns one row
per authenticated session on SQL Server.
2.1.
Go to Service Manager and restart the SQL server service
2.2.
E.g.
--SELECT  is_user_process ,
--        original_login_name ,
--           session_id
--FROM    sys.dm_exec_sessions
--WHERE   is_user_process = 1
--        AND original_login_name = ORIGINAL_LOGIN()
--ORDER BY login_time DESC;
Return all the sessions which is using by current user.
2.3.
--SELECT  COUNT(*)
--FROM    sys.dm_exec_sessions
--WHERE   is_user_process = 1
--        AND original_login_name = ORIGINAL_LOGIN()
Return the number of the sessions which is using by current user.
2.4.
Columns in sys.dm_exec_sessions
2.4.1.
--sys.dm_exec_sessions
Reference:
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-
sessions-transact-sql
sys.dm_exec_sessions returns one row
per authenticated session on SQL Server.
2.4.2.
--is_user_process=1
0 if the session is a system session.
Otherwise, it is 1. Is not nullable.
2.4.3.
--original_login_name
SQL Server login name that the client used to create this session.
2.4.4.
--session_id
The current connection ID,
If using SSMS, you may find the session on the bottom bar.
*/
--==========================================================================
--T029_05_04
--sys.dm_exec_sessions
--Drop ALL SERVER scope trigger if it exists
```

```
--Reference: http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgLogonAuditTriggers' )
    BEGIN
        DROP TRIGGER trgLogonAuditTriggers ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgLogonAuditTriggers ON ALL SERVER
    FOR LOGON
AS
    BEGIN
        DECLARE @LoginName NVARCHAR(100);
        SET @LoginName = ORIGINAL_LOGIN();
        IF ( SELECT COUNT(*)
             FROM   sys.dm_exec_sessions
             WHERE  is_user_process = 1
                    AND original_login_name = @LoginName
           ) > 3
            BEGIN
                PRINT 'LoginName : ' + @LoginName + '. The connections can not more than 3';
                ROLLBACK;
            END;
    END;
GO -- Run the previous command and begins new batch
```




```
/*
Go to Service Manager and restart the SQL server service
If there are more than 3 sessions/connections by current user,
then we block it by rollback.
*/
--==========================================================================
--T029_05_05
--The system store procedure to read the Error Log.
Execute sp_readerrorlog 0, 1, 'The connections can not more than 3'
```

Results | Messages

	LogDate	ProcessInfo	Text
1	2017-10-10 15:05:07.450	spid54	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
2	2017-10-10 15:05:07.460	spid54	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
3	2017-10-10 15:05:07.470	spid54	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
4	2017-10-10 15:05:24.790	spid55	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
5	2017-10-10 15:05:24.800	spid55	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
6	2017-10-10 15:05:24.800	spid55	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
7	2017-10-10 15:05:24.810	spid55	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3
8	2017-10-10 15:05:24.840	spid55	LoginName : MicrosoftAccount\lpmplpmp01@hotmail.com. The connections can not more than 3

```
/*
1.
--sp_readerrorlog
1.1.
Reference:
https://www.mssqltips.com/sqlservertip/1476/reading-the-sql-server-log-files-using-tsql/
```

```
1.1.1.
1st parameter: The session Number.
Value of error log file you want to read:
0 = current, 1 = Archive #1, 2 = Archive #2, etc...
1.1.2.
2nd parameter: Log File Type:
1 or NULL = error log, 2 = SQL Agent log
1.1.3.
3rd parameter:
Search string 1:
String one you want to search for
1.1.4.
4th parameter:
Search string 2:
String two you want to search for to further refine the results
1.2.
E.g.
Execute sp_readerrorlog 0, 1, 'The connections can not more than 3'
Searching the Text 'The connections can not more than 3'
in the Error log in Current log file
*/
--==========================================================================
--T029_05_06
--Clean up
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgLogonAuditTriggers' )
    BEGIN
        DROP TRIGGER trgLogonAuditTriggers ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
```