**Execute As**

**By [Patrick Cahill](http://www.sqlservercentral.com/Authors/Articles/Patrick_Cahill/293201/), 2013/10/11 (first published: 2011/02/02)**

"Execute As" is a nice solution for testing user permissions or controlling security within stored procedures while still allowing execute permission to the users that need the item. In this article I am going to use an example involving data encryption. I will create an application user, and will only allow the user to decrypt the encrypted data through a stored procedure. The user will be able to select from the table, but will not be able to decrypt the data using a select statement.

Let's start by creating an application user and an encryption user. The application user will have execute and select permissions. The encryption user will only have permission to encrypt and dencrypt the data, but no execute and select permissions.

The following script will create a database called MyTesting. It will then create two users, EncryptionUser and ApplicationUser. Next it will create the master key, certificate and symmetric key in the MyTesting database. It will create the table EncryptionTest and two stored procedures that simply select from the table while decrypting the encrypted column. The first stored procedure getEncryption will not use "Execute As" while the second stored procedure getEncryptionWithExecute will use "Execute As". Last it will populate the table.

USE [master]

GO

--Create a database called TestDB

IF EXISTS (SELECT name FROM sys.databases WHERE name = N'TestDB')

BEGIN

ALTER DATABASE [TestDB] SET RESTRICTED\_USER WITH ROLLBACK IMMEDIATE

DROP DATABASE [TestDB]

END

GO

CREATE DATABASE [TestDB]

--Create two sql users

IF NOT EXISTS (SELECT \* FROM sys.server\_principals WHERE name = N'EncryptionUser')

CREATE LOGIN [EncryptionUser] WITH PASSWORD=N'123456abc', DEFAULT\_DATABASE=[master], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF;

GO

IF NOT EXISTS (SELECT \* FROM sys.server\_principals WHERE name = N'ApplicationUser')

CREATE LOGIN [ApplicationUser] WITH PASSWORD=N'123456xyz', DEFAULT\_DATABASE=[master], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF;

GO

USE [TestDB]

GO

--Create Users

CREATE USER [EncryptionUser] FOR LOGIN [EncryptionUser]

GO

CREATE USER [ApplicationUser] FOR LOGIN [ApplicationUser]

GO

--Create Keys and certificates for encryption

CREATE MASTER KEY Encryption By Password = '7B35E6CF-16F4-4A9A-A8AB-7B150BE531EA';

GO

CREATE CERTIFICATE CertTest

WITH SUBJECT = 'Test Cert' ;

GO

CREATE SYMMETRIC KEY SymKeyTest

WITH ALGORITHM = AES\_256

ENCRYPTION BY CERTIFICATE CertTest ;

GO

--Grant Permissions

GRANT CONTROL ON CERTIFICATE::CertTest TO EncryptionUser;

GO

GRANT VIEW DEFINITION ON SYMMETRIC KEY::SymKeyTest TO EncryptionUser;

GO

GRANT EXECUTE ON SCHEMA::[dbo] TO [ApplicationUser]

GO

GRANT SELECT ON SCHEMA::[dbo] TO [ApplicationUser]

GO

--Create Table

CREATE TABLE [dbo].[EncryptionTest](

[EncryptionTestId] [int] IDENTITY(1,1) NOT NULL,

[EncryptionNumber] [varbinary](100) NOT NULL,

[PlainTextNumber] [int] NOT NULL)

GO

--Create Test procedures

CREATE PROCEDURE dbo.getEncryption

@EncryptionTestId INT

AS

BEGIN

SELECT EncryptionTestId

,PlainTextNumber

,CONVERT(VARCHAR(10),DecryptByKeyAutoCert(cert\_id('CertTest'),NULL,EncryptionNumber)) DecryptedNumber

FROM dbo.EncryptionTest

WHERE EncryptionTestId = @EncryptionTestId

END

GO

CREATE PROCEDURE dbo.getEncryptionWithExecute

@EncryptionTestId INT

WITH EXECUTE AS 'EncryptionUser'

AS

BEGIN

SELECT EncryptionTestId

,PlainTextNumber

,CONVERT(VARCHAR(10),DecryptByKeyAutoCert(cert\_id('CertTest'),NULL,EncryptionNumber)) DecryptedNumber

FROM dbo.EncryptionTest

WHERE EncryptionTestId = @EncryptionTestId

END

GO

--Populate table

OPEN SYMMETRIC KEY SymKeyTest

DECRYPTION BY CERTIFICATE CertTest ;

DECLARE @Encrypt VARCHAR(10) ;

DECLARE @i INT = 10

WHILE @i > 0

BEGIN

SELECT @Encrypt = ROUND(((99999) \* RAND()),0)

INSERT INTO dbo.EncryptionTest

(EncryptionNumber

,PlainTextNumber

)

VALUES (ENCRYPTBYKEY(KEY\_GUID('SymKeyTest'),@Encrypt)

,@Encrypt

)

SET @i = @i - 1

END

CLOSE SYMMETRIC KEY SymKeyTest ;

GO

Now we can run a test for each user to see how they act when selecting from the table or running the stored procedures. Here we will use the Execute As Login to test running everything as the EncryptionUser.

EXECUTE AS LOGIN = 'EncryptionUser'

EXEC dbo.getEncryption

EXEC dbo.getEncryptionWithExecute

SELECT \* FROM dbo.EncryptionTest

REVERT

Since this user has no select or execute permissions all three statements will fail. Here are the results.

Msg 229, Level 14, State 5, Procedure getEncryption, Line 1  
The EXECUTE permission was denied on the object 'getEncryption', database 'MyTesting', schema 'dbo'.  
Msg 229, Level 14, State 5, Procedure getEncryptionWithEXECUTE, Line 1  
The EXECUTE permission was denied on the object 'getEncryptionWithEXECUTE', database 'MyTesting', schema 'dbo'.  
Msg 229, Level 14, State 5, Line 5  
The SELECT permission was denied on the object 'EncryptionTest', database 'MyTesting', schema 'dbo'.

Now let's try the running the code with the Execute As Login for the ApplicationUser.

EXECUTE AS LOGIN = 'ApplicationUser'

SELECT EncryptionTestId

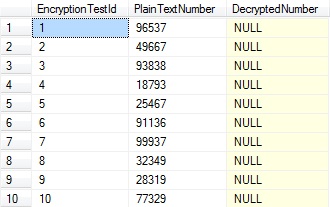
,PlainTextNumber

,CONVERT(VARCHAR(10),DecryptByKeyAutoCert(cert\_id('CertTest'),NULL,EncryptionNumber)) DecryptedNumber

FROM dbo.EncryptionTest

REVERT

The Application user has permissions to run the select query, but it does not have permissions to decrypt the encrypted column. This is why the results return a null value for the column.



Now when you execute the stored procedure getEncryption (this is the procedure without EXECUTE AS 'EncryptionUser') as the ApplicationUser, it runs without error, but it is unable to decrypt the encrypted number.

EXECUTE AS LOGIN = 'ApplicationUser'

EXEC dbo.getEncryption @EncryptionTestId = 3

REVERT

**Results**

http://www.sqlservercentral.com/Images/7763.jpg

When ApplicationUser runs the stored procedure getEncryptionWithExecute (this is the procedure with EXECUTE AS 'EncryptionUser'), it is able to decrypt the encrypted number.

EXECUTE AS LOGIN = 'ApplicationUser'

EXEC dbo.getEncryptionWithExecute @EncryptionTestId = 3

REVERT

**Results**

http://www.sqlservercentral.com/Images/7764.jpg

By using the "execute as" I am better able to control which users have access to the encrypted data, and how they have access to this data. "Execute As" also gives me the ability to test running a stored procedure or select statement as a different user, without having to login as that user.