

TDE

Last updated by | Soma Jagadeesh | Jan 10, 2021 at 9:46 PM PST

Contents

- [Issue](#)
- [Investigation/Analysis](#)
- [Mitigation](#)
- [RCA \(optional\)](#)
- [More Information \(optional\)](#)
- [Public Doc Reference \(optional\)](#)
- [Internal Reference \(optional\)](#)
- [Classification](#)

Issue

Transparent Data Encryption (TDE)

Investigation/Analysis

Describes the steps/queries to use for confirming the issue

Mitigation

What to do to resolve the issue. Maybe file an ICM, might have a canned RCA template to be used, etc.

RCA (optional)

More Information (optional)

Monitor encryption state

MonDatabaseEncryptionKeys | where AppName contains "b98c77aff304" and logical_database_id == "6F020107-90FB-4C42-80B0-01929FE022E0" | sort by TIMESTAMP asc | project TIMESTAMP, end_utc_date , start_utc_date , is_encrypted, encryption_state

Azure SQL Database TDE

Update June 2016 Since May 16, all newly created SQL Databases are encrypted at rest by default with Transparent Data Encryption (TDE).

The effort to automatically encrypt all new databases at rest is part of a broader Azure-wide "Get Secure" initiative and is another step towards ensuring that customer data is secure and protected by default without extra configuration by the user.

The encryption by default update reached all clusters on May 16, 2017 and since then, approx. 100,000 databases have been encrypted with TDE by default. This means the # of databases encrypted with TDE has increased by ~25%. Because TDE can have a performance impact for read/write-heavy analytics workloads, we allow customers to opt out of TDE once the database has been created. Even then, more than 99% of databases have kept TDE on after creation time.

Update May 2015 Feature is in Public Preview now. Public information can be found [here](#).


Summary Azure SQL DB is introducing optional encryption-at-rest for all database files using the Transparent Data Encryption (TDE) feature from SQL Server. This provides additional protection for databases files, transaction logs and backups beyond the existing physical and operational security controls already in place. The initial release focuses on zero-admin management capabilities by providing automatic management of keys and certificates used in encryption. Private Preview Details For entry into the Private Preview, please email azsqltde@microsoft.com. Allow listing is done at the server level, so you'll need a list of servers you'd like allow listed. Servers must already be at V12 to be allow listed. If you need any help at all send an email. Current State

- T-SQL API Only - No portal/rest API – Will be released with Public Preview
- No Geo-DR / Geo-Restore support – some other Private Preview features may also not be supported – full feature support before Public Preview - ~Late Feb
- Backups can be delayed – Planned for ~Late Feb

Availability Encryption at rest will be available for new Basic, Standard, and Premium databases. It is only available for servers upgraded to V12. Additional notes

- Encryption is per database, but the allow listing for TDE happens per server, which contains or will contain your database. It must be a V12 server.
- In the future all databases may be encrypted by default, for this reason you should write any deployment scripts with no assumptions as to the state of the database (good practice in general). An example deployment script is shown below. How it works To encrypt your database, you just run the command: (note this command may change, but it will only be cosmetic)

```
-- From Database to be encrypted CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256
ENCPTION BY SERVER CERTIFICATE ##MS_TdeCertificate## ALTER DATABASE Foo SET ENCRYPTION ON
```

You can monitor your encryption status with the normal DMV. <http://msdn.microsoft.com/en-us/library/bb677274.aspx> 

```
SELECT * FROM sys.dm_database_encryption_keys
```

You'll be able see the progress on long encryption scans and the current thumbprint (allowing you to detect when we rotate it, if you need to report on this for compliance purposes, rotation is entirely transparent to you otherwise)

If for some reason you need to unencrypt your database you'll be able to run:

```
ALTER DATABASE Foo SET ENCRYPTION OFF
```

Notes: any backups taken before you enable Encryption at Rest will remain unencrypted and if you restore that unencrypted database, it will remain unencrypted until you choose to encrypt it.

Example Scripts Deployment If you want to be sure your database is encrypted/decrypted at any given point, most likely during deployment, you'll want to check the current state before attempting to change it. This is good practice for cloud environments in general where you may have multiple actors trying to change a database and default configurations change over time. The pattern for this with TDE is:

1. Database is online.
2. Get the current state (none, encrypting, encrypted, decrypting, decrypted)

3. If the state matches the desired state, continue deployment
4. If the state is encrypting/decrypting, wait or continue //(see comment below)
5. //You can make a choice here based on your desires. If a database is encrypting, it isn't entirely encrypted meaning some data will be exposed in backups/files/etc. One solution would be to let your database continue set up but before you exit deployment and start the database in production, you check to see that encryption has finished.
6. Alter database to the desired state
7. //The same principles that affected 2.b are presented here as well. You don't necessarily need to block deployment on encryption (it's an online operation that only has some restrictions), but before you put sensitive data on the database, it's a good idea to let it finish encrypting. Below is a code sample of how you could set it up with just T-SQL. Ideally you'd return the result to your deployment logic and make a decision based off the advice I've given in the above pseudocode. It returns the current state and only attempts to change the state if it is different from the desired state.

```
-- # DROP PROCEDURE IF EXISTS
```

```
IF (OBJECT_ID('setEncryption') IS NOT NULL) BEGIN DROP PROCEDURE setEncryption END
```

```
GO -- /DROP PROCEDURE IF EXISTS
```

```
/* NAME: 'setEncryption' DESCRIPTION: 'Will look up the current state of the database and attempt to change it to the desired state' PARAMS: @database_name - 'Needed because Azure SQL DB doesn't support ALTER DATABASE CURRENT' @desire_encrypted - 'TRUE'/1 will attempt to change the database to encrypted, 'FALSE'/0 will attempt to change the database to decrypted' */ CREATE PROCEDURE setEncryption @database_name varchar(128), @desire_encrypted BIT AS DECLARE @state INT
```

```

-- get the current state of the database
SELECT @state = [encryption_state]
FROM [sys].[dm_database_encryption_keys]
WHERE
    [database_id] = (
        SELECT TOP 1 [database_id]
        FROM [sys].[databases]
        WHERE [name] = @database_name
    )

-- If there is no record, the database doesn't have a key present, so set the state = 0
SELECT @state = ISNULL(@state, 0)

IF (4 <= @state OR 2 = @state)
BEGIN
    PRINT 'STATE ' + FORMAT(@state, 'G')
    PRINT 'NO OP'
END

-- #Check @state = no Key (0)
IF (0 = @state)
BEGIN
    PRINT 'STATE ' + FORMAT(@state, 'G')
    -- #Create Key and encrypt if encryption desired
    IF ('TRUE' = @desire_encrypted)
    BEGIN
        PRINT 'CREATING KEY'
        CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256 ENCRYPTION BY SERVER CERTIFICATE #
        PRINT 'ATTEMPT ENCRYPTING'
        EXEC ('Alter Database [' + @database_name + '] set encryption on')
    END -- /Create Key if encryption desired
    ELSE
        PRINT 'NO CHANGE REQUIRED'
END -- /Check @state = no Key (0)

-- #Check @state = Encrypted (3)
IF (3 = @state)
BEGIN
    PRINT 'STATE ' + FORMAT(@state, 'G')
    -- #Turn off encryption, if no encryption is desired
    IF ('FALSE' = @desire_encrypted)
    BEGIN
        PRINT 'ATTEMPT DECRYPTING'
        EXEC ('Alter Database [' + @database_name + '] set encryption off')
    END -- /Turn off encryption, if no encryption is desired
    ELSE
        PRINT 'NO CHANGE REQUIRED'
END -- /Check @state = Encrypted (3)

-- #Check @state = Decrypted (1)
IF (1 = @state)
BEGIN
    PRINT 'STATE ' + FORMAT(@state, 'G')
    -- #Turn on Encryption, if encryption is desired
    IF ('TRUE' = @desire_encrypted)
    BEGIN
        PRINT 'ATTEMPT ENCRYPTING'
        EXEC ('Alter Database [' + @database_name + '] set encryption on')
    END -- /Turn on Encryption, if encryption is desired
    ELSE
        PRINT 'NO CHANGE REQUIRED'
END -- /Check @state = Decrypted (1)

RETURN (
    SELECT [encryption_state]
    FROM [sys].[dm_database_encryption_keys]
    WHERE [database_id] = (
        SELECT TOP 1 [database_id]

```

```
FROM [sys].[databases]
WHERE [name] = @database_name
)
)
```


```
GO -- /CREATE PROCEDURE
```

```
-- # RUN PROCEDURE DECLARE @Results INT EXEC @Results = setEncryption '<DB NAME GOES HERE>',
'FALSE' SELECT @Results as 'Results'
```

```
-- # Additional Info SELECT * FROM [sys].[dm_database_encryption_keys] SELECT * FROM [sys].[databases]
```

Errors to be aware of Certs not being available If you receive a message similar to: Msg 15151, Level 16, State 1, Line 1 Cannot find the certificate '##MS_TdeCertificate##', because it does not exist or you do not have permission. Send a message to azsqltde@microsoft.com. This means the server certificate for TDE doesn't exist and we'll need to perform a rollover. This is a quick operation and we're planning on doing a service wide rollover to prevent customers from hitting this issue, but in the event you hit it, the reason is a cert not provisioned. Alter statement fails due to backup being required If you receive a message similar to: Msg 33122, Level 16, State 1, Line 3 This command requires a database encryption scan on database 'TestSecurity'. However, the database has changes from previous encryption scans that are pending log backup. Take a log backup and retry the command. Msg 5069, Level 16, State 1, Line 3 ALTER DATABASE statement failed. Wait 5 minutes and try again. If it fails again, please send an email to azsqltde@microsoft.com. Other errors If you receive any other errors and aren't sure what they mean, do not hesitate to message azsqltde@microsoft.com; we'll respond as fast as possible to any errors and help you debug them.

Public Doc Reference (optional)

<https://docs.microsoft.com/en-us/azure/sql-database/transparent-data-encryption-azure-sql?redirectedfrom=MSDN&view=sql-server-ver15&tabs=azure-portal> 

Internal Reference (optional)

<https://microsoft.sharepoint.com/teams/bidpwiki/Pages1/Azure SQL DB TDE.aspx> 

Classification

Root cause tree: Security/User Request/How-to/advisory

How good have you found this content?

