User Creation Scripts On Premises

The following documentation allows you to create a user with the necessary permissions for SQL Diagnostic Manager, allowing the new user to gather diagnostic data from a monitored instance successfully.

Recommended permissions on-premises

Follow the next steps to create a new user with recommended permissions for SQL Diagnostic Manager to function.

1. Open the script with the source code editor of your preference.
2. Find and replace all the following instances:
    a. `'$SQLdmMonitorRole'` with the desired name of the custom server role.
    b. `'$username'` with the account name that SQLDM will use to collect diagnostic data. You may use a SQL account (e.g. `SQLdmService`) or a domain account (e.g. `DOMAIN\SQLdmService`).
    c. `'$password'` with the password related to the account. Only if you are using a SQL account.

    > Use the Find/Replace function to locate and replace all references from the script.

3. Execute the script against the target monitored instance using an account with sysadmin rights. This means you should execute the script on each instance you intend to monitor with the user created.

Below you can find the script for user creation on-premises, also you can download it by clicking here.

Create a user with recommended permission on-premises

```
USE master
IF NOT EXISTS(SELECT [name] FROM [sys].[server_principals] WHERE [type]='R' AND [name]='$SQLdmMonitorRole')
        CREATE SERVER ROLE [$SQLdmMonitorRole]
IF NOT EXISTS(SELECT [name] FROM [sys].[server_principals] WHERE ([type]=N'S' OR [type]=N'U') AND [name]
='$username')
        BEGIN TRY
                IF CHARINDEX('\','$username') <> 0
                        CREATE LOGIN [$username] FROM WINDOWS WITH DEFAULT_DATABASE=[master]
                ELSE
                        CREATE LOGIN [$username] WITH PASSWORD=N'$password', DEFAULT_DATABASE=[master],
DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=ON, CHECK_POLICY=ON
        END TRY
        BEGIN CATCH
                SELECT ERROR_NUMBER() AS ErrorNumber
                        , ERROR_SEVERITY() AS ErrorSeverity
                        , ERROR_STATE() AS ErrorState
                        , ERROR_PROCEDURE() AS ErrorProcedure
                        , ERROR_LINE() AS ErrorLine
                        , ERROR_MESSAGE() AS ErrorMessage;
                RETURN;
        END CATCH

IF NOT EXISTS(SELECT [name] FROM [sys].[database_principals] WHERE [type] = N'S' AND [name] ='$username')
                CREATE USER [$username] FOR LOGIN [$username]

ALTER SERVER ROLE [$SQLdmMonitorRole] ADD MEMBER [$username]

GRANT EXECUTE on xp_loginconfig to [$username]
GRANT EXECUTE on xp_regread to [$username]
GRANT EXECUTE on xp_readerrorlog to [$username]
GRANT EXECUTE on xp_instance_regread to [$username]
GRANT EXECUTE on sp_OACreate to [$username]

-- GRANT SERVER LEVEL PERMISSIONS --
GRANT VIEW ANY DEFINITION TO [$SQLdmMonitorRole]
GRANT VIEW ANY DATABASE TO [$SQLdmMonitorRole]
GRANT VIEW SERVER STATE TO [$SQLdmMonitorRole]
GRANT ALTER SETTINGS TO [$SQLdmMonitorRole]

-- GRANT ADDITIONAL XEVENTS AND TRACE PERMISSIONS --
GRANT ALTER ANY EVENT SESSION TO [$SQLdmMonitorRole]
GRANT ALTER TRACE TO [$SQLdmMonitorRole]
GRANT CREATE ANY DATABASE TO [$SQLdmMonitorRole]

/** The following script adds the user to the database with the required permissions**/

SET @cmd = '
    USE [?]
        IF ((SELECT is_read_only FROM sys.databases WHERE name = DB_NAME()) = 0)
                BEGIN
                        IF ((SELECT state FROM sys.databases WHERE name = DB_NAME())=0)
                                BEGIN
                                        IF NOT EXISTS(SELECT [name] FROM [sys].[database_principals] WHERE
([type] = N''S'' OR [type] = N''U'') AND [name]=''$username'')
                                                CREATE USER [$username] FOR LOGIN [$username]

                                        ALTER ROLE [db_owner] ADD MEMBER
[$username]

                                        IF (DB_NAME()=''msdb'')
                                                ALTER ROLE [SQLAgentOperatorRole] ADD MEMBER
[$username]

                                        IF (SELECT COUNT(*)FROM [sys].[database_mirroring] WHERE
[mirroring_guid] IS NOT NULL) > 0
                                                ALTER SERVER ROLE [dbm_monitor] ADD MEMBER
[$SQLdmMonitorRole]
                                END
                END'

EXEC sp_MSforeachdb @cmd
```

When running the script above, it is added:

- A new user (a Windows or a SQL Server account) to the SQL Server instance.
- A new SQL Server level role.
- The `db_owner` role to view Databases > Summary.
- The `SQLAgentOperatorRole` role to the `msdb` database. Which lets you view, start, or stop SQL Agent Jobs.
- The `dbm_monitor` role, allowing you to monitor the database mirroring. Only required if there are any databases participating in mirroring.

GRANT DATABASE PERMISSIONS

Restored databases onto the instance later must have the monitor account added as a bd_owner.