Diagnostic connection for database administrators

Article03/31/202512 contributors Feedback

In this article

Applies to: ✓ SQL Server ✓ Azure SQL Database ✓ Azure SQL Managed Instance

SQL Server provides a special diagnostic connection for administrators when standard connections to the server aren't possible. This diagnostic connection allows an administrator to access SQL Server to execute diagnostic queries and troubleshoot problems even when SQL Server isn't responding to standard connection requests.

This dedicated administrator connection (DAC) supports encryption and other security features of SQL Server. The DAC only allows changing the user context to another admin user.

SQL Server makes every attempt to make DAC connect successfully, but under extreme situations it might not be successful.

Connect with DAC

By default, the connection is only allowed from a client running on the server. Network connections aren't permitted unless they're configured by using the `sp_configure` stored procedure with the remote admin connections option.

Only members of the SQL Server sysadmin role can connect using the DAC.

The DAC is available and supported through the `sqlcmd` command-prompt utility using a special administrator switch (`-A`). For more information about using `sqlcmd`, see sqlcmd - Use with scripting variables. You can also connect prefixing `admin:` to the instance name in the format `sqlcmd -S admin:<instance_name>`. You can also initiate a DAC from a SQL Server Management Studio Query Editor by connecting to `admin:<instance_name>`.

To establish a DAC from SQL Server Management Studio:

- Disconnect all connections to the related SQL Server instance, including the Object Explorer and all open query windows.
- From the menu, select **File** > **New** > **Database Engine Query**
- From the connection dialog box in the Server Name field, enter `admin:<server_name>` if using the default instance or `admin:<server_name>\<instance_name>` if using a named instance.

DAC port

SQL Server listens for the DAC on TCP port 1434 if available or a TCP port dynamically assigned upon Database Engine startup. The error log contains the port number the DAC is listening on. By default the DAC listener accepts connection on only the local port. For a code sample that activates remote administration connections, see remote admin connections Server Configuration Option.

After the remote administration connection is configured, the DAC listener is enabled without restarting SQL Server and a client can now connect to the DAC remotely. You can enable the DAC listener to accept connections remotely even if SQL Server is unresponsive by first connecting to SQL Server using the DAC locally, and then executing the `sp_configure` stored procedure to accept connection from remote connections.

On cluster configurations, the DAC will be off by default. Users can execute the remote admin connection option of `sp_configure` to enable the DAC listener to access a remote connection. If SQL Server is unresponsive and the DAC listener isn't enabled, you might have to restart SQL Server to connect with the DAC. Therefore, we recommend that you enable the remote admin connections configuration option on clustered systems.

The DAC port is assigned dynamically by SQL Server during startup. When connecting to the default instance, the DAC avoids using a SQL Server Resolution Protocol (SSRP) request to the SQL Server Browser Service when connecting. It first connects over TCP port 1434. If that fails, it makes an SSRP call to get the port. If SQL Server Browser isn't listening for SSRP requests, the connection request returns an error. Refer to the error log to find the port number DAC is listening on. If SQL Server is configured to accept remote administration connections, the DAC must be initiated with an explicit port number:

```
sqlcmd -S tcp:<server>,<port>
```

The SQL Server error log lists the port number for the DAC, which is 1434 by default. If SQL Server is configured to accept local DAC connections only, connect using the loopback adapter using the following command:

PowerShell Copy

```powershell
sqlcmd -S 127.0.0.1,1434
```

Restrictions

Because the DAC exists solely for diagnosing server problems in rare circumstances, there are some restrictions on the connection:

- To guarantee that there are resources available for the connection, only one DAC is allowed per instance of SQL Server. If a DAC connection is already active, any new request to connect through the DAC is denied with error 17810.

- To conserve resources, SQL Server Express doesn't listen on the DAC port unless started with a trace flag 7806.

- The DAC initially attempts to connect to the default database associated with the login. After it is successfully connected, you can connect to the `master` database. If the default database is offline or otherwise not available, the connection will return error 4060. However, it will succeed if you override the default database to connect to the `master` database instead using the following command:

  PowerShell

  ```
  sqlcmd -A -d master
  ```

  We recommend that you connect to the `master` database with the DAC because `master` is guaranteed to be available if the instance of the Database Engine is started.

- SQL Server prohibits running parallel queries or commands with the DAC. For example, error 3637 is generated if you execute either of the following statements with the DAC:

  - `RESTORE...`
  - `BACKUP...`

- Only limited resources are guaranteed to be available with the DAC. Don't use the DAC to run resource-intensive queries or queries that can block other queries. This helps prevent the DAC from compounding any existing server problems. To avoid potential blocking scenarios, if you have to run queries that might block, run the query under snapshot-based isolation levels if possible; otherwise, set the transaction isolation level to READ UNCOMMITTED and set the LOCK_TIMEOUT value to a short value such as 2000 milliseconds, or both. This will prevent the DAC session from getting blocked. However, depending on the state that the SQL Server is in, the DAC session might get blocked on a latch. You might be able to terminate the DAC session using CTRL-C but it isn't guaranteed. In that case, your only option might be to restart SQL Server.

- To guarantee connectivity and troubleshooting with the DAC, SQL Server reserves limited resources to process commands run on the DAC. These resources are typically only enough for simple diagnostic and troubleshooting functions, such as those listed below.

Although you can theoretically run any Transact-SQL statement that doesn't have to execute in parallel on the DAC, we strongly recommend that you restrict usage to the following diagnostic and troubleshooting commands:

- Querying dynamic management views for basic diagnostics such as sys.dm_tran_locks for the locking status, sys.dm_os_memory_cache_counters to check the health of caches, and sys.dm_exec_requests and sys.dm_exec_sessions for active sessions and requests. Avoid dynamic management views that are resource intensive (for example, sys.dm_tran_version_store scans the full version store and can cause extensive I/O) or that use complex joins. For information about performance implications, see the documentation for the specific dynamic management view.

- Querying catalog views.

- Basic DBCC commands such as DBCC FREEPROCCACHE, DBCC FREESYSTEMCACHE, DBCC DROPCLEANBUFFERS, and DBCC SQLPERF. Avoid resource-intensive commands such as DBCC CHECKDB, DBCC DBREINDEX, or DBCC SHRINKDATABASE.

- Transact-SQL `KILL <spid>` command. Depending on the state of SQL Server, the `KILL` command might not succeed. The only option might be to restart the instance, in the case of SQL Server or Azure SQL Managed Instance. The following are some general guidelines:

  - Verify that the SPID was killed by querying `SELECT * FROM sys.dm_exec_sessions WHERE session_id = <spid>;`. If it returns no rows, it means the session was killed.

  - If the session is still there, verify whether there are tasks assigned to this session by running the query `SELECT * FROM sys.dm_os_tasks WHERE session_id = <spid>;`. If you see the task there, most likely your session is currently being killed. This can take considerable amount of time and might not succeed at all.

  - If there are no tasks in the `sys.dm_os_tasks` associated with this session, but the session remains in `sys.dm_exec_sessions` after executing the `KILL` command, it means that you don't have a worker available. Select one of the currently running tasks (a task listed in the `sys.dm_os_tasks` view with a `sessions_id <> NULL`), and kill the session associated with it to free up the worker. It might not be enough to kill a single session: you might have to kill multiple ones.

Limitation in Azure SQL Database

When connecting to the Azure SQL Database with the DAC, you must also specify the database name in the connection string by using the `-d` option.

Limitation in Azure SQL Managed Instance

The DAC doesn't work over a private endpoint for Azure SQL Managed Instance. On SQL managed instances, the DAC listens on port 1434. Because private endpoints to SQL managed instances only allow connections on port 1433, you cannot use a private endpoint to establish a DAC connection. You must be in the same virtual network as the SQL managed instance to connect with DAC.

Examples

In this example, an administrator notices that server `contoso-server` isn't responding and wants to diagnose the problem. To do this, the user activates the `sqlcmd` command prompt utility and connects to server `contoso-server` using `-A` to indicate the DAC.

PowerShell

```powershell
sqlcmd -S contoso-server -U sa -P <StrongPassword> -A
```

The administrator can now execute queries to diagnose the problem and possibly terminate the unresponsive sessions.

A similar example connecting to SQL Database would use the following command including the -d parameter to specify the database:

PowerShell

```powershell
sqlcmd -S serverName.database.windows.net,1434 -U sa -P <StrongPassword> -d AdventureWorks
```