Table of Contents

EKM provider enabled Server Configuration Option

external scripts enabled Server Configuration Option

filestream access level Server Configuration Option

Configure the fill factor Server Configuration Option

ft crawl bandwidth Server Configuration Option

ft notify bandwidth Server Configuration Option

PolyBase Connectivity Configuration (Transact-SQL)

in-doubt xact resolution Server Configuration Option

Configure the index create memory Server Configuration Option

lightweight pooling Server Configuration Option

Configure the locks Server Configuration Option

Configure the max degree of parallelism Server Configuration Option

max full-text crawl range Server Configuration Option

Configure the max text repl size Server Configuration Option

Configure the max worker threads Server Configuration Option

Configure the media retention Server Configuration Option

Configure the min memory per query Server Configuration Option

Configure the nested triggers Server Configuration Option

Configure the network packet size Server Configuration Option

Ole Automation Procedures Server Configuration Option

open objects Server Configuration Option

optimize for ad hoc workloads Server Configuration Option

PH timeout Server Configuration Option

precompute rank Server Configuration Option

Configure the priority boost Server Configuration Option

Configure the query governor cost limit Server Configuration Option

Configure the query wait Server Configuration Option

Configure the recovery interval Server Configuration Option

Configure the remote access Server Configuration Option

remote admin connections Server Configuration Option

Configure the remote data archive Server Configuration Option

Configure the remote login timeout Server Configuration Option

SQL Server Database Engine

3/24/2017 • 1 min to read • Edit Online

The Database Engine is the core service for storing, processing, and securing data. The Database Engine provides controlled access and rapid transaction processing to meet the requirements of the most demanding data consuming applications within your enterprise.

Use the Database Engine to create relational databases for online transaction processing or online analytical processing data. This includes creating tables for storing data, and database objects such as indexes, views, and stored procedures for viewing, managing, and securing data. You can use SQL Server Management Studio to manage the database objects, and SQL Server Profiler to capture server events.

See Also

SQL Server Resource Center

What's New in SQL Server vNext (Database Engine)

3/24/2017 • 4 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with vNext) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Note: SQL Server vNext also includes the features added in SQL Server 2016 service packs. For those items, see What's New in SQL Server 2016 (Database Engine).

SQL Server Database Engine (CTP 1.4)

- There are no new Database Engine features in this CTP.
- This CTP contains bug fixes for the Database Engine.

SQL Server Database Engine (CTP 1.3)

- Indirect checkpoint performance improvements.
- Cluster-less Availability Groups support added.
- Minimum Replica Commit Availability Groups setting added.
- Availability Groups can now work across Windows-Linux to enable cross-OS migrations and testing.
- Temporal Tables Retention Policy support added,
- New DMV SYS.DM_DB_STATS_HISTOGRAM
- Online non-clustered columnstore index buill and rebuild support added
- 5 new dynamic management views to return information about Linux process. For more information, see Linux Process Dynamic Management Views.
- sys.dm_db_stats_histogram (Transact-SQL) is added for examining statistics.

SQL Server Database Engine (CTP 1.2)

- The Database Tuning Advisor (DTA) released with SQL Server Management Studio version 16.4, when analyzing SQL Server 2016 and later, has additional options.
  - Improved performance. For more information, see Performance Improvements using Database Engine Tuning Advisor (DTA) recommendations.
  - The `-fc` option for allowing recommendations of columnstore indexes. For more information, see DTA Utility and Columnstore index recommendations in Database Engine Tuning Advisor (DTA).
  - The `-iq` option for allowing the DTA to review a workload from the Query Store. For more information, see Tuning Database Using Workload from Query Store.

SQL Server Database Engine (CTP 1.1)

- For In-Memory functionality, additional enhancements to memory-optimized tables and natively compiled functions are listed next, and code samples are available in subsequent text:
  - Support for computed columns in memory-optimized tables, including indexes on computed columns.
  - Full support for JSON functions in natively compiled modules, and in check constraints.
  - `CROSS APPLY` operator in natively compiled modules.
- New string functions CONCAT_WS, TRANSLATE, and TRIM are added.
- The `WITHIN GROUP` clause is now supported for the STRING_AGG function.
- Two new Japanese collation families (Japanese_Bushu_Kakusu_140 and Japanese_XJIS_140) were added, and

the collation option Variation-selector-sensitive (_VSS) was added for use in Japanese collations. For more detail see Collation and Unicode Support

- New bulk access options (BULK INSERT and OPENROWSET(BULK...)) enable access data directly from a file specified as CSV format, and from files stored in Azure Blob storage through the new `BLOB_STORAGE` option of EXTERNAL DATA SOURCE.

SQL Server Database Engine (CTP 1.0)

- Database **COMPATIBILITY_LEVEL** 140 has been added. Customers running in this level will get the latest language features and query optimizer behaviors. This includes changes in each pre-release version Microsoft releases.
- Improvements to the way incremental statistics update thresholds are computed (140 compat mode required).
- sys.dm_exec_query_statistics_xml is added.
- We have added many performance and language enhancements to In-Memory Tables:
  - `sp_spaceused` is now supported for in-memory tables.
  - `sp_rename` is now supported for native modules.
  - `CASE` statements are now supported for native modules.
  - The limitation of 8 indexes on in-memory tables has been removed.
  - `TOP (N) WITH TIES` is now supported in native modules.
  - `ALTER TABLE` against in-memory tables is now substantially faster in some cases.
  - Transaction redo In-memory tables is now done in parallel. This substantially reduces the time to do failovers or in some cases restarts.
  - In-memory checkpoint files can now be stored on Azure Storage. This provides equivalent capabilities to MDF compared to LDF files, which already have this capability.
- Clustered Columnstore Indexes now support LOB columns (nvarchar(max), varchar(max), varbinary(max)).
- The STRING_AGG aggregate function has been added.
- New Permissions: `DATABASE SCOPED CREDENTIAL` is now a class of securable, supporting `CONTROL`, `ALTER`, `REFERENCES`, `TAKE OWNERSHIP`, and `VIEW DEFINITION` permissions. `ADMINISTER DATABASE BULK OPERATIONS` which is restricted to SQL Database is now visible in `sys.fn_builtin_permissions`.
- The sys.dm_os_host_info DMV is added to provide operating system information for both Windows and Linux.
- The database roles are created with R Services for managing permissions associated with packages. For more information, see R Package management for SQL Server.

Code Samples for new In-Memory Enhancements

The following subsections provide Transact-SQL code samples which illustrate new In-Memory features which bullet listed in preceding text in this article.

The CTP 1.1 bullet list for In-Memory is here.

Computed column in a memory-optimized table

This CREATE TABLE statement illustrates the following features which were mentioned in preceding text about CTP 1.1:

- JSON check constraint on a column.
- New computed columns.
- An index on a computed column.

```
CREATE TABLE Product(
    ProductID  int            PRIMARY KEY NONCLUSTERED,
    Name       nvarchar(400)  NOT NULL,
    Price      float,

    Data      nvarchar(4000)  CONSTRAINT [Data contains JSON]  CHECK (ISJSON(Data)=1),

    MadeIn  AS CAST(JSON_VALUE(Data, '$.MadeIn')            as NVARCHAR(50)) PERSISTED,
    Cost    AS CAST(JSON_VALUE(Data, '$.ManufacturingCost') as float       ),

    INDEX [idx_Product_MadeIn] NONCLUSTERED (MadeIn)
)
        WITH (MEMORY_OPTIMIZED=ON);
```

CROSS APPLY, and JSON functions

This CREATE PROCEDURE statement, for a natively compiled stored procedure, illustrates the following features which were mentioned in preceding text about CTP 1.1:

- CROSS APPLY operator.
- JSON functions.

```
CREATE OR ALTER PROCEDURE ProductList()
    WITH SCHEMABINDING, NATIVE_COMPILATION
as begin
    atomic with (transaction isolation level = snapshot,  language = N'English')

    SELECT
            ProductID, Name, Price, Tags,
            Data,
            JSON_VALUE(Data,'$.MadeIn') AS MadeIn,
            value
        FROM Product
        CROSS APPLY OPENJSON(Data, '$.SalesReasons')
        FOR JSON PATH
end;
```

What's New in SQL Server 2016 (Database Engine)

3/24/2017 • 23 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2016) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic summarizes the enhancements introduced in the SQL Server 2016 release of the SQL Server Database Engine. The new features and enhancements increase the power and productivity of architects, developers, and administrators who design, develop, and maintain data storage systems.

To review what is new in the other SQL Server components, see What's New in SQL Server 2016.

> **NOTE**
> SQL Server 2016 is a 64-bit application. 32-bit installation is discontinued, though some elements run as 32-bit components.

Try it out

- To download SQL Server 2016, go to **Evaluation Center**.
- Have an Azure account? Then go **Here** to spin up a Virtual Machine with SQL Server 2016 already installed.

Tip For the current release notes, see SQL Server 2016 Release Notes.

SQL Server 2016 Service Pack 1 (SP1)

- `CREATE OR ALTER <object>` syntax is now available for procedures, views, functions, and triggers.
- Support for a more generic query hinting model has been added: `OPTION (USE HINT('<hint1>', '<hint2>'))`. For more information, see Query Hints (Transact-SQL).
- The sys.dm_exec_valid_use_hints DMV is added to list hints.
- The sys.dm_exec_query_statistics_xml DMV is added to return showplan XML transient statistics.
- The sys.dm_db_incremental_stats_properties DMV is added to incremental statistics for the specified table.
- The `instant_file_initialization_enabled` column is added to sys.dm_server_services.
- The of `estimated_read_row_count` column is added to sys.dm_exec_query_profiles.
- The `sql_memory_model` and `sql_memory_model_desc` columns are added to sys.dm_os_sys_info to provide information about the locking model for memory pages.
- The editions that support a number of features has been expanded. This includes adding row-level security, Always Encrypted, dynamic data masking, database auditing, in-memory OLTP and several other features to all editions. For more information see Editions and Supported Features for SQL Server 2016.
- sp_refresh_parameter_encryption allows Always On encryption to update metadata, when objects encrypted using Always On are redefined.

SQL Server 2016 RTM

This section contains the following subsections:

- Columnstore Indexes
- Database Scoped Configurations
- In-Memory OLTP
- Query Optimizer

- Live Query Statistics
- Query Store
- Temporal Tables
- Striped Backups to Microsoft Azure Blob Storage
- File-Snapshot Backups to Microsoft Azure Blob Storage
- Managed Backup
- TempDB Database
- Built-in JSON Support
- PolyBase
- Stretch Database
- Support for UTF-8
- New Default Database Size and Autogrow Values
- Transact-SQL Enhancements
- System View Enhancements
- Security Enhancements
- High Availability Enhancements
- Replication Enhancements
- Tools Enhancements

Columnstore Indexes

This release offers improvements for columnstore indexes including updateable nonclustered columnstore indexes, columnstore indexes on in-memory tables, and many more new features for operational analytics.

- A read-only nonclustered columnstore index is updateable after upgrade. A rebuild of the index is not required to make it updateable.
- There are performance improvements for analytics queries on columnstore indexes, especially for aggregates and string predicates.
- DMVs and XEvents have supportability improvements.

For more details, see these topics in the Columnstore Indexes Guide section of Books Online:

- Columnstore Indexes Versioned Feature Summary – includes what's new.
- Columnstore Indexes Data Loading
- Columnstore Indexes Query Performance
- Get started with Columnstore for real time operational analytics
- Columnstore Indexes for Data Warehousing
- Columnstore Indexes Defragmentation

Database Scoped Configurations

The new ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL) statement gives you control of certain configurations for your particular database. The configuration settings affect application behavior.

The new statement is available in both SQL Server 2016 and SQL Database.

In-Memory OLTP

Storage format change

The storage format for memory-optimized tables is changed between SQL Server 2014 and 2016. For upgrade and attach/restore from SQL Server 2014, the new storage format is serialized and the database is restarted once during database recovery.

- Upgrade to SQL Server 2016

ALTER TABLE is log-optimized, and runs in parallel

Now when you execute an ALTER TABLE statement on a memory-optimized table, only the metadata changes are written to the log. This greatly reduces log IO. Also, most ALTER TABLE scenarios now run in parallel, which can greatly shorten the duration of the statement.

- For non-parallel exceptions, including LOBs, see Altering Memory-Optimized Tables.

Statistics

Statistics for memory-optimized tables are now updated automatically. In addition, sampling is now a supported method to collect statistics, allowing you to avoid the more expensive fullscan method.

Parallel and heap scan for memory-optimized tables

Memory-optimized tables, and indexes on memory-optimized tables, now support parallel scan. This improves the performance of analytical queries.

In addition, heap scan is supported, and can be performed in parallel. In the case of a memory-optimized table, a heap scan refers to scanning all the rows in a table using the in-memory heap data structure used for storing the rows. For a full table scan, heap scan is more efficient than using an index.

Transact-SQL Improvements for memory-optimized tables

There are several Transact-SQL elements that were not supported for memory-optimized tables in SQL Server 2014, which are now supported in SQL Server 2016:

- UNIQUE constraints and indexes are supported.
- FOREIGN KEY references between memory-optimized tables are supported.
    - These foreign keys can reference only a primary key, and cannot reference a unique key.
- CHECK constraints are supported.
- A non-unique index can allow NULL values in its key.
- TRIGGERs are supported on memory-optimized tables.
    - Only AFTER triggers are supported. INSTEADOF triggers are not supported.
    - Any trigger on a memory-optimized table must use WITH NATIVE_COMPILATION.
- Full support for all SQL Server code pages and collations with indexes and other artifacts in memory-optimized tables and natively compiled T-SQL modules.
- Support for Altering Memory-Optimized Tables:
    - ADD and DROP indexes. Change bucket_count of hash indexes.
    - Make schema changes: add/drop/alter columns; add/drop constraint.
- A memory-optimized table can now have several columns whose combined lengths are longer than the length of the 8060 byte page. An example is a table that has three columns of type `nvarchar(4000)`. In such examples, some columns are now stored off-row. Your queries are blissfully unaware of whether a column is on-row or off-row.
- LOB (large object) types `varbinary(max)`, `nvarchar(max)`, and `varchar(max)` are now supported in memory-optimized tables.

For overall information, see:

- Transact-SQL Constructs Not Supported by In-Memory OLTP
- Unsupported SQL Server Features for In-Memory OLTP

Transact-SQL Improvements for natively compiled modules

There are some Transact-SQL elements that were not supported for natively compiled modules in SQL Server

2014, which are now supported in SQL Server 2016:

- Query constructs:
  - UNION and UNION ALL
  - SELECT DISTINCT
  - OUTER JOIN
  - Subqueries in SELECT
- INSERT, UPDATE and DELETE statements can now include the OUTPUT clause.
- LOBs can now be used in the following ways in a native proc:
  - Declaration of variables.
  - Input parameters received.
  - Parameters passed into string functions, such as into LTrim or Substring, in a native proc.
- Inline (meaning single statement) table-valued functions (TVFs) can now be natively compiled.
- Scalar user-defined functions (UDFs) can now be natively compiled.
- Increased support for a native proc to call:
  - Built-in security functions.
  - Built-in math functions.
  - Built-in function `@@SPID`.
- EXECUTE AS CALLER is now support, which means the EXECUTE AS clause is no longer required when creating a natively compiled T-SQL module.

For overall information, see:

- Supported Features for Natively Compiled T-SQL Modules
- Altering Natively Compiled T-SQL Modules

Performance and scaling improvements

- There is no longer any limitation on data size. See Estimate Memory Requirements for Memory-Optimized Tables.
- There are now multiple concurrent threads responsible to persist to disk the changes to memory-optimized tables.
- Parallel plan support for Accessing Memory-Optimized Tables Using Interpreted Transact-SQL.

Enhancements in SQL Server Management Studio

- The Determining if a Table or Stored Procedure Should Be Ported to In-Memory OLTP no longer requires the configuration of data collectors or management data warehouse. The report can now run directly on a production database.
- PowerShell Cmdlet for Migration Evaluation for evaluating the migration fitness of multiple objects in a SQL Server database.
- Generate migration checklists by right-clicking on a database, and selecting Tasks > Generate In-Memory OLTP migration checklists.

Cross-feature support

- Support for using temporal system-versioning with In-Memory OLTP. For more information, see System-Versioned Temporal Tables with Memory-Optimized Tables
- Query store support for natively compiled code from In-Memory OLTP workloads. For more information, see Using the Query Store with In-Memory OLTP.

- Row-Level Security in Memory-Optimized Tables
- Using Multiple Active Result Sets (MARS) connections can now access memory-optimized tables and natively compiled stored procedures.
- Transparent Data Encryption (TDE) support. If a database is configured for ENCRYPTION, files in theThe Memory Optimized Filegroup are now also encrypted.

For more information, see In-Memory OLTP (In-Memory Optimization).

Query Optimizer

Compatibility Level Guarantees

When you upgrade your database to SQL Server 2016, there will be no plan changes seen if you remain at the older compatibility levels that you were using (for example, 120 or 110). New features and improvements related to query optimizer, will be available only under latest compatibility level.

Trace Flag 4199

In general, you do not need to use trace flag 4199 in SQL Server 2016 since most of the query optimizer behaviors controlled by this trace flag are enabled unconditionally under the latest compatibility level (130) in SQL Server 2016.

New Referential Integrity Operator

A table can reference a maximum of 253 other tables and columns as foreign keys (outgoing references). SQL Server 2016 increases the limit for the number of other table and columns that can reference columns in a single table (incoming references), from 253 to 10,000. For restrictions, see Create Foreign Key Relationships. A new referential integrity operator is introduced (under compatibility level 130), which performs the referential integrity checks in place. This improves overall performance for UPDATE and DELETE operations, on tables that have a large number of incoming references, thereby making it feasible to have large number of incoming references. For more information, see Query Optimizer Additions in SQL Server 2016

Parallel update of sampled statistics

Data sampling to build statistics is now done in parallel (under compatibility level 130), to improve the performance of statistics collection. For more information, see Update Statistics.

Sublinear threshold for update of statistics

Automatic update of statistics is now more aggressive on large tables (under compatibility level 130). The threshold to trigger auto-update of statistics is 20%, starting SQL Server 2016, for larger tables, this threshold will start decreasing (still a percentage) as the number of rows increase in the table. You will no longer need to set trace flag 2371 to reduce the threshold.

Other enhancements

The Insert in an Insert-select statement is multi-threaded or can have a parallel plan (under compatibility level 130). To get a parallel plan, INSERT … SELECT statement must use the TABLOCK hint. For more information, see Parallel Insert Select

Live Query Statistics

Management Studio provides the ability to view the live execution plan of an active query. This live query plan provides real-time insights into the query execution process as the controls flow from one query plan operator to another. For more information, see Live Query Statistics.

Query Store

Query store is a new feature that provides DBAs with insight on query plan choice and performance. It simplifies performance troubleshooting by enabling you to quickly find performance differences caused by changes in query plans. The feature automatically captures a history of queries, plans, and runtime statistics, and retains these for your review. It separates data by time windows, allowing you to see database usage patterns and understand when query plan changes happened on the server. The query store presents information by using a Management Studio dialog box, and lets you force the query to one of the selected query plans. For more information, see Monitoring Performance By Using the Query Store.

Temporal Tables

SQL Server 2016 now supports system-versioned temporal tables. A temporal table is a new type of table that provides correct information about stored facts at any point in time. Each temporal table consists of two tables actually, one for the current data and one for the historical data. The system ensures that when the data changes in the table with the current data the previous values are stored in the historical table. Querying constructs are provided to hide this complexity from users. For more information, see Temporal Tables.

Striped Backups to Microsoft Azure Blob Storage

In SQL Server 2016, SQL Server backup to URL using the Microsoft Azure Blob storage service now supports striped backups sets using block blobs to support a maximum backup size of 12.8 TB. For examples, see Code Examples.

File-Snapshot Backups to Microsoft Azure Blob Storage

In SQL Server 2016, SQL Server backup to URL now supports using Azure snapshots to backup databases in which all database files are stored using the Microsoft Azure Blob storage service. For more information, see File-Snapshot Backups for Database Files in Azure.

Managed Backup

In SQL Server 2016 SQL Server Managed Backup to Microsoft Azure uses the new block blob storage for backup files. There are also several changes and enhancements to Managed Backup.

- Support for both automated and custom scheduling of backups.
- Support backups for system databases.
- Support for databases that are using the Simple recovery model.

  For more information, see SQL Server Managed Backup to Microsoft Azure

> **NOTE**
> For SQL Server 2016, these new managed backup features do not yet have corresponding UI support in SQL Server Management Studio.

TempDB Database

There are several enhancements to TempDB:

- Trace Flags 1117 and 1118 are not required for tempdb anymore. If there are multiple tempdb database files all files will grow at the same time depending on growth settings. In addition, all allocations in tempdb will use uniform extents.
- By default, setup adds as many tempdb files as the CPU count or 8, whichever is lower.
- During setup, you can configure the number of tempdb database files, initial size, autogrowth and directory placement using the new UI input control on the Database Engine Configuration - TempDB section of SQL Server Installation Wizard.
- The default initial size is 8MB and the default autogrowth is 64MB.
- You can specify multiple volumes for tempdb database files. If multiple directories are specified tempdb data files will be spread across the directories in a round-robin fashion.

Built-in JSON Support

SQL Server 2016 adds built-in support for importing and exporting JSON and working with JSON strings. This built-in support includes the following statements and functions.

- Format query results as JSON, or export JSON, by adding the **FOR JSON** clause to a **SELECT** statement. Use the **FOR JSON** clause, for example, to delegate the formatting of JSON output from your client applications to SQL Server. For more info, see Format Query Results as JSON with FOR JSON (SQL Server).

- Convert JSON data to rows and columns, or import JSON, by calling the **OPENJSON** rowset provider function. Use **OPENJSON** to import JSON data into SQL Server, or convert JSON data to rows and columns for an app or service that can't currently consume JSON directly. For more info, see Convert JSON Data to Rows and Columns with OPENJSON (SQL Server).
- The **ISJSON** function tests whether a string contains valid JSON. For more info, see ISJSON (Transact-SQL)
- The **JSON_VALUE** function extracts a scalar value from a JSON string.For more info, see JSON_VALUE (Transact-SQL).
- The **JSON_QUERY** function extracts an object or an array from a JSON string. For more info, see JSON_QUERY (Transact-SQL).
- The **JSON_MODIFY** function updates the value of a property in a JSON string and return the updated JSON string. For more info, see JSON_MODIFY (Transact-SQL).

PolyBase

PolyBase allows you to use T-SQL statements to access data stored in Hadoop or Azure Blob Storage and query it in an adhoc fashion. It also lets you query semi-structured data and join the results with relational data sets stored in SQL Server. PolyBase is optimized for data warehousing workloads and intended for analytical query scenarios.

For more information, see PolyBase Guide.

Stretch Database

Stretch Database is a new feature in SQL Server 2016 that migrates your historical data transparently and securely to the Microsoft Azure cloud. You can access your SQL Server data seamlessly regardless of whether it's on-premises or stretched to the cloud. You set the policy that determines where data is stored, and SQL Server handles the data movement in the background. The entire table is always online and queryable. And, Stretch Database doesn't require any changes to existing queries or applications – the location of the data is completely transparent to the application. For more info, see Stretch Database.

Support for UTF-8

bcp Utility, BULK INSERT, and OPENROWSET now support the UTF-8 code page. For more information, see those topics and Create a Format File (SQL Server).

New Default Database Size and Autogrow Values

New values for the model database and default values for new databases (which are based on model). The initial size of the data and log files is now 8 MB. The default auto-growth of data and log files is now 64MB.

Transact-SQL Enhancements

Numerous enhancements support the features described in the other sections of this topic. The following additional enhancements are available.

- The TRUNCATE TABLE statement now permits the truncation of specified partitions. For more information, see TRUNCATE TABLE (Transact-SQL).
- ALTER TABLE (Transact-SQL) now allows many alter column actions to be performed while the table remains available.
- The full-text index DMV sys.dm_fts_index_keywords_position_by_document (Transact-SQL) returns the location of keywords in documents. This DMV has also been added in SQL Server 2012 SP2 and SQL Server 2014 SP1.
- A new query hint **NO_PERFORMANCE_SPOOL** can prevent a spool operator from being added to query plans. This can improve performance when many concurrent queries are running with spool operations. For more information, see Query Hints (Transact-SQL).
- The FORMATMESSAGE (Transact-SQL) statement is enhances to accept a msg_string argument.- The maximum index key size for NONCLUSTERED indexes has been increased to 1700 bytes.
- New DROP IF syntax is added for drop statements related to AGGREGATE, ASSEMBLY, COLUMN, CONSTRAINT, DATABASE, DEFAULT, FUNCTION, INDEX, PROCEDURE, ROLE, RULE, SCHEMA, SECURITY POLICY, SEQUENCE,

SYNONYM, TABLE, TRIGGER, TYPE, USER, and VIEW. See individual syntax topics for syntax.

- A MAXDOP option has been added to DBCC CHECKTABLE (Transact-SQL), DBCC CHECKDB (Transact-SQL), and DBCC CHECKFILEGROUP (Transact-SQL) to specify the degree of parallelism.
- SESSION_CONTEXT can now be set. Includes the SESSION_CONTEXT (Transact-SQL) function, CURRENT_TRANSACTION_ID (Transact-SQL) function, and the sp_set_session_context (Transact-SQL) procedure.
- Advanced Analytics Extensions allow users to execute scripts written in a supported language such as R. Transact-SQL supports R by introducing the sp_execute_external_script (Transact-SQL) stored procedure, and the external scripts enabled Server Configuration Option. For more information, see SQL Server R Services.
- Also to support R, the ability to create an external resource pool. For more information, see CREATE EXTERNAL RESOURCE POOL (Transact-SQL). New catalog views and DMVs (sys.resource_governor_external_resource_pools (Transact-SQL) and sys.dm_resource_governor_external_resource_pool_affinity (Transact-SQL)). Additional arguments are available for sp_execute_external_script (Transact-SQL) and CREATE WORKLOAD GROUP (Transact-SQL). Additional columns are added to some of the existing resource governor catalog views and DMVs.
- The CREATE USER syntax is enhanced with the ALLOW_ENCRYPTED_VALUE_MODIFICATIONS option to support the Always Encrypted feature. For more information see Migrate Sensitive Data Protected by Always Encrypted.
- The COMPRESS (Transact-SQL) and DECOMPRESS (Transact-SQL) functions convert values into and out of the GZIP algorithm.
- The DATEDIFF_BIG (Transact-SQL) and AT TIME ZONE (Transact-SQL) functions and the sys.time_zone_info (Transact-SQL) view are added to support date and time interactions.
- A credential can now be created at the database level (in addition to the server level credential that was previously available). For more information, see CREATE DATABASE SCOPED CREDENTIAL (Transact-SQL).
- Eight new properties are added to SERVERPROPERTY (Transact-SQL): InstanceDefaultDataPath, InstanceDefaultLogPath, ProductBuild, ProductBuildType, ProductMajorVersion, ProductMinorVersion, ProductUpdateLevel, and ProductUpdateReference.
- The input length limit of 8,000 bytes for the HASHBYTES (Transact-SQL) function is removed.
- New string functions STRING_SPLIT (Transact-SQL) and STRING_ESCAPE (Transact-SQL) are added.
- Autogrow options: Trace flag 1117 is replaced by the AUTOGROW_SINGLE_FILE and AUTOGROW_ALL_FILES option of ALTER DATABASE, and trace flag 1117 has no affect. For more information, see ALTER DATABASE File and Filegroup Options (Transact-SQL) and the new is_autogrow_all_files column of sys.filegroups (Transact-SQL).
- Allocation of mixed extents: For user databases, default allocation for the first 8 pages of an object will change from using mixed page extents to using uniform extents. Trace flag 1118 is replaced with the SET MIXED_PAGE_ALLOCATION option of ALTER DATABASE, and trace flag 1118 has no affect. For more information, see ALTER DATABASE SET Options (Transact-SQL), and the new `is_mixed_page_allocation_on` column of sys.databases (Transact-SQL).

System View Enhancements

- Two new views support row level security. For more information, see sys.security_predicates (Transact-SQL) and sys.security_policies (Transact-SQL).
- Seven new views support the Query Store feature. For more information, see Query Store Catalog Views (Transact-SQL).
- 24 new columns are added to sys.dm_exec_query_stats (Transact-SQL) provide information about memory grants.
- Two new query hints (MIN_GRANT_PERCENT and MAX_GRANT_PERCENT) are added to specify memory grants. See Query Hints (Transact-SQL).
- sys.dm_exec_session_wait_stats (Transact-SQL) provides a per session report similar to the server wide sys.dm_os_wait_stats (Transact-SQL).
- sys.dm_exec_function_stats (Transact-SQL) provides execution statistics regarding scalar valued functions.

- Beginning with SQL Server 2016, entries in sys.dm_db_index_usage_stats (Transact-SQL) are retained as they were prior to SQL Server 2008 R2.
- Information about statements submitted to an instance of SQL Server can be returned by the new dynamic management function sys.dm_exec_input_buffer (Transact-SQL).
- Two new views support SQL Server R Services: sys.dm_external_script_requests and sys.dm_external_script_execution_stats.

Security Enhancements

Row-Level Security

Row-level security introduces predicate based access control. It features a flexible, centralized, predicate-based evaluation that can take into consideration metadata (such as labels) or any other criteria the administrator determines as appropriate. The predicate is used as a criterion to determine whether or not the user has the appropriate access to the data based on user attributes. Label based access control can be implemented by using predicate based access control. For more information, see Row-Level Security.

Always Encrypted

With Always Encrypted, SQL Server can perform operations on encrypted data, and best of all the encryption key resides with the application inside the customer's trusted environment and not on the server. Always Encrypted secures customer data so DBAs do not have access to plain text data. Encryption and decryption of data happens transparently at the driver level minimizing changes that have to be made to existing applications. For more information, see Always Encrypted (Database Engine).

Dynamic Data Masking

Dynamic data masking limits sensitive data exposure by masking it to non-privileged users. Dynamic data masking helps prevent unauthorized access to sensitive data by enabling customers to designate how much of the sensitive data to reveal with minimal impact on the application layer. It's a policy-based security feature that hides the sensitive data in the result set of a query over designated database fields, while the data in the database is not changed. For more information, see Dynamic Data Masking.

New Permissions

- The **ALTER ANY SECURITY POLICY** permission is available as part of the implementation of row level security.
- The **ALTER ANY MASK** and **UNMASK** permissions are available as part of the implementation of dynamic data masking.
- The **ALTER ANY COLUMN ENCRYPTION KEY**, **VIEW ANY COLUMN ENCRYPTION KEY**, **ALTER ANY COLUMN MASTER KEY DEFINITION**, and **VIEW ANY COLUMN MASTER KEY DEFINITION** permissions are available as part of the implementation of the Always Encrypted feature.
- The **ALTER ANY EXTERNAL DATA SOURCE** and **ALTER ANY EXTERNAL FILE FORMAT** permissions are visible in SQL Server 2016 but only apply to the Analytics Platform System (SQL Data Warehouse).
- The **EXECUTE ANY EXTERNAL SCRIPT** permissions are available as part of the support for R scripts.
  - The **ALTER ANY DATABASE SCOPED CONFIGURATION** permissions is available to authorize the use of the ALTER DATABASE SCOPED CONFIGURATION (Transact-SQL) statement.

Transparent Data Encryption

- Transparent Data Encryption has been enhanced with support for Intel AES-NI hardware acceleration of encryption. This will reduce the CPU overhead of turning on Transparent Data Encryption.

AES Encryption for Endpoints

- The default encryption for endpoints is changed from RC4 to AES.

New Credential Type

- A credential can now be created at the database level (in addition to the server level credential that was previously available). For more information, see CREATE DATABASE SCOPED CREDENTIAL (Transact-SQL).

High Availability Enhancements

SQL Server 2016 Standard Edition now supports Always On Basic Availability Groups. Basic availability groups provide support for a primary and secondary replica. This capability replaces the obsolete Database Mirroring technology for high availability. For more information about the differences between basic and advanced availability groups, see Basic Availability Groups (Always On Availability Groups).

Load-balancing of read-intent connection requests is now supported across a set of read-only replicas. The previous behavior always directed connections to the first available read-only replica in the routing list. For more information, see Configure load-balancing across read-only replicas.

The number of replicas that support automatic failover has been increased from two to three.

Group Managed Service Accounts are now supported for Always On Failover Clusters. For more information, see Group Managed Service Accounts. For Windows Server 2012 R2, an update is required to avoid temporary downtime after a password change. To obtain the update, see gMSA-based services can't log on after a password change in a Windows Server 2012 R2 domain.

Always On Availability Groups supports distributed transactions and the DTC on Windows Server 2016. For more information, see Support for distributed transactions.

You can now configure Always On Availability Groups to failover when a database goes offline. This change requires the setting the **DB_FAILOVER** option to **ON** in the CREATE AVAILABILITY GROUP (Transact-SQL) or ALTER AVAILABILITY GROUP (Transact-SQL) statements.

Always On now supports encrypted databases. The Availability Group wizards now prompt you for a password for any databases that contain a database master key when you create a new Availability Group or when you add databases or add replicas to an existing Availability Group.

Two availability groups in two separate Windows Server Failover Clusters (WSFC) can now be combined into a Distributed Availability Group. For more information, see Distributed Availability Groups (Always On Availability Groups).

Direct seeding allows a secondary replica to be automatically seeded over the network (rather than manual seeding that requires a physical backup of the target database to be restored on the secondary). Direct seeding is specified by setting **SEEDING_MODE=AUTOMATIC** in the CREATE AVAILABILITY GROUP (Transact-SQL) or ALTER AVAILABILITY GROUP (Transact-SQL) statements. You must also specify **GRANT CREATE ANY DATABASE** with ALTER AVAILABILITY GROUP (Transact-SQL) on each secondary replica that is used with direct seeding.

Performance improvements – The synchronization throughput of availability groups has been increased ~10x through parallel and faster compression of log blocks on the primary replica, an optimized synchronization protocol, and parallel decompression and redo of log records on the secondary replica. This increases the freshness of readable secondaries and reduces database recovery time in case of failover. Note that redo for memory-optimized tables is not yet parallel in SQL Server 2016.

Replication Enhancements

- Replication of memory-optimized tables are now supported. For more information, see Replication to Memory-Optimized Table Subscribers.
- Replication is now supported to Azure SQL Database. For more information, see Replication to SQL Database.

Tools Enhancements

Management Studio

Download the latest SQL Server Management Studio (SSMS)

- SQL Server Management Studio supports the Active Directory Authentication Library (ADAL) which is under development for connecting to Microsoft Azure. This replaces the certificate-based authentication used in SQL Server 2014 Management Studio.
- SQL Server Management Studio installation requires installing .NET 4.6 as a pre-requisite. .NET 4.6 will be automatically installed by setup when SQL Server Management Studio is installed.
- A new query result grid option supports keeping Carriage Return/Line Feed (newline characters) when copying

or saving text from the results grid. Set this from the Tools/Options menu.

- SQL Server Management Tools is no longer installed from the main feature tree; for details see Install SQL Server Management Tools with SSMS.
- SQL Server Management Studio installation requires installing .NET 4.6.1 as a pre-requisite. .NET 4.6.1 will be automatically installed by setup when SQL Server Management Studio is installed.

Upgrade Advisor

SQL Server 2016 Upgrade Advisor Preview is a standalone tool that enables users of prior versions to run a set of upgrade rules against their SQL Server database to pinpoint breaking and behavior changes and deprecated features as well as providing help with the adoption of new features such as Stretch Database.

You can download Upgrade Advisor Preview here or you can install it by using the Web Platform Installer.

See Also

What's New in SQL Server 2016

SQL Server 2016 Release Notes

Install SQL Server Management Tools with SSMS

Database Engine Instances (SQL Server)

3/24/2017 • 5 min to read • Edit Online

An instance of the Database Engine is a copy of the **sqlservr.exe** executable that runs as an operating system service. Each instance manages several system databases and one or more user databases. Each computer can run multiple instances of the Database Engine. Applications connect to the instance in order to perform work in a database managed by the instance.

Instances

An instance of the Database Engine operates as a service that handles all application requests to work with the data in any of the databases managed by that instance. It is the target of the connection requests (logins) from applications. The connection runs through a network connection if the application and instance are on separate computers. If the application and instance are on the same computer, the SQL Server connection can run as either a network connection or an in-memory connection. When a connection has been completed, an application sends Transact-SQL statements across the connection to the instance. The instance resolves the Transact-SQL statements into operations against the data and objects in the databases, and if the required permissions have been granted to the login credentials, performs the work. Any data retrieved is returned to the application, along with any messages such as errors.

You can run multiple instances of the Database Engine on a computer. One instance can be the default instance. The default instance has no name. If a connection request specifies only the name of the computer, the connection is made to the default instance. A named instance is one where you specify an instance name when installing the instance. A connection request must specify both the computer name and instance name in order to connect to the instance. There is no requirement to install a default instance; all of the instances running on a computer can be named instances.

Related Tasks

TASK DESCRIPTION	TOPIC
Describes how to configure the properties of an instance. Configure defaults such as file locations and date formats, or how the instance uses operating system resources, such as memory or threads.	Configure Database Engine Instances (SQL Server)
Describes how to manage the collation for an instance of the Database Engine. Collations define the bit patterns used to represent characters, and associated behaviors such as sorting, and case or accent sensitivity in comparison operations.	Collation and Unicode Support
Describes how to configure linked server definitions, which allow Transact-SQL statements run in an instance to work with data stored in separate OLE DB data sources.	Linked Servers (Database Engine)

TASK DESCRIPTION	TOPIC
Describes how to create a logon trigger, which specifies actions to be taken after a logon attempt has been validated, but before it starts working with resources in the instance. Logon triggers support actions such as logging connection activity, or restricting logins based on logic in addition to the credential authentication performed by Windows and SQL Server.	Logon Triggers
Describes how to manage the service associated with an instance of the Database Engine. This includes actions such as starting and stopping the service, or configuring startup options.	Manage the Database Engine Services
Describes how to perform server network configuration tasks such as enabling protocols, modifying the port or pipe used by a protocol, configuring encryption, configuring the SQL Server Browser service, exposing or hiding the SQL Server Database Engine on the network, and registering the Server Principal Name.	Server Network Configuration
Describes how to perform client network configuration tasks such as configuring client protocols and creating or deleting a Server Alias.	Client Network Configuration
Describes the SQL Server Management Studio editors that can be used to design, debug, and run scripts such as Transact-SQL scripts. Also describes how to code Windows PowerShell scripts to work with SQL Server components.	Database Engine Scripting
Describes how to use maintenance plans to specify a workflow of common administration tasks for an instance. Workflows include tasks such as backing up databases and updating statistics to improve performance.	Maintenance Plans
Describes how to use the resource governor to manage resource consumption and workloads by specifying limits to the amount of CPU and memory that application requests can use.	Resource Governor
Describes how database applications can use database mail to send e-mail messages from the Database Engine.	Database Mail
Describes how to use extended events to capture performance data can be used to build performance baselines or to diagnose performance problems. Extended events are a light-weight, highly scalable system for gathering performance data.	Extended Events
Describes how to use SQL Trace to build a customized system for capturing and recording events in the Database Engine.	SQL Trace
Describes how to use SQL Server Profiler to capture traces of application requests coming in to an instance of the Database Engine. These traces can later be replayed for activities such as performance testing or problem diagnosis.	SQL Server Profiler

TASK DESCRIPTION	TOPIC
Describes the Change Data Capture (CDC) and Change Tracking features and describes how to use these features to track changes to data in a database.	Track Data Changes (SQL Server)
Describes how to use the Log File viewer to find and view SQL Server errors and messages in various logs, such as the SQL Server job history, the SQL Server logs, and Windows event logs.	Log File Viewer
Describes how to use the Database Engine Tuning Advisor to analyze databases and make recommendations for addressing potential performance problems.	Database Engine Tuning Advisor
Describes how the production database administrators can make a diagnostic connection to instances when standard connections are not being accepted.	Diagnostic Connection for Database Administrators
Describes how to use the deprecated remote servers feature to enable access from one instance of the Database Engine to another. The preferred mechanism for this functionality is a linked server.	Remote Servers
Describes the capabilities of Service Broker for messaging and queueing applications and provides pointers to the Service Broker documentation.	Service Broker
Describes how the buffer pool extension can be used to provide seamless integration of nonvolatile random access storage (solid-state drives) to the Database Engine buffer pool to significantly improve I/O throughput.	Buffer Pool Extension File

See Also

sqlservr Application
Database Features
Database Engine Cross-Instance Features

Configure Database Engine Instances (SQL Server)

3/24/2017 • 1 min to read • Edit Online

Each instance of the Database Engine must be configured to meet the performance and availability requirements defined for the databases hosted by the instance. The Database Engine includes configuration options that control behaviors such as resource usage and the availability of features such as auditing or trigger recursion.

Instance Configuration

When a database is deployed into production there is often a service level agreement (SLA) defining areas such as the levels of performance required from the database and the required availability level of the database. The terms of the SLA typically drive configuration requirements for the instance.

An instance is usually configured immediately after it has been installed. The initial configuration is usually determined by the SLA requirements of the types of databases planned to be deployed to the instance. After the databases have been deployed, the database administrators monitor the performance of the instance and adjust the configuration settings as needed if the performance metrics show the instance is not meeting the SLA requirements.

Configuration Tasks

TASK DESCRIPTION	TOPIC
Describes the various instance configuration options and how to view or change these options.	Server Configuration Options (SQL Server)
Describes how to view and configure the default locations of new data and log files in the instance.	View or Change the Default Locations for Data and Log Files (SQL Server Management Studio)
Describes how to configure SQL Server to use soft-NUMA.	Soft-NUMA (SQL Server)
Describes how to map a TCP/IP port to non-uniform memory access (NUMA) node affinity.	Map TCP IP Ports to NUMA Nodes (SQL Server)
Describes how to enable the Windows Lock Pages In Memory policy. This policy determines which accounts can use a process to keep data in physical memory, preventing the system from paging the data to virtual memory on disk.	Enable the Lock Pages in Memory Option (Windows)

See Also

Database Engine Instances (SQL Server)

Server Configuration Options (SQL Server)

3/24/2017 • 5 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

You can manage and optimize SQL Server resources through configuration options by using SQL Server Management Studio or the sp_configure system stored procedure. The most commonly used server configuration options are available through SQL Server Management Studio; all configuration options are accessible through sp_configure. Consider the effects on your system carefully before setting these options. For more information, see View or Change Server Properties (SQL Server).

> **IMPORTANT!!** Advanced options should be changed only by an experienced database administrator or certified SQL Server technician.

Categories of Configuration Options

Configuration options take effect either:

- Immediately after setting the option and issuing the **RECONFIGURE** (or in some cases, **RECONFIGURE WITH OVERRIDE**) statement. Reconfiguring certain options will invalidate plans in the plan cache, causing new plans to be compiled. For more information, see DBCC FREEPROCCACHE (Transact-SQL).

  -or-

- After performing the above actions and restarting the instance of SQL Server.

Options that require SQL Server to restart will initially show the changed value only in the value column. After restart, the new value will appear in both the value column and the value_in_use column.

Some options require a server restart before the new configuration value takes effect. If you set the new value and run sp_configure before restarting the server, the new value appears in the configuration options **value** column, but not in the **value_in_use** column. After restarting the server, the new value appears in the **value_in_use** column.

Self-configuring options are those that SQL Server adjusts according to the needs of the system. In most cases, this eliminates the need for setting the values manually. Examples include the **min server memory** and **max server memory** options and the user connections option.

Configuration Options Table

The following table lists all available configuration options, the range of possible settings, and default values. Configuration options are marked with letter codes as follows:

- A= Advanced options, which should be changed only by an experienced database administrator or a certified SQL Server technician, and which require setting show advanced options to 1.
- RR = Options requiring a restart of the Database Engine.
- RP = Options that require a restart of the PolyBase Engine.
- SC = Self-configuring options.

CONFIGURATION OPTION	MINIMUM VALUE	MAXIMUM VALUE	DEFAULT
access check cache bucket count (A)	0	16384	0
access check cache quota (A)	0	2147483647	0
ad hoc distributed queries (A)	0	1	0
affinity I/O mask (A, RR)	-2147483648	2147483647	0
affinity64 I/O mask (A, only available on 64-bit version of SQL Server)	-2147483648	2147483647	0
affinity mask (A)	-2147483648	2147483647	0
affinity64 mask (A, RR), only available on 64-bit version of SQL Server	-2147483648	2147483647	0
Agent XPs (A)	0	1	0 (Changes to 1 when SQL Server Agent is started. Default value is 0 if SQL Server Agent is set to automatic start during Setup.)
allow updates (Obsolete. Do not use. Will cause an error during reconfigure.)	0	1	0
automatic soft-NUMA disabled	0	1	0
backup checksum default	0	1	0
backup compression default	0	1	0
blocked process threshold (A)	0	86400	0
c2 audit mode (A, RR)	0	1	0
clr enabled	0	1	0

CONFIGURATION OPTION	MINIMUM VALUE	MAXIMUM VALUE	DEFAULT
common criteria compliance enabled (A, RR)	0	1	0
contained database authentication	0	1	0
cost threshold for parallelism (A)	0	32767	5
cross db ownership chaining	0	1	0
cursor threshold (A)	-1	2147483647	-1
Database Mail XPs (A)	0	1	0
default full-text language (A)	0	2147483647	1033
default language	0	9999	0
default trace enabled (A)	0	1	1
disallow results from triggers (A)	0	1	0
EKM provider enabled	0	1	0
external scripts enabled (RR) **Applies to**: SQL Server (SQL Server 2016 through current version.	0	1	0
filestream_access_level	0	2	0
fill factor (A, RR)	0	100	0
ft crawl bandwidth (max), see ft crawl bandwidth(A)	0	32767	100
ft crawl bandwidth (min), see ft crawl bandwidth(A)	0	32767	0
ft notify bandwidth (max), see ft notify bandwidth(A)	0	32767	100

CONFIGURATION OPTION	MINIMUM VALUE	MAXIMUM VALUE	DEFAULT
ft notify bandwidth (min), see ft notify bandwidth(A)	0	32767	0
index create memory (A, SC)	704	2147483647	0
in-doubt xact resolution (A)	0	2	0
lightweight pooling (A, RR)	0	1	0
locks (A, RR, SC)	5000	2147483647	0
max degree of parallelism (A)	0	32767	0
max full-text crawl range (A)	0	256	4
max server memory (A, SC)	16	2147483647	2147483647
max text repl size	0	2147483647	65536
max worker threads (A)	128	32767 (1024 is the maximum recommended for 32-bit SQL Server, 2048 for 64-bit SQL Server.)(SQL Server 2014 was the last version available on 32-bit operating system.)	0 Zero auto-configures the number of max worker threads depending on the number of processors, using the formula (256+(<*processors*> -4) * 8) for 32-bit SQL Server and twice that for 64-bit SQL Server. (SQL Server 2014 was the last version available on 32-bit operating system.)
media retention (A, RR)	0	365	0
min memory per query (A)	512	2147483647	1024
min server memory (A, SC)	0	2147483647	0
nested triggers	0	1	1
network packet size (A)	512	32767	4096

CONFIGURATION OPTION	MINIMUM VALUE	MAXIMUM VALUE	DEFAULT
Ole Automation Procedures (A)	0	1	0
open objects (A, RR, obsolete)	0	2147483647	0
optimize for ad hoc workloads (A)	0	1	0
PH_timeout (A)	1	3600	60
PolyBase Hadoop and Azure blob storage (RP) **Applies to**: SQL Server (SQL Server 2016 through current version.	0	7	0
precompute rank (A)	0	1	0
priority boost (A, RR)	0	1	0
query governor cost limit (A)	0	2147483647	0
query wait (A)	-1	2147483647	-1
recovery interval (A, SC)	0	32767	0
remote access (RR)	0	1	1
remote admin connections	0	1	0
remote data archive	0	1	0
remote login timeout	0	2147483647	10
remote proc trans	0	1	0
remote query timeout	0	2147483647	0
Replication XPs Option (A)	0	1	0
scan for startup procs (A, RR)	0	1	0
server trigger recursion	0	1	1

CONFIGURATION OPTION	MINIMUM VALUE	MAXIMUM VALUE	DEFAULT
set working set size (A, RR, obsolete)	0	1	0
show advanced options	0	1	0
SMO and DMO XPs (A)	0	1	1
transform noise words (A)	0	1	0
two digit year cutoff (A)	1753	9999	2049
user connections (A, RR, SC)	0	32767	0
user options	0	32767	0
xp_cmdshell (A)	0	1	0

See also

sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)
DBCC FREEPROCCACHE (Transact-SQL)

access check cache Server Configuration Options

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

When database objects are accessed by SQL Server, the access check is cached in an internal structure called the **access check result cache**. The **access check cache quota** and **access check cache bucket count** options control the number of entries and number of hash buckets used for **access check result cache**. In rare circumstances, performance can be improved by changing these options.

The default values of 0 indicates that SQL Server is managing these options. Microsoft recommends only changing these options when directed by Microsoft Customer Support Services.

See Also

Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

ad hoc distributed queries Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

By default, SQL Server does not allow ad hoc distributed queries using OPENROWSET and OPENDATASOURCE. When this option is set to 1, SQL Server allows ad hoc access. When this option is not set or is set to 0, SQL Server does not allow ad hoc access.

Ad hoc distributed queries use the OPENROWSET and OPENDATASOURCE functions to connect to remote data sources that use OLE DB. OPENROWSET and OPENDATASOURCE should be used only to reference OLE DB data sources that are accessed infrequently. For any data sources that will be accessed more than several times, define a linked server.

IMPORTANT

Enabling the use of ad hoc names means that any authenticated login to SQL Server can access the provider. SQL Server administrators should enable this feature for providers that are safe to be accessed by any local login.

Remarks

Attempting to make an ad hoc connection with **Ad Hoc Distributed Queries** not enabled results in error: Msg 7415, Level 16, State 1, Line 1

Ad hoc access to OLE DB provider 'Microsoft.ACE.OLEDB.12.0' has been denied. You must access this provider through a linked server.

Examples

The following example enables ad hoc distributed queries and then queries a server named `Seattle1` using the `OPENROWSET` function.

```
sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
sp_configure 'Ad Hoc Distributed Queries', 1;
RECONFIGURE;
GO

SELECT a.*
FROM OPENROWSET('SQLNCLI', 'Server=Seattle1;Trusted_Connection=yes;',
     'SELECT GroupName, Name, DepartmentID
      FROM AdventureWorks2012.HumanResources.Department
      ORDER BY GroupName, Name') AS a;
GO
```

See Also

Server Configuration Options (SQL Server)

Linked Servers (Database Engine)
OPENROWSET (Transact-SQL)
OPENDATASOURCE (Transact-SQL)
sp_addlinkedserver (Transact-SQL)

affinity Input-Output mask Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

To carry out multitasking, Microsoft Windows 2000 and Windows Server 2003 sometimes move process threads among different processors. Although efficient from an operating system point of view, this activity can reduce Microsoft SQL Server performance under heavy system loads, as each processor cache is repeatedly reloaded with data. Assigning processors to specific threads can improve performance under these conditions by eliminating processor reloads; such an association between a thread and a processor is called processor affinity.

SQL Server supports processor affinity by means of two affinity mask options: **affinity mask** (also known as **CPU affinity mask**) and **affinity I/O mask**. For more information on the **affinity mask** option, see affinity mask Server Configuration Option. CPU and I/O affinity support for servers with 33 to 64 processors requires the additional use of the affinity64 mask Server Configuration Option and affinity64 Input-Output mask Server Configuration Option respectively.

NOTE

Affinity support for servers with 33 to 64 processors is only available on 64-bit operating systems.

The **affinity I/O mask** option binds SQL Server disk I/O to a specified subset of CPUs. In high-end SQL Server online transactional processing (OLTP) environments, this extension can enhance the performance of SQL Server threads issuing I/Os. This enhancement does not support hardware affinity for individual disks or disk controllers.

The value for **affinity I/O mask** specifies which CPUs in a multiprocessor computer are eligible to process SQL Server disk I/O operations. The mask is a bitmap in which the rightmost bit specifies the lowest-order CPU(0), the bit to its immediate left specifies the next-lowest-order CPU(1), and so on. To configure more than 32 processors, set both the **affinity I/O mask** and the **affinity64 I/O mask**.

The values for **affinity I/O mask** are as follows:

- A 1-byte **affinity I/O mask** covers up to 8 CPUs in a multiprocessor computer.
- A 2-byte **affinity I/O mask** covers up to 16 CPUs in a multiprocessor computer.
- A 3-byte **affinity I/O mask** covers up to 24 CPUs in a multiprocessor computer.
- A 4-byte **affinity I/O mask** covers up to 32 CPUs in a multiprocessor computer.
- To cover more than 32 CPUs, configure a four-byte **affinity I/O mask** for the first 32 CPUs and up to a four-byte **affinity64 I/O mask** for the remaining CPUs.

  A 1 bit in the affinity I/O pattern specifies that the corresponding CPU is eligible to perform SQL Server disk I/O operations; a 0 bit specifies that no SQL Server disk I/O operations should be scheduled for the corresponding CPU. When all bits are set to zero, or **affinity I/O mask** is not specified, SQL Server disk I/O is scheduled to any of the CPUs eligible to process SQL Server threads.

  Because setting the SQL Server **affinity I/O mask** option is a specialized operation, it should be used only when necessary. In most cases, the Windows 2000 or Windows Server 2003 default affinity provides the best performance.

When specifying the **affinity I/O mask** option, you must use it with the **affinity mask** configuration option. Do not enable the same CPU in both the **affinity I/O mask** switch and the **affinity mask** option. The bits corresponding to each CPU should be in one of the following three states:

- 0 in both the **affinity I/O mask** option and the **affinity mask** option.
- 1 in the **affinity I/O mask** option and 0 in the **affinity mask** option.
- 0 in the **affinity I/O mask** option and 1 in the **affinity mask** option.

The **affinity I/O mask** option is an advanced option. If you are using the **sp_configure** system stored procedure to change the setting, you can change **affinity I/O mask** only when **show advanced options** is set to 1. In SQL Server, reconfiguring the **affinity I/O mask** option requires a restart of the SQL Server instance.

Caution

Do not configure CPU affinity in the Windows operating system and also configure the affinity mask in SQL Server. These settings are attempting to achieve the same result, and if the configurations are inconsistent, you may have unpredictable results. SQL Server CPU affinity is best configured using the **sp_configure** option in SQL Server.

See Also

Monitor Resource Usage (System Monitor)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

affinity64 Input-Output mask Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

The **affinity64 I/O mask** binds SQL Server disk I/O to a specified subset of CPUs, similar to the **affinity I/O mask** option. Use **affinity I/O mask** to bind the first 32 processors, and use **affinity64 I/O mask** to bind the remaining processors on the computer. If you reconfigure the **affinity64 I/O mask**, you must restart the instance of SQL Server. This option is only visible on the 64-bit version of SQL Server.

See Also

affinity Input-Output mask Server Configuration Option
Monitor Resource Usage (System Monitor)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)

affinity mask Server Configuration Option

3/24/2017 • 8 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

> **NOTE**
>
> This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. Use ALTER SERVER CONFIGURATION (Transact-SQL) instead.

To carry out multitasking, Microsoft Windows sometimes move process threads among different processors. Although efficient from an operating system point of view, this activity can reduce SQL Server performance under heavy system loads, as each processor cache is repeatedly reloaded with data. Assigning processors to specific threads can improve performance under these conditions by eliminating processor reloads and reducing thread migration across processors (thereby reducing context switching); such an association between a thread and a processor is called processor affinity.

SQL Server supports processor affinity by means of two affinity mask options: affinity mask (also known as **CPU affinity mask**) and affinity I/O mask. For more information on the affinity I/O maskoption, see affinity Input-Output mask Server Configuration Option. CPU and I/O affinity support for servers with 33 to 64 processors requires the additional use of the affinity64 mask Server Configuration Option and affinity64 Input-Output mask Server Configuration Option, respectively.

> **NOTE**
>
> Affinity support for servers with 33 to 64 processors is only available on 64-bit operating systems.

The affinity mask option, which existed in earlier releases of SQL Server, dynamically controls CPU affinity.

In SQL Server, the affinity mask option can be configured without requiring a restart of the instance of SQL Server. When you are using sp_configure, you must run either RECONFIGURE or RECONFIGURE WITH OVERRIDE after setting a configuration option. When you are using SQL Server Express, changing the affinity mask option does require a restart.

Changes to the affinity masks occur dynamically, allowing for on-demand startup and shutdown of the CPU schedulers that bind process threads within SQL Server. This can occur as conditions change on the server. For example, if a new instance of SQL Server is added to the server, it may be necessary to make adjustments to the affinity mask option to redistribute processor load.

Modifications to the affinity bitmasks require SQL Server to enable a new CPU scheduler and disable the existing CPU scheduler. New batches can then be processed on the new or remaining schedulers.

To start a new CPU scheduler, SQL Server creates a new scheduler and adds it to the list of its standard schedulers. The new scheduler is considered only for the new incoming batches. Current batches continue to run on the same scheduler. The workers migrate to the new scheduler as they free up, or as new workers are created.

Shutting down a scheduler requires all batches on the scheduler to complete their activities and exit. A scheduler that has been shut down is marked as offline so that no new batch is scheduled on it.

Whether a new scheduler is added or removed, the permanent system tasks such as lockmonitor, checkpoint, system task thread (processing DTC), and signal process continue to run on the scheduler while the server is operational. These permanent system tasks do not dynamically migrate. To redistribute processor load for these system tasks across schedulers, it is necessary to restart the SQL Server instance. If SQL Server attempts to shut down a scheduler associated with a permanent system task, the task continues to run on the offline scheduler (no migration). This scheduler is bound to the processors in the modified affinity mask and should not put any load on the processor it was affinitized with before the change. Having extra offline schedulers, should not significantly affect the load of the system. If this is not the case, a database server reboot is required to reconfigure these tasks.

The I/O affinity tasks (such as lazywriter and logwriter) are directly affected by the I/O affinity mask. If the lazywriter and logwriter tasks are not affinitized, they follow the same rules defined for the other permanent tasks such as lockmonitor or checkpoint.

To ensure that the new affinity mask is valid, the RECONFIGURE command verifies that the normal CPU and I/O affinities are mutually exclusive. If this is not the case, an error message is reported to the client session and to the SQL Server error log, indicating that such a setting is not recommended. Running RECONFIGURE WITH OVERRIDE options allows CPU and I/O affinities that are not mutually exclusive.

If you specify an affinity mask that attempts to map to a nonexistent CPU, the RECONFIGURE command reports an error message to both the client session and the SQL Server error log. Using the RECONFIGURE WITH OVERRIDE option has no effect in this case, and the same configuration error is reported again.

You can also exclude SQL Server activity from processors assigned specific workload assignments by the Windows 2000 or Windows Server 2003 operating system. If you set a bit representing a processor to 1, that processor is selected by the SQL Server Database Engine for thread assignment. When you set **affinity mask** to 0 (the default), the Microsoft Windows 2000 or Windows Server 2003 scheduling algorithms set the thread's affinity. When you set **affinity mask** to any nonzero value, SQL Server affinity interprets the value as a bitmask that specifies those processors eligible for selection.

By segregating SQL Server threads from running on particular processors, Microsoft Windows 2000 or Windows Server 2003 can better evaluate the system's handling of processes specific to Windows. For example, on an 8-CPU server running two instances of SQL Server (instance A and B), the system administrator could use the affinity mask option to assign the first set of 4 CPUs to instance A and the second set of 4 to instance B. To configure more than 32 processors, set both the affinity mask and the affinity64 mask. The values for **affinity mask** are as follows:

- A one-byte **affinity mask** covers up to 8 CPUs in a multiprocessor computer.
- A two-byte **affinity mask** covers up to 16 CPUs in a multiprocessor computer.
- A three-byte **affinity mask** covers up to 24 CPUs in a multiprocessor computer.
- A four-byte **affinity mask** covers up to 32 CPUs in a multiprocessor computer.
- To cover more than 32 CPUs, configure a four-byte affinity mask for the first 32 CPUs and up to a four-byte affinity64 mask for the remaining CPUs.

  Because setting SQL Server processor affinity is a specialized operation, it is recommended that it be used only when necessary. In most cases, the Microsoft Windows 2000 or Windows Server 2003 default affinity provides the best performance. You should also consider the CPU requirements for other applications when setting the affinity masks. For more information, see your Windows operating system documentation.

NOTE

You can use the Windows System Monitor to view and analyze individual processor usage.

When specifying the affinity I/O mask option, you must use it in connection with the affinity mask configuration option. Do not enable the same CPU in both the **affinity mask** switch and the affinity I/O mask option. The bits corresponding to each CPU should be in one of these three states:

- 0 in both the affinity mask option and the affinity I/O mask option.
- 1 in the affinity mask option and 0 in the affinity I/O mask option.
- 0 in the affinity mask option and 1 in the affinity I/O mask option.

Caution

Do not configure CPU affinity in the Windows operating system and also configure the affinity mask in SQL Server. These settings are attempting to achieve the same result, and if the configurations are inconsistent, you may have unpredictable results. SQL Server CPU affinity is best configured using the sp_configure option in SQL Server.

Example

As an example of setting the affinity mask option, if processors 1, 2, and 5 are selected as available with bits 1, 2, and 5 set to 1 and bits 0, 3, 4, 6, and 7 set to 0, a hexadecimal value of 0x26 or the decimal equivalent of `38` is specified. Number the bits from right to left. The affinity mask option starts counting processors from 0 to 31, so that in the following example the counter `1` represents the second processor on the server.

```
sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
sp_configure 'affinity mask', 38;
RECONFIGURE;
GO
```

These are **affinity mask** values for an 8-CPU system.

DECIMAL VALUE	BINARY BIT MASK	ALLOW SQL SERVER THREADS ON PROCESSORS
1	00000001	0
3	00000011	0 and 1
7	00000111	0, 1, and 2
15	00001111	0, 1, 2, and 3
31	00011111	0, 1, 2, 3, and 4
63	00111111	0, 1, 2, 3, 4, and 5
127	01111111	0, 1, 2, 3, 4, 5, and 6
255	11111111	0, 1, 2, 3, 4, 5, 6, and 7

The affinity mask option is an advanced option. If you are using the sp_configure system stored procedure to change the setting, you can change **affinity mask** only when **show advanced options** is set to 1. After executing the Transact-SQL RECONFIGURE command, the new setting takes effect immediately without requiring a restart of the SQL Server instance.

Non-uniform Memory Access (NUMA)

When using hardware based non-uniform memory access (NUMA) and the affinity mask is set, every scheduler in a node will be affinitized to its own CPU. When the affinity mask is not set, each scheduler is affinitized to the group of CPUs within the NUMA node and a scheduler mapped to NUMA node N1 can schedule work on any CPU in the node, but not on CPUs associated with another node.

Any operation running on a single NUMA node can only use buffer pages from that node. When an operation is run in parallel on CPUs from multiple nodes, memory can be used from any node involved.

Licensing Issues

Dynamic affinity is tightly constrained by CPU licensing. SQL Server does not allow any configuration of affinity mask options that violates the licensing policy.

Startup

If a specified affinity mask violates the licensing policy during SQL Server startup or during database attach, the engine layer will complete the startup process or database attach/restore operation, and then it will reset the sp_configure run value for the affinity mask to zero, issuing an error message to the SQL Server error log.

Reconfigure

If a specified affinity mask violates the licensing policy when running Transact-SQL RECONFIGURE command, an error message is reported to the client session and to the SQL Server error log, requiring the database administrator to reconfigure the affinity mask. No RECONFIGURE WITH OVERRIDE command is accepted in this case.

See Also

Monitor Resource Usage (System Monitor)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
ALTER SERVER CONFIGURATION (Transact-SQL)

affinity64 mask Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

The affinity64 mask binds processors to specific threads, similar to the affinity mask option. Use affinity mask to bind the first 32 processors, and use affinity64 mask to bind the remaining processors on the computer. This option is only visible on the 64-bit version of SQL Server.

NOTE

This feature will be removed in the next version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature. Use ALTER SERVER CONFIGURATION (Transact-SQL) instead.

See Also

affinity mask Server Configuration Option
Monitor Resource Usage (System Monitor)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)

Agent XPs Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **Agent XPs** option to enable the SQL Server Agent extended stored procedures on this server. When this option is not enabled, the SQL Server Agent node is not available in SQL Server Management Studio Object Explorer.

When you use the SQL Server Management Studio tool to start the SQL Server Agent service, these extended stored procedures are enabled automatically. For more information, see Surface Area Configuration.

> **NOTE**
>
> Management Studio Object Explorer does not display the contents of the SQL ServerAgent node unless these extended stored procedures are enabled regardless of the SQL Server Agent service state.

The possible values are:

- **0**, indicating that SQL Server Agent extended stored procedures are not available (the default).
- **1**, indicating that SQL Server Agent extended stored procedures are available.

  The setting takes effect immediately without a server stop and restart.

Example

The following example enables the SQL Server Agent extended stored procedures.

1. From Microsoft SQL Server Management Studio connect to the Database Engine.
2. From the Standard bar, click **New Query.**
3. Copy and paste the following example into the query window and click **Execute**.

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Agent XPs', 1;
GO
RECONFIGURE
GO
```

See Also

Automated Administration Tasks (SQL Server Agent)
Start, Stop, or Pause the SQL Server Agent Service

allow updates Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This option is still present in the **sp_configure** stored procedure, although its functionality is unavailable in SQL Server. The setting has no effect. Direct updates to the system tables are not supported.

> **IMPORTANT**
>
> This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible.

Changing the **allow updates** option will cause the RECONFIGURE statement to fail. Changes to the **allow updates** option should be removed from all scripts.

See Also

Server Configuration Options (SQL Server)

backup checksum default

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the backup checksum default setting to enable or disable backup checksum during backup and restore.

The possible values are described in the following table:

Value	**Meaning**
0	Disabled. This is the default setting.
1	Enabled

The setting takes effect immediately.

See Also

Enable or Disable Backup Checksums During Backup or Restore (SQL Server)

View or Configure the backup compression default Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to view or configure the **backup compression default** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **backup compression default** option determines whether the server instance creates compressed backups by default. When SQL Server is installed, the **backup compression default** option is off.

In This Topic

Before You Begin

Limitations and Restrictions

- Backup compression is not available in all editions of SQL Server. For more information, see Features Supported by the Editions of SQL Server 2016.

- By default, compression significantly increases CPU usage, and the additional CPU consumed by the compression process might adversely impact concurrent operations. Therefore, you might want to create low-priority compressed backups in a session whose CPU usage is limited by Resource Governor. For more information, see Use Resource Governor to Limit CPU Usage by Backup Compression (Transact-SQL).

Recommendations

- When you are creating an individual backup, configuring a log shipping configuration, or creating a maintenance plan, you can override the server-level default.

- Backup compression is supported for both disk backup devices and tape backup devices.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To view or configure the backup compression default option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Database settings** node.

3. Under **Backup and restore**, **Compress backup** shows the current setting of the **backup compression default** option. This setting determines the server-level default for compressing backups, as follows:

   - If the **Compress backup** box is blank, new backups are uncompressed by default.

   - If the **Compress backup** box is checked, new backups are compressed by default.

     If you are a member of the **sysadmin** or **serveradmin** fixed server role, you can also change the default setting by clicking the **Compress backup** box.

Using Transact-SQL

To view the backup compression default option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example queries the sys.configurations catalog view to determine the value for `backup compression default`. A value of 0 means that backup compression is off, and a value of 1 means that backup compression is enabled.

```
SELECT value
FROM sys.configurations
WHERE name = 'backup compression default' ;
GO
```

To configure the backup compression default option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the server instance to create compressed backups by default.

```
EXEC sp_configure 'backup compression default', 1 ;
RECONFIGURE WITH OVERRIDE ;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the backup compression default option

The setting takes effect immediately without restarting the server.

See Also

BACKUP (Transact-SQL)
Server Configuration Options (SQL Server)
RECONFIGURE (Transact-SQL)
sp_configure (Transact-SQL)

Backup Overview (SQL Server)

blocked process threshold Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **blocked process threshold** option to specify the threshold, in seconds, at which blocked process reports are generated. The threshold can be set from 0 to 86,400. By default, no blocked process reports are produced. This event is not generated for system tasks or for tasks that are waiting on resources that do not generate detectable deadlocks.

You can define an alert to be executed when this event is generated. So for example, you can choose to page the administrator to take appropriate action to handle the blocking situation.

Blocked process threshold uses the deadlock monitor background thread to walk through the list of tasks waiting for a time greater than or multiples of the configured threshold. The event is generated once per reporting interval for each of the blocked tasks.

The blocked process report is done on a best effort basis. There is no guarantee of any real-time or even close to real-time reporting.

The setting takes effect immediately without a server stop and restart.

Examples

The following example sets the `blocked process threshold` to `20` seconds, generating a blocked process report for each task that is blocked.

```
sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE ;
GO
sp_configure 'blocked process threshold', 20 ;
GO
RECONFIGURE ;
GO
```

See Also

sp_trace_setevent (Transact-SQL)
Blocked Process Report Event Class

c2 audit mode Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

C2 audit mode can be configured through SQL Server Management Studio or with the **c2 audit mode** option in **sp_configure**. Selecting this option will configure the server to record both failed and successful attempts to access statements and objects. This information can help you profile system activity and track possible security policy violations.

NOTE

This feature will be removed in a future version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature. The C2 security standard has been superseded by Common Criteria Certification. See the common criteria compliance enabled Server Configuration Option.

Audit Log File

C2 audit mode data is saved in a file in the default data directory of the instance. If the audit log file reaches its size limit of 200 megabytes (MB), SQL Server will create a new file, close the old file, and write all new audit records to the new file. This process will continue until the audit data directory fills up or auditing is turned off. To determine the status of a C2 trace, query the sys.traces catalog view.

IMPORTANT

C2 audit mode saves a large amount of event information to the log file, which can grow quickly. If the data directory in which logs are being saved runs out of space, SQL Server will shut itself down. If auditing is set to start automatically, you must either restart the instance with the **-f** flag (which bypasses auditing), or free up additional disk space for the audit log.

Permissions

Requires membership in the **sysadmin** fixed server role.

Example

The following example turns on C2 audit mode.

```
sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE ;
GO

sp_configure 'c2 audit mode', 1 ;
GO
RECONFIGURE ;
GO
```

See Also

RECONFIGURE (Transact-SQL)

Server Configuration Options (SQL Server)

sp_configure (Transact-SQL)

clr enabled Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the clr enabled option to specify whether user assemblies can be run by SQL Server. The clr enabled option provides the following values:

VALUE	DESCRIPTION
0	Assembly execution not allowed on SQL Server.
1	Assembly execution allowed on SQL Server.

WOW64 only. Restart WOW64 servers to effect the settings changes. No restart required for other server types.

When you run RECONFIGURE, and the run value of the clr enabled option is changed from 1 to 0, all application domains containing user assemblies are immediately unloaded.

> **Common language runtime (CLR) execution is not supported under lightweight pooling** Disable one of two options: "clr enabled" or "lightweight pooling. Features that rely upon CLR and that do not work properly in fiber mode include the **hierarchy** data type, replication, and Policy-Based Management.

Example

The following example first displays the current setting of the clr enabled option and then enables the option by setting the option value to 1. To disable the option, set the value to 0.

```
EXEC sp_configure 'clr enabled';
EXEC sp_configure 'clr enabled' , '1';
RECONFIGURE;
```

See Also

lightweight pooling Server Configuration Option
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
lightweight pooling Server Configuration Option

common criteria compliance enabled Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

The common criteria compliance enabled option enables the following elements that are required for the Common Criteria.

CRITERIA	DESCRIPTION
Residual Information Protection (RIP)	RIP requires a memory allocation to be overwritten with a known pattern of bits before memory is reallocated to a new resource. Meeting the RIP standard can contribute to improved security; however, overwriting the memory allocation can slow performance. After the common criteria compliance enabled option is enabled, the overwriting occurs.
The ability to view login statistics	After the common criteria compliance enabled option is enabled, login auditing is enabled. Each time a user successfully logs in to SQL Server, information about the last successful login time, the last unsuccessful login time, and the number of attempts between the last successful and current login times is made available. These login statistics can be viewed by querying the sys.dm_exec_sessions dynamic management view.
That column `GRANT` should not override table `DENY`	After the common criteria compliance enabled option is enabled, a table-level `DENY` takes precedence over a column-level `GRANT`. When the option is not enabled, a column-level `GRANT` takes precedence over a table-level `DENY`.

The common criteria compliance enabled option is an advanced option. Common criteria is only evaluated and certified for the Enterprise edition and Datacenter edition. For the latest status of common criteria certification, see the Microsoft SQL Server Common Criteria Web site.

IMPORTANT
In addition to enabling the common criteria compliance enabled option, you also must download and run a script that finishes configuring SQL Server to comply with Common Criteria Evaluation Assurance Level 4+ (EAL4+). You can download this script from the Microsoft SQL Server Common Criteria Web site.

If you are using the `sp_configure` system stored procedure to change the setting, you can change common criteria compliance enabled only when show advanced options is set to 1. The setting takes effect after the server is restarted. The possible values are 0 and 1:

- 0 indicates that common criteria compliance is not enabled. This is the default.
- 1 indicates that common criteria compliance is enabled.

Examples

The following example enables common criteria compliance.

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'common criteria compliance enabled', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

Restart SQL Server.

See Also

Server Configuration Options (SQL Server)

contained database authentication Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **contained database authentication** option to enable contained databases on the instance of SQL Server Database Engine.

This server option allows you to control **contained database authentication**.

- When **contained database authentication** is off (0) for the instance, contained databases cannot be created, or attached to the Database Engine.
- When **contained database authentication** is on (1) for the instance, contained databases can be created, or attached to the Database Engine.

  A contained database includes all database settings and metadata required to define the database and has no configuration dependencies on the instance of the Database Engine where the database is installed. Users can connect to the database without authenticating a login at the Database Engine level. Isolating the database from the Database Engine makes it possible to easily move the database to another instance of SQL Server. Including all the database settings in the database enables database owners to manage all the configuration settings for the database. For more information about contained databases, see Contained Databases.

NOTE

Contained databases are always enabled for SQL Database and SQL Data Warehouse and cannot be disabled.

If an instance of SQL Server has any contained databases the **contained database authentication** setting can be set to 0 by using the **RECONFIGURE WITH OVERRIDE** statement. Setting **contained database authentication** to 0 will disable contained database authentication for the contained databases.

IMPORTANT

When contained databases are enabled, database users with the ALTER ANY USER permission, such as members of the db_owner and db_accessadmin database roles, can grant access to databases and by doing so, grant access to the instance of SQL Server. This means that control over access to the server is no longer limited to members of the sysadmin and securityadmin fixed server role, and logins with the server level CONTROL SERVER and ALTER ANY LOGIN permission. Before allowing contained databases, you should understand the risks associated with contained databases. For more information, see Security Best Practices with Contained Databases.

Examples

The following example enables contained databases on the instance of the Database Engine.

```
sp_configure 'contained database authentication', 1;
GO
RECONFIGURE;
GO
```

See Also

sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)

Configure the cost threshold for parallelism Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **cost threshold for parallelism** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **cost threshold for parallelism** option specifies the threshold at which SQL Server creates and runs parallel plans for queries. SQL Server creates and runs a parallel plan for a query only when the estimated cost to run a serial plan for the same query is higher than the value set in **cost threshold for parallelism**. The cost refers to an estimated cost required to run the serial plan on a specific hardware configuration, and is not a unit of time. The **cost threshold for parallelism** option can be set to any value from 0 through 32767. The default value is 5.

In This Topic

Before You Begin

Limitations and Restrictions

- The cost refers to an abstracted unit of cost and not a unit of estimated time. Only set **cost threshold for parallelism** on symmetric multiprocessors.
- SQL Server ignores the **cost threshold for parallelism** value under the following conditions:
  - Your computer has only one logical processor.
  - Only a single logical processor is available to SQL Server because of the **affinity mask** configuration option.
  - The **max degree of parallelism** option is set to 1.

  A logical processor is the basic unit of processor hardware that allows the operating system to dispatch a task or execute a thread context. Each logical processor can execute only one thread context at a time. The processor core is the circuitry that provides ability to decode and execute instructions. A processor core may contain one or more logical processors. The following Transact-SQL query can be used for obtaining CPU information for the system.

```
SELECT (cpu_count / hyperthread_ratio) AS PhysicalCPUs,
cpu_count AS logicalCPUs
FROM sys.dm_os_sys_info
```

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.
- In certain cases, a parallel plan may be chosen even though the query's cost plan is less than the current **cost threshold for parallelism** value. This can happen because the decision to use a parallel or serial plan is based on a cost estimate provided before the full optimization is complete.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the cost threshold for parallelism option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Advanced** node.
3. Under **Parallelism**, change the **CostThresholdForParallelism** option to the value you want. Type or select a value from 0 to 32767.

Using Transact-SQL

To configure the cost threshold for parallelism option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `cost threshold for parallelism` option to `10`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE
GO
EXEC sp_configure 'cost threshold for parallelism', 10 ;
GO
RECONFIGURE
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the cost threshold for parallelism option

The setting takes effect immediately without restarting the server.

See Also

Configure Parallel Index Operations
Query Hints (Transact-SQL)
ALTER WORKLOAD GROUP (Transact-SQL)
affinity mask Server Configuration Option
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the cursor threshold Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure the **cursor threshold** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **cursor threshold** option specifies the number of rows in the cursor set at which cursor keysets are generated asynchronously. When cursors generate a keyset for a result set, the query optimizer estimates the number of rows that will be returned for that result set. If the query optimizer estimates that the number of returned rows is greater than this threshold, the cursor is generated asynchronously, allowing the user to fetch rows from the cursor while the cursor continues to be populated. Otherwise, the cursor is generated synchronously, and the query waits until all rows are returned.

In This Topic

Before You Begin

Limitations and Restrictions

- SQL Server does not support generating keyset-driven or static Transact-SQL cursors asynchronously. Transact-SQL cursor operations such as OPEN or FETCH are batched, so there is no need for the asynchronous generation of Transact-SQL cursors. SQL Server continues to support asynchronous keyset-driven or static application programming interface (API) server cursors where low latency OPEN is a concern, due to client round trips for each cursor operation.

- The accuracy of the query optimizer to determine an estimate for the number of rows in a keyset depends on the currency of the statistics for each of the tables in the cursor.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- If you set **cursor threshold** to -1, all keysets are generated synchronously, which benefits small cursor sets. If you set **cursor threshold** to 0, all cursor keysets are generated asynchronously. With other values, the query optimizer compares the number of expected rows in the cursor set and builds the keyset asynchronously if it exceeds the number set in **cursor threshold**. Do not set **cursor threshold** too low, because small result sets are better built synchronously.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the cursor threshold option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Advanced** node.
3. Under **Miscellaneous**, change the **Cursor Threshold** option to the value you want.

Using Transact-SQL

To configure the cursor threshold option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the `cursor threshold` option to `0` so that cursor keysets are generated asynchronously.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE
GO
EXEC sp_configure 'cursor threshold', 0 ;
GO
RECONFIGURE
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the cursor threshold option

The setting takes effect immediately without restarting the server.

See Also

@@CURSOR_ROWS (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
UPDATE STATISTICS (Transact-SQL)

cross db ownership chaining Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **cross db ownership chaining** option to configure cross-database ownership chaining for an instance of Microsoft SQL Server.

This server option allows you to control cross-database ownership chaining at the database level or to allow cross-database ownership chaining for all databases:

- When **cross db ownership chaining** is off (0) for the instance, cross-database ownership chaining is disabled for all databases.
- When **cross db ownership chaining** is on (1) for the instance, cross-database ownership chaining is on for all databases.
- You can set cross-database ownership chaining for individual databases using the SET clause of the ALTER DATABASE statement. If you are creating a new database, you can set the cross-database ownership chaining option for the new database using the CREATE DATABASE statement.

  Setting **cross db ownership chaining** to 1 is not recommended unless all of the databases hosted by the instance of SQL Server must participate in cross-database ownership chaining and you are aware of the security implications of this setting.

Controlling Cross-Database Ownership Chaining

Before turning cross-database ownership chaining on or off, consider the following:

- You must be a member of the **sysadmin** fixed server role to turn cross-database ownership chaining on or off.
- Before turning off cross-database ownership chaining on a production server, fully test all applications, including third-party applications, to ensure that the changes do not affect application functionality.
- You can change the **cross db ownership chaining** option while the server is running if you specify RECONFIGURE with **sp_configure**.
- If you have databases that require cross-database ownership chaining, the recommended practice is to turn off the **cross db ownership chaining** option for the instance using **sp_configure**; then turn on cross-database ownership chaining for individual databases that require it using the ALTER DATABASE statement.

See Also

ALTER DATABASE (Transact-SQL)
CREATE DATABASE (SQL Server Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)

Database Mail XPs Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **DatabaseMail XPs** option to enable Database Mail on this server. The possible values are:

- **0** indicating Database Mail is not available (default).
- **1** indicating Database Mail is available.

  The setting takes effect immediately without a server stop and restart.

  After enabling Database Mail, you must configure a Database Mail host database to use Database Mail.

  Configuring Database Mail using the **Database Mail Configuration Wizard** enables the Database Mail extended stored procedures in the **msdb** database. If you use the **Database Mail Configuration Wizard**, you do not have to use the **sp_configure** example below.

  Setting the **Database Mail XPs** option to 0 prevents Database Mail from starting. If it is running when the option is set to 0, it continues to run and send mail until it is idle for the time configured in the **DatabaseMailExeMinimumLifeTime** option.

Examples

The following example enables the Database Mail extended stored procedures.

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Database Mail XPs', 1;
GO
RECONFIGURE
GO
```

See Also

Database Mail

Configure the default full-text language Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **default full-text language** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **default full-text language** option specifies a default language value for full-text indexes. Linguistic analysis is performed on all data that is full-text indexed and is dependent on the language of the data. The default value of this option is the language of the server. For a localized version of SQL Server, SQL Server Setup sets the **default full-text language** option to the language of the server if an appropriate match exists. For a non-localized version of SQL Server, the **default full-text language** option is English.

In This Topic

Before You Begin

Limitations and Restrictions

- The value of the **default full-text language** option is used in a full-text index when no language is specified for a column through the LANGUAGE **language_term** option in the CREATE FULLTEXT INDEX or ALTER FULLTEXT INDEX statements. If the default full-text language is not supported or the linguistic analysis package is not available, the CREATE or ALTER operation will fail and SQL Server will return an error message stating that the language specified is not valid.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- The **default full-text language** option requires an LCID value. For a list of supported LCIDs and their related languages, see sys.fulltext_languages (Transact-SQL). Other languages may also be available from independent software vendors, for example. If no specific language dialect is found, the Full-Text Engine will automatically switch to the primary language.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the default full-text language option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Advanced** node.
3. Under Miscellaneous, use **Default Full Text Language** to specify a default language value for full-text indexed columns.

Using Transact-SQL

To configure the default full-text language option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `default full-text` option to Dutch (`1043`).

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE
GO
EXEC sp_configure 'default full-text language', 1043 ;
GO
RECONFIGURE
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the default full-text language option

The setting takes effect immediately without restarting the server.

See Also

sys.fulltext_languages (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
CREATE FULLTEXT INDEX (Transact-SQL)
ALTER FULLTEXT INDEX (Transact-SQL)

Configure the default language Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **default language** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **default language** option specifies the default language for all newly created logins. To set default language, specify the **langid** value of the language you want. The **langid** value can be obtained by querying the **sys.syslanguages** compatibility view.

In This Topic

Before You Begin

Recommendations

- The default language for a login can be overridden by using CREATE LOGIN or ALTER LOGIN. The default language for a session is the language for that session's login, unless overridden on a per-session basis by using the Open Database Connectivity (ODBC) or OLE DB APIs. Note that you can only set the **default language** option to a language ID defined in sys.syslanguages (0-32). When you are using contained databases, a default language can be set for a database by using CREATE DATABASE or ALTER DATABASE, and for contained database users by using CREATE USER or ALTER USER. Setting default languages in a contained database accepts **langid** value, the language name, or a language alias as listed in **sys.syslanguages**.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the default language option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **General settings** node.

3. In the **Default language for users** box, choose the language in which Microsoft SQL Server should display system messages.

   The default language is English.

Using Transact-SQL

To configure the default language option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the `default language` option to French (`2`).

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'default language', 2 ;
GO
RECONFIGURE ;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the default language option

The setting takes effect immediately without restarting the server.

See Also

CREATE LOGIN (Transact-SQL)
ALTER LOGIN (Transact-SQL)
CREATE USER (Transact-SQL)
ALTER USER (Transact-SQL)
CREATE DATABASE (SQL Server Transact-SQL)
ALTER DATABASE (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

default trace enabled Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **default trace enabled** option to enable or disable the default trace log files. The default trace functionality provides a rich, persistent log of activity and changes primarily related to the configuration options.

WARNING

This feature will be removed in a future version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature. Use Extended Events instead.

Purpose

Default trace provides troubleshooting assistance to database administrators by ensuring that they have the log data necessary to diagnose problems the first time they occur.

Viewing

The default trace logs can be opened and examined by SQL Server Profiler or queried with Transact-SQL by using the `fn_trace_gettable` system function. SQL Server Profiler can open the default trace log files just as it does normal trace output files. The default trace log is stored by default in the `\MSSQL\LOG` directory using a rollover trace file. The base file name for the default trace log file is `log.trc`. In a typical installation of SQL Server, the default trace is enabled and thus becomes TraceID 1. If enabled after installation and after creating other traces, the TraceID can become a larger number.

For more information about using SQL Server Profiler to view this trace file, see Open a Trace File (SQL Server Profiler)

Example:

The following statement opens the default trace log in the default location:

```
SELECT *
FROM fn_trace_gettable
('C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\LOG\log.trc', default);
GO
```

Configuring

When set to 1, the **default trace enabled** option enables **Default Trace**. The default setting for this option is 1 (ON). A value of 0 turns off the trace.

The **default trace enabled** option is an advanced option. If you are using the **sp_configure** system stored procedure to change the setting, you can change the **default trace enabled** option only when **show advanced options** is set to 1. The setting takes effect immediately without a server restart.

See Also

RECONFIGURE (Transact-SQL)

Server Configuration Options (SQL Server)

sp_configure (Transact-SQL)

disallow results from triggers Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **disallow results from triggers** option to control whether triggers return result sets. Triggers that return result sets may cause unexpected behavior in applications that are not designed to work with them.

> **IMPORTANT**
>
> This feature will be removed in the next version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. We recommend that you set this value to 1.

When set to 1, the **disallow results from triggers** option is set to ON. The default setting for this option is 0 (OFF). If this option is set to 1 (ON), any attempt by a trigger to return a result set fails, and the user receives the following error message:

"Msg 524, Level 16, State 1, Procedure <Procedure Name>, Line <Line#>

"A trigger returned a resultset and the server option 'disallow_results_from_triggers' is true."

The **disallow results from triggers** option is applied at the Microsoft SQL Server instance level, and it will determine behavior for all existing triggers within the instance.

The **disallow results from triggers** option is an advanced option. If you are using the **sp_configure** system stored procedure to change the setting, you can change disallow results from triggers only when **show advanced options** is set to 1. The setting takes effect immediately without a server restart.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

EKM provider enabled Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

The **EKM provider enabled** option controls Extensible Key Management device support in SQL Server. By default this option is off.

To enable or disable the feature, issue one of the following **sp_configure** commands:

```
/* Enable the external encryption provider option */
sp_configure 'EKM provider enabled', 1
/* Disable the external encryption provider option */
sp_configure 'EKM provider enabled', 0
```

NOTE

This option is not enabled in every edition of Microsoft SQL Server. For a list of features that are supported by the editions of SQL Server, see Features Supported by the Editions of SQL Server 2016.

See Also

Extensible Key Management (EKM)
Server Configuration Options (SQL Server)
Monitor Resource Usage (System Monitor)
sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)

external scripts enabled Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2016) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **external scripts enabled** option to enable the execution of scripts with certain remote language extensions. This property will be OFF by default. Setup can optionally set this property to true if **Advanced Analytics Services** is installed.

You must enable the external script enabled option before you can execute an external script using the **sp_execute_external_script** procedure. Use **sp_execute_external_script** to execute scripts written in a supported language such as R. In SQL Server 2016, R Services (In-Database) is comprised of a server component installed with SQL Server 2016, and a set of workstation tools and connectivity libraries that connect the data scientist to the high-performance environment of SQL Server. Install the **Advanced Analytics Extensions** feature during SQL Server setup to enable the execution of R scripts. For more information, see Installing Previous Versions of SQL Server R Services.

Execute the following script to enable external scripts.

```
sp_configure 'external scripts enabled', 1;
RECONFIGURE;
```

You must restart SQL Server to make this change effective.

See Also

sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)
sp_execute_external_script (Transact-SQL)
SQL Server R Services

filestream access level Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the filestream_access_level option to change the FILESTREAM access level for this instance of SQL Server.

NOTE

Before this option has any effect, the Windows administration settings for FILESTREAM must be enabled. You can enable these settings when you install SQL Server or by using SQL Server Configuration Manager.

VALUE	DEFINITION
0	Disables FILESTREAM support for this instance.
1	Enables FILESTREAM for Transact-SQL access.
2	Enables FILESTREAM for Transact-SQL and Win32 streaming access.

See Also

Database Engine Configuration - Filestream
Enable and Configure FILESTREAM

Configure the fill factor Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **fill factor** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. Fill factor is provided for fine-tuning index data storage and performance. When an index is created or rebuilt, the fill-factor value determines the percentage of space on each leaf-level page to be filled with data, reserving the rest as free space for future growth. For more information, see Specify Fill Factor for an Index.

In This Topic

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the fill factor option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Database Settings** node.

3. In the **Default index fill factor** box, type or select the index fill factor that you want.

Using Transact-SQL

To configure the fill factor option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `fill factor` option to `100`.

```
Use AdventureWorks2012;
GO
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'fill factor', 100;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the fill factor option

The server must be restarted before the setting can take effect.

See Also

RECONFIGURE (Transact-SQL)
ALTER INDEX (Transact-SQL)
CREATE INDEX (Transact-SQL)
Specify Fill Factor for an Index
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
sys.indexes (Transact-SQL)

ft crawl bandwidth Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **ft crawl bandwidth** option to specify the size to which the pool of large memory buffers can grow. Large memory buffers are 4 megabytes (MB) in size. The **max** parameter value specifies the maximum number of buffers that the full-text memory manager should maintain in a large buffer pool. If the **max** value is zero, then there is no upper limit to the number of buffers that can be in a large buffer pool.

The **min** parameter specifies the minimum number of memory buffers that must be maintained in the pool of large memory buffers. Upon request from the Microsoft SQL Server memory manager, all extra buffer pools will be released but this minimum number of buffers will be maintained. If, however, the **min** value specified is zero, then all memory buffers are released.

Under certain circumstances, the number of buffers currently allocated is less than the value specified by the **min** parameter.

NOTE

This feature will be removed in a future version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature.

See Also

Server Configuration Options (SQL Server)
ft notify bandwidth Server Configuration Option
sp_configure (Transact-SQL)

ft notify bandwidth Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **ft notify bandwidth** option to specify the size to which the pool of small memory buffers can grow. Small memory buffers are 64 kilobytes (KB) in size. The *max* parameter value specifies the maximum number of buffers that the full-text memory manager should maintain in a small buffer pool. If the **max** value is zero, then there is no upper limit to the number of buffers that can be in a small buffer pool.

The **min** parameter specifies the minimum number of memory buffers that must be maintained in the pool of small memory buffers. Upon request from the Microsoft SQL Server memory manager, all extra buffer pools will be released but this minimum number of buffers will be maintained. If, however, the **min** value specified is zero, then all memory buffers are released.

Under certain circumstances the number of buffers currently allocated is less than the value specified by the **min** parameter.

NOTE

This feature will be removed in a future version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature.

See Also

Server Configuration Options (SQL Server)
ft crawl bandwidth Server Configuration Option
sp_configure (Transact-SQL)

PolyBase Connectivity Configuration (Transact-SQL)

3/24/2017 • 4 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Displays or changes global configuration settings for PolyBase Hadoop and Azure blob storage connectivity.

Transact-SQL Syntax Conventions

Syntax

```
--List all of the configuration options
sp_configure
[;]

--Configure Hadoop connectivity
sp_configure [ @configname = ] 'hadoop connectivity',
             [ @configvalue = ] { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 }
[;]

RECONFIGURE
[;]
```

Arguments

[**@configname=**] '*option_name*'
Is the name of a configuration option. *option_name* is **varchar(35)**, with a default of NULL. If not specified, the complete list of options is returned.

[**@configvalue=**] '*value*'
Is the new configuration setting. *value* is **int**, with a default of NULL. The maximum value depends on the individual option.

'hadoop connectivity'
Specifies the type of Hadoop data source for all connections from PolyBase to Hadoop clusters or Azure blob storage (WASB). This setting is required in order to create an external data source for an external table. For more information, see CREATE EXTERNAL DATA SOURCE (Transact-SQL),

These are the Hadoop connectivity settings and their corresponding supported Hadoop data sources. Only one setting can be in effect at a time. Options 1, 4, and 7 allow multiple types of external data sources to be created and used across all sessions on the server.

- Option 0: Disable Hadoop connectivity
- Option 1: Hortonworks HDP 1.3 on Windows Server
- Option 1: Azure blob storage (WASB[S])
- Option 2: Hortonworks HDP 1.3 on Linux
- Option 3: Cloudera CDH 4.3 on Linux
- Option 4: Hortonworks HDP 2.0 on Windows Server

- Option 4: Azure blob storage (WASB[S])
- Option 5: Hortonworks HDP 2.0 on Linux
- Option 6: Cloudera 5.1, 5.2, 5.3, 5.4, 5.5,5.9, and 5.10 on Linux
- Option 7: Hortonworks 2.1, 2.2, 2.3, 2.4, and 2.5 on Linux
- Option 7: Hortonworks 2.1, 2.2, and 2.3 on Windows Server
- Option 7: Azure blob storage (WASB[S])

  RECONFIGURE
  Updates the run value (run_value) to match the configuration value (config_value). See Result Sets for definitions of run_value and config_value. The new configuration value that is set by sp_configure does not become effective until the run value is set by the RECONFIGURE statement.

  After running RECONFIGURE, you must stop and restart the SQL Server service. Note that when stopping the SQL Server service, the two additional PolyBase Engine and Data Movement Service will automatically stop. After restarting the SQL Server engine service, re-start these two services again (they won't start automatically).

Return Code Values

0 (success) or 1 (failure)

Result Sets

When executed with no parameters, **sp_configure** returns a result set with five columns.

COLUMN NAME	DATA TYPE	DESCRIPTION
name	**nvarchar(35)**	Name of the configuration option.
minimum	**int**	Minimum value of the configuration option.
maximum	**int**	Maximum value of the configuration option.
config_value	**int**	Value that was set using **sp_configure**.
run_value	**int**	Current value in use by PolyBase. This value is set by running RECONFIGURE. The **config_value** and **run_value** are usually the same unless the value is in the process of being changed. A restart might be required before this run value is accurate, if the reconfiguration is in progress.

General Remarks

In SQL Server, after running RECONFIGURE, for the run value of the 'hadoop connectivity' to take effect, you need to restart SQL Server.
In Parallel Data Warehouse, after running RECONFIGURE, for the run value of the 'hadoop connectivity' to take

effect, you need to restart the Parallel Data Warehouse region.

Limitations and Restrictions

RECONFIGURE is not allowed in an explicit or implicit transaction.

Permissions

All users can execute **sp_configure** with no parameters or the @configname parameter.

Requires **ALTER SETTINGS** server-level permission or membership in the **sysadmin** fixed server role to change a configuration value or to run RECONFIGURE.

Examples

A. List all available configuration settings

The following example shows how to list all configuration options.

```
EXEC sp_configure;
```

The result returns the option name followed by the minimum and maximum values for the option. The **config_value** is the value that SQL, or PolyBase, will use when reconfiguration is complete. The **run_value** is the value that is currently being used. The **config_value** and **run_value** are usually the same unless the value is in the process of being changed.

B. List the configuration settings for one configuration name

```
EXEC sp_configure @configname='hadoop connectivity';
```

C. Set Hadoop connectivity

This example sets PolyBase to option 7. This option allows PolyBase to create and use external tables on Hortonworks 2.1, 2.2, and 2.3 on Linux and Windows Server, and Azure blob storage. For example, SQL could have 30 external tables with 7 of them referencing data on Hortonworks 2.1 on Linux, 4 on Hortonworks 2.2 on Linux, 7 on Hortonworks 2.3 on Linux, and the other 12 referencing Azure blob storage.

```
--Configure external tables to reference data on Hortonworks 2.1, 2.2, and 2.3 on Linux, and Azure blob
storage

sp_configure @configname = 'hadoop connectivity', @configvalue = 7;
GO

RECONFIGURE
GO
```

See Also

sp_configure (Transact-SQL)
CREATE EXTERNAL TABLE (Transact-SQL)
CREATE EXTERNAL FILE FORMAT (Transact-SQL)
CREATE EXTERNAL DATA SOURCE (Transact-SQL)

in-doubt xact resolution Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **in-doubt xact resolution** option to control the default outcome of transactions that the Microsoft Distributed Transaction Coordinator (MS DTC) is unable to resolve. Inability to resolve transactions may be related to the MS DTC down time or an unknown transaction outcome at the time of recovery.

The following table lists the possible outcome values for resolving an in-doubt transaction.

OUTCOME VALUE	DESCRIPTION
0	No presumption. Recovery fails if MS DTC cannot resolve any in-doubt transactions.
1	Presume commit. Any MS DTC in-doubt transactions are presumed to have committed.
2	Presume abort. Any MS DTC in-doubt transactions are presumed to have aborted.

To minimize the possibility of extended down time, an administrator might choose to configure this option either to presume commit or presume abort, as shown in the following example.

```
sp_configure 'show advanced options', 1
GO
RECONFIGURE
GO
sp_configure 'in-doubt xact resolution', 2 -- presume abort
GO
RECONFIGURE
GO
sp_configure 'show advanced options', 0
GO
RECONFIGURE
GO
```

Alternatively, the administrator might want to leave the default (no presumption) and allow recovery to fail in order to be made aware of a DTC failure, as shown in the following example.

```
sp_configure 'show advanced options', 1
GO
RECONFIGURE
GO
sp_configure 'in-doubt xact resolution', 1 -- presume commit
GO
reconfigure
GO
ALTER DATABASE pubs SET ONLINE -- run recovery again
GO
sp_configure 'in-doubt xact resolution', 0 -- back to no assumptions
GO
sp_configure 'show advanced options', 0
GO
RECONFIGURE
GO
```

The **in-doubt xact resolution** option is an advanced option. If you are using the **sp_configure** system stored procedure to change the setting, you can change **in-doubt xact resolution** only when **show advanced options** is set to 1. The setting takes effect immediately without a server restart.

NOTE

Consistent configuration of this option across all Microsoft SQL Server instances involved in any distributed transactions will help avoid data inconsistencies.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the index create memory Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure the **index create memory** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **index create memory** option controls the maximum amount of memory initially allocated for creating indexes. The default value for this option is 0 (self-configuring). If more memory is later needed for index creation and the memory is available, the server will use it; thereby, exceeding the setting of this option. If additional memory is not available, the index creation will continue using the memory already allocated.

In This Topic

Before You Begin

Limitations and Restrictions

- The setting of the **min memory per query** option has precedence over the **index create memory** option. If you change both options and the **index create memory** is less than **min memory per query**, you receive a warning message, but the value is set. During query execution, you receive a similar warning.

- When using partitioned tables and indexes, the minimum memory requirements for index creation may increase significantly if there are non-aligned partitioned indexes and a high degree of parallelism. This option controls the total initial amount of memory allocated for all index partitions in a single index creation operation. The query will terminate with an error message if the amount set by this option is less than the minimum required to run the query.

- The run value for this option will not exceed the actual amount of memory that can be used for the operating system and hardware platform on which SQL Server is running.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- The **index create memory** option is self-configuring and usually works without requiring adjustment. However, if you experience difficulties creating indexes, consider increasing the value of this option from its run value.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the index create memory option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Memory** node.

3. Under **Index creation memory**, type or select the desired value for the index create memory option.

   Use the **index create memory** option to control the amount of memory used by index creation sorts. The **index create memory** option is self-configuring and should work in most cases without requiring adjustment. However, if you experience difficulties creating indexes, consider increasing the value of this option from its run value. Query sorts are controlled through the **min memory per query** option.

Using Transact-SQL

To configure the index create memory option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `index create memory` option to `4096`.

```
USE AdventureWorks2012 ;
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
EXEC sp_configure 'index create memory', 4096
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the index create memory option

The setting takes effect immediately without restarting the server.

See Also

sys.configurations (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Memory Server Configuration Options
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

lightweight pooling Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **lightweight pooling** option to provide a means of reducing the system overhead associated with the excessive context switching sometimes seen in symmetric multiprocessing (SMP) environments. When excessive context switching is present, lightweight pooling can provide better throughput by performing the context switching inline, thus helping to reduce user/kernel ring transitions.

Fiber mode is intended for certain situations in which the context switching of the UMS workers are the critical bottleneck in performance. Because this is rare, fiber mode rarely enhances performance or scalability on the typical system. Improved context switching in Microsoft Windows Server 2003 has also reduced the need for fiber mode. We do not recommend that you use fiber mode scheduling for routine operation. This is because it can decrease performance by inhibiting the regular benefits of context switching, and because some components of SQL Server that use Thread Local Storage (TLS) or thread-owned objects, such as mutexes (a type of Win32 kernel object), cannot function correctly in fiber mode.

Setting **lightweight pooling** to 1 causes SQL Server to switch to fiber mode scheduling. The default value for this option is 0.

The **lightweight pooling** option is an advanced option. If you are using the **sp_configure** system stored procedure to change the setting, you can change **lightweight pooling** only when **show advanced options** is set to 1. The setting takes effect after the server is restarted.

NOTE

Lightweight pooling is not supported for Microsoft Windows 2000 and Microsoft Windows XP. Windows Server 2003 provides full support for lightweight pooling.

NOTE

Common language runtime (CLR) execution is not supported under lightweight pooling. Disable one of two options: "clr enabled" or "lightweight pooling". Features that rely upon CLR and that do not work properly in fiber mode include the hierarchy data type, replication, and Policy-Based Management.

See Also

clr enabled Server Configuration Option
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
clr enabled Server Configuration Option

Configure the locks Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure the **locks** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **locks** option sets the maximum number of available locks, thereby limiting the amount of memory the SQL Server Database Engine uses for them. The default setting is 0, which allows the Database Engine to allocate and deallocate lock structures dynamically, based on changing system requirements.

IMPORTANT

This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible.

In This Topic

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- When the server is started with **locks** set to 0, the lock manager acquires sufficient memory from the Database Engine for an initial pool of 2,500 lock structures. As the lock pool is exhausted, additional memory is acquired for the pool.

  Generally, if more memory is required for the lock pool than is available in the Database Engine memory pool, and more computer memory is available (the **max server memory** threshold has not been reached), the Database Engine allocates memory dynamically to satisfy the request for locks. However, if allocating that memory would cause paging at the operating system level (for example, if another application is running on the same computer as an instance of SQL Server and using that memory), more lock space is not allocated. The dynamic lock pool does not acquire more than 60 percent of the memory allocated to the Database Engine. After the lock pool has reached 60 percent of the memory acquired by an instance of the Database Engine, or no more memory is available on the computer, further requests for locks generate an error.

Allowing SQL Server to use locks dynamically is the recommended configuration. However, you can set **locks** and override the ability of SQL Server to allocate lock resources dynamically. When **locks** is set to a value other than 0, the Database Engine cannot allocate more locks than the value specified in **locks**. Increase this value if SQL Server displays a message that you have exceeded the number of available locks. Because each lock consumes memory (96 bytes per lock), increasing this value can require increasing the amount of memory dedicated to the server.

- The **locks** option also affects when lock escalation occurs. When **locks** is set to 0, lock escalation occurs when the memory used by the current lock structures reaches 40 percent of the Database Engine memory pool. When **locks** is not set to 0, lock escalation occurs when the number of locks reaches 40 percent of the value specified for **locks**.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the locks option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Advanced** node.
3. Under **Parallelism**, type the desired value for the **locks** option.

   Use the **locks** option to set the maximum number of available locks, thereby limiting the amount of memory SQL Server uses for them.

Using Transact-SQL

To configure the locks option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `locks` option to set the number of locks available for all users to `20000`.

```
Use AdventureWorks2012 ;
GO
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'locks', 20000;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the locks option

The server must be restarted before the setting can take effect.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the max degree of parallelism Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure the **max degree of parallelism (MAXDOP)** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. When an instance of SQL Server runs on a computer that has more than one microprocessor or CPU, it detects the best degree of parallelism, that is, the number of processors employed to run a single statement, for each parallel plan execution. You can use the **max degree of parallelism** option to limit the number of processors to use in parallel plan execution. SQL Server considers parallel execution plans for queries, index data definition language (DDL) operations, parallel insert, online alter column, parallel stats collectiuon, and static and keyset-driven cursor population.

Before You Begin

Limitations and Restrictions

- If the affinity mask option is not set to the default, it may restrict the number of processors available to SQL Server on symmetric multiprocessing (SMP) systems.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.
- To enable the server to determine the maximum degree of parallelism, set this option to 0, the default value. Setting maximum degree of parallelism to 0 allows SQL Server to use all the available processors up to 64 processors. To suppress parallel plan generation, set **max degree of parallelism** to 1. Set the value to a number from 1 to 32,767 to specify the maximum number of processor cores that can be used by a single query execution. If a value greater than the number of available processors is specified, the actual number of available processors is used. If the computer has only one processor, the **max degree of parallelism** value is ignored.
- You can override the max degree of parallelism value in queries by specifying the MAXDOP query hint in the query statement. For more information, see Query Hints (Transact-SQL).
- Index operations that create or rebuild an index, or that drop a clustered index, can be resource intensive. You can override the max degree of parallelism value for index operations by specifying the MAXDOP index option in the index statement. The MAXDOP value is applied to the statement at execution time and is not stored in the index metadata. For more information, see Configure Parallel Index Operations.
- In addition to queries and index operations, this option also controls the parallelism of DBCC CHECKTABLE, DBCC CHECKDB, and DBCC CHECKFILEGROUP. You can disable parallel execution plans for these statements by using trace flag 2528. For more information, see Trace Flags (Transact-SQL).

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the max degree of parallelism option

1. In **Object Explorer**, right-click a server and select **Properties**.
2. Click the **Advanced** node.
3. In the **Max Degree of Parallelism** box, select the maximum number of processors to use in parallel plan execution.

Using Transact-SQL

To configure the max degree of parallelism option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the `max degree of parallelism` option to `8`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
EXEC sp_configure 'max degree of parallelism', 8;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the max degree of parallelism option

The setting takes effect immediately without restarting the server.

See Also

affinity mask Server Configuration Option
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
CREATE INDEX (Transact-SQL)
ALTER INDEX (Transact-SQL)
ALTER TABLE (Transact-SQL)
DBCC CHECKTABLE (Transact-SQL)
DBCC CHECKDB (Transact-SQL)
DBCC CHECKFILEGROUP (Transact-SQL)
Configure Parallel Index Operations Query Hints (Transact-SQL)
Set Index Options

max full-text crawl range Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **max full-text crawl range** option to optimize CPU utilization, which improves crawl performance during a full crawl. Using this option, you can specify the number of partitions that Microsoft SQL Server should use during a full index crawl. For example, if there are many CPUs and their utilization is not optimal, you can increase the maximum value of this option. In addition to this option, SQL Server uses a number of other factors, such as the number of rows in the table and the number of CPUs, to determine the actual number of partitions used.

The default value of this option is 4; the minimum value is 1, and the maximum value is 256. Changes made to this option affect only subsequent crawls. Crawls in process will not be affected by a change in this option setting.

The **max full-text crawl range** option is an advanced option. If you are using the **sp_configure** system stored procedure to change the setting, you can change **max full-text crawl range** only when **show advanced options** is set to 1. The setting takes effect immediately without a server restart.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the max text repl size Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **max text repl size** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **max text repl size** option specifies the maximum size (in bytes) of **text**, **ntext**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, **xml**, and **image** data that can be added to a replicated column or captured column in a single INSERT, UPDATE, WRITETEXT, or UPDATETEXT statement. The default value is 65536 bytes. A value of -1 indicates that there is no size limit, other than the limit imposed by the data type.

In This Topic

Before You Begin

Limitations and Restrictions

- This option applies to transactional replication and Change Data Capture. When a server is configured for both transactional replication and Change Data Capture, the specified value applies to both features. This option is ignored by snapshot replication and merge replication.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the max text repl size option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Advanced** node.

3. Under **Miscellaneous**, change the **Max Text Replication Size** option to the desired value.

Using Transact-SQL

To configure the max text repl size option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the `max text repl size` option to `-1`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1 ;
RECONFIGURE ;
GO
EXEC sp_configure 'max text repl size', -1 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the max text repl size option

The setting takes effect immediately without restarting the server.

See Also

Replication Features and Tasks
INSERT (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
UPDATE (Transact-SQL)
UPDATETEXT (Transact-SQL)
WRITETEXT (Transact-SQL)

Configure the max worker threads Server Configuration Option

3/24/2017 • 4 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **max worker threads** server configuration option in SQL Server by using SQL Server Management Studio or Transact-SQL. The **max worker threads** option configures the number of worker threads that are available to SQL Server processes. SQL Server uses the native thread services of the operating systems so that one or more threads support each network that SQL Server supports simultaneously, another thread handles database checkpoints, and a pool of threads handles all users. The default value for **max worker threads** is 0. This enables SQL Server to automatically configure the number of worker threads at startup. The default setting is best for most systems. However, depending on your system configuration, setting **max worker threads** to a specific value sometimes improves performance.

In This Topic

Before You Begin

Limitations and Restrictions

- When the actual number of query requests is less than the amount set in **max worker threads**, one thread handles each query request. However, if the actual number of query request exceeds the amount set in **max worker threads**, SQL Server pools the worker threads so that the next available worker thread can handle the request.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- Thread pooling helps optimize performance when large numbers of clients are connected to the server. Usually, a separate operating system thread is created for each query request. However, with hundreds of connections to the server, using one thread per query request can consume large amounts of system resources. The **max worker threads** option enables SQL Server to create a pool of worker threads to service a larger number of query requests, which improves performance.

- The following table shows the automatically configured number of max worker threads for various combinations of CPUs and versions of SQL Server.

NUMBER OF CPUS	32-BIT COMPUTER	64-BIT COMPUTER
<= 4 processors	256	512
8 processors	288	576
16 processors	352	704
32 processors	480	960
64 processors	736	1472
128 processors	4224	4480
256 processors	8320	8576

NOTE

SQL Server can no longer be installed on a 32-bit operating system. 32-bit computer values are listed for the assistance of customers running SQL Server 2014 and earlier. We recommend 1024 as the maximum number of worker threads for an instance of SQL Server that is running on a 32-bit computer.

NOTE

For recommendations on using more than 64 CPUs, refer to Best Practices for Running SQL Server on Computers That Have More Than 64 CPUs.

- When all worker threads are active with long running queries, SQL Server might appear unresponsive until a worker thread completes and becomes available. Although this is not a defect, it can sometimes be undesirable. If a process appears to be unresponsive and no new queries can be processed, then connect to SQL Server using the dedicated administrator connection (DAC), and kill the process. To prevent this, increase the number of max worker threads.

  The**max worker threads** server configuration option does not take into account threads that are required for all the system tasks such as Availibility Groups, Service Broker, Lock Manager, and others. If the number of threads configured are being exceeded, the following query will provide information about the system tasks that have spawned the additional threads.

```
SELECT
s.session_id,
r.command,
r.status,
r.wait_type,
r.scheduler_id,
w.worker_address,
w.is_preemptive,
w.state,
t.task_state,
t.session_id,
t.exec_context_id,
t.request_id
FROM sys.dm_exec_sessions AS s
INNER JOIN sys.dm_exec_requests AS r
    ON s.session_id = r.session_id
INNER JOIN sys.dm_os_tasks AS t
    ON r.task_address = t.task_address
INNER JOIN sys.dm_os_workers AS w
    ON t.worker_address = w.worker_address
WHERE s.is_user_process = 0;
```

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the max worker threads option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Processors** node.

3. In the **Max worker threads** box, type or select a value from 128 through 32767.

   Use the **max worker threads** option to configure the number of worker threads available to SQL Server processes. The default setting for **max worker threads** is best for most systems. However, depending on your system configuration, setting **max worker threads** to a smaller value sometimes improves performance.

Using Transact-SQL

To configure the max worker threads option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the `max worker threads` option to `900`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'max worker threads', 900 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the max worker threads option

The change will take effect immediately without requiring the Database Engine to restart.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
Diagnostic Connection for Database Administrators

Configure the media retention Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **media retention** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **media retention** option specifies the length of time to retain each backup set. The option helps protect backups from being overwritten until the specified number of days has elapsed. After you configure **media retention** option, you do not have to specify the length of time to retain system backups each time you perform a backup. The default value is 0 days, and the maximum value is 365 days.

In This Topic

Before You Begin

Limitations and Restrictions

- If you use the backup medium before the set number of days has passed, SQL Server issues a warning message. SQL Server does not issue a warning unless you change the default.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- The **media retention** option can be overridden by using the RETAINDAYS clause of the BACKUP statement.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the media retention option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Database settings** node.

3. Under **Backup/Restore**, in the **Default backup media retention** box, type or select a value from 0 through 365 to set the number of days the backup medium will be retained after a database or transaction log backup.

Using Transact-SQL

To configure the media retention option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `media retention` option to `60` days.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'media retention', 60 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the media retention option

The setting takes effect immediately without restarting the server.

See Also

Back Up and Restore of SQL Server Databases
BACKUP (Transact-SQL)
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the min memory per query Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **min memory per query** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **min memory per query** option specifies the minimum amount of memory (in kilobytes) that will be allocated for the execution of a query. For example, if **min memory per query** is set to 2,048 KB, the query is guaranteed to get at least that much total memory. The default value is 1,024 KB. The minimum value 512 KB, and the maximum is 2,147,483,647 KB (2 GB).

In This Topic

- **Before you begin:**

  Limitations and Restrictions

  Recommendations

  Security

- **To configure the min memory per query option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the min memory per query option

Before You Begin

Limitations and Restrictions

- The amount of min memory per query has precedence over the index create memory Option. If you modify both options and the index create memory is less than min memory per query, you receive a warning message, but the value is set. During query execution you receive another similar warning.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- The SQL Server query processor tries to determine the optimal amount of memory to allocate to a query. The min memory per query option lets the administrator specify the minimum amount of memory any single query receives. Queries generally receive more memory than this if they have hash and sort operations on a large volume of data. Increasing the value of min memory per query may improve performance for some small to medium-sized queries, but doing so could lead to increased competition for memory resources. The min memory per query option includes memory allocated for sorting.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the

RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the min memory per query option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Memory** node.
3. In the **Minimum memory per query** box, enter the minimum amount of memory (in kilobytes) that will be allocated for the execution of a query.

Using Transact-SQL

To configure the min memory per query option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `min memory per query` option to `3500` KB.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'min memory per query', 3500 ;
GO
RECONFIGURE;
GO
```

Follow Up: After you configure the min memory per query option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
Configure the index create memory Server Configuration Option

Configure the nested triggers Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **nested triggers** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **nested triggers** option controls whether an AFTER trigger can cascade. That is, perform an action that initiates another trigger, which initiates another trigger, and so on. When **nested triggers** is set to 0, AFTER triggers cannot cascade. When **nested triggers** is set to 1 (the default), AFTER triggers can cascade to as many as 32 levels. INSTEAD OF triggers can be nested regardless of the setting of this option.

In This Topic

- **Before you begin:**

  Security

- **To configure the nested triggers option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the nested triggers option

Before You Begin

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the nested triggers option

1. In **Object Explorer**, right-click a server, and then select **Properties**.

2. On the **Advanced** page, set the **Allow Triggers to Fire Others** option to **True** (the default) or **False**.

Using Transact-SQL

To configure the nested triggers option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `nested triggers` option to `0`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'nested triggers', 0 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the nested triggers option

The setting takes effect immediately without restarting the server.

See Also

Create Nested Triggers
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the network packet size Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **network packet size** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **network packet size** option sets the packet size (in bytes) that is used across the whole network. Packets are the fixed-size chunks of data that transfer requests and results between clients and servers. The default packet size is 4,096 bytes.

NOTE

Do not change the packet size unless you are certain that it will improve performance. For most applications, the default packet size is best.

In This Topic

- **Before you begin:**

  Limitations and Restrictions

  Recommendations

  Security

- **To configure the network packet size option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the network packet size option

Before You Begin

Limitations and Restrictions

- The maximum network packet size for encrypted connections is 16,383 bytes.

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- If an application does bulk copy operations or sends or receives large amounts of text or image data, a packet size larger than the default might improve efficiency because it results in fewer network read-and-write operations. If an application sends and receives small amounts of information, the packet size can be set to 512 bytes, which is sufficient for most data transfers.

- On systems that are using different network protocols, set network packet size to the size for the most common protocol used. The network packet size option improves network performance when network protocols support larger packets. Client applications can override this value.

- You can also call OLE DB, Open Database Connectivity (ODBC), and DB-Library functions request a change the packet size. If the server cannot support the requested packet size, the Database Engine will send a warning message to the client. In some circumstances, changing the packet size might lead to a communication link failure, such as the following:

  `Native Error: 233, no process is on the other end of the pipe.`

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the network packet size option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Advanced** node.
3. Under **Network**, select a value for the **Network Packet Size** box.

Using Transact-SQL

To configure the network packet size option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `network packet size` option to `6500` bytes.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'network packet size', 6500 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the network packet size option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Ole Automation Procedures Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **Ole Automation Procedures** option to specify whether OLE Automation objects can be instantiated within Transact-SQL batches. This option can also be configured using the Policy-Based Management or the **sp_configure** stored procedure. For more information, see Surface Area Configuration.

The **Ole Automation Procedures** option can be set to the following values.

0
OLE Automation Procedures are disabled. Default for new instances of SQL Server.

1
OLE Automation Procedures are enabled.

When OLE Automation Procedures are enabled, a call to **sp_OACreate** will start the OLE shared execution environment.

The current value of the **Ole Automation Procedures** option can be viewed and changed by using the **sp_configure** system stored procedure.

Examples

The following example shows how to view the current setting of OLE Automation procedures.

```
EXEC sp_configure 'Ole Automation Procedures';
GO
```

The following example shows how to enable OLE Automation procedures.

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Ole Automation Procedures', 1;
GO
RECONFIGURE;
GO
```

See Also

sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)
Surface Area Configuration
Server Configuration Options (SQL Server)

open objects Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This option is still present in **sp_configure**, although its functionality has been disabled in Microsoft SQL Server. (The setting has no effect.) In SQL Server, the number of open database objects is managed dynamically and is limited only by the available memory. The **open objects** option available in **sp_configure** for backward compatibility with existing scripts.

IMPORTANT

This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible.

See Also

Server Configuration Options (SQL Server)

optimize for ad hoc workloads Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

The **optimize for ad hoc workloads** option is used to improve the efficiency of the plan cache for workloads that contain many single use ad hoc batches. When this option is set to 1, the Database Engine stores a small compiled plan stub in the plan cache when a batch is compiled for the first time, instead of the full compiled plan. This helps to relieve memory pressure by not allowing the plan cache to become filled with compiled plans that are not reused.

The compiled plan stub allows the Database Engine to recognize that this ad hoc batch has been compiled before but has only stored a compiled plan stub, so when this batch is invoked (compiled or executed) again, the Database Engine compiles the batch, removes the compiled plan stub from the plan cache, and adds the full compiled plan to the plan cache.

Setting the **optimize for ad hoc workloads** to 1 affects only new plans; plans that are already in the plan cache are unaffected.

The compiled plan stub is one of the cacheobjtypes displayed by the sys.dm_exec_cached_plans catalog view. It has a unique sql handle and plan handle. The compiled plan stub does not have an execution plan associated with it and querying for the plan handle will not return an XML Showplan.

Trace flag 8032 reverts the cache limit parameters to the SQL Server 2005 RTM setting which in general allows caches to be larger. Use this setting when frequently reused cache entries do not fit into the cache and when the optimize for ad hoc workloads Server Configuration Option has failed to resolve the problem with plan cache.

WARNING

Trace flag 8032 can cause poor performance if large caches make less memory available for other memory consumers, such as the buffer pool.

See Also

sys.dm_exec_cached_plans (Transact-SQL)
Server Configuration Options (SQL Server)

PH timeout Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the PH timeout option to specify the time, in seconds, that the full-text protocol handler should wait to connect to a database before timing out. The default value is 60 seconds. Increase the ph timeout value when connection attempts are timing out due to temporary network issues.

The full-text protocol handler is hosted in the filter daemon host and is used to fetch from SQL Server the data to be full-text indexed. For more information about full-text search components, see Full-Text Search.

When attempting to fetch a data row, if the protocol handler cannot connect to SQL Server within the specified time, it reports a time-out error for that row. The full-text gatherer will retry the row later. For more information about the full-text gatherer, see Populate Full-Text Indexes.

See Also

Full-Text Search
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

precompute rank Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This option is not implemented in SQL Server 2016. This is a breaking change. Modify applications that currently use this feature as soon as possible.

Configure the priority boost Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **priority boost** configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. Use the **priority boost** option to specify whether Microsoft SQL Server should run at a higher Microsoft Windows 2008 or Windows 2008 R2 scheduling priority than other processes on the same computer. If you set this option to 1, SQL Server runs at a priority base of 13 in the Windows 2008 or Windows Server 2008 R2 scheduler. The default is 0, which is a priority base of 7.

> **IMPORTANT**
>
> This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible.

In This Topic

Before You Begin

Limitations and Restrictions

- Raising the priority too high may drain resources from essential operating system and network functions, resulting in problems shutting down SQL Server or using other operating system tasks on the server.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the priority boost option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Processors** node.

3. Under **Threads**, select the **Boost SQL Server priority** check box.

4. Stop and restart SQL Server.

Using Transact-SQL

To configure the priority boost option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `priority boost` option to `1`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'priority boost', 1 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the priority boost option

The server must be restarted before the setting can take effect.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the query governor cost limit Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **query governor cost limit** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The query governor cost limit option specifies an upper limit on the time period in which a query can run. Query cost refers to the estimated elapsed time, in seconds, that is required to complete a query on a specific hardware configuration. The default value for this option is 0, which sets the query governor to off. This allows all queries to run without any time limitation. If you specify a nonzero, nonnegative value, the query governor disallows execution of any query that has an estimated cost that exceeds that value.

In This Topic

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.
- To change the value query governor cost limit on a per-connection basis, use the SET QUERY_GOVERNOR_COST_LIMIT statement.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the query governor cost limit option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Connections** page.

3. Select or clear the **Use query governor to prevent long-running queries** check box.

   If you select this check box, in the box below, enter a positive value, which the query governor uses to disallow execution of any query with a running length exceeding that value.

Using Transact-SQL

To configure the query governor cost limit option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `query governor cost limit` option to `120` seconds.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'query governor cost limit', 120 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the query governor cost limit option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
SET QUERY_GOVERNOR_COST_LIMIT (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the query wait Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **query wait** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. Memory-intensive queries (such as those involving sorting and hashing) are queued when there is not enough memory available to run the query. The **query wait** option specifies the time, in seconds (from 0 through 2147483647), that a query waits for resources before it times out. The default value for this option is -1. This means the time-out is calculated as 25 times the estimated query cost.

IMPORTANT
A transaction that contains the waiting query might hold locks while the query waits for memory. In rare situations, it is possible for an undetectable deadlock to occur. Decreasing the query wait time lowers the probability of such deadlocks. Eventually, a waiting query will be terminated and the transaction locks released. However, increasing the maximum wait time may increase the amount of time for the query to be terminated. Changes to this option are not recommended.

In This Topic

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the query wait option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Advanced** node.

3. Under **Parallelism**, type the desired value for the **query wait** option.

Using Transact-SQL

To configure the query wait option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `query wait` option to `7500` seconds.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'query wait', 7500 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the query wait option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the recovery interval Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **recovery interval** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **recovery interval** option defines an upper limit on the time recovering a database should take. The SQL Server Database Engine uses the value specified for this option to determine approximately how often automatic checkpoints to issue automatic checkpoints on a given database.

The default recovery-interval value is 0, which allows the Database Engine to automatically configure the recovery interval. Typically, the default recovery interval results in automatic checkpoints occurring approximately once a minute for active databases and a recovery time of less than one minute. Higher values indicate the approximate maximum recovery time, in minutes. For example, setting the recovery interval to 3 indicates a maximum recovery time of approximately three minutes.

In This Topic

Before You Begin

Limitations and Restrictions

- The recovery interval affects only databases that use the default target recovery time (0). To override the server recovery interval on a database, configure a non-default target recovery time on the database. For more information, see Change the Target Recovery Time of a Database (SQL Server).

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- Typically, we recommend that you keep the recovery interval at 0, unless you experience performance problems. If you decide to increase the recovery-interval setting, we recommend increasing it gradually by small increments and evaluating the effect of each incremental increase on recovery performance.

- If you use **sp_configure** to change the value of the **recovery interval** option to more than 60 (minutes), specify RECONFIGURE WITH OVERRIDE. WITH OVERRIDE disables configuration value checking (for values that are not valid or are nonrecommended values).

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To set the recovery interval

1. In Object Explorer, right-click server instance and select **Properties**.

2. Click the **Database settings** node.

3. Under **Recovery**, in the **Recovery interval (minutes)** box, type or select a value from 0 through 32767 to set the maximum amount of time, in minutes, that SQL Server should spend recovering each database at startup. The default is 0, indicating automatic configuration by SQL Server. In practice, this means a recovery time of less than one minute and a checkpoint approximately every one minute for active databases.

Using Transact-SQL

To set the recovery interval

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `recovery interval` option to `3` minutes.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'recovery interval', 3 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the recovery internal option

The setting takes effect immediately without restarting the server.

See Also

Change the Target Recovery Time of a Database (SQL Server)
Database Checkpoints (SQL Server)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
show advanced options Server Configuration Option
RECONFIGURE (Transact-SQL)

Configure the remote access Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic is about the "Remote Access" feature. This is an obscure SQL Server to SQL Server communication feature that is deprecated, and you probably shouldn't be using it. If you reached this page because you are having trouble connecting to SQL Server, see one of the following topics instead:

- Tutorial: Getting Started with the Database Engine
- Logging In to SQL Server
- Connect to SQL Server When System Administrators Are Locked Out
- Connect to a Registered Server (SQL Server Management Studio)
- Connect to Any SQL Server Component from SQL Server Management Studio
- Connect to the Database Engine With sqlcmd
- How to Troubleshoot Connecting to the SQL Server Database Engine

  Programmers may be interested in the following topics:

- How To: Connect to SQL Server Using SQL Authentication in ASP.NET 2.0
- Connecting to an Instance of SQL Server
- How to: Create Connections to SQL Server Databases

  The main body of this topic starts here.

  This topic describes how to configure the **remote access** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **remote access** option controls the execution of stored procedures from local or remote servers on which instances of SQL Server are running. This default value for this option is 1. This grants permission to run local stored procedures from remote servers or remote stored procedures from the local server. To prevent local stored procedures from being run from a remote server or remote stored procedures from being run on the local server, set the option to 0.

> **IMPORTANT**
>
> This feature will be removed in the next version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. Use sp_addlinkedserver instead.

In This Topic

- **Before you begin:**

  Limitations and Restrictions

  Security

- **To configure the remote access option, using:**

SQL Server Management Studio

Transact-SQL

- **Follow Up:** After you configure the remote access option

Before You Begin

Limitations and Restrictions

- The **remote access** option only applies to servers that are added by using sp_addserver, and is included for backward compatibility.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the remote access option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Connections** node.

3. Under **Remote server connections**, select or clear the **Allow remote connections to this server** check box.

Using Transact-SQL

To configure the remote access option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `remote access` option to `0`.

```
EXEC sp_configure 'remote access', 0 ;
GO
RECONFIGURE ;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the remote access option

This setting does not take effect until you restart SQL Server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

remote admin connections Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

SQL Server provides a dedicated administrator connection (DAC). The DAC lets an administrator access a running server to execute diagnostic functions or Transact-SQL statements, or to troubleshoot problems on the server, even when the server is locked or running in an abnormal state and not responding to a SQL Server Database Engine connection. By default, the DAC is only available from a client on the server. To enable client applications on remote computers to use the DAC, use the remote admin connections option of sp_configure.

By default, the DAC only listens on the loop-back IP address (127.0.0.1), port 1434. If TCP port 1434 is not available, a TCP port is dynamically assigned when the Database Engine starts up. When more than one instance of SQL Server is installed on a computer, check the error log for the TCP port number.

The following table lists the possible values for the remote admin connections option.

VALUE	DESCRIPTION
0	Indicates only local connections are allowed by using the DAC.
1	Indicates remote connections are allowed by using the DAC.

Example

The following example enables the DAC from a remote computer.

```
sp_configure 'remote admin connections', 1;
GO
RECONFIGURE;
GO
```

See Also

Diagnostic Connection for Database Administrators

Configure the remote data archive Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **remote data archive** option to specify whether databases and tables on the server can be enabled for Stretch. For more info, see Enable Stretch Database for a database.

The **remote data archive** option can have the following values.

VALUE	DESCRIPTION
0	Databases and tables on the server cannot be enabled for Stretch.
1	Databases and tables on the server can be enabled for Stretch.

Running **sp_configure** to set the value of the **remote data archive** option requires sysadmin or serveradmin permissions.

Example

The following example first displays the current setting of the **remote data archive** option. Then the example enables the **remote data archive** option by setting its value to 1.

```
EXEC sp_configure 'remote data archive';
GO
EXEC sp_configure 'remote data archive' , '1';
GO
RECONFIGURE;
GO
```

To disable the option, set the value to 0.

See Also

Enable Stretch Database for a database
Stretch Database
sp_configure (Transact-SQL)

Configure the remote login timeout Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure the **remote login timeout** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **remote login timeout** option specifies the number of seconds to wait before returning from a failed attempt to log in to a remote server. For example, if you are trying to log in to a remote server and that server is down, **remote login timeout** helps make sure that you do not have to wait indefinitely before your computer stops trying to log in. The default value for this option is 10 seconds. A value of 0 allows for an infinite wait.

NOTE

The default value for this option is 20 seconds in SQL Server 2008.

In This Topic

- **Before you begin:**

  Limitations and Restrictions

  Security

- **To configure the remote login timeout option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the remote login timeout option

Before You Begin

Limitations and Restrictions

- The **remote login timeout** option affects connections to OLE DB providers made for heterogeneous queries.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the remote login timeout option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Advanced** node.

3. Under **Network**, select a value for the **Remote Login Timeout** box.

   Use the **remote login timeout** option to specify the number of seconds to wait before returning from a failed remote login attempt.

Using Transact-SQL

To configure the remote login timeout option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `remote login timeout` option to `35` seconds.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'remote login timeout', 35 ;
GO
RECONFIGURE ;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the remote login timeout option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the remote proc trans Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **remote proc trans** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **remote proc trans** option helps protect the actions of a server-to-server procedure through a Microsoft Distributed Transaction Coordinator (MS DTC) transaction.

Set the value of **remote proc trans** to 1 to provide an MS DTC-coordinated distributed transaction that protects the ACID (atomic, consistent, isolated, and durable) properties of transactions. Sessions begun after setting this option to 1 inherit the configuration setting as their default.

> **IMPORTANT**
> This feature will be removed in the next version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature.

In This Topic

Before You Begin

Prerequisites

- Remote server connections must be allowed before this value can be set.

Recommendations

- This option is provided for compatibility with earlier versions of Microsoft SQL Server for applications that use remote stored procedures. Instead of issuing remote stored procedure calls, use distributed queries that reference linked servers, which are defined by using sp_addlinkedserver.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER

SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the remote proc trans option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Connections** node.
3. Under **Remote server connections**, select the **Require Distributed Transactions for server to server communication** check box.

Using Transact-SQL

To configure the remote proc trans option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `remote proc trans` option to `1`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'remote proc trans', 1 ;
GO
RECONFIGURE ;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the remote proc trans option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Configure the remote query timeout Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure the **remote query timeout** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **remote query timeout** option specifies how long, in seconds, a remote operation can take before SQL Server times out. The default value for this option is 600, which allows a 10-minute wait. This value applies to an outgoing connection initiated by the Database Engine as a remote query. This value has no effect on queries received by the Database Engine. To disable the time-out, set the value to 0. A query will wait until it completes.

For heterogeneous queries, **remote query timeout** specifies the number of seconds (initialized in the command object using the DBPROP_COMMANDTIMEOUT rowset property) that a remote provider should wait for result sets before the query times out. This value is also used to set DBPROP_GENERALTIMEOUT if supported by the remote provider. This will cause any other operations to time out after the specified number of seconds.

For remote stored procedures, **remote query timeout** specifies the number of seconds that must elapse after sending a remote `EXEC` statement before the remote stored procedure times out.

In This Topic

Before You Begin

Prerequisites

- Remote server connections must be allowed before this value can be set.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the remote query timeout option

1. In Object Explorer, right-click a server and select **Properties**.
2. Click the **Connections** node.
3. Under **Remote server connections**, in the **Remote query timeout** box, type or select a value from 0 through 2,147,483,647 to set the maximum number seconds for SQL Server to wait before timing out.

Using Transact-SQL

To configure the remote query timeout option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `remote query timeout` option to `0` to disable the time-out.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'remote query timeout', 0 ;
GO
RECONFIGURE ;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the remote query timeout option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Rowset Properties and Behaviors
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

Replication XPs Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This option is for internal use only.

See Also

Server Configuration Options (SQL Server)

Configure the scan for startup procs Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **scan for startup procs** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. Use the **scan for startup procs** option to scan for automatic execution of stored procedures at SQL Server startup time. If this option is set to 1, SQL Server scans for and runs all automatically run stored procedures that are defined on the server. The default value for **scan for startup procs** is 0 (do not scan).

In This Topic

- **Before you begin:**

  Recommendations

  Security

- **To configure the scan for startup procs option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the scan for startup procs option

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- The value for this option can be set by using **sp_configure**; however, it will be set automatically if you use **sp_procoption**, which is used to mark or unmark automatically run stored procedures. When **sp_procoption** is used to mark the first stored procedure as an autoproc, this option is set automatically to a value of 1. When **sp_procoption** is used to unmark the last stored procedure as an autoproc, this option is automatically set to a value of 0. If you use **sp_procoption** to mark and unmark autoprocs, and if you always unmark autoprocs before dropping them, there is no need to set this option manually.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the scan for startup procs option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Advanced** node.

3. Under **Miscellaneous**, change the **Scan for Startup Procs** option to True or False by selecting the value you want from the drop-down list box.

Using Transact-SQL

To configure the scan for startup procs option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `scan for startup procs` option to `1`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1 ;
GO
RECONFIGURE
GO
EXEC sp_configure 'scan for startup procs', 1 ;
GO
RECONFIGURE
GO
```

Follow Up: After you configure the scan for startup procs option

The server must be restarted before the setting can take effect.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
sp_procoption (Transact-SQL)

Server Memory Server Configuration Options

3/24/2017 • 8 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the two server memory options, **min server memory** and **max server memory**, to reconfigure the amount of memory (in megabytes) that is managed by the SQL Server Memory Manager for a SQL Server process used by an instance of SQL Server.

The default setting for **min server memory** is 0, and the default setting for **max server memory** is 2147483647 MB. By default, SQL Server can change its memory requirements dynamically based on available system resources.

> **NOTE**
> Setting **max server memory** to the minimum value can severely reduce SQL Server performance and even prevent it from starting. If you cannot start SQL Server after changing this option, start it using the **–f** startup option and reset **max server memory** to its previous value. For more information, see Database Engine Service Startup Options.

When SQL Server is using memory dynamically, it queries the system periodically to determine the amount of free memory. Maintaining this free memory prevents the operating system (OS) from paging. If less memory is free, SQL Server releases memory to the OS. If more memory is free, SQL Server may allocate more memory. SQL Server adds memory only when its workload requires more memory; a server at rest does not increase the size of its virtual address space.

See example B for a query to return the currently used memory. **max server memory** controls the SQL Server memory allocation, including the buffer pool, compile memory, all caches, qe memory grants, lock manager memory, and clr memory (essentially any memory clerk found in **sys.dm_os_memory_clerks**). Memory for thread stacks, memory heaps, linked server providers other than SQL Server, and any memory allocated by a non SQL Server DLL are not controlled by max server memory.

SQL Server uses the memory notification API **QueryMemoryResourceNotification** to determine when the SQL Server Memory Manager may allocate memory and release memory.

Allowing SQL Server to use memory dynamically is recommended; however, you can set the memory options manually and restrict the amount of memory that SQL Server can access. Before you set the amount of memory for SQL Server, determine the appropriate memory setting by subtracting, from the total physical memory, the memory required for the OS and any other instances of SQL Server (and other system uses, if the computer is not wholly dedicated to SQL Server). This difference is the maximum amount of memory you can assign to SQL Server.

Setting the Memory Options Manually

Set **min server memory** and **max server memory** to span a range of memory values. This method is useful for system or database administrators to configure an instance of SQL Server in conjunction with the memory requirements of other applications that run on the same computer.

Use **min server memory** to guarantee a minimum amount of memory available to the SQL Server Memory Manager for an instance of SQL Server. SQL Server will not immediately allocate the amount of memory specified in **min server memory** on startup. However, after memory usage has reached this value due to client load, SQL Server cannot free memory unless the value of **min server memory** is reduced.

NOTE

SQL Server is not guaranteed to allocate the amount of memory specified in **min server memory**. If the load on the server never requires allocating the amount of memory specified in **min server memory**, SQL Server will run with less memory.

The minimum memory amount allowable for **max server memory** is 128 MB.

How to configure memory options using SQL Server Management Studio

Use the two server memory options, **min server memory** and **max server memory**, to reconfigure the amount of memory (in megabytes) managed by the SQL Server Memory Manager for an instance of SQL Server. By default, SQL Server can change its memory requirements dynamically based on available system resources.

Procedure for configuring a fixed amount of memory

To set a fixed amount of memory

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Memory** node.

3. Under **Server Memory Options**, enter the amount that you want for **Minimum server memory** and **Maximum server memory**.

   Use the default settings to allow SQL Server to change its memory requirements dynamically based on available system resources. The default setting for **min server memory** is 0, and the default setting for **max server memory** is 2147483647 megabytes (MB).

Maximize Data Throughput for Network Applications

To optimize system memory use for SQL Server, you should limit the amount of memory that is used by the system for file caching. To limit the file system cache, make sure that **Maximize data throughput for file sharing** is not selected. You can specify the smallest file system cache by selecting **Minimize memory used** or **Balance**.

To check the current setting on your operating system

1. Click **Start**, click **Control Panel**, double-click **Network Connections**, and then double-click **Local Area Connection**.

2. On the **General** tab, click **Properties**, select **File and Printer Sharing Microsoft Networks**, and then click **Properties**.

3. If **Maximize data throughput for network applications** is selected, choose any other option, click **OK**, and then close the rest of the dialog boxes.

Lock Pages in Memory

This Windows policy determines which accounts can use a process to keep data in physical memory, preventing the system from paging the data to virtual memory on disk. Locking pages in memory may keep the server responsive when paging memory to disk occurs. The SQL Server **Lock Pages in Memory** option is set to ON in instances of SQL Server 2016 Standard edition and higher when the account with privileges to run sqlservr.exe has been granted the Windows "Locked Pages in Memory" (LPIM) user right.

To disable the **Lock Pages In Memory** option for SQL Server, remove the "Locked Pages in Memory" user right for the SQL Server startup account.

To Disable Lock Pages in Memory

To disable the lock pages in memory option

1. On the **Start** menu, click **Run**. In the **Open** box, type **gpedit.msc**.

   The **Group Policy** dialog box opens.

2. On the **Group Policy** console, expand **Computer Configuration**, and then expand **Windows Settings**.

3. Expand **Security Settings**, and then expand **Local Policies**.

4. Select the **User Rights Assignment** folder.

   The policies will be displayed in the details pane.

5. In the pane, double-click **Lock pages in memory**.

6. In the **Local Security Policy Setting** dialog box, select the account with privileges to run sqlservr.exe and click **Remove**.

Virtual Memory Manager

The committed regions of address space are mapped to the available physical memory by the Windows Virtual Memory Manager (VMM).

For more information on the amount of physical memory supported by different operating systems, see the Windows documentation "Memory Limits for Windows Releases".

Virtual memory systems allow the over-commitment of physical memory, so that the ratio of virtual-to-physical memory can exceed 1:1. As a result, larger programs can run on computers with a variety of physical memory configurations. However, using significantly more virtual memory than the combined average working sets of all the processes can cause poor performance.

The **min server memory** and **max server memory** options are advanced options. If you are using the **sp_configure** system stored procedure to change these settings, you can change them only when **show advanced options** is set to 1. These settings take effect immediately without a server restart.

Running Multiple Instances of SQL Server

When you are running multiple instances of the Database Engine, there are three approaches you can use to manage memory:

- Use **max server memory** to control memory usage. Establish maximum settings for each instance, being careful that the total allowance is not more than the total physical memory on your machine. You might want to give each instance memory proportional to its expected workload or database size. This approach has the advantage that when new processes or instances start up, free memory will be available to them immediately. The drawback is that if you are not running all of the instances, none of the running instances will be able to utilize the remaining free memory.

- Use **min server memory** to control memory usage. Establish minimum settings for each instance, so that the sum of these minimums is 1-2 GB less than the total physical memory on your machine. Again, you may establish these minimums proportionately to the expected load of that instance. This approach has the advantage that if not all instances are running at the same time, the ones that are running can use the remaining free memory. This approach is also useful when there is another memory-intensive process on the computer, since it would insure that SQL Server would at least get a reasonable amount of memory. The drawback is that when a new instance (or any other process) starts, it may take some time for the running instances to release memory, especially if they must write modified pages back to their databases to do so.

- Do nothing (not recommended). The first instances presented with a workload will tend to allocate all of memory. Idle instances, or instances started later, may end up running with only a minimal amount of memory available. SQL Server makes no attempt to balance memory usage across instances. All instances will, however, respond to Windows Memory Notification signals to adjust the size of their memory footprint. Windows does not balance memory across applications with the Memory Notification API. It merely provides global feedback as to the availability of memory on the system.

  You can change these settings without restarting the instances, so you can easily experiment to find the best settings for your usage pattern.

Providing the Maximum Amount of Memory to SQL Server

Memory can be configured up to the process virtual address space limit in all SQL Server editions (8 TB).

***/3gb** is an operating-system boot parameter. For more information, visit the MSDN Library.

Examples

Example A

The following example sets the `max server memory` option to 4 GB:

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'max server memory', 4096;
GO
RECONFIGURE;
GO
```

Example B. Determining Current Memory Allocation

The following query returns information about currently allocated memory.

```
SELECT
(physical_memory_in_use_kb/1024) AS Memory_usedby_Sqlserver_MB,
(locked_page_allocations_kb/1024) AS Locked_pages_used_Sqlserver_MB,
(total_virtual_address_space_kb/1024) AS Total_VAS_in_MB,
process_physical_memory_low,
process_virtual_memory_low
FROM sys.dm_os_process_memory;
```

See Also

Monitor and Tune for Performance
RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

server trigger recursion Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **server trigger recursion** option to specify whether to allow server-level triggers to fire recursively. When this option is set to 1 (ON), server-level triggers will be allowed to fire recursively. When set to 0 (OFF), server-level triggers cannot be fired recursively. Only direct recursion is prevented when the server trigger recursion option is set to 0 (OFF). (To disable indirect recursion, set the **nested triggers** option to 0.) The default value for this option is 1 (ON). The setting takes effect immediately (without a server restart).

For more information, see Create Nested Triggers.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

set working set size Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This option is still present in the **sp_configure** stored procedure, but its functionality is unavailable in Microsoft SQL Server. (The setting has no effect.)

> **IMPORTANT**
>
> This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible.

See Also

Server Configuration Options (SQL Server)

show advanced options Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the **show advanced options** option to display the **sp_configure** system stored procedure advanced options. When you set **show advanced options** to 1, you can list the advanced options by using **sp_configure**. The default is 0.

The setting takes effect immediately without a server restart.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)

SMO and DMO XPs Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Use the SMO and DMO XPs option to enable SQL Server Management Object (SMO) extended stored procedures on this server.

Note than beginning in SQL Server 2012, DMO has been removed from SQL Server.

The possible values are described in the following table:

VALUE	MEANING
0	SMO XPs are not available.
1	SMO XPs are available. This is the default.

The setting takes effect immediately.

Examples

The following example enables SMO extended stored procedures.

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'SMO and DMO XPs', 1;
GO
RECONFIGURE
GO
```

See Also

SQL Server Management Objects (SMO) Programming Guide

transform noise words Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

Use the **transform noise words** server configuration option to suppress an error message if noise words, that is stopwords, cause a Boolean operation on a full-text query to return zero rows. This option is useful for full-text queries that use the CONTAINS predicate in which Boolean operations or NEAR operations include noise words. The possible values are described in the following table.

VALUE	DESCRIPTION
0	Noise words (or stopwords) are not transformed. When a full-text query contains noise words, the query returns zero rows, and SQL Server raises a warning. This is the default behavior. Note: The warning is a run-time warning. Therefore, if the full-text clause in the query is not executed, the warning is not raised. For a local query, only one warning is raised, even when there are multiple full-text query clauses. For a remote query, the linked server might not relay the error; therefore, the warning might not be raised.
1	Noise words (or stopwords) are transformed. They are ignored, and the rest of the query is evaluated. If noise words are specified in a proximity term, SQL Server removes them. For example, the noise word `is` is removed from `CONTAINS(<column_name>, 'NEAR (hello,is,goodbye)')`, transforming the search query into `CONTAINS(<column_name>, 'NEAR(hello,goodbye)')`. Notice that `CONTAINS(<column_name>, 'NEAR(hello,is)')` would be transformed into simply `CONTAINS(<column_name>, hello)` because there is only one valid search term.

Effects of the transform noise words Setting

This section illustrates the behavior of queries containing a noise word, "`the`", under the alternate settings of **transform noise words**. The sample full-text query strings are assumed to be run against a table row containing the following data: `[1, "The black cat"]`.

> **NOTE**
>
> All such scenarios can generate a noise word warning.

- With transform noise words set to 0:

QUERY STRING	RESULT
" `cat` " AND " `the` "	No results (The behavior is the same for " `the` " AND " `cat` ".)
" `cat` " NEAR " `the` "	No results (The behavior is the same for " `the` " NEAR " `cat` ".)
" `the` " AND NOT " `black` "	No results
" `black` " AND NOT " `the` "	No results

- With transform noise words set to 1:

QUERY STRING	RESULT
" `cat` " AND " `the` "	Hit for row with ID 1
" `cat` " NEAR " `the` "	Hit for row with ID 1
" `the` " AND NOT " `black` "	No results
" `black` " AND NOT " `the` "	Hit for row with ID 1

Example

The following example sets **transform noise words** to `1`.

```
sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
sp_configure 'transform noise words', 1;
RECONFIGURE;
GO
```

See Also

Server Configuration Options (SQL Server)
CONTAINS (Transact-SQL)

Configure the two digit year cutoff Server Configuration Option

3/24/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **two digit year cutoff** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **two digit year cutoff** option specifies an integer from 1753 to 9999 that represents the cutoff year for interpreting two-digit years as four-digit years. The default time span for SQL Server is 1950-2049, which represents a cutoff year of 2049. This means that SQL Server interprets a two-digit year of 49 as 2049, a two-digit year of 50 as 1950, and a two-digit year of 99 as 1999. To maintain backward compatibility, leave the setting at the default value.

In This Topic

- **Before you begin:**

  Recommendations

  Security

- **To configure the two digit year cutoff option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the two digit year cutoff option

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- OLE Automation objects use 2030 as the two-digit cutoff year. You can use the **two digit year cutoff** option to provide consistency in date values between SQL Server and client applications. However, to avoid ambiguity with dates, use four-digit years in your data.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the two digit year cutoff option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Misc server settings** node.

3. Under **Two digit year support**, in the **When a two digit year is entered**, **interpret it as a year between** box, type or select a value that is the ending year of the time span.

Using Transact-SQL

To configure the two digit year cutoff option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to set the value of the `two digit year cutoff` option to `2030`.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'two digit year cutoff', 2030 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the two digit year cutoff option

The setting takes effect immediately without restarting the server.

See Also

Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)

Configure the user connections Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to set the **user connections** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **user connections** option specifies the maximum number of simultaneous user connections that are allowed on an instance of SQL Server. The actual number of user connections allowed also depends on the version of SQL Server that you are using, and also the limits of your application or applications and hardware. SQL Server allows a maximum of 32,767 user connections. Because **user connections** is a dynamic (self-configuring) option, SQL Server adjusts the maximum number of user connections automatically as needed, up to the maximum value allowable. For example, if only 10 users are logged in, 10 user connection objects are allocated. In most cases, you do not have to change the value for this option. The default is 0, which means that the maximum (32,767) user connections are allowed.

To determine the maximum number of user connections that your system allows, you can execute sp_configure or query the sys.configuration catalog view.

In This Topic

Before You Begin

Recommendations

- This option is an advanced option and should be changed only by an experienced database administrator or certified SQL Server technician.

- Using the **user connections** option helps avoid overloading the server with too many concurrent connections. You can estimate the number of connections based on system and user requirements. For example, on a system with many users, each user would not usually require a unique connection. Connections can be shared among users. Users running OLE DB applications need a connection for each open connection object, users running Open Database Connectivity (ODBC) applications need a connection for each active connection handle in the application, and users running DB-Library applications need one connection for each process started that calls the DB-Library **dbopen** function.

IMPORTANT

If you must use this option, do not set the value too high, because each connection has overhead regardless of whether the connection is being used. If you exceed the maximum number of user connections, you receive an error message and are not able to connect until another connection becomes available.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the user connections option

1. In Object Explorer, right-click a server and click **Properties**.
2. Click the **Connections** node.
3. Under **Connections**, in the **Max number of concurrent connections** box, type or select a value from 0 through 32767 to set the maximum number of users that are allowed to connect simultaneously to the instance of SQL Server.
4. Restart SQL Server.

Using Transact-SQL

To configure the user connections option

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the value of the `user connections` option to `325` users.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'user connections', 325 ;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Follow Up: After you configure the user connections option

The server must be restarted before the setting can take effect.

See Also

RECONFIGURE (Transact-SQL)

Server Configuration Options (SQL Server)

sp_configure (Transact-SQL)

Configure the user options Server Configuration Option

3/24/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

This topic describes how to configure the **user options** server configuration option in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. The **user options** option specifies global defaults for all users. A list of default query processing options is established for the duration of a user's work session. The **user options** option allows you to change the default values of the SET options (if the server's default settings are not appropriate).

A user can override these defaults by using the SET statement. You can configure **user options** dynamically for new logins. After you change the setting of **user options**, new login sessions use the new setting; current login sessions are not affected.

In This Topic

- **Before you begin:**

  Recommendations

  Security

- **To configure the user options configuration option, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure the user options configuration option

Before You Begin

Recommendations

- The following table lists and describes the configuration values for **user options**. Not all configuration values are compatible with each other. For example, ANSI_NULL_DFLT_ON and ANSI_NULL_DFLT_OFF cannot be set at the same time.

VALUE	CONFIGURATION	DESCRIPTION
1	DISABLE_DEF_CNST_CHK	Controls interim or deferred constraint checking.
2	IMPLICIT_TRANSACTIONS	For dblib network library connections, controls whether a transaction is started implicitly when a statement is executed. The IMPLICIT_TRANSACTIONS setting has no effect on ODBC or OLEDB connections.

VALUE	CONFIGURATION	DESCRIPTION
4	CURSOR_CLOSE_ON_COMMIT	Controls behavior of cursors after a commit operation has been performed.
8	ANSI_WARNINGS	Controls truncation and NULL in aggregate warnings.
16	ANSI_PADDING	Controls padding of fixed-length variables.
32	ANSI_NULLS	Controls NULL handling when using equality operators.
64	ARITHABORT	Terminates a query when an overflow or divide-by-zero error occurs during query execution.
128	ARITHIGNORE	Returns NULL when an overflow or divide-by-zero error occurs during a query.
256	QUOTED_IDENTIFIER	Differentiates between single and double quotation marks when evaluating an expression.
512	NOCOUNT	Turns off the message returned at the end of each statement that states how many rows were affected.
1024	ANSI_NULL_DFLT_ON	Alters the session's behavior to use ANSI compatibility for nullability. New columns defined without explicit nullability are defined to allow nulls.
2048	ANSI_NULL_DFLT_OFF	Alters the session's behavior not to use ANSI compatibility for nullability. New columns defined without explicit nullability do not allow nulls.
4096	CONCAT_NULL_YIELDS_NULL	Returns NULL when concatenating a NULL value with a string.
8192	NUMERIC_ROUNDABORT	Generates an error when a loss of precision occurs in an expression.
16384	XACT_ABORT	Rolls back a transaction if a Transact-SQL statement raises a run-time error.

- The bit positions in **user options** are identical to those in @@OPTIONS. Each connection has its own @@OPTIONS function, which represents the configuration environment. When logging in to an instance of \ SQL Server, a user receives a default environment that assigns the current **user options** value to @@OPTIONS. Executing SET statements for **user options** affects the corresponding value in the session's @@OPTIONS function. All connections created after this setting is changed receive the new value.

Security

Permissions

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To configure the user options configuration option

1. In Object Explorer, right-click a server and select **Properties**.

2. Click the **Connections** node.

3. In the **Default connection options** box, select one or more attributes to configure the default query-processing options for all connected users.

   By default, no user options are configured.

Using Transact-SQL

To configure the user options configuration option

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_configure to configure the `user options` to change the setting for the ANSI_WARNINGS server option.

```
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'user options', 8 ;
GO
RECONFIGURE ;
GO
```

Follow Up: After you configure the user options configuration option

The setting takes effect immediately without restarting the server.

See Also

RECONFIGURE (Transact-SQL)
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
SET Statements (Transact-SQL)

xp_cmdshell Server Configuration Option

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

The **xp_cmdshell** option is a SQL Server server configuration option that enables system administrators to control whether the **xp_cmdshell** extended stored procedure can be executed on a system. By default, the **xp_cmdshell** option is disabled on new installations and can be enabled by using the Policy-Based Management or by running the **sp_configure** system stored procedure as shown in the following code example:

```
-- To allow advanced options to be changed.
EXEC sp_configure 'show advanced options', 1;
GO
-- To update the currently configured value for advanced options.
RECONFIGURE;
GO
-- To enable the feature.
EXEC sp_configure 'xp_cmdshell', 1;
GO
-- To update the currently configured value for this feature.
RECONFIGURE;
GO
```

See Also

Server Configuration Options (SQL Server)
Administer Servers by Using Policy-Based Management

View or Change the Default Locations for Data and Log Files

3/29/2017 • 1 min to read • Edit Online

This topic describes how to view and change the default locations of new data and log files in SQL Server 2016 using SQL Server Management Studio. The default path is obtained from the registry. After you change the location all new databases created in the instance of SQL Server will use that location if a different location is not specified.

Recommendations

The best practice for protecting your data files and log files is to ensure that they are protected by access control lists (ACLs). Set the ACLs on the directory root under which the files are created.

View or change the default locations for database files

1. In Object Explorer, right-click on your server and click **Properties**.
2. In the left panel on that Properties page, click the **Database settings** tab.
3. In **Database default locations**, view the current default locations for new data files and new log files. To change a default location, enter a new default pathname in the **Data** or **Log** field, or click the browse button to find and select a pathname.

> **NOTE:** After changing the default locations, you must stop and start the SQL Server service to complete the change.

See also

CREATE DATABASE (SQL Server Transact-SQL)
Create a Database

View or Configure Remote Server Connection Options (SQL Server)

3/24/2017 • 1 min to read • Edit Online

This topic describes how to view or configure remote server connection options at the server level in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL.

In This Topic

- **Before you begin:**

  Security

- **To view or configure remote server connection options, using:**

  SQL Server Management Studio

  Transact-SQL

- **Follow Up:** After you configure remote server connection options

Before You Begin

Security

Permissions

Executing **sp_serveroption** requires ALTER ANY LINKED SERVER permission on the server.

Using SQL Server Management Studio

To view or configure remote server connection options

1. In Object Explorer, right-click a server, and then click **Properties**.
2. In the **SQL Server Properties - <***server_name***>** dialog box, click **Connections**.
3. On the **Connections** page, review the **Remote server connections** settings, and modify them if necessary.
4. Repeat steps 1 through 3 on the other server of the remote server pair.

Using Transact-SQL

To view remote server connection options

1. Connect to the Database Engine.
2. From the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. This example uses sp_helpserver to return information about all remote servers.

```
USE master;
GO
EXEC sp_helpserver ;
```

To configure remote server connection options

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how to use sp_serveroption to configure a remote server. The example configures a remote server corresponding to another instance of SQL Server, `SEATTLE3`, to be collation compatible with the local instance of SQL Server.

```
USE master;
EXEC sp_serveroption 'SEATTLE3', 'collation compatible', 'true';
```

Follow Up: After you configure remote server connection options

The remote server must be stopped and restarted before the setting can take effect.

See Also

Server Configuration Options (SQL Server)
Remote Servers
Linked Servers (Database Engine)
sp_linkedservers (Transact-SQL)
sp_helpserver (Transact-SQL)
sp_serveroption (Transact-SQL)

View or Change Server Properties (SQL Server)

3/24/2017 • 4 min to read • Edit Online

This topic describes how to view or change the properties of an instance of SQL Server by using SQL Server Management Studio, Transact-SQL, or SQL Server Configuration Manager.

In This Topic

Before You Begin

Limitations and Restrictions

- When using sp_configure, you must run either RECONFIGURE or RECONFIGURE WITH OVERRIDE after setting a configuration option. The RECONFIGURE WITH OVERRIDE statement is usually reserved for configuration options that should be used with extreme caution. However, RECONFIGURE WITH OVERRIDE works for all configuration options, and you can use it in place of RECONFIGURE.

  > **NOTE**
  >
  > RECONFIGURE executes within a transaction. If any of the reconfigure operations fail, none of the reconfigure operations will take effect.

- Some property pages present information obtained via Windows Management Instrumentation (WMI). To display those pages, WMI must be installed on the computer running SQL Server Management Studio.

Security

Permissions

For more information, see Server-Level Roles.

Execute permissions on **sp_configure** with no parameters or with only the first parameter are granted to all users by default. To execute **sp_configure** with both parameters to change a configuration option or to run the RECONFIGURE statement, a user must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.

Using SQL Server Management Studio

To view or change server properties

1. In Object Explorer, right-click a server, and then click **Properties**.

2. In the **Server Properties** dialog box, click a page to view or change server information about that page. Some properties are read-only.

Using Transact-SQL

To view server properties by using the SERVERPROPERTY built-in function

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example uses the SERVERPROPERTY built-in function in a `SELECT` statement to return information about the current server. This scenario is useful when there are multiple instances of SQL Server installed on a Windows-based server, and the client must open another connection to the same instance that is used by the current connection.

```
SELECT CONVERT( sysname, SERVERPROPERTY('servername'));
GO
```

To view server properties by using the sys.servers catalog view

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example queries the sys.servers catalog view to return the name (`name`) and ID (`server_id`) of the current server, and the name of the OLE DB provider (`provider`) for connecting to a linked server.

```
USE AdventureWorks2012;
GO
SELECT name, server_id, provider
FROM sys.servers ;
GO
```

To view server properties by using the sys.configurations catalog view

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example queries the sys.configurations catalog view to return information about each server configuration option on the current server. The example returns the name (`name`) and description (`description`) of the option and whether the option is an advanced option (`is_advanced`).

```
USE AdventureWorks2012;
GO
SELECT name, description, is_advanced
FROM sys.configurations ;
GO
```

To change a server property by using sp_configure

1. Connect to the Database Engine.

2. From the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. This example shows how

to use sp_configure to change a server property. The example changes the value of the `fill factor` option to `100`. The server must be restarted before the change can take effect.

```
Use AdventureWorks2012;
GO
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'fill factor', 100;
GO
RECONFIGURE;
GO
```

For more information, see Server Configuration Options (SQL Server).

Using SQL Server Configuration Manager

Some server properties can be viewed or changed by using SQL Server Configuration Manager. For example, you can view the version and edition of the instance of SQL Server, or change the location where error log files are stored. These properties can also be viewed by querying the Server-Related Dynamic Management Views and Functions.

To view or change server properties

1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

2. In **SQL Server Configuration Manager**, click **SQL Server Services**.

3. In the details pane, right-click **SQL Server (<***instancename***>)**, and then click **Properties**.

4. In the **SQL Server (<***instancename***>) Properties** dialog box, change the server properties on the **Service** tab or the **Advanced** tab, and then click **OK**.

Follow Up: After you change server properties

For some properties, the server might have to be restarted before the change can take effect.

See Also

Server Configuration Options (SQL Server)
SET Statements (Transact-SQL)
SERVERPROPERTY (Transact-SQL)
sp_configure (Transact-SQL)
RECONFIGURE (Transact-SQL)
SELECT (Transact-SQL)
Configure WMI to Show Server Status in SQL Server Tools
SQL Server Configuration Manager
Configuration Functions (Transact-SQL)
Server-Related Dynamic Management Views and Functions (Transact-SQL)

Server Properties - General Page

3/24/2017 • 1 min to read • Edit Online

Use this page to view read-only information about your Microsoft SQL Server installation.

Property Grid

View properties for the selected server, such as the server name, server operating system, or number of processors.

Name
Displays the name of the server instance.

Product
Displays the edition of SQL Server currently running.

Operating System
Displays the Microsoft Windows operating system currently running.

Platform
Describes the operating system and hardware running SQL Server.

Version
Displays the version number of the SQL Server edition currently running.

Language
Displays the language supported by the running instance of SQL Server.

Memory
Lists the amount of RAM installed on the server.

Processors
Displays the number of CPUs installed.

Root Directory
Displays the path to the location of the SQL Server instance, typically C:\Program Files\Microsoft SQL Server\.

Server Collation
Displays the collation supported by the server. A collation specifies the particular code page and sort order to use for Unicode and non-Unicode data.

Is Clustered
Displays **True** if the server instance is configured in a SQL Server failover cluster or **False** if the server instance is not clustered.

Is HADR Enabled
Displays **True** if the Always On Availability Groups feature is enabled or **False** if the Always On Availability Groups feature is disabled. For information about enabling or disabling Always On Availability Groups, see Enable and Disable Always On Availability Groups (SQL Server).

Description Field

View additional information on the server properties.

See Also

Server Configuration Options (SQL Server)

Server Properties - Memory Page

3/24/2017 • 2 min to read • Edit Online

Use this page to view or modify your server memory options. When **Minimum server memory** is set to 0 and **Maximum server memory** is set to 2147483647 MB, SQL Server can take advantage of the optimum amount of memory at any given time, subject to how much memory the operating system and other applications are currently using. As the load on the computer and SQL Server changes, so does the memory allocated. You can further limit this dynamic memory allocation to the minimum and maximum values specified below.

Options

Minimum server memory (in MB)
Specifies that SQL Server should start with at least the minimum amount of allocated memory and not release memory below this value. Set this value based on the size and activity of your instance of SQL Server. Always set the option to a reasonable value to ensure that the operating system does not request too much memory from SQL Server and inhibit Windows performance.

Maximum server memory (in MB)
Specifies the maximum amount of memory SQL Server can allocate when it starts and while it runs. This configuration option can be set to a specific value if you know there are multiple applications running at the same time as SQL Server and you want to guarantee that these applications have sufficient memory to run. If these other applications, such as Web or e-mail servers, request memory only as needed, then do not set the option, because SQL Server will release memory to them as needed. However, applications often use whatever memory is available when they start and do not request more if needed. If an application that behaves in this manner runs on the same computer at the same time as SQL Server, set the option to a value that guarantees that the memory required by the application is not allocated by SQL Server. The minimum amounts of memory you can specify for **max server memory** is 128 MB. (64 megabytes (MB) for older 32-bit systems.)

Index creation memory (in KB, 0 = dynamic memory)
Specifies the amount of memory (in KB) to use during index creation sorts. The default value of zero enables dynamic allocation and should work in most cases without additional adjustment; however, the user can enter a different value from 704 to 2147483647.

NOTE

Values from 1 to 703 are not allowed. If a value in this range is entered, the field overrides the entered value with 704.

Minimum memory per query (in KB)
Specifies the amount of memory (in KB) to allocate for the execution of a query. The user can set the value from 512 to 2147483647 KB. The default value is 1024 KB.

Configured Values
Displays the configured values for the options on this pane. If you change these values, click **Running Values** to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restarted first.

Running Values
Shows the currently running values for the options on this pane. These values are read-only.

See Also

Server Configuration Options (SQL Server)
Server Memory Server Configuration Options

Server Properties - Processors Page

3/24/2017 • 1 min to read • Edit Online

Use this page to view or modify your processor options. Processor affinity settings are only enabled when more than one processor is installed.

Options

Processor Affinity
Assigns processors to specific threads to eliminating processor reloads and reduce thread migration across processors. For more information, see affinity mask Server Configuration Option.

I/O Affinity
Binds Microsoft SQL Server disk I/Os to a specified subset of CPUs. For more information, see affinity Input-Output mask Server Configuration Option.

Automatically set processor affinity mask for all processors
Allows SQL Server to set the processor affinity.

Automatically set I/O affinity mask for all processors
Allows SQL Server to set the I/O affinity.

Maximum worker threads
0 allows SQL Server to dynamically set the number of worker threads. This setting is best for most systems. However, depending on your system configuration, setting this option to a specific value sometimes improves performance. For more information, see Configure the max worker threads Server Configuration Option.

Boost SQL Server priority
Specifies whether SQL Server should run at a higher Microsoft Windows scheduling priority than other processes on the same computer. For more information, see Configure the priority boost Server Configuration Option.

Use Windows fibers (lightweight pooling)
Use Windows fibers instead of threads for the SQL Server service. Note that this is only available in Windows 2003 Server Edition. For more information, see lightweight pooling Server Configuration Option.

Configured Values
Displays the configured values for the options on this pane. If you change these values, click **Running Values** to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restarted first.

Running Values
View the currently running values for the options on this pane. These values are read-only.

See Also

Server Configuration Options (SQL Server)

Server Properties - Security Page

3/24/2017 • 1 min to read • Edit Online

Use this page to view or modify your server security options.

Server Authentication

Windows Authentication mode
Uses Windows Authentication to validate attempted connections. If the **sa** password is blank when the security mode is being changed, the user is prompted to enter an **sa** password.

> **IMPORTANT**
> Windows Authentication is much more secure than SQL Server Authentication. When possible, you should use Windows Authentication.

SQL Server and Windows Authentication mode
Uses mixed mode authentication to verify attempted connections, for backward compatibility with earlier versions of SQL Server. If the **sa** password is blank when the security mode is being changed, the user is prompted to enter an **sa** password.

> **NOTE**
> Changing the security configuration requires a restart of the service. When changing the Server Authentication to SQL Server and Windows Authentication mode the SA account is not automatically enabled. To use the SA account, execute ALTER LOGIN with the ENABLE option.

Login Auditing

None
Turns off login auditing.

Failed logins only
Audits unsuccessful logins only.

Successful logins only
Audits successful logins only.

Both failed and successful logins
Audits all login attempts.

> **NOTE**
> Changing the audit level requires restarting the service.

Server Proxy Account

Enable server proxy account
Enables an account for use by **xp_cmdshell**. Proxy accounts allow for the impersonation of logins, server roles, and

database roles when an operating system command is being executed.

Caution

The login used by the server proxy account should have the least privileges required to perform the intended work. Excessive privileges for the proxy account could be used by a malicious user to compromise your system security.

Proxy account
Specify the proxy account used.

Password
Specify the password for the proxy account.

Options

Enable C2 audit tracing
Audits all attempts to access statements and objects and records them to a file in the \MSSQL\Data directory for default instances of SQL Server, or the \MSSQL$*instancename*\Data directory for named instances of SQL Server. For more information, see c2 audit mode Server Configuration Option.

Cross database ownership chaining
Select to allow the database to be the source or target of a cross-database ownership chain. For more information, see cross db ownership chaining Server Configuration Option.

See Also

Server Configuration Options (SQL Server)

Server Properties - Connections Page

3/24/2017 • 2 min to read • Edit Online

Use this page to view or modify your connection options.

Connections

Maximum number of concurrent connections (0 = unlimited)
If set to a value other than zero, limits the number of connections that SQL Server will allow.

Caution
Setting this to a small value, such as 1 or 2, can prevent administrators from connecting to administer the server; however, the Dedicated Admin Connection can always connect.

Default Connection Options

Default connection options
Specifies the default connection options, as described in the following table.

CONFIGURATION OPTION	DESCRIPTION
disable deferred constraint checking	Controls interim or deferred constraint checking.
implicit transactions	Controls whether a transaction is started implicitly when a statement is run.
cursor close on commit	Controls behavior of cursors after a commit operation has been performed.
ansi warnings	Controls truncation and NULL in aggregate warnings.
ansi padding	Controls padding of fixed-length variables.
ansi nulls	Controls `NULL` handling when using equality operators.
arithmetic abort	Terminates a query when an overflow or divide-by-zero error occurs during query execution.
arithmetic ignore	Returns NULL when an overflow or divide-by-zero error occurs during a query.
quoted identifier	Differentiates between single and double quotation marks when evaluating an expression.
no count	Turns off the message returned at the end of each statement that states how many rows were affected.
ansi null default on	Alters the session's behavior to use ANSI compatibility for nullability. New columns defined without explicit nullability are defined to allow nulls.

CONFIGURATION OPTION	DESCRIPTION
ansi null default off	Alters the session's behavior not to use ANSI compatibility for nullability. New columns defined without explicit nullability are defined not to allow nulls.
concat null yields null	Returns NULL when concatenating a NULL value with a string.
numeric round abort	Generates an error when a loss of precision occurs in an expression.
xact abort	Rolls back a transaction if a Transact-SQL statement raises a run-time error.

For more information on connection options, search Books Online for the specific option.

Remote Server Connections

Allow remote connections to this server
Controls the execution of stored procedures from remote servers running instances of SQL Server. Selecting this check box has the same effect as setting the **sp_configureremote access** option to 1. Clearing it prevents execution of stored procedures from a remote server.

Remote query timeout (in seconds, 0 = no timeout)
Specifies how long (in seconds) a remote operation may take before SQL Server times out. The default is 600 seconds, or a 10-minute wait.

Require distributed transactions for server-to-server communication
Protects the actions of a server-to-server procedure through a Microsoft Distributed Transaction Coordinator (MS DTC) transaction. For more information, see Configure the remote proc trans Server Configuration Option.

Property Page Display Options

Configured Values
Displays the configured values for the options on this pane. If you change these values, click **Running Values** to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restarted first.

Running Values
View the currently running values for the options on this pane. These values are read-only.

See Also

Options (Query Execution: SQL Server: Advanced Page)
Server Configuration Options (SQL Server)

Server Properties - Database Settings Page

3/24/2017 • 2 min to read • Edit Online

Use this page to view or modify your database settings.

Options

Default index fill factor
Specifies how full SQL Server should make each page when it creates a new index using existing data. The fill factor affects performance because SQL Server must take time to split pages when they fill up.

The default value is 0; valid values range from 0 through 100. A fill factor of 0 or 100 creates clustered indexes with full data pages and nonclustered indexes with full leaf pages, but it leaves some space within the upper level of the index tree. Fill factor values 0 and 100 are identical in all respects.

Small fill factor values cause SQL Server to create indexes with pages that are not full. Each index takes more storage space, but there is more room for subsequent insertions without requiring page splits.

Wait indefinitely
Specifies that SQL Server will never time out while waiting for a new backup tape.

Try once
Specifies that SQL Server will time out if a backup tape is not available when needed.

Try for minute(s)
Specifies that SQL Server will time out if a backup tape is not available within the period specified.

Default backup media retention (in days)
Provides a system-wide default for the length of time to retain each backup medium after it has been used for a database or transaction log backup. This option helps protect backups from being overwritten until the specified number of days has elapsed.

Compress backup
In SQL Server 2008 Enterprise (or later versions), indicates the current setting of the **backup compression default** option. This option determines the server-level default for compressing backups, as follows:

- If the **Compress backup** box is blank, new backups are uncompressed by default.
- If the **Compress backup** box is checked, new backups are compressed by default.

> IMPORTANT
> By default, compression significantly increases CPU usage, and the additional CPU consumed by the compression process might adversely affect concurrent operations. Therefore, you might want to create low-priority compressed backups in a session whose CPU usage is limited by Resource Governor. For more information, see Use Resource Governor to Limit CPU Usage by Backup Compression (Transact-SQL).

If you are a member of the **sysadmin** or **serveradmin** fixed server role, you can change the setting by clicking the **Compress backup** box.

For more information, see View or Configure the backup compression default Server Configuration Option and Backup Compression (SQL Server).

Recovery interval (minutes)

Sets the maximum number of minutes per database to recover databases. The default is 0, indicating automatic configuration by SQL Server. In practice, this means a recovery time of less than one minute and a checkpoint approximately every one minute for active databases. For more information, see Configure the recovery interval Server Configuration Option.

Data
Specifies the default location for data files. Click the browse button to navigate to a new default location. Does not take effect until SQL Server is restarted.

Log
Specifies the default location for log files. Click the browse button to navigate to a new default location. Does not take effect until SQL Server is restarted.

Configured Values
Displays the configured values for the options on this pane. If you change these values, click **Running Values** to see whether the changes have taken effect. If they have not, the instance of SQL Server must be restated first.

Running Values
View the currently running values for the options on this pane. These values are read-only.

See Also

Server Configuration Options (SQL Server)
Specify Fill Factor for an Index

Server Properties - Misc Server Settings Page

3/24/2017 • 1 min to read • Edit Online

Use this page to view or modify your server settings.

Options

Default language for users
Specifies the default language for all newly created logins.

Allow triggers to fire other triggers
Controls whether a trigger can perform an action that initiates another trigger. When cleared, triggers cannot be fired by another trigger. When selected, triggers can be fired by other triggers to as many as 32 levels.

Use query governor to prevent long-running queries
Specifies an upper limit on the time within which a query can run. Query cost refers to the estimated elapsed time, in seconds, required to execute a query on a specific hardware configuration. By default, the query governor is turned off and all queries are allowed to run. If this option is selected, you must enter a time limit in the text box below. If you specify a nonzero, nonnegative value, the query governor disallows execution of any query that has an estimated cost exceeding that value.

Interpret a two-digit year as falling between
Specifies the 100-year date range for interpreting two-digit year values. Microsoft SQL Server will interpret two-digit date values to refer to the year ending in those digits that falls within the specified range.

Set the right box with the ending year. When the ending year is saved, SQL Server will automatically populate the left box with the beginning year.

Configured Values
Displays the configured values for the options on this pane. If you change these values, click **Running Values** to see whether the changes have taken effect. If the changes have not taken effect, the instance of SQL Server must be restated first.

Running Values
View the currently running values for the options on this pane. These values are read-only.

See Also

Server Configuration Options (SQL Server)

Server Properties - FILESTREAM Page

3/24/2017 • 1 min to read • Edit Online

Use this page to enable FILESTREAM for this installation of SQL Server 2016.

UIElement List

Enable FILESTREAM for Transact-SQL access
Select to enable FILESTREAM for Transact-SQL access. This control must be checked before the other control options will be available.

Enable FILESTREAM for file I/O streaming access
Select to enable Win32 streaming access for FILESTREAM.

Windows share name
Use this control to enter the name of the Windows share in which the FILESTREAM data will be stored.

Allow remote clients to have streaming access to FILESTREAM data
Select this control to allow remote clients to access this FILESTREAM data on this server.

See Also

FILESTREAM (SQL Server)
sp_configure (Transact-SQL)

Server Properties - Advanced Page

3/24/2017 • 7 min to read • Edit Online

Use this page to view or modify your advanced server settings.

To view the Server Properties pages

- View or Change Server Properties (SQL Server)

Containment

Enable Contained Databases
Indicates if this instance of SQL Server permits contained databases. When **True**, a contained database can be created, restored, or attached. When **False**, a contained database cannot be created, restored, or attached to this instance of SQL Server. Changing the containment property can have an impact on the security of the database. Enabling contained databases lets database owners grant access to this SQL Server. Disabling contained databases can prevent users from connecting. To understand the impact of the containment property, see Contained Databases and Security Best Practices with Contained Databases.

FILESTREAM

FILESTREAM Access Level
Shows the current level of FILESTREAM support on the instance of SQL Server. To change the access level, select one of the following values:

Disabled
Binary large object (BLOB) data cannot be stored on the file system. This is the default value.

Transact-SQL access enabled
FILESTREAM data is accessible by using Transact-SQL, but not through the file system.

Full access enabled
FILESTREAM data is accessible by using Transact-SQL and through the file system.

When you enable FILESTREAM for the first time, you might have to restart the computer to configure drivers.

FILESTREAM Share Name
Displays the read-only name of the FILESTREAM share that was selected during setup. For more information, see FILESTREAM (SQL Server).

Miscellaneous

Allow Triggers to Fire Others
Allows triggers to fire other triggers. Triggers can be nested to a maximum of 32 levels. For more information, see the "Nested Triggers" section in CREATE TRIGGER (Transact-SQL).

Blocked Process Threshold
The threshold, in seconds, at which blocked process reports are generated. The threshold can be set from 0 to 86,400. By default, no blocked process reports are produced. For more information, see blocked process threshold Server Configuration Option.

Cursor Threshold
Specifies the number of rows in the cursor set at which cursor keysets are generated asynchronously. When

cursors generate a keyset for a result set, the query optimizer estimates the number of rows that will be returned for that result set. If the query optimizer estimates that the number of returned rows is greater than this threshold, the cursor is generated asynchronously, allowing the user to fetch rows from the cursor while the cursor continues to be populated. Otherwise, the cursor is generated synchronously, and the query waits until all rows are returned.

If set to -1, all keysets are generated synchronously; this benefits small cursor sets. If set to 0, all cursor keysets are generated asynchronously. With other values, the query optimizer compares the number of expected rows in the cursor set and builds the keyset asynchronously if it exceeds the number set. For more information, see Configure the cursor threshold Server Configuration Option.

Default Full Text Language
Specifies a default language for full-text indexed columns. Linguistic analysis of full-text indexed data is dependent on the language of the data. The default value of this option is the language of the server. For the language that corresponds to the displayed setting, see sys.fulltext_languages (Transact-SQL).

Default Language
The default language for all new logins, unless otherwise specified.

Full-Text Upgrade Option
Controls how full-text indexes are migrated when upgrading a database from SQL Server 2005. This property applies to upgrading by attaching a database, restoring a database backup, restoring a file backup, or copying the database by using the Copy Database Wizard.

The alternatives are as follows:

Import
Full-text catalogs are imported. This operation is significantly faster than **Rebuild**. However, an imported full-text catalog does not use the new and enhanced word breakers that are introduced in SQL Server 2008. Therefore, you might want to rebuild your full-text catalogs eventually.

If a full-text catalog is not available, the associated full-text indexes are rebuilt. This option is available for only SQL Server 2005 databases.

Rebuild
Full-text catalogs are rebuilt using the new and enhanced word breakers. Rebuilding indexes can take awhile, and a significant amount of CPU and memory might be required after the upgrade.

Reset
Full-text catalogs are reset. SQL Server 2005 full-text catalog files are removed, but the metadata for full-text catalogs and full-text indexes is retained. After being upgraded, all full-text indexes are disabled for change tracking and crawls are not started automatically. The catalog will remain empty until you manually issue a full population, after the upgrade completes.

For information about how to choose the full-text upgrade option, see Upgrade Full-Text Search.

NOTE

The full-text upgrade option can also be set by using the sp_fulltext_serviceupgrade_option action.

After you attach, restore, or copy a SQL Server 2005 database to SQL Server 2016, the database becomes available immediately and is then automatically upgraded. If the database has full-text indexes, the upgrade process either imports, resets, or rebuilds them, depending on the setting of the **Full-Text Upgrade Option** server property. If the upgrade option is set to **Import** or **Rebuild**, the full-text indexes will be unavailable during the upgrade. Depending on the amount of data being indexed, importing can take several hours, and rebuilding can take up to ten times longer. Note also that when the upgrade option is set to **Import**, if a full-text catalog is not available, the associated full-text indexes are rebuilt. For information about viewing or changing the setting of the **Full-Text**

Upgrade Option property, see Manage and Monitor Full-Text Search for a Server Instance.

Max Text Replication Size
Specifies the maximum size (in bytes) of **text**, **ntext**, **varchar(max)**, **nvarchar(max)**, **xml**, and **image** data that can be added to a replicated column or captured column in a single INSERT, UPDATE, WRITETEXT, or UPDATETEXT statement. Changing the setting takes effect immediately. For more information, see Configure the max text repl size Server Configuration Option.

Scan For Startup Procs
Specifies that SQL Server will scan for automatic execution of stored procedures at startup. If set to **True**, SQL Server scans for and runs all automatically run stored procedures defined on the server. If set to **False** (the default), no scan is performed. For more information, see Configure the scan for startup procs Server Configuration Option.

Two Digit Year Cutoff
Indicates the highest year number that can be entered as a two-digit year. The year listed and the previous 99 years can be entered as a two-digit year. All other years must be entered as a four-digit year.

For example, the default setting of 2049 indicates that a date entered as '3/14/49' will be interpreted as March 14, 2049, and a date entered as '3/14/50' will be interpreted as March 14, 1950. For more information, see Configure the two digit year cutoff Server Configuration Option.

Network

Network Packet Size
Sets the packet size (in bytes) used across the whole network. The default packet size is 4096 bytes. If an application does bulk-copy operations or sends or receives large amounts of **text** or **image** data, a packet size larger than the default may improve efficiency, because it results in fewer network reads and writes. If an application sends and receives small amounts of information, you can set the packet size to 512 bytes, which is sufficient for most data transfers. For more information, see Configure the network packet size Server Configuration Option.

> **NOTE**
>
> Do not change the packet size unless you are certain that it will improve performance. For most applications, the default packet size is best.

Remote Login Timeout
Specifies the number of seconds SQL Server waits before returning from a failed remote login attempt. This setting affects connections to OLE DB providers made for heterogeneous queries. The default value is 20 seconds. A value of 0 allows for an infinite wait. For more information, see Configure the remote login timeout Server Configuration Option.

Changing the setting takes effect immediately.

Parallelism:

Cost Threshold for Parallelism
Specifies the threshold above which SQL Server creates and runs parallel plans for queries. The cost refers to an estimated elapsed time in seconds required to run the serial plan on a specific hardware configuration. Only set this option on symmetric multiprocessors. For more information, see Configure the cost threshold for parallelism Server Configuration Option.

Locks
Sets the maximum number of available locks, thereby limiting the amount of memory SQL Server uses for them. The default setting is 0, which allows SQL Server to allocate and deallocate locks dynamically based on changing system requirements.

Allowing SQL Server to use locks dynamically is the recommended configuration. For more information, see Configure the locks Server Configuration Option.

Max Degree of Parallelism
Limits the number of processors (up to a maximum of 64) to use in parallel plan execution. The default value of 0 uses all available processors. A value of 1 suppresses parallel plan generation. A number greater than 1 restricts the maximum number of processors used by a single query execution. If a value greater than the number of available processors is specified, the actual number of available processors is used. For more information, see Configure the max degree of parallelism Server Configuration Option.

Query Wait
Specifies the time in seconds (from 0 through 2147483647) that a query waits for resources before timing out. If the default value of -1 is used, the time-out is calculated as 25 times of the estimated query cost. For more information, see Configure the query wait Server Configuration Option.

See Also

Server Configuration Options (SQL Server)

Soft-NUMA (SQL Server)

3/24/2017 • 6 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

Modern processors have multiple to many cores per socket. Each socket is represented, usually, as a single NUMA node. The SQL Server database engine partitions various internal structures and partitions service threads per NUMA node. With processors containing 10 or more cores per socket, using software NUMA to split hardware NUMA nodes generally increases scalability and performance. Prior to SQL Server 2014 SP2, software-based NUMA (soft-NUMA) required you to edit the registry to add a node configuration affinity mask and was configured per computer rather than per instance. With SQL Server 2014 SP2 and SQL Server 2016, soft-NUMA is configured automatically at the database-instance level when the SQL Server service starts.

NOTE

Hot-add processors are not supported by soft-NUMA.

Automatic Soft-NUMA

With SQL Server 2016, whenever the database engine server detects more than 8 physical cores per NUMA node or socket at startup, soft-NUMA nodes are created automatically by default. Hyper-threaded processor cores are not differentiated when counting physical cores in a node. When the number of physical cores detected is more than 8 per socket, the database engine service will create soft-NUMA nodes that ideally contain 8 cores, but can go down to 5 or up to 9 logical cores per node. The size of the hardware node can be limited by a CPU affinity mask. See
ALTER SERVER CONFIGURATION (Transact-SQL). The number of NUMA nodes will never exceed the maximum number of supported NUMA nodes.

You can disable or re-enable soft-NUMA using the ALTER SERVER CONFIGURATION (Transact-SQL) statement with the SET SOFTNUMA argument. Changing the value of this setting requires a restart of the database engine to take effect.

The figure below shows the type of information regarding soft-NUMA that you will see in the SQL Server error log when SQL Server detects hardware NUMA nodes with greater than 8 physical cores in each node or socket.

```
2016-11-14 13:39:43.17 Server SQL Server detected 2 sockets with 12 cores per socket and 24 logical processors
per socket, 48 total logical processors; using 48 logical processors based on SQL Server licensing. This is an
informational message; no user action is required.

2016-11-14 13:39:43.35 Server Automatic soft-NUMA was enabled because SQL Server has detected hardware NUMA
nodes with greater than 8 physical cores.

2016-11-14 13:39:43.63 Server Node configuration: node 0: CPU mask: 0x0000000000555555:0 Active CPU mask:
0x0000000000555555:0. This message provides a description of the NUMA configuration for this computer. This is
an informational message only. No user action is required.

2016-11-14 13:39:43.63 Server Node configuration: node 1: CPU mask: 0x0000000000aaaaaa:0 Active CPU mask:
0x0000000000aaaaaa:0. This message provides a description of the NUMA configuration for this computer. This is
an informational message only. No user action is required.

2016-11-14 13:39:43.63 Server Node configuration: node 2: CPU mask: 0x0000555555000000:0 Active CPU mask:
0x0000555555000000:0. This message provides a description of the NUMA configuration for this computer. This is
an informational message only. No user action is required.

2016-11-14 13:39:43.63 Server Node configuration: node 3: CPU mask: 0x0000aaaaaa000000:0 Active CPU mask:
0x0000aaaaaa000000:0. This message provides a description of the NUMA configuration for this computer. This is
an informational message only. No user action is required.
```

Manual Soft-NUMA

To manually configure SQL Server to use soft-NUMA by disabling automatic soft_NUMA and editing the registry to add a node configuration affinity mask. When using this method, the soft-NUMA mask can be stated as a binary, DWORD (hexadecimal or decimal), or QWORD (hexadecimal or decimal) registry entry. To configure more than the first 32 CPUs use QWORD or BINARY registry values. (QWORD values cannot be used prior to SQL Server 2012.) After modifying the registry, you must restart the Database Engine for the soft-NUMA configuration to take effect.

TIP

CPUs are numbered starting with 0.

WARNING

Incorrectly editing the registry can severely damage your system. Before making changes to the registry, we recommend that you back up any valued data on the computer.

Consider the following example. A computer with eight CPUs does not have hardware NUMA. Three soft-NUMA nodes are configured.
Database Engine instance A is configured to use CPUs 0 through 3. A second instance of the Database Engine is installed and configured to use CPUs 4 through 7. The example can be visually represented as:

```
CPUs 0 1 2 3 4 5 6 7
```

```
Soft-NUMA <-N0--><-N1-><----N2---->
```

```
SQL Server <instance A ><instance B>
```

Instance A, which experiences significant I/O, now has two I/O threads and one lazy writer thread, while instance B, which performs processor-intensive operations, has only one I/O thread and one lazy writer thread. Differing amounts of memory can be assigned to the instances, but unlike hardware NUMA, they both receive memory from the same operating system memory block and there is no memory-to-processor affinity.

The lazy writer thread is tied to the SQL OS view of the physical NUMA memory nodes. Therefore, whatever the hardware presents as physical NUMA nodes will equate to the number of lazy writer threads that are created. For more information, see
How It Works: Soft NUMA, I/O Completion Thread, Lazy Writer Workers and Memory Nodes.

NOTE

The **Soft-NUMA** registry keys are not copied when you upgrade an instance of SQL Server.

Set the CPU affinity mask

Run the following statement on instance A to configure it to use CPUs 0, 1, 2, and 3 by setting the CPU affinity mask:

```
ALTER SERVER CONFIGURATION
SET PROCESS AFFINITY CPU=0 TO 3;
```

Run the following statement on instance B to configure it to use CPUs 4, 5, 6, and 7 by setting the CPU affinity mask:

```
ALTER SERVER CONFIGURATION
SET PROCESS AFFINITY CPU=4 TO 7;
```

Map soft-NUMA nodes to CPUs

Using the Registry Editor program (regedit.exe), add the following registry keys to map soft-NUMA node 0 to CPUs 0 and 1, soft-NUMA node 1 to CPUs 2 and 3, and soft-NUMA node 2 to CPUs 4. 5, 6, and 7.

TIP

To specify CPUs 60 through 63, use a QWORD value of F000000000000000 or a BINARY value of 1111000000000000000000000000000000000000000000000000000000000000.

In the following example, assume you have a DL580 G9 server, with 18 cores per socket (in 4 sockets), and each socket is in its own K-group. A soft-numa configuration that you might create would look something like following. (6 cores per Node, 3 nodes per group, 4 groups).

EXAMPLE FOR A SQL SERVER 2016 SERVER WITH MULTIPLE K-GROUPS	TYPE	VALUE NAME	VALUE DATA
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node0	DWORD	CPUMask	0x3F
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node0	DWORD	Group	0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node1	DWORD	CPUMask	0x0fc0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node1	DWORD	Group	0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node2	DWORD	CPUMask	0x3f000
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node2	DWORD	Group	0

EXAMPLE FOR A SQL SERVER 2016 SERVER WITH MULTIPLE K-GROUPS	TYPE	VALUE NAME	VALUE DATA
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node3	DWORD	CPUMask	0x3F
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node3	DWORD	Group	1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node4	DWORD	CPUMask	0x0fc0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node4	DWORD	Group	1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node5	DWORD	CPUMask	0x3f000
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node5	DWORD	Group	1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node6	DWORD	CPUMask	0x3F
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node6	DWORD	Group	2
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node7	DWORD	CPUMask	0x0fc0

EXAMPLE FOR A SQL SERVER 2016 SERVER WITH MULTIPLE K-GROUPS	TYPE	VALUE NAME	VALUE DATA
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node7	DWORD	Group	2
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node8	DWORD	CPUMask	0x3f000
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node8	DWORD	Group	2
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node9	DWORD	CPUMask	0x3F
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node9	DWORD	Group	3
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node10	DWORD	CPUMask	0x0fc0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node10	DWORD	Group	3
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node11	DWORD	CPUMask	0x3f000
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\130\NodeConfiguration\Node11	DWORD	Group	3

Metadata

You can use the following DMVs to view the current state and configuration of soft_NUMA.

- sp_configure (Transact-SQL): Displays the current value (0 or 1) for SOFTNUMA
- sys.dm_os_nodes (Transact-SQL): The node_state_desc column for each NUMA node will show whether the node was created by soft-NUMA or not.
- sys.dm_os_sys_info (Transact-SQL): The softnuma and softnuma_desc columns will show the configuration values.

NOTE

While you can view the running value for automatic soft-NUMA using sp_configure (Transact-SQL), you cannot change its value using **sp_configure**. You must use the ALTER SERVER CONFIGURATION (Transact-SQL) statement with the SET SOFTNUMA argument.

See Also

Map TCP IP Ports to NUMA Nodes (SQL Server)
affinity mask Server Configuration Option
ALTER SERVER CONFIGURATION (Transact-SQL)

Map TCP IP Ports to NUMA Nodes (SQL Server)

3/24/2017 • 1 min to read • Edit Online

This topic describes how to map TCP/IP ports to non-uniform memory access (NUMA) nodes by using SQL Server Configuration Manager. On startup, the Database Engine writes the node information to the error log.

To determine the node number of the node you want to use, either read the node information from the error log, or from the **sys.dm_os_schedulers** view. To set a TCP/IP address and port to single or multiple nodes, append a node identification bitmap (an affinity mask) in brackets after the port number. Nodes can be specified in either decimal or hexadecimal format. To create the bitmap, first number the nodes from right to left starting with zero, as in 76543210. Create a binary representation of the node list, providing 1 for nodes you want to use, and 0 for nodes you do not want to use. For example, to use NUMA nodes 0, 2, and 5, specify 00100101.

NUMA node number	76543210
Mask for 0, 2, and 5 counting from right	00100101

Convert the binary representation (00100101), into decimal `[37]`, or hexadecimal `[0x25]`. To listen on all nodes, provide no node identifier.

If a port is mapped to more than one NUMA node, SQL Server assigns connections to nodes in a round-robin fashion without attempting to balance load across the nodes.

NOTE

To enable SQL Server to listen on multiple TCP ports for each IP address, see Configure the Database Engine to Listen on Multiple TCP Ports.

Using SQL Server Configuration Manager

To map a TCP/IP port to a NUMA node

1. In SQL Server Configuration Manager, expand **SQL Server Network Configuration**, and then click **Protocols for** <*instance name*>.

2. In the details pane, double-click **TCP/IP**.

3. On the **IP Addresses** tab, in the section corresponding to the IP address to configure, in the **TCP Port** box, add the NUMA node identifier in brackets after the port number. For example, for TCP port 1500 and nodes 0, 2, and 5, use **1500[37]**, or **1500[0x25]**.

See Also

Soft-NUMA (SQL Server)

Enable the Lock Pages in Memory Option (Windows)

3/24/2017 • 1 min to read • Edit Online

This Windows policy determines which accounts can use a process to keep data in physical memory, preventing the system from paging the data to virtual memory on disk.

NOTE

Locking pages in memory may boost performance when paging memory to disk is expected.

Use the Windows Group Policy tool (gpedit.msc) to enable this policy for the account used by SQL Server. You must be a system administrator to change this policy.

To enable the lock pages in memory option

1. On the **Start** menu, click **Run**. In the **Open** box, type **gpedit.msc**.
2. On the **Local Group Policy Editor** console, expand **Computer Configuration**, and then expand **Windows Settings**.
3. Expand **Security Settings**, and then expand **Local Policies**.
4. Select the **User Rights Assignment** folder.

   The policies will be displayed in the details pane.
5. In the pane, double-click **Lock pages in memory**.
6. In the **Local Security Setting – Lock pages in memory** dialog box, click **Add User or Group**.
7. In the **Select Users, Service Accounts, or Groups** dialog box, add an account with privileges to run sqlservr.exe.
8. Log out and then log back in for this change to take effect.

See Also

Server Memory Server Configuration Options

Manage the Database Engine Services

3/24/2017 • 2 min to read • Edit Online

Microsoft SQL Server runs on the operating systems as a service. A service is a type of application that runs in the system background. Services usually provide core operating system features, such as Web serving, event logging, or file serving. Services can run without showing a user interface on the computer desktop. The SQL Server Database Engine, SQL Server Agent, and several other SQL Server components run as services. These services typically are started when the operating system starts. This depends on what is specified during setup; some services are not started by default. This section describes the management of the various SQL Server services. Before you log in to an instance of SQL Server, you need to know how to start, stop, pause, resume, and restart an instance of SQL Server. After you are logged in, you can perform tasks such as administering the server or querying a database.

Using the SQL Server Service

When you start an instance of SQL Server Database Engine, you are starting the SQL Server service. After you start the SQL Server service, users can establish new connections to the server. The SQL Server service can be started and stopped as a service, either locally or remotely. The SQL Server service is referred to as SQL Server (MSSQLSERVER) if it is the default instance, or MSSQL$<*instancename*>if it is a named instance.

Using SQL Server Configuration Manager

SQL Server Configuration Manager allows you to stop, start, or pause various SQL Server services.

NOTE

SQL Server Configuration Manager cannot manage SQL Server 2000 services.

You can also use SQL Server Configuration Manager to view the properties of the selected service. SQL Server Configuration Manager is a Microsoft Management Console (MMC) snap-in. For more information about MMC and how a snap-in works, see Windows Help.

To access SQL Server Configuration Manager

- On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

  To access SQL Server Configuration Manager Using Windows 8

  Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager not does not appear as an application when running Windows 8.0. To open SQL Server Configuration Manager, in the **Search** charm, under **Apps**, type **SQLServerManager12.msc** (for SQL Server 2014), **SQLServerManager11.msc** (for SQL Server 2012), or **SQLServerManager10.msc** for (SQL Server 2008), and then press **Enter**.

In this Section

Security Requirements for Managing Services	Prevent Automatic Startup of an Instance of SQL Server (SQL Server Configuration Manager)

Configure Windows Service Accounts and Permissions	Change the Service Startup Account for SQL Server (SQL Server Configuration Manager)
Run SQL Server With or Without a Network	Configure Server Startup Options (SQL Server Configuration Manager)
SQL Server Browser Service (Database Engine and SSAS)	Change the Password of the Accounts Used by SQL Server (SQL Server Configuration Manager)
Database Engine Service Startup Options	Configure SQL Server Error Logs
Start, Stop, Pause, Resume, Restart the Database Engine, SQL Server Agent, or SQL Server Browser Service	Change Server Authentication Mode
Start SQL Server in Single-User Mode	SQL Writer Service
Start SQL Server with Minimal Configuration	Broadcast a Shutdown Message (Command Prompt)
Connect to Another Computer (SQL Server Configuration Manager)	Log In to an Instance of SQL Server (Command Prompt)
Set an Instance of SQL Server to Start Automatically (SQL Server Configuration Manager)	Configure File System Permissions for Database Engine Access

Related Content

Configure SQL Server Agent

Logging In to SQL Server

Security Requirements for Managing Services

3/24/2017 • 1 min to read • Edit Online

To manage the Microsoft SQL Server and SQL Server Agent Services, use either SQL Server Configuration Manager or SQL Server Management Studio. Manage the services on clustered servers with the Cluster Administrator.

To manage the SQL Server service and set the server configuration options, you must be a member of the **serveradmin** fixed server role or the **sysadmin** fixed server role. Members of the Windows **Administrators** group can start and stop services and configure the server options that Windows provides.

NOTE

To operate properly, the accounts used for the services must be configured with the correct domain, file system, and registry permissions. For information about the required permissions, see Configure Windows Service Accounts and Permissions.

Windows Management Instrumentation

SQL Server Configuration Manager and SQL Server Management Studio use Windows Management Instrumentation (WMI) to display and modify some of the server properties. To manage services and obtain the status of the services, the user must have rights to access the WMI object. In SQL Server Management Studio, the following server property pages use WMI:

- Autostart Services
- Startup Parameters
- Security
- Misc Server Settings

Related Tasks

Configure WMI to Show Server Status in SQL Server Tools

Related Content

WMI Provider for Configuration Management Concepts

Configure Windows Service Accounts and Permissions

3/24/2017 • 28 min to read • Edit Online

Each service in SQL Server represents a process or a set of processes to manage authentication of SQL Server operations with Windows. This topic describes the default configuration of services in this release of SQL Server, and configuration options for SQL Server services that you can set during and after SQL Server installation. This topic helps advanced users understand the details of the service accounts.

Most services and their properties can be configured by using SQL Server Configuration Manager. Here are the paths to the last four versions when Windows in installed on the C drive.

SQL Server 2016	C:\Windows\SysWOW64\SQLServerManager13.msc
SQL Server 2014	C:\Windows\SysWOW64\SQLServerManager12.msc
SQL Server 2012	C:\Windows\SysWOW64\SQLServerManager11.msc
SQL Server 2008	C:\Windows\SysWOW64\SQLServerManager10.msc

Services Installed by SQL Server

Depending on the components that you decide to install, SQL Server Setup installs the following services:

- **SQL Server Database Services** - The service for the SQL Server relational Database Engine. The executable file is <MSSQLPATH>\MSSQL\Binn\sqlservr.exe.
- **SQL Server Agent** - Executes jobs, monitors SQL Server, fires alerts, and enables automation of some administrative tasks. The SQL Server Agent service is present but disabled on instances of SQL Server Express. The executable file is <MSSQLPATH>\MSSQL\Binn\sqlagent.exe.
- **Analysis Services** - Provides online analytical processing (OLAP) and data mining functionality for business intelligence applications. The executable file is <MSSQLPATH>\OLAP\Bin\msmdsrv.exe.
- **Reporting Services** - Manages, executes, creates, schedules, and delivers reports. The executable file is <MSSQLPATH>\Reporting Services\ReportServer\Bin\ReportingServicesService.exe.
- **Integration Services** - Provides management support for Integration Services package storage and execution. The executable path is <MSSQLPATH>\130\DTS\Binn\MsDtsSrvr.exe
- **SQL Server Browser** - The name resolution service that provides SQL Server connection information for client computers. The executable path is c:\Program Files (x86)\Microsoft SQL Server\90\Shared\sqlbrowser.exe
- **Full-text search** - Quickly creates full-text indexes on content and properties of structured and semistructured data to provide document filtering and word-breaking for SQL Server.
- **SQL Writer** - Allows backup and restore applications to operate in the Volume Shadow Copy Service (VSS) framework.
- **SQL Server Distributed Replay Controller** - Provides trace replay orchestration across multiple

Distributed Replay client computers.

- **SQL Server Distributed Replay Client** - One or more Distributed Replay client computers that work together with a Distributed Replay controller to simulate concurrent workloads against an instance of the SQL Server Database Engine.
- **SQL Server Trusted Launchpad** - A trusted service that hosts external executables that are provided by Microsoft, such as the R runtime installed as part of R Services (In-Database). Satellite processes can be launched by the Launchpad process but will be resource governed based on the configuration of the individual instance. The Launchpad service runs under its own user account, and each satellite process for a specific, registered runtime will inherit the user account of the Launchpad. Satellite processes are created and destroyed on demand during execution time.
    - **SQL Server PolyBase Engine** - Provides distributed query capabilities to external data sources.
    - **SQL Server Polybase Data Movement Service** - Enables data movement between SQL Server and External Data Sources and between SQL nodes in PolyBase Scaleout Groups.

Service Properties and Configuration

Startup accounts used to start and run SQL Server can be domain user accounts, local user accounts, managed service accounts, virtual accounts, or built-in system accounts. To start and run, each service in SQL Server must have a startup account configured during installation.

This section describes the accounts that can be configured to start SQL Server services, the default values used by SQL Server Setup, the concept of per-service SID's, the startup options, and configuring the firewall.

- Default Service Accounts
- Automatic Startup
- Configuring Service StartupType
- Firewall Port

Default Service Accounts

The following table lists the default service accounts used by setup when installing all components. The default accounts listed are the recommended accounts, except as noted.

Stand-alone Server or Domain Controller

COMPONENT	WINDOWS SERVER 2008	WINDOWS 7 AND WINDOWS SERVER 2008 R2 AND HIGHER
Database Engine	NETWORK SERVICE	Virtual Account*
SQL Server Agent	NETWORK SERVICE	Virtual Account*
SSAS	NETWORK SERVICE	Virtual Account*
SSIS	NETWORK SERVICE	Virtual Account*
SSRS	NETWORK SERVICE	Virtual Account*
SQL Server Distributed Replay Controller	NETWORK SERVICE	Virtual Account*

COMPONENT	WINDOWS SERVER 2008	WINDOWS 7 AND WINDOWS SERVER 2008 R2 AND HIGHER
SQL Server Distributed Replay Client	NETWORK SERVICE	Virtual Account*
FD Launcher (Full-text Search)	LOCAL SERVICE	Virtual Account
SQL Server Browser	LOCAL SERVICE	LOCAL SERVICE
SQL Server VSS Writer	LOCAL SYSTEM	LOCAL SYSTEM
Advanced Analytics Extensions	NTSERVICE\MSSQLLaunchpad	NTSERVICE\MSSQLLaunchpad
PolyBase Engine	NETWORK SERVICE	NETWORK SERVICE
PolyBase Data Movement Service	NETWORK SERVICE	NETWORK SERVICE

*When resources external to the SQL Server computer are needed, Microsoft recommends using a Managed Service Account (MSA), configured with the minimum privileges necessary.

SQL Server Failover Cluster Instance

COMPONENT	WINDOWS SERVER 2008	WINDOWS SERVER 2008 R2
Database Engine	None. Provide a domain user account.	Provide a domain user account.
SQL Server Agent	None. Provide a domain user account.	Provide a domain user account.
SSAS	None. Provide a domain user account.	Provide a domain user account.
SSIS	NETWORK SERVICE	Virtual Account
SSRS	NETWORK SERVICE	Virtual Account
FD Launcher (Full-text Search)	LOCAL SERVICE	Virtual Account
SQL Server Browser	LOCAL SERVICE	LOCAL SERVICE
SQL Server VSS Writer	LOCAL SYSTEM	LOCAL SYSTEM

Changing Account Properties

IMPORTANT

- Always use SQL Server tools such as SQL Server Configuration Manager to change the account used by the SQL Server Database Engine or SQL Server Agent services, or to change the password for the account. In addition to changing the account name, SQL Server Configuration Manager performs additional configuration such as updating the Windows local security store which protects the service master key for the Database Engine. Other tools such as the Windows Services Control Manager can change the account name but do not change all the required settings.
  - For Analysis Services instances that you deploy in a SharePoint farm, always use SharePoint Central Administration to change the server accounts for Power Pivot service applications and the Analysis Services service. Associated settings and permissions are updated to use the new account information when you use Central Administration.
  - To change Reporting Services options, use the Reporting Services Configuration Tool.

Managed Service Accounts, Group Managed Service Accounts, and Virtual Accounts

Managed service accounts, group managed service accounts, and virtual accounts are designed to provide crucial applications such as SQL Server with the isolation of their own accounts, while eliminating the need for an administrator to manually administer the Service Principal Name (SPN) and credentials for these accounts. These make long term management of service account users, passwords and SPNs much easier.

- **Managed Service Accounts**

  A Managed Service Account (MSA) is a type of domain account created and managed by the domain controller. It is assigned to a single member computer for use running a service. The password is managed automatically by the domain controller. You cannot use a MSA to log into a computer, but a computer can use a MSA to start a Windows service. An MSA has the ability to register Service Principal Name (SPN) with the Active Directory. A MSA is named with a **$** suffix, for example **DOMAIN\ACCOUNTNAME$**. When specifying a MSA, leave the password blank. Because a MSA is assigned to a single computer, it cannot be used on different nodes of a Windows cluster.

  > **NOTE**
  > The MSA must be created in the Active Directory by the domain administrator before SQL Server setup can use it for SQL Server services.

- **Group Managed Service Accounts**

  A Group Managed Service Account is an MSA for multiple servers. Windows manages a service account for services running on a group of servers. Active Directory automatically updates the group managed service account password without restarting services. You can configure SQL Server services to use a group managed service account principal. Beginning with SQL Server 2014, SQL Server supports group managed service accounts on Windows Server 2012 R2 and later for standalone instances, failover cluster instances, and availability groups.

  To use a group managed service account for SQL Server 2014 or later, the operating system must be Windows Server 2012 R2 or later. Servers with Windows Server 2012 R2 require KB 2998082 applied so that the services can log in without disruption immediately after a password change.

  For more information, see Group Manged Service Accounts

  > **NOTE**
  > The group managed service account must be created in the Active Directory by the domain administrator before SQL Server setup can use it for SQL Server services.

- **Virtual Accounts**

  Virtual accounts (beginning with Windows Server 2008 R2 and Windows 7) are *managed local accounts* that provide the following features to simplify service administration. The virtual account is auto-managed, and the virtual account can access the network in a domain environment. If the default value is used for the service accounts during SQL Server setup, a virtual account using the instance name as the service name is used, in the format **NT SERVICE\**<*SERVICENAME*>. Services that run as virtual accounts access network resources by using the credentials of the computer account in the format **\\$**. When specifying a virtual account to start SQL Server, leave the password blank. If the virtual account fails to register the Service Principal Name (SPN), register the SPN manually. For more information on registering a SPN manually, see Manual SPN Registration.

NOTE

Virtual accounts cannot be used for SQL Server Failover Cluster Instance, because the virtual account would not have the same SID on each node of the cluster.

The following table lists examples of virtual account names.

SERVICE	VIRTUAL ACCOUNT NAME
Default instance of the Database Engine service	**NT SERVICE\MSSQLSERVER**
Named instance of a Database Engine service named **PAYROLL**	**NT SERVICE\MSSQL$PAYROLL**
SQL Server Agent service on the default instance of SQL Server	**NT SERVICE\SQLSERVERAGENT**
SQL Server Agent service on an instance of SQL Server named **PAYROLL**	**NT SERVICE\SQLAGENT$PAYROLL**

For more information on Managed Service Accounts and Virtual Accounts, see the **Managed service account and virtual account concepts** section of Service Accounts Step-by-Step Guide and Managed Service Accounts Frequently Asked Questions (FAQ).

Security Note: Always run SQL Server services by using the lowest possible user rights. Use a MSA or virtual account when possible. When MSA and virtual accounts are not possible, use a specific low-privilege user account or domain account instead of a shared account for SQL Server services. Use separate accounts for different SQL Server services. Do not grant additional permissions to the SQL Server service account or the service groups. Permissions will be granted through group membership or granted directly to a service SID, where a service SID is supported.

Automatic Startup

In addition to having user accounts, every service has three possible startup states that users can control:

- **Disabled** The service is installed but not currently running.
- **Manual** The service is installed, but will start only when another service or application needs its functionality.
- **Automatic** The service is automatically started by the operating system.

  The startup state is selected during setup. When installing a named instance, the SQL Server Browser service should be set to start automatically.

Configuring Services During Unattended Installation

The following table shows the SQL Server services that can be configured during installation. For unattended installations, you can use the switches in a configuration file or at a command prompt.

SQL SERVER SERVICE NAME	SWITCHES FOR UNATTENDED INSTALLATIONS*
MSSQLSERVER	SQLSVCACCOUNT, SQLSVCPASSWORD, SQLSVCSTARTUPTYPE
SQLServerAgent**	AGTSVCACCOUNT, AGTSVCPASSWORD, AGTSVCSTARTUPTYPE

SQL SERVER SERVICE NAME	SWITCHES FOR UNATTENDED INSTALLATIONS*
MSSQLServerOLAPService	ASSVCACCOUNT, ASSVCPASSWORD, ASSVCSTARTUPTYPE
ReportServer	RSSVCACCOUNT, RSSVCPASSWORD, RSSVCSTARTUPTYPE
Integration Services	ISSVCACCOUNT, ISSVCPASSWORD, ISSVCSTARTUPTYPE
SQL Server Distributed Replay Controller	DRU_CTLR, CTLRSVCACCOUNT,CTLRSVCPASSWORD, CTLRSTARTUPTYPE, CTLRUSERS
SQL Server Distributed Replay Client	DRU_CLT, CLTSVCACCOUNT, CLTSVCPASSWORD, CLTSTARTUPTYPE, CLTCTLRNAME, CLTWORKINGDIR, CLTRESULTDIR
R Services (In-Database)	EXTSVCACCOUNT, EXTSVCPASSWORD, ADVANCEDANALYTICS
PolyBase Engine	PBENGSVCACCOUNT, PBENGSVCPASSWORD, PBENGSVCSTARTUPTYPE, PBDMSSVCACCOUNT,PBDMSSVCPASSWORD, PBDMSSVCSTARTUPTYPE, PBSCALEOUT, PBPORTRANGE

*For more information and sample syntax for unattended installations, see Install SQL Server 2016 from the Command Prompt.

**The SQL Server Agent service is disabled on instances of SQL Server Express and SQL Server Express with Advanced Services.

Firewall Port

In most cases, when initially installed, the Database Engine can be connected to by tools such as SQL Server Management Studio installed on the same computer as SQL Server. SQL Server Setup does not open ports in the Windows firewall. Connections from other computers may not be possible until the Database Engine is configured to listen on a TCP port, and the appropriate port is opened for connections in the Windows firewall. For more information, see Configure the Windows Firewall to Allow SQL Server Access.

Service Permissions

This section describes the permissions that SQL Server Setup configures for the per-service SID's of the SQL Server services.

- Service Configuration and Access Control
- Windows Privileges and Rights
- File System Permissions Granted to SQL Server Per-service SIDs or SQL Server Local Windows Groups
- File System Permissions Granted to Other Windows User Accounts or Groups
- File System Permissions Related to Unusual Disk Locations
- Reviewing Additional Considerations
- Registry Permissions
- WMI
- Named Pipes

Service Configuration and Access Control

SQL Server 2016 enables per-service SID for each of its services to provide service isolation and defense in depth. The per-service SID is derived from the service name and is unique to that service. For example, a service SID name for the Database Engine service might be **NT Service\MSSQL$**<*InstanceName*>. Service isolation enables access to specific objects without the need to run a high-privilege account or weaken the security protection of the object. By using an access control entry that contains a service SID, a SQL Server service can restrict access to its resources.

NOTE

On Windows 7 and Windows Server 2008 R2 (and later) the per-service SID can be the virtual account used by the service.

For most components SQL Server configures the ACL for the per-service account directly, so changing the service account can be done without having to repeat the resource ACL process.

When installing SSAS, a per-service SID for the Analysis Services service is created. A local Windows group is created, named in the format **SQLServerMSASUser$***computer_name***$***instance_name*. The per-service SID **NT SERVICE\MSSQLServerOLAPService** is granted membership in the local Windows group, and the local Windows group is granted the appropriate permissions in the ACL. If the account used to start the Analysis Services service is changed, SQL Server Configuration Manager must change some Windows permissions (such as the right to log on as a service), but the permissions assigned to the local Windows group will still be available without any updating, because the per-service SID has not changed. This method allows the Analysis Services service to be renamed during upgrades.

During SQL Server installation, SQL Server Setup creates a local Windows groups for SSAS and the SQL Server Browser service. For these services, SQL Server configures the ACL for the local Windows groups.

Depending on the service configuration, the service account for a service or service SID is added as a member of the service group during install or upgrade.

Windows Privileges and Rights

The account assigned to start a service needs the **Start, stop and pause permission** for the service. The SQL Server Setup program automatically assigns this. First install Remote Server Administration Tools (RSAT). See Remote Server Administration Tools for Windows 7.

The following table shows permissions that SQL Server Setup requests for the per-service SIDs or local Windows groups used by SQL Server components.

SQL SERVER SERVICE	PERMISSIONS GRANTED BY SQL SERVER SETUP
SQL Server Database Engine: (All rights are granted to the per-service SID. Default instance: **NT SERVICE\MSSQLSERVER**. Named instance: **NT SERVICE\MSSQL$**InstanceName.)	**Log on as a service** (SeServiceLogonRight) **Replace a process-level token** (SeAssignPrimaryTokenPrivilege) **Bypass traverse checking** (SeChangeNotifyPrivilege) **Adjust memory quotas for a process** (SeIncreaseQuotaPrivilege) Permission to start SQL Writer Permission to read the Event Log service Permission to read the Remote Procedure Call service

SQL SERVER SERVICE	PERMISSIONS GRANTED BY SQL SERVER SETUP
SQL Server Agent: * (All rights are granted to the per-service SID. Default instance: **NT Service\SQLSERVERAGENT**. Named instance: **NT Service\SQLAGENT$***InstanceName*.)	**Log on as a service** (SeServiceLogonRight) **Replace a process-level token** (SeAssignPrimaryTokenPrivilege) **Bypass traverse checking** (SeChangeNotifyPrivilege) **Adjust memory quotas for a process** (SeIncreaseQuotaPrivilege)
SSAS: (All rights are granted to a local Windows group. Default instance: **SQLServerMSASUser$***ComputerName***$MSSQLSERVER**. Named instance: **SQLServerMSASUser$***ComputerName***$***InstanceName*. Power Pivot for SharePoint instance: **SQLServerMSASUser$***ComputerName***$***PowerPivot*.)	**Log on as a service** (SeServiceLogonRight) For tabular only: **Increase a process working set** (SeIncreaseWorkingSetPrivilege) **Adjust memory quotas for a process** (SeIncreaseQuotaSizePrivilege) **Lock pages in memory** (SeLockMemoryPrivilege) – this is needed only when paging is turned off entirely. For failover cluster installations only: **Increase scheduling priority** (SeIncreaseBasePriorityPrivilege)
SSRS: (All rights are granted to the per-service SID. Default instance: **NT SERVICE\ReportServer**. Named instance: **NT SERVICE\$***InstanceName*.)	**Log on as a service** (SeServiceLogonRight)
SSIS: (All rights are granted to the per-service SID. Default instance and named instance: **NT SERVICE\MsDtsServer130**. Integration Services does not have a separate process for a named instance.)	**Log on as a service** (SeServiceLogonRight) Permission to write to application event log. **Bypass traverse checking** (SeChangeNotifyPrivilege) **Impersonate a client after authentication** (SeImpersonatePrivilege)
Full-text search: (All rights are granted to the per-service SID. Default instance: **NT Service\MSSQLFDLauncher**. Named instance: **NT Service\ MSSQLFDLauncher$***InstanceName*.)	**Log on as a service** (SeServiceLogonRight) **Adjust memory quotas for a process** (SeIncreaseQuotaPrivilege) **Bypass traverse checking** (SeChangeNotifyPrivilege)
SQL Server Browser: (All rights are granted to a local Windows group. Default or named instance: **SQLServer2005SQLBrowserUser***$ComputerName*. SQL Server Browser does not have a separate process for a named instance.)	**Log on as a service** (SeServiceLogonRight)

SQL SERVER SERVICE	PERMISSIONS GRANTED BY SQL SERVER SETUP
SQL Server VSS Writer: (All rights are granted to the per-service SID. Default or named instance: **NT Service\SQLWriter**. SQL Server VSS Writer does not have a separate process for a named instance.)	The SQLWriter service runs under the LOCAL SYSTEM account which has all the required permissions. SQL Server setup does not check or grant permissions for this service.
SQL Server Distributed Replay Controller:	**Log on as a service** (SeServiceLogonRight)
SQL Server Distributed Replay Client:	**Log on as a service** (SeServiceLogonRight)
PolyBase Engine and DMS	**Log on as a service** (SeServiceLogonRight)
Launchpad:	**Log on as a service** (SeServiceLogonRight) **Replace a process-level token** (SeAssignPrimaryTokenPrivilege) **Bypass traverse checking** (SeChangeNotifyPrivilege) **Adjust memory quotas for a process** (SeIncreaseQuotaPrivilege)
R Services: SQLRUserGroup	**Allow Log on locally**

*The SQL Server Agent service is disabled on instances of SQL Server Express.

File System Permissions Granted to SQL Server Per-service SIDs or Local Windows Groups

SQL Server service accounts must have access to resources. Access control lists are set for the per-service SID or the local Windows group.

IMPORTANT

For failover cluster installations, resources on shared disks must be set to an ACL for a local account.

The following table shows the ACLs that are set by SQL Server Setup:

SERVICE ACCOUNT FOR	FILES AND FOLDERS	ACCESS
MSSQLServer	Instid\MSSQL\backup	Full control
	Instid\MSSQL\binn	Read, Execute
	Instid\MSSQL\data	Full control
	Instid\MSSQL\FTData	Full control
	Instid\MSSQL\Install	Read, Execute
	Instid\MSSQL\Log	Full control
	Instid\MSSQL\Repldata	Full control

SERVICE ACCOUNT FOR	FILES AND FOLDERS	ACCESS
	130\shared	Read, Execute
	Instid\MSSQL\Template Data (SQL Server Express only)	Read
SQLServerAgent*	Instid\MSSQL\binn	Full control
	Instid\MSSQL\binn	Full control
	Instid\MSSQL\Log	Read, Write, Delete, Execute
	130\com	Read, Execute
	130\shared	Read, Execute
	130\shared\Errordumps	Read, Write
	ServerName\EventLog	Full control
FTS	Instid\MSSQL\FTData	Full control
	Instid\MSSQL\FTRef	Read, Execute
	130\shared	Read, Execute
	130\shared\Errordumps	Read, Write
	Instid\MSSQL\Install	Read, Execute
	Instid\MSSQL\jobs	Read, Write
MSSQLServerOLAPservice	130\shared\ASConfig	Full control
	Instid\OLAP	Read, Execute
	Instid\Olap\Data	Full control
	Instid\Olap\Log	Read, Write
	Instid\OLAP\Backup	Read, Write
	Instid\OLAP\Temp	Read, Write
	130\shared\Errordumps	Read, Write
SQLServerReportServerUser	Instid\Reporting Services\Log Files	Read, Write, Delete
	Instid\Reporting Services\ReportServer	Read, Execute

SERVICE ACCOUNT FOR	FILES AND FOLDERS	ACCESS
	Instid\Reportingservices\Reportserver\global.asax	Full control
	Instid\Reportingservices\Reportserver\Reportserver.config	Read
	Instid\Reporting Services\reportManager	Read, Execute
	Instid\Reporting Services\RSTempfiles	Read, Write, Execute, Delete
	130\shared	Read, Execute
	130\shared\Errordumps	Read, Write
MSDTSServer100	130\dts\binn\MsDtsSrvr.ini.xml	Read
	130\dts\binn	Read, Execute
	130\shared	Read, Execute
	130\shared\Errordumps	Read, Write
SQL Server Browser	130\shared\ASConfig	Read
	130\shared	Read, Execute
	130\shared\Errordumps	Read, Write
SQLWriter	N/A (Runs as local system)	
User	Instid\MSSQL\binn	Read, Execute
	Instid\Reporting Services\ReportServer	Read, Execute, List Folder Contents
	Instid\Reportingservices\Reportserver\global.asax	Read
	Instid\Reporting Services\ReportManager	Read, Execute
	Instid\Reporting Services\ReportManager\pages	Read
	Instid\Reporting Services\ReportManager\Styles	Read
	130\dts	Read, Execute
	130\tools	Read, Execute

SERVICE ACCOUNT FOR	FILES AND FOLDERS	ACCESS
	100\tools	Read, Execute
	90\tools	Read, Execute
	80\tools	Read, Execute
	130\sdk	Read
	Microsoft SQL Server\130\Setup Bootstrap	Read, Execute
SQL Server Distributed Replay Controller	<ToolsDir>\DReplayController\Log\ (empty directory)	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayController\DReplayController.exe	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayController\resources|Read, Execute, List Folder Contents	
	<ToolsDir>\DReplayController\{all dlls}	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayController\DReplayController.config	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayController\IRTemplate.tdf	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayController\IRDefinition.xml	Read, Execute, List Folder Contents
SQL Server Distributed Replay Client	<ToolsDir>\DReplayClient\Log|Read, Execute, List Folder Contents	
	<ToolsDir>\DReplayClient\DReplayClient.exe	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayClient\resources|Read, Execute, List Folder Contents	
	<ToolsDir>\DReplayClient\ (all dlls)	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayClient\DReplayClient.config	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayClient\IRTemplate.tdf	Read, Execute, List Folder Contents
	<ToolsDir>\DReplayClient\IRDefinition.xml	Read, Execute, List Folder Contents
Launchpad	%binn	Read, Execute

SERVICE ACCOUNT FOR	FILES AND FOLDERS	ACCESS
	ExtensiblilityData	Full control
	Log\ExtensibiltityLog	Full control

*The SQL Server Agent service is disabled on instances of SQL Server Express and SQL Server Express with Advanced Services.

When database files are stored in a user-defined location, you must grant the per-service SID access to that location. For more information about granting file system permissions to a per-service SID, see Configure File System Permissions for Database Engine Access.

File System Permissions Granted to Other Windows User Accounts or Groups

Some access control permissions might have to be granted to built-in accounts or other SQL Server service accounts. The following table lists additional ACLs that are set by SQL Server Setup.

REQUESTING COMPONENT	ACCOUNT	RESOURCE	PERMISSIONS
MSSQLServer	Performance Log Users	Instid\MSSQL\binn	List folder contents
	Performance Monitor Users	Instid\MSSQL\binn	List folder contents
	Performance Log Users, Performance Monitor Users	\WINNT\system32\sqlctr130.dll	Read, Execute
	Administrator only	\\.\root\Microsoft\SqlServer\ServerEvents\*	Full control
	Administrators, System	\tools\binn\schemas\sqlserver\2004\07\showplan	Full control
	Users	\tools\binn\schemas\sqlserver\2004\07\showplan	Read, Execute
Reporting Services	<Report Server Web Service Account>	<*install*>\Reporting Services\LogFiles	DELETE READ_CONTROL SYNCHRONIZE FILE_GENERIC_READ FILE_GENERIC_WRITE FILE_READ_DATA FILE_WRITE_DATA FILE_APPEND_DATA FILE_READ_EA FILE_WRITE_EA FILE_READ_ATTRIBUTES FILE_WRITE_ATTRIBUTES

REQUESTING COMPONENT	ACCOUNT	RESOURCE	PERMISSIONS
	Report Manager Application pool identity, ASP.NET account, Everyone	<*install*>\Reporting Services\ReportManager, <*install*>\Reporting Services\ReportManager\Pages\*.*, <*install*>\Reporting Services\ReportManager\Styles\*.*, <*install*>\Reporting Services\ReportManager\webctrl_client\1_0\*.*	Read
	Report Manager Application pool identity	<*install*>\Reporting Services\ReportManager\Pages\*.*	Read
	<Report Server Web Service Account>	<*install*>\Reporting Services\ReportServer	Read
	<Report Server Web Service Account>	<*install*>\Reporting Services\ReportServer\global.asax	Full
	Everyone	<*install*>\Reporting Services\ReportServer\global.asax	READ_CONTROL FILE_READ_DATA FILE_READ_EA FILE_READ_ATTRIBUTES
	Network service	<*install*>\Reporting Services\ReportServer\ReportService.asmx	Full

REQUESTING COMPONENT	ACCOUNT	RESOURCE	PERMISSIONS
	Everyone	<*install*>\Reporting Services\ReportServer\ReportService.asmx	READ_CONTROL SYNCHRONIZE FILE_GENERIC_READ FILE_GENERIC_EXECUTE FILE_READ_DATA FILE_READ_EA FILE_EXECUTE FILE_READ_ATTRIBUTES
	ReportServer Windows Services Account	<*install*>\Reporting Services\ReportServer\RSReportServer.config	DELETE READ_CONTROL SYNCHRONIZE FILE_GENERIC_READ FILE_GENERIC_WRITE FILE_READ_DATA FILE_WRITE_DATA FILE_APPEND_DATA FILE_READ_EA FILE_WRITE_EA FILE_READ_ATTRIBUTES FILE_WRITE_ATTRIBUTES
	Everyone	Report Server keys (Instid hive)	Query Value Enumerate SubKeys Notify Read Control
	Terminal Services User	Report Server keys (Instid hive)	Query Value Set Value Create SubKey Enumerate SubKey Notify Delete Read Control

REQUESTING COMPONENT	ACCOUNT	RESOURCE	PERMISSIONS
	Power Users	Report Server keys (Instid hive)	Query Value Set Value Create Subkey Enumerate Subkeys Notify Delete Read Control

*This is the WMI provider namespace.

File System Permissions Related to Unusual Disk Locations

The default drive for locations for installation is **systemdrive**, normally drive C. When tempdb or user databases are installed

Non-default Drive

When installed to a local drive that is not the default drive, the per-service SID must have access to the file location. SQL Server Setup will provision the required access.

Network Share

When databases are installed to a network share, the service account must have access to the file location of the user and tempdb databases. SQL Server Setup cannot provision access to a network share. The user must provision access to a tempdb location for the service account before running setup. The user must provision access to the user database location before creating the database.

NOTE

Virtual accounts cannot be authenticated to a remote location. All virtual accounts use the permission of machine account. Provision the machine account in the format **\$**.

Reviewing Additional Considerations

The following table shows the permissions that are required for SQL Server services to provide additional functionality.

SERVICE/APPLICATION	FUNCTIONALITY	REQUIRED PERMISSION
SQL Server (MSSQLSERVER)	Write to a mail slot using xp_sendmail.	Network write permissions.
SQL Server (MSSQLSERVER)	Run xp_cmdshell for a user other than a SQL Server administrator.	Act as part of operating system and replace a process-level token.
SQL Server Agent (MSSQLSERVER)	Use the autorestart feature.	Must be a member of the Administrators local group.

SERVICE/APPLICATION	FUNCTIONALITY	REQUIRED PERMISSION
Database Engine Tuning Advisor	Tunes databases for optimal query performance.	On first use, a user who has system administrative credentials must initialize the application. After initialization, dbo users can use the Database Engine Tuning Advisor to tune only those tables that they own. For more information, see "Initializing Database Engine Tuning Advisor on First Use" in SQL Server Books Online.

IMPORTANT
Before you upgrade SQL Server, enable Windows Authentication for SQL Server Agent and verify the required default configuration: that the SQL Server Agent service account is a member of the SQL Serversysadmin group.

Registry Permissions

The registry hive is created under **HKLM\Software\Microsoft\Microsoft SQL Server** for instance-aware components. For example

- **HKLM\Software\Microsoft\Microsoft SQL Server\MSSQL13.MyInstance**
- **HKLM\Software\Microsoft\Microsoft SQL Server\MSASSQL13.MyInstance**
- **HKLM\Software\Microsoft\Microsoft SQL Server\MSSQL.130**

  The registry also maintains a mapping of instance ID to instance name. Instance ID to instance name mapping is maintained as follows:

- **[HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\Instance Names\SQL] "InstanceName"="MSSQL13"**
- **[HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\Instance Names\OLAP] "InstanceName"="MSASSQL13"**
- **[HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\Instance Names\RS] "InstanceName"="MSRSSQL13"**

WMI

Windows Management Instrumentation (WMI) must be able to connect to the Database Engine. To support this, the per-service SID of the Windows WMI provider (**NT SERVICE\winmgmt**) is provisioned in the Database Engine.

The SQL WMI provider requires the following permissions:

- Membership in the **db_ddladmin** or **db_owner** fixed database roles in the msdb database.
- **CREATE DDL EVENT NOTIFICATION** permission in the server.
- **CREATE TRACE EVENT NOTIFICATION** permission in the Database Engine.
- **VIEW ANY DATABASE** server-level permission.

  SQL Server setup creates a SQL WMI namespace and grants read permission to the SQL Server Agent service-SID.

Named Pipes

In all installation, SQL Server Setup provides access to the SQL Server Database Engine through the shared memory protocol, which is a local named pipe.

Provisioning

This section describes how accounts are provisioned inside the various SQL Server components.

- Database Engine Provisioning
  - Windows Principals
  - sa Account
  - SQL Server Per-service SID Login and Privileges
  - SQL Server Agent Login and Privileges
  - HADRON and SQL Failover Cluster Instance and Privileges
  - SQL Writer and Privileges
  - SQL WMI and Privileges
- SSAS Provisioning
- SSRS Provisioning

Database Engine Provisioning

The following accounts are added as logins in the SQL Server Database Engine.

Windows Principals

During setup, SQL Server Setup requires at least one user account to be named as a member of the **sysadmin** fixed server role.

sa Account

The **sa** account is always present as a Database Engine login and is a member of the **sysadmin** fixed server role. When the Database Engine is installed using only Windows Authentication (that is when SQL Server Authentication is not enabled), the **sa** login is still present but is disabled. For information about enabling the **sa** account, see Change Server Authentication Mode.

SQL Server Per-service SID Login and Privileges

The per-service SID of the SQL Server service is provisioned as a Database Engine login. The per-service SID login is a member of the **sysadmin** fixed server role.

SQL Server Agent Login and Privileges

The per-service SID of the SQL Server Agent service is provisioned as a Database Engine login. The per-service SID login is a member of the **sysadmin** fixed server role.

Always On Availability Groups and SQL Failover Cluster Instance and Privileges

When installing the Database Engine as a Always On Availability Groups or SQL Failover Cluster Instance (SQL FCI), **LOCAL SYSTEM** is provisioned in the Database Engine. The **LOCAL SYSTEM** login is granted the **ALTER ANY AVAILABILITY GROUP** permission (for Always On Availability Groups) and the **VIEW SERVER STATE** permission (for SQL FCI).

SQL Writer and Privileges

The per-service SID of the SQL Server VSS Writer service is provisioned as a Database Engine login. The per-service SID login is a member of the **sysadmin** fixed server role.

SQL WMI and Privileges

SQL Server Setup provisions the **NT SERVICE\Winmgmt** account as a Database Engine login and adds it to the **sysadmin** fixed server role.

SSRS Provisioning

The account specified during setup is provisioned as a member of the **RSExecRole** database role. For more information, see Configure the Report Server Service Account (SSRS Configuration Manager).

SSAS Provisioning

SSAS service account requirements vary depending on how you deploy the server. If you are installing Power Pivot for SharePoint, SQL Server Setup requires that you configure the Analysis Services service to run under a domain account. Domain accounts are required to support the managed account facility that is built into SharePoint. For this reason, SQL Server Setup does not provide a default service account, such as a virtual account, for a Power Pivot for SharePoint installation. For more information about provisioning Power Pivot for SharePoint, see Configure Power Pivot Service Accounts.

For all other standalone SSAS installations, you can provision the service to run under a domain account, built-in system account, managed account, or virtual account. For more information about account provisioning, see Configure Service Accounts (Analysis Services).

For clustered installations, you must specify a domain account or a built-in system account. Neither managed accounts nor virtual accounts are supported for SSAS failover clusters.

All SSAS installations require that you specify a system administrator of the Analysis Services instance. Administrator privileges are provisioned in the Analysis Services **Server** role.

SSRS Provisioning

The account specified during setup is provisioned in the Database Engine as a member of the **RSExecRole** database role. For more information, see Configure the Report Server Service Account (SSRS Configuration Manager).

Upgrading From Previous Versions

This section describes the changes made during upgrade from a previous version of SQL Server.

- SQL Server 2016 requires Windows Server 2008 R2 SP1, Windows Server 2012, Windows 8.0, Windows Server 2012 R2, or Windows 8.1, . Any previous version of SQL Server running on a lower operating system version must have the operating system upgraded before upgrading SQL Server.

- During upgrade of SQL Server 2005 to SQL Server 2016, SQL Server Setup will configure SQL Server in the following way.

    - The Database Engine runs with the security context of the per-service SID. The per-service SID is granted access to the file folders of the SQL Server instance (such as DATA), and the SQL Server registry keys.

    - The per-service SID of the Database Engine is provisioned in the Database Engine as a member of the **sysadmin** fixed server role.

    - The per-service SID's are added to the local SQL Server Windows groups, unless SQL Server is a Failover Cluster Instance.

    - The SQL Server resources remain provisioned to the local SQL Server Windows groups.

    - The local Windows group for services is renamed from **SQLServer2005MSSQLUser$$** to **SQLServerMSSQLUser$$**. File locations for migrated databases will have Access Control Entries (ACE) for the local Windows groups. The file locations for new databases will have ACE's for the per-service SID.

- During upgrade from SQL Server 2008, SQL Server Setup will be preserve the ACE's for the SQL Server 2008 per-service SID.

- For a SQL Server Failover Cluster Instance, the ACE for the domain account configured for the service will be retained.

Appendix

This section contains additional information about SQL Server services.

- Description of Service Accounts
- Identifying Instance-Aware and Instance-Unaware Services
- Localized Service Names

Description of Service Accounts

The service account is the account used to start a Windows service, such as the SQL Server Database Engine.

Accounts Available With Any Operating System

In addition to the new MSA and virtual accounts described earlier, the following accounts can be used.

Domain User Account

If the service must interact with network services, access domain resources like file shares or if it uses linked server connections to other computers running SQL Server, you might use a minimally-privileged domain account. Many server-to-server activities can be performed only with a domain user account. This account should be pre-created by domain administration in your environment.

> **NOTE**
>
> If you configure the application to use a domain account, you can isolate the privileges for the application, but must manually manage passwords or create a custom solution for managing these passwords. Many server applications use this strategy to enhance security, but this strategy requires additional administration and complexity. In these deployments, service administrators spend a considerable amount of time on maintenance tasks such as managing service passwords and service principal names (SPNs), which are required for Kerberos authentication. In addition, these maintenance tasks can disrupt service.

Local User Accounts

If the computer is not part of a domain, a local user account without Windows administrator permissions is recommended.

Local Service Account

The Local Service account is a built-in account that has the same level of access to resources and objects as members of the Users group. This limited access helps safeguard the system if individual services or processes are compromised. Services that run as the Local Service account access network resources as a null session without credentials. Be aware that the Local Service account is not supported for the SQL Server or SQL Server Agent services. Local Service is not supported as the account running those services because it is a shared service and any other services running under local service would have system administrator access to SQL Server. The actual name of the account is **NT AUTHORITY\LOCAL SERVICE**.

Network Service Account

The Network Service account is a built-in account that has more access to resources and objects than members of the Users group. Services that run as the Network Service account access network resources by using the credentials of the computer account in the format **\$**. The actual name of the account is **NT AUTHORITY\NETWORK SERVICE**.

Local System Account

Local System is a very high-privileged built-in account. It has extensive privileges on the local system and acts as the computer on the network. The actual name of the account is **NT AUTHORITY\SYSTEM**.

Identifying Instance-Aware and Instance-Unaware Services

Instance-aware services are associated with a specific instance of SQL Server, and have their own registry hives. You can install multiple copies of instance-aware services by running SQL Server Setup for each component or service. Instance-unaware services are shared among all installed SQL Server instances. They are not associated with a specific instance, are installed only once, and cannot be installed side-by-side.

Instance-aware services in SQL Server include the following:

- SQL Server
- SQL Server Agent

  Be aware that the SQL Server Agent service is disabled on instances of SQL Server Express and SQL Server Express with Advanced Services.

- Analysis Services*
- Reporting Services
- Full-text search

  Instance-unaware services in SQL Server include the following:

- Integration Services
- SQL Server Browser
- SQL Writer

  *Analysis Services in SharePoint integrated mode runs as ' Power Pivot' as a single, named instance. The instance name is fixed. You cannot specify a different name. You can install only one instance of Analysis Services running as ' Power Pivot' on each physical server.

Localized Service Names

The following table shows service names that are displayed by localized versions of Windows.

LANGUAGE	NAME FOR LOCAL SERVICE	NAME FOR NETWORK SERVICE	NAME FOR LOCAL SYSTEM	NAME FOR ADMIN GROUP
English Simplified Chinese Traditional Chinese Korean Japanese	NT AUTHORITY\LOCAL SERVICE	NT AUTHORITY\NETWORK SERVICE	NT AUTHORITY\SYSTEM	BUILTIN\Administrators
German	NT-AUTORITÄT\LOKALER DIENST	NT-AUTORITÄT\NETZWERKDIENST	NT-AUTORITÄT\SYSTEM	VORDEFINIERT\Administratoren
French	AUTORITE NT\SERVICE LOCAL	AUTORITE NT\SERVICE RÉSEAU	AUTORITE NT\SYSTEM	BUILTIN\Administrators

LANGUAGE	NAME FOR LOCAL SERVICE	NAME FOR NETWORK SERVICE	NAME FOR LOCAL SYSTEM	NAME FOR ADMIN GROUP
Italian	NT AUTHORITY\SERVIZIO LOCALE	NT AUTHORITY\SERVIZIO DI RETE	NT AUTHORITY\SYSTEM	BUILTIN\Administrators
Spanish	NT AUTHORITY\SERVICIO LOC	NT AUTHORITY\SERVICIO DE RED	NT AUTHORITY\SYSTEM	BUILTIN\Administradores
Russian	NT AUTHORITY\LOCAL SERVICE	NT AUTHORITY\NETWORK SERVICE	NT AUTHORITY\SYSTEM	BUILTIN\Администраторы

Related Content

Security Considerations for a SQL Server Installation

File Locations for Default and Named Instances of SQL Server

Install Master Data Services

Configure File System Permissions for Database Engine Access

3/24/2017 • 1 min to read • Edit Online

This topic describes how to grant the SQL Server Database Engine, file system access to the location where database files are stored. The Database Engine service must have permission of the Windows file system to access the file folder where database files are stored. Permission to the default location is configured during setup. If you place your database files in a different location, you might need to follow these steps to grant the Database Engine the full control permission to that location.

Beginning with SQL Server 2012 permissions are assigned to the per-service SID for each of its services. This system helps provide service isolation and defense in depth. The per-service SID is derived from the service name and is unique to each service. The topic Configure Windows Service Accounts and Permissions describes the per-service SID and provides the names in the section **Windows Privileges and Rights**. It is the per-service SID that must be assigned the access permission on the file location.

To Grant File System Permission to the Per-service SID

1. Using Windows Explorer, navigate to the file system location where the database files are stored. Right-click the file system folder, and then click **Properties**.

2. On the **Security** tab, click **Edit**, and then **Add**.

3. In the **Select Users, Computer, Service Account, or Groups** dialog box, click **Locations**, at the top of the location list, select your computer name, and then click **OK**.

4. In the **Enter the object names to select** box, type the name of the per-service SID name listed in the Books Online topic **Configure Windows Service Accounts and Permissions**. (For the Database Engine per service SID name, use **NT SERVICE\MSSQLSERVER** for a default instance, or **NT SERVICE\MSSQL$InstanceName** for a named instance.)

5. Click **Check Names** to validate the entry. (If the validation fails, it might advise you that the name was not found. When you click **OK**, a **Multiple Names Found** dialog box appears. Now select the per-service SID name, either **MSSQLSERVER** or **NT SERVICE\MSSQL$InstanceName**, and then click **OK**. Click **OK** again to return to the **Permissions** dialog box.)

6. In the **Group or user** names box, select the per-service SID name, and then in the **Permissions for** <name> box, select the **Allow** check box for **Full control**.

7. Click **Apply**, and then click **OK** twice to exit.

See Also

Manage the Database Engine Services
Move System Databases
Move User Databases

Run SQL Server With or Without a Network

3/24/2017 • 1 min to read • Edit Online

Microsoft SQL Server can run on a network, or it can function without a network.

Running SQL Server on a Network

For SQL Server to communicate over a network, the SQL Server service must be running. By default, Microsoft Windows automatically starts the built-in SQL Server service. To find out whether the SQL Server service has been started, at the command prompt, type the following:

net start

If the services associated with SQL Server have been started, the following services will appear in the **net start** output:

- Analysis Services (MSSQLSERVER)
- SQL Server (MSSQLSERVER)
- SQL Server Agent (MSSQLSERVER)

Running SQL Server Without a Network

When running an instance of SQL Server without a network, you do not need to start the built-in SQL Server service. Because SQL Server Management Studio, SQL Server Configuration Manager, and the **net start** and **net stop** commands are functional even without a network, the procedures for starting and stopping an instance of SQL Server are identical for a network or stand-alone operation.

When connecting to an instance of a stand-alone SQL Server from a local client such as **sqlcmd**, you bypass the network and connect directly to the instance of SQL Server by using a local pipe. The difference between a local pipe and a network pipe is whether you are using a network. Both local and network pipes establish a connection with an instance of SQL Server by using the standard pipe (\\.\pipe\sql\query), unless otherwise directed.

When you connect to an instance of a local SQL Server without specifying a server name, you are using a local pipe. When you connect to an instance of a local SQL Server and specify a server name explicitly, you are using either a network pipe or another network interprocess communication (IPC) mechanism, such as Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) (assuming you have configured SQL Server to use multiple networks). Because a stand-alone SQL Server does not support network pipes, you must omit the unnecessary **/** argument when connecting to the instance of SQL Server from a client. For example, to connect to a stand-alone instance of SQL Server from **osql**, type:

osql /Usa /P *<saPassword>*

SQL Server Browser Service (Database Engine and SSAS)

3/24/2017 • 6 min to read • Edit Online

The SQL ServerBrowser program runs as a Windows service. SQL Server Browser listens for incoming requests for Microsoft SQL Server resources and provides information about SQL Server instances installed on the computer. SQL Server Browser contributes to the following actions:

- Browsing a list of available servers
- Connecting to the correct server instance
- Connecting to dedicated administrator connection (DAC) endpoints

  For each instance of the Database Engine and SSAS, the SQL Server Browser service (sqlbrowser) provides the instance name and the version number. SQL Server Browser is installed with SQL Server.

  SQL Server Browser can be configured during setup or by using SQL Server Configuration Manager. By default, the SQL Server Browser service starts automatically:

- When upgrading an installation.
- When installing on a cluster.
- When installing a named instance of the Database Engine including all instances of SQL Server Express.
- When installing a named instance of Analysis Services.

Background

Prior to SQL Server 2000, only one instance of SQL Server could be installed on a computer. SQL Server listened for incoming requests on port 1433, assigned to SQL Server by the official Internet Assigned Numbers Authority (IANA). Only one instance of SQL Server can use a port, so when SQL Server 2000 introduced support for multiple instances of SQL Server, SQL Server Resolution Protocol (SSRP) was developed to listen on UDP port 1434. This listener service responded to client requests with the names of the installed instances, and the ports or named pipes used by the instance. To resolve limitations of the SSRP system, SQL Server 2005 introduced the SQL Server Browser service as a replacement for SSRP.

How SQL Server Browser Works

When an instance of SQL Server starts, if the TCP/IP protocol is enabled for SQL Server, the server is assigned a TCP/IP port. If the named pipes protocol is enabled, SQL Server listens on a specific named pipe. This port, or "pipe," is used by that specific instance to exchange data with client applications. During installation, TCP port 1433 and pipe `\sql\query` are assigned to the default instance, but those can be changed later by the server administrator using SQL Server Configuration Manager. Because only one instance of SQL Server can use a port or pipe, different port numbers and pipe names are assigned for named instances, including SQL Server Express. By default, when enabled, both named instances and SQL Server Express are configured to use dynamic ports, that is, an available port is assigned when SQL Server starts. If you want, a specific port can be assigned to an instance of SQL Server. When connecting, clients can specify a specific port; but if the port is dynamically assigned, the port number can change anytime SQL Server is restarted, so the correct port number is unknown to the client.

Upon startup, SQL Server Browser starts and claims UDP port 1434. SQL Server Browser reads the registry, identifies all instances of SQL Server on the computer, and notes the ports and named pipes that they use. When a server has two or more network cards, SQL Server Browser returns the first enabled port it encounters for SQL Server. SQL Server Browser support ipv6 and ipv4.

When SQL Server clients request SQL Server resources, the client network library sends a UDP message to the server using port 1434. SQL Server Browser responds with the TCP/IP port or named pipe of the requested instance. The network library on the client application then completes the connection by sending a request to the server using the port or named pipe of the desired instance. The SQL Server Browser does not return port information for the default instance.

For information about starting and stopping the SQL Server Browser service, see Start, Stop, Pause, Resume, Restart the Database Engine, SQL Server Agent, or SQL Server Browser Service.

Using SQL Server Browser

If the SQL Server Browser service is not running, you are still able to connect to SQL Server if you provide the correct port number or named pipe. For instance, you can connect to the default instance of SQL Server with TCP/IP if it is running on port 1433.

However, if the SQL Server Browser service is not running, the following connections do not work:

- Any component that tries to connect to a named instance without fully specifying all the parameters (such as the TCP/IP port or named pipe).
- Any component that generates or passes server\instance information that could later be used by other components to reconnect.
- Connecting to a named instance without providing the port number or pipe.
- DAC to a named instance or the default instance if not using TCP/IP port 1433.
- The OLAP redirector service.
- Enumerating servers in SQL Server Management Studio, Enterprise Manager, or Query Analyzer.

  If you are using SQL Server in a client-server scenario (for example, when your application is accessing SQL Server across a network), if you stop or disable the SQL Server Browser service, you must assign a specific port number to each instance and write your client application code to always use that port number. This approach has the following problems:

- You must update and maintain client application code to ensure it is connecting to the proper port.
- The port you choose for each instance may be used by another service or application on the server, causing the instance of SQL Server to be unavailable.

Clustering

SQL Server Browser is not a clustered resource and does not support failover from one cluster node to the other. Therefore, in the case of a cluster, SQL Server Browser should be installed and turned on for each node of the cluster. On clusters, SQL Server Browser listens on IP_ANY.

> **NOTE**
>
> When listening on IP_ANY, when you enable listening on specific IPs, the user must configure the same TCP port on each IP, because SQL Server Browser returns the first IP/port pair that it encounters.

Installing, Uninstalling, and Running from the Command Line

By default, the SQL Server Browser program is installed at C:\Program Files (x86)\Microsoft SQL Server\90\Shared\sqlbrowser.exe.

The SQL Server Browser service is uninstalled when the last instance of SQL Server is removed.

SQL Server Browser can be started from the command prompt for troubleshooting, by using the **-c** switch:

```
<drive>\<path>\sqlbrowser.exe -c
```

Security

Account Privileges

SQL Server Browser listens on a UDP port and accepts unauthenticated requests by using SQL Server Resolution Protocol (SSRP). SQL Server Browser should be run in the security context of a low privileged user to minimize exposure to a malicious attack. The logon account can be changed by using the SQL Server Configuration Manager. The minimum user rights for SQL Server Browser are the following:

- Deny access to this computer from the network
- Deny logon locally
- Deny Log on as a batch job
- Deny Log On Through Terminal Services
- Log on as a service
- Read and write the SQL Server registry keys related to network communication (ports and pipes)

Default Account

Setup configures SQL Server Browser to use the account selected for services during setup. Other possible accounts include the following:

- Any **domain\local** account
- The **local service** account
- The **local system** account (not recommended as has unnecessary privileges)

Hiding SQL Server

Hidden instances are instances of SQL Server that support only shared memory connections. For SQL Server, set the `HideInstance` flag to indicate that SQL Server Browser should not respond with information about this server instance.

Using a Firewall

To communicate with the SQL Server Browser service on a server behind a firewall, open UDP port 1434, in addition to the TCP port used by SQL Server (e.g., 1433). For information about working with a firewall, see "How to: Configure a Firewall for SQL Server Access" in SQL Server Books Online.

See Also

Network Protocols and Network Libraries

Database Engine Service Startup Options

3/24/2017 • 7 min to read • Edit Online

Startup options designate certain file locations needed during startup, and specify some server wide conditions. Most users do not need to specify startup options unless you are troubleshooting the Database Engine or you have an unusual problem and are directed to use a startup option by SQL Server Customer Support.

WARNING

Improper use of startup options can affect server performance and can prevent SQL Server from starting.

About Startup Options

When you install SQL Server, Setup writes a set of default startup options in the Microsoft Windows registry. You can use these startup options to specify an alternate master database file, master database log file, or error log file. If the Database Engine cannot locate the necessary files, SQL Server will not start.

Startup options can be set by using SQL Server Configuration Manager. For information, see Configure Server Startup Options (SQL Server Configuration Manager).

List of Startup Options

DEFAULT STARTUP OPTIONS	DESCRIPTION
-d *master_file_path*	Is the fully qualified path for the master database file (typically, C:\Program Files\Microsoft SQL Server\MSSQL.*n*\MSSQL\Data\master.mdf). If you do not provide this option, the existing registry parameters are used.
-e *error_log_path*	Is the fully qualified path for the error log file (typically, C:\Program Files\Microsoft SQL Server\MSSQL.*n*\MSSQL\LOG\ERRORLOG). If you do not provide this option, the existing registry parameters are used.
-l *master_log_path*	Is the fully qualified path for the master database log file (typically C:\Program Files\Microsoft SQL Server\MSSQL.*n*\MSSQL\Data\mastlog.ldf). If you do not specify this option, the existing registry parameters are used.

OTHER STARTUP OPTIONS	DESCRIPTION
-c	Shortens startup time when starting SQL Server from the command prompt. Typically, the SQL Server Database Engine starts as a service by calling the Service Control Manager. Because the SQL Server Database Engine does not start as a service when starting from the command prompt, use **-c** to skip this step.

OTHER STARTUP OPTIONS	DESCRIPTION
-f	Starts an instance of SQL Server with minimal configuration. This is useful if the setting of a configuration value (for example, over-committing memory) has prevented the server from starting. Starting SQL Server in minimal configuration mode places SQL Server in single-user mode. For more information, see the description for **-m** that follows.
-g *memory_to_reserve*	Specifies an integer number of megabytes (MB) of memory that SQL Server will leave available for memory allocations within the SQL Server process, but outside the SQL Server memory pool. The memory outside of the memory pool is the area used by SQL Server for loading items, such as extended procedure .dll files, the OLE DB providers referenced by distributed queries, and automation objects referenced in Transact-SQL statements. The default is 256 MB. Use of this option might help tune memory allocation, but only when physical memory exceeds the configured limit set by the operating system on virtual memory available to applications. Use of this option might be appropriate in large memory configurations in which the memory usage requirements of SQL Server are atypical and the virtual address space of the SQL Server process is totally in use. Incorrect use of this option can lead to conditions under which an instance of SQL Server may not start or may encounter run-time errors. Use the default for the **-g** parameter unless you see any of the following warnings in the SQL Server error log: "Failed Virtual Allocate Bytes: FAIL_VIRTUAL_RESERVE <size>" "Failed Virtual Allocate Bytes: FAIL_VIRTUAL_COMMIT <size>" These messages might indicate that SQL Server is trying to free parts of the SQL Server memory pool in order to find space for items, such as extended stored procedure .dll files or automation objects. In this case, consider increasing the amount of memory reserved by the **-g** switch. Using a value lower than the default will increase the amount of memory available to the memory pool managed by the SQL Server Memory Manager and thread stacks; this may, in turn, provide some performance benefit to memory-intensive workloads in systems that do not use many extended stored procedures, distributed queries, or automation objects.

OTHER STARTUP OPTIONS	DESCRIPTION
-m	Starts an instance of SQL Server in single-user mode. When you start an instance of SQL Server in single-user mode, only a single user can connect, and the CHECKPOINT process is not started. CHECKPOINT guarantees that completed transactions are regularly written from the disk cache to the database device. (Typically, this option is used if you experience problems with system databases that should be repaired.) Enables the sp_configure allow updates option. By default, allow updates is disabled. Starting SQL Server in single-user mode enables any member of the computer's local Administrators group to connect to the instance of SQL Server as a member of the sysadmin fixed server role. For more information, see Connect to SQL Server When System Administrators Are Locked Out. For more information about single-user mode, see Start SQL Server in Single-User Mode.

-mClient Application Name	Limits the connections to a specified client application. For example, `-mSQLCMD` limits connections to a single connection and that connection must identify itself as the SQLCMD client program. Use this option when you are starting SQL Server in single-user mode and an unknown client application is taking the only available connection. Use `"Microsoft SQL Server Management Studio - Query"` to connect with the SSMS Query Editor. The SSMS Query Editor option cannot be configured by using SQL Server Configuration Manager because it includes the dash character which is rejected by the tool.
	
	Client Application Name is case sensitive. Double quotes are required if the application name contains spaces or special characters.
	
	Examples when starting from the command line:
	
	`C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Binn\sqlserver -s MSSQLSERVER -m"Microsoft SQL Server Management Studio - Query"`
	
	`C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Binn\sqlserver -s MSSQLSERVER -mSQLCMD`
	
	**** Security Note *\*** Do not use this option as a security feature. The client application provides the client application name, and can provide a false name as part of the connection string.

-n	Does not use the Windows application log to record SQL Server events. If you start an instance of SQL Server with **-n**, we recommend that you also use the **-e** startup option. Otherwise, SQL Server events are not logged.
-s	Allows you to start a named instance of SQL Server. Without the **-s** parameter set, the default instance will try to start. You must switch to the appropriate BINN directory for the instance at a command prompt before starting **sqlservr.exe**. For example, if Instance1 were to use \mssql$Instance1 for its binaries, the user must be in the \mssql$Instance1\binn directory to start **sqlservr.exe -s instance1**.

OTHER STARTUP OPTIONS	DESCRIPTION
-T *trace#*	Indicates that an instance of SQL Server should be started with a specified trace flag (*trace#*) in effect. Trace flags are used to start the server with nonstandard behavior. For more information, see Trace Flags (Transact-SQL). ** **Important** ** When specifying a trace flag with the **-T** option, use an uppercase "T" to pass the trace flag number. A lowercase "t" is accepted by SQL Server, but this sets other internal trace flags that are required only by SQL Server support engineers. (Parameters specified in the Control Panel startup window are not read.)
-x	Disables the following monitoring features: SQL Server performance monitor counters Keeping CPU time and cache-hit ratio statistics Collecting information for the DBCC SQLPERF command Collecting information for some dynamic management views Many extended-events event points ** **Warning** ** When you use the **–x** startup option, the information that is available for you to diagnose performance and functional problems with SQL Server is greatly reduced.
-E	Increases the number of extents that are allocated for each file in a filegroup. This option may be helpful for data warehouse applications that have a limited number of users running index or data scans. It should not be used in other applications because it might adversely affect performance. This option is not supported in 32-bit releases of SQL Server.

Using Startup Options for Troubleshooting

Some startup options, such as single-user mode and minimal configuration mode, are principally used during troubleshooting. Starting the server for troubleshooting with the **–m** or **–f** options is easiest at the command line, while manually starting sqlservr.exe.

> **NOTE**
>
> When SQL Server is started by using **net start**, startup options use a slash (/) instead of a hyphen (-).

Using Startup Options During Normal Operations

You may want to use some startup options every time you start SQL Server. These options, such as **–g** or starting with a trace flag, are most easily done by configuring the startup parameters by using SQL Server Configuration Manager. These tool saves the startup options as registry keys, enabling SQL Server to always start with the startup options.

Compatibility Support

The **-h** parameter is not supported in SQL Server 2016. This parameter was used in earlier versions of 32-bit instances of SQL Server to reserve virtual memory address space for Hot Add memory metadata when AWE is enabled. For more information, see Discontinued SQL Server Features in SQL Server 2016.

Related Tasks

Configure the scan for startup procs Server Configuration Option

Start, Stop, Pause, Resume, Restart the Database Engine, SQL Server Agent, or SQL Server Browser Service

See Also

CHECKPOINT (Transact-SQL)
sqlservr Application

Start, Stop, Pause, Resume, Restart SQL Server Services

3/29/2017 • 9 min to read • Edit Online

This topic describes how to start, stop, pause, resume, or restart the SQL Server Database Engine, the SQL Server Agent, or the SQL Server Browser service by using SQL Server Configuration Manager, SQL Server Management Studio, **net** commands from a command prompt, Transact-SQL, or PowerShell.

- **Before you begin:**
  - What are these services?
  - Additional Information
  - Security
- **Instructions using:**
  - SQL Server Configuration Manager
  - SQL Server Management Studio
  - net Commands from a Command Prompt window
  - Transact-SQL
  - PowerShell

Before You Begin

What is the SQL Server Database Engine service, the SQL Server Agent service, and the SQL Server Browser service?

SQL Server components are executable programs that run as a Windows service. Programs that run as a Windows service can continue to operate without displaying any activity on the computer screen.

Database Engine service
The executable process that is the SQL Server Database Engine. The Database Engine can be the default instance (limit one per computer), or can be one of many named instances of the Database Engine. Use SQL Server Configuration Manager to determine which instances of Database Engine are installed on the computer. The default instance (if you install it) is listed as **SQL Server (MSSQLSERVER)**. Named instances (if you install them) are listed as **SQL Server ()**. By default, SQL Server Express is installed as **SQL Server (SQLEXPRESS)**.

SQL Server Agent service
A Windows service that executes scheduled administrative tasks, which are called jobs and alerts. For more information, see SQL Server Agent. SQL Server Agent is not available in every edition of SQL Server. For a list of features that are supported by the editions of SQL Server, see Features Supported by the Editions of SQL Server 2016.

SQL Server Browser service
A Windows service that listens for incoming requests for SQL Server resources and provides clients information about SQL Server instances installed on the computer. A single instance of the SQL Server Browser service is used for all instances of SQL Server installed on the computer.

Additional Information

- Pausing the Database Engine service prevents new users from connecting to the Database Engine, but users who are already connected can continue to work until their connections are broken. Use pause when you want to wait for users to complete work before you stop the service. This enables them to complete transactions that are in progress. Resume allows the Database Engine to accept new connections again. The SQL Server Agent service cannot be paused or resumed.
- The SQL Server Configuration Manager and SQL Server Management Studio display the current status of services by using the following icons.

  SQL Server Configuration Manager

  - A green arrow on the icon next to the service name indicates that the service is started.
  - A red square on the icon next to the service name indicates that the service is stopped.
  - Two vertical blue lines on the icon next to the service name indicates that the service is paused.
  - When restarting the Database Engine, a red square will indicate that the service stopped, and then a green arrow will indicate that he service started successfully.

    SQL Server Management Studio

  - A white arrow on a green circle icon next to the service name indicates that the service is started.
  - A white square on a red circle icon next to the service name indicates that the service is stopped.
  - Two vertical white lines on a blue circle icon next to the service name indicates that the service is paused.
- When using SQL Server Configuration Manager or SQL Server Management Studio, only options that are possible will be available. For example, if the service is already started, **Start** will be unavailable.
- When running on a cluster, the SQL Server Database Engine service is best managed by using Cluster Administrator.

Security

Permissions

By default, only members of the local administrators group can start, stop, pause, resume or restart a service. To grant non-administrators the ability to manage services, see How to grant users rights to manage services in Windows Server 2003. (The process is similar on other versions of Windows.)

Stopping the Database Engine by using the Transact-SQL**SHUTDOWN** command requires membership in the **sysadmin** or **serveradmin** fixed server roles, and is not transferable.

Using SQL Server Configuration Manager

Starting SQL Server Configuration Manager

1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

   Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager does not appear as an application in newer versions of Windows. Here are the paths to the last four versions when Windows in installed on the C drive.

SQL Server 2016	C:\Windows\SysWOW64\SQLServerManager13.msc

SQL Server 2014	C:\Windows\SysWOW64\SQLServerManager12.msc
SQL Server 2012	C:\Windows\SysWOW64\SQLServerManager11.msc
SQL Server 2008	C:\Windows\SysWOW64\SQLServerManager10.msc

To start, stop, pause, resume, or restart the an instance of the SQL Server Database Engine

1. Start SQL Server Configuration Manager, using the instructions above.
2. If the **User Account Control** dialog box appears, click **Yes**.
3. In SQL Server Configuration Manager, in the left pane, click **SQL Server Services**.
4. In the results pane, right-click **SQL Server (MSSQLServer)** or a named instance, and then click **Start**, **Stop**, **Pause**, **Resume**, or **Restart**.
5. Click **OK** to close SQL Server Configuration Manager.

> **NOTE**
> To start an instance of the SQL Server Database Engine with startup options, see Configure Server Startup Options (SQL Server Configuration Manager).

To start, stop, pause, resume, or restart the SQL Server Browser or an instance of the SQL Server Agent

1. Start SQL Server Configuration Manager, using the instructions above.
2. If the **User Account Control** dialog box appears, click **Yes**.
3. In SQL Server Configuration Manager, in the left pane, click **SQL Server Services**.
4. In the results pane, right-click **SQL Server Browser**, or **SQL Server Agent (MSSQLServer)** or **SQL Server Agent ()** for a named instance, and then click **Start**, **Stop**, **Pause**, **Resume**, or **Restart**.
5. Click **OK** to close SQL Server Configuration Manager.

> **NOTE**
> SQL Server Agent cannot be paused.

Using SQL Server Management Studio

To start, stop, pause, resume, or restart the an instance of the SQL Server Database Engine

1. In Object Explorer, connect to the instance of the Database Engine, right-click the instance of the Database Engine you want to start, and then click **Start**, **Stop**, **Pause**, **Resume**, or **Restart**.

   Or, in Registered Servers, right-click the instance of the Database Engine you want to start, point to **Service Control**, and then click **Start**, **Stop**, **Pause**, **Resume**, or **Restart**.
2. If the **User Account Control** dialog box appears, click **Yes**.
3. When prompted if you want to perform the action, click **Yes**.

To start, stop, or restart the an instance of the SQL Server Agent

1. In Object Explorer, connect to the instance of the Database Engine, right-click **SQL Server Agent**, and then click **Start**, **Stop**, or **Restart**.

2. If the **User Account Control** dialog box appears, click **Yes**.

3. When prompted if you want to perform the action, click **Yes**.

From the Command Prompt Window using net Commands

The Microsoft SQL Server services can be started, stopped, or paused by using Microsoft Windows **net** commands.

To start the default instance of the Database Engine

- From a command prompt, enter one of the following commands:

  net start "SQL Server (MSSQLSERVER)"

  -or-

  net start MSSQLSERVER

To start a named instance of the Database Engine

- From a command prompt, enter one of the following commands. Replace <*instancename*> with the name of the instance you want to manage.

  net start "SQL Server (*instancename* **)"**

  -or-

  net start MSSQL$ *instancename*

To start the Database Engine with startup options

- Add startup options to the end of the **net start "SQL Server (MSSQLSERVER)"** statement, separated by a space. When started using **net start**, startup options use a slash (/) instead of a hyphen (-).

  net start "SQL Server (MSSQLSERVER)" /f /m

  -or-

  net start MSSQLSERVER /f /m

  > **NOTE**
  >
  > For more information about startup options, see Database Engine Service Startup Options.

To start the SQL Server Agent on the default instance of SQL Server

- From a command prompt, enter one of the following commands:

  net start "SQL Server Agent (MSSQLSERVER)"

  -or-

  net start SQLSERVERAGENT

To start the SQL Server Agent on a named instance of SQL Server

- From a command prompt, enter one of the following commands. Replace *instancename* with the name of the instance you want to manage.

  net start "SQL Server Agent(*instancename* **)"**

  -or-

  net start SQLAgent$ *instancename*

For information about how to run SQL Server Agent in verbose mode for troubleshooting, see sqlagent90 Application.

To start the SQL Server Browser

- From a command prompt, enter one of the following commands:

  net start "SQL Server Browser"

  -or-

  net start SQLBrowser

To pause or stop services from the Command Prompt window

- To pause or stop services modify the commands in the following ways.

  - To pause a service, replace **net start** with **net pause**.

  - To stop a service, replace **net start** with use **net stop**.

Using Transact-SQL

The Database Engine can be stopped by using the **SHUTDOWN** statement.

To stop the Database Engine using Transact-SQL

- To wait for currently running Transact-SQL statements and stored procedures to finish, and then stop the Database Engine, execute the following statement.

  ```
  SHUTDOWN;
  ```

- To stop the Database Engine immediately, execute the following statement.

  ```
  SHUTDOWN WITH NOWAIT;
  ```

  For more information about the **SHUTDOWN** statement, see SHUTDOWN (Transact-SQL).

Using PowerShell

To start and stop Database Engine services

1. In a Command Prompt window, start SQL Server PowerShell by executing the following command.

   ```
   sqlps
   ```

2. At a SQL Server PowerShell command prompt, by executing the following command. Replace `computername` with the name of your computer.

   ```
   # Get a reference to the ManagedComputer class.
   CD SQLSERVER:\SQL\computername
   $Wmi = (get-item .).ManagedComputer
   ```

3. Identify the service that you want to stop or start. Pick one of the following lines. Replace `instancename` with the name of the named instance.

   - To get a reference to the default instance of the Database Engine.

```
$DfltInstance = $Wmi.Services['MSSQLSERVER']
```

- To get a reference to a named instance of the Database Engine.

```
$DfltInstance = $Wmi.Services['MSSQL$instancename']
```

- To get a reference to the SQL Server Agent service on the default instance of the Database Engine.

```
$DfltInstance = $Wmi.Services['SQLSERVERAGENT']
```

- To get a reference to the SQL Server Agent service on a named instance of the Database Engine.

```
$DfltInstance = $Wmi.Services['SQLAGENT$instancename']
```

- To get a reference to the SQL Server Browser service.

```
$DfltInstance = $Wmi.Services['SQLBROWSER']
```

4. Complete the example to start and then stop the selected service.

```
# Display the state of the service.
$DfltInstance
# Start the service.
$DfltInstance.Start();
# Wait until the service has time to start.
# Refresh the cache.
$DfltInstance.Refresh();
# Display the state of the service.
$DfltInstance
# Stop the service.
$DfltInstance.Stop();
# Wait until the service has time to stop.
# Refresh the cache.
$DfltInstance.Refresh();
# Display the state of the service.
$DfltInstance
```

See Also

Overview of SQL Server Setup Documentation
View and Read SQL Server Setup Log Files
SQL Server Configuration Manager
Start SQL Server with Minimal Configuration
Features Supported by the Editions of SQL Server 2016

Start SQL Server in Single-User Mode

3/24/2017 • 2 min to read • Edit Online

Under certain circumstances, you may have to start an instance of SQL Server in single-user mode by using the **startup option -m.** For example, you may want to change server configuration options or recover a damaged master database or other system database. Both actions require starting an instance of SQL Server in single-user mode.

Starting SQL Server in single-user mode enables any member of the computer's local Administrators group to connect to the instance of SQL Server as a member of the sysadmin fixed server role. For more information, see Connect to SQL Server When System Administrators Are Locked Out.

When you start an instance of SQL Server in single-user mode, note the following:

- Only one user can connect to the server.
- The CHECKPOINT process is not executed. By default, it is executed automatically at startup.

NOTE

Stop the SQL Server Agent service before connecting to an instance of SQL Server in single-user mode; otherwise, the SQL Server Agent service uses the connection, thereby blocking it.

When you start an instance of SQL Server in single-user mode, SQL Server Management Studio can connect to SQL Server. Object Explorer in Management Studio might fail because it requires more than one connection for some operations. To manage SQL Server in single-user mode, execute Transact-SQL statements by connecting only through the Query Editor in Management Studio, or use the sqlcmd utility.

When you use the **-m** option with **sqlcmd** or Management Studio, you can limit the connections to a specified client application. For example, **-m"sqlcmd"** limits connections to a single connection and that connection must identify itself as the **sqlcmd** client program. Use this option when you are starting SQL Server in single-user mode and an unknown client application is taking the only available connection. To connect through the Query Editor in Management Studio, use **-m"Microsoft SQL Server Management Studio - Query"**.

IMPORTANT

Do not use this option as a security feature. The client application provides the client application name, and can provide a false name as part of the connection string.

Note for Clustered installations

For SQL Server installation in a clustered environment, when SQL Server is started in single user mode, the cluster resource dll uses up the available connection thereby blocking any other connections to the server. When SQL Server is in this state, if you try to bring SQL Server Agent resource online, it may fail over the SQL resource to a different node if the resource is configured to affect the group.

To get around the problem use the following procedure:

1. Remove the –m startup parameter from the SQL Server advanced Properties.
2. Take the SQL Server resource offline.

3. From the current owner node of this group, issue the following command from the command prompt: net start MSSQLSERVER /m.

4. Verify from the cluster administrator or failover cluster management console that the SQL Server resource is still offline.

5. Connect to the SQL Server now using the following command and do the necessary operation: SQLCMD -E -S<servername>.

6. Once the operation is complete, close the command prompt and bring back the SQL and other resources online through cluster administrator.

See Also

Start, Stop, or Pause the SQL Server Agent Service
Diagnostic Connection for Database Administrators
sqlcmd Utility
CHECKPOINT (Transact-SQL)
sp_configure (Transact-SQL)
Database Engine Service Startup Options

Start SQL Server with Minimal Configuration

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

If you have configuration problems that prevent the server from starting, you can start an instance of Microsoft SQL Server by using the minimal configuration startup option. This is the startup option **-f**. Starting an instance of SQL Server with minimal configuration automatically puts the server in single-user mode.

When you start an instance of SQL Server in minimal configuration mode, note the following:

- Only a single user can connect, and the `CHECKPOINT` process is not executed.
- Remote access and read-ahead are disabled.
- Startup stored procedures do not run.
- `tempdb` is configured at the smallest possible size.

  After the server has been started with minimal configuration, you should change the appropriate server option value or values, stop, and then restart the server.

IMPORTANT

Use the **sqlcmd** utility and the dedicated administrator connection (DAC) to connect to SQL Server. If you use a typical connection, stop the SQL Server Agent service before connecting to an instance of SQL Server in minimal configuration mode. Otherwise, the SQL Server Agent service uses the connection, thereby blocking it.

See Also

Start, Stop, or Pause the SQL Server Agent Service
Diagnostic Connection for Database Administrators
sqlcmd Utility
Server Configuration Options (SQL Server)
sp_configure (Transact-SQL)
Database Engine Service Startup Options

SCM Services - Connect to Another Computer

3/24/2017 • 2 min to read • Edit Online

This topic describes how to connect to another computer in SQL Server 2016. Follow the first procedure to open the Windows Computer Management Microsoft Management Console (mmc), connect to the computer, and expand the Services and Applications tree. Follow the second procedure to create a file with a link to the SQL Server Configuration Manager on a remote computer.

> **NOTE**
> Some actions cannot be performed by Configuration Manager when connecting remotely.

To start, stop, pause, or resume services on another computer, you can also connect to the server with SQL Server Management Studio, right-click the server or SQL Server Agent and then click the desired action.

To connect to another computer with Windows Computer Management

1. On the **Start** menu, right-click **My Computer**, and then click **Manage.**
2. In **Computer Management**, right-click **Computer Management (Local)**, and then click **Connect to another computer**.
3. In the **Select Computer** dialog box, in the **Another computer** text box, type the name of the computer you want to manage, and then click **OK**.

   Computer Management displays the services running on the remote computer. The top-level node changes to **Computer Management** *<remotecomputer>*.
4. In the console tree, expand **Services and Applications**, and then expand **SQL Server Configuration Manager** to manage the remote computer's services.

To save a link to SQL Server Configuration Manager for another computer

1. On the **Start** menu, click **Run**.
2. In the **Open** box, type **mmc -a** to open the Microsoft Management Console in author mode.
3. On the **File** menu, click **Add/Remove Snap-in**.
4. In the **Add/Remove Snap-in** window, click **Add**.
5. In the **Add Standalone Snap-in** window, click **Computer Management** and then click **Add**.
6. In the **Computer Management** window, click **Another computer**, type the name of the remote computer you wish to manage, and then click **Finish**.
7. In the **Add Standalone Snap-in** window, click **Close**.
8. In the **Add/Remove Snap-in** window, click **OK**.
9. Expand **Computer Management (***<computer name>***)**, and **Services and Applications**.
10. Right-click **SQL Server Configuration Manager**, and then click **New Window from here**.
11. On the **Window** menu, click **Console Root**, to switch back to the first widow, and delete the window.
12. On the **File** menu, click **Save As**, and save the file in the desired folder, with an appropriate name with the

.msc file extension. Close the Microsoft Management Console.

13. To open SQL Server Configuration Manager on the target computer, double-click the file. If desired, save a link to the file on the desktop or in the **Start** menu.

Caution

When using SQL Server Configuration Manager on a remote computer, the computer name is not obvious and it is possible to mistakenly stop or configure the wrong computer. On the **Service** tab, check the **Host Name** box to confirm the computer name before modifying a service.

See Also

Configure WMI to Show Server Status in SQL Server Tools

SCM Services - Set an Instance to Start Automatically

3/29/2017 • 1 min to read • Edit Online

This topic describes how to set an instance of SQL Server to start automatically in SQL Server 2016 by using SQL Server Configuration Manager. During setup, SQL Server is normally configured to start automatically. If this was not done, you can change that setting at any time.

Using SQL Server Configuration Manager

To set an instance of SQL Server to start automatically

1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

   > **NOTE**
   >
   > Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager does not appear as an application in newer versions of Windows.
   >
   > - **Windows 10**:
   >   To open SQL Server Configuration Manager, on the **Start Page**, type SQLServerManager13.msc (for SQL Server 2016). For previous versions of SQL Server replace 13 with a smaller number. Clicking SQLServerManager13.msc opens the Configuration Manager. To pin the Configuration Manager to the Start Page or Task Bar, right-click SQLServerManager13.msc, and then click **Open file location**. In the Windows File Explorer, right-click SQLServerManager13.msc, and then click **Pin to Start** or **Pin to taskbar**.
   >   - **Windows 8**:
   >   To open SQL Server Configuration Manager, in the **Search** charm, under **Apps**, type **SQLServerManager<version>.msc** such as **SQLServerManager13.msc**, and then press **Enter**.

2. In **SQL Server Configuration Manager**, expand **Services**, and then click **SQL Server**.

3. In the details pane, right-click the name of the instance you want to start automatically, and then click **Properties**.

4. In the **SQL Server <*instancename*> Properties** dialog box, set **Start Mode** to **Automatic**.

5. Click **OK**, and then close SQL Server Configuration Manager.

See Also

Prevent Automatic Startup of an Instance of SQL Server (SQL Server Configuration Manager)
Connect to Another Computer (SQL Server Configuration Manager)
Configure WMI to Show Server Status in SQL Server Tools

SCM Services - Prevent Automatic Startup of an Instance

3/29/2017 • 1 min to read • Edit Online

This topic describes how prevent an instance of SQL Server from starting automatically in SQL Server 2016 by using SQL Server Configuration Manager. SQL Server is normally configured to start automatically. You can change that by setting the start mode for the instance to manual.

Using SQL Server Configuration Manager

To prevent automatic startup of an instance of SQL Server

1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

   > **NOTE**
   >
   > Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager does not appear as an application in newer versions of Windows.
   >
   > - **Windows 10**:
   >   To open SQL Server Configuration Manager, on the **Start Page**, type SQLServerManager13.msc (for SQL Server 2016). For previous versions of SQL Server replace 13 with a smaller number. Clicking SQLServerManager13.msc opens the Configuration Manager. To pin the Configuration Manager to the Start Page or Task Bar, right-click SQLServerManager13.msc, and then click **Open file location**. In the Windows File Explorer, right-click SQLServerManager13.msc, and then click **Pin to Start** or **Pin to taskbar**.
   >   - **Windows 8**:
   >   To open SQL Server Configuration Manager, in the **Search** charm, under **Apps**, type **SQLServerManager<version>.msc** such as **SQLServerManager13.msc**, and then press **Enter**.

2. In SQL Server Configuration Manager, expand **Services**, and then click **SQL Server**.

3. In the details pane, right-click **MSSQLServer**, and then click **Properties.**

4. In the **SQL Server <*instancename*> Properties** dialog box, in the **Properties** box, set the value of **Start Mode** to **Manual**.

5. Click **OK** to close the **SQL Server <*instancename*> Properties** dialog box, and then close SQL Server Configuration Manager.

See Also

Start, Stop, Pause, Resume, Restart the Database Engine, SQL Server Agent, or SQL Server Browser Service

SCM Services - Change the Service Startup Account

3/29/2017 • 3 min to read • Edit Online

This topic describes how to Use the SQL Server Configuration Manager to change the start up options of SQL Server services and to change the service accounts that are used by the SQL Server Database Engine, SQL Server Agent, SQL Server Browser, SQL Server Analysis Services, and SQL Server Integration Services. in SQL Server 2016 by using SQL Server Management Studio, Transact-SQL, or PowerShell. For more information about how to select an appropriate service account, see Configure Windows Service Accounts and Permissions.

IMPORTANT
When you change the service startup account for the Database Engine and SQL Server Agent, the SQL Server service (the Database Engine) must be restarted for the change to take effect. When the service is restarted, all databases associated with that instance of SQL Server will be unavailable until the service successfully restarts. If you have to change the service startup account of SQL Server or SQL Server Agent, make sure that you do so during regularly scheduled maintenance or when the databases can be taken offline without interrupting daily operations.

Before You Begin

Limitations and Restrictions

- Clustered servers

  Changing the service account that is used by SQL Server or SQL Server Agent must be performed from the active node of the SQL Server cluster.

  When running on Windows Server 2008 (in a non-default configuration using Domain groups), changing the service account that is used by SQL Server or SQL Server Agent requires SQL Server Configuration Manager to stop SQL Server by taking the resource groups offline.

- SKU Upgrade (SQL Server Express to non-Express SKU)

  During SQL Server Express installation, the SQL Server Agent service is configured to use the Network Service account but disabled. SQL Server Configuration Manager can change the account assigned for the SQL Server Agent service but the service cannot be enabled or started. After SKU upgrade from SQL Server Express to non-Express, the SQL Server Agent service is not automatically enabled, but can be enabled when needed by using the SQL Server Configuration Manager and changing the service start mode to Manual or Automatic.

Using SQL Server Configuration Manager

To change the SQL Server service startup account

1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

NOTE

Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager does not appear as an application in newer versions of Windows.

- **Windows 10**:
  To open SQL Server Configuration Manager, on the **Start Page**, type SQLServerManager13.msc (for SQL Server 2016). For previous versions of SQL Server replace 13 with a smaller number. Clicking SQLServerManager13.msc opens the Configuration Manager. To pin the Configuration Manager to the Start Page or Task Bar, right-click SQLServerManager13.msc, and then click **Open file location**. In the Windows File Explorer, right-click SQLServerManager13.msc, and then click **Pin to Start** or **Pin to taskbar**.
  - **Windows 8**:
    To open SQL Server Configuration Manager, in the **Search** charm, under **Apps**, type **SQLServerManager<version>.msc** such as **SQLServerManager13.msc**, and then press **Enter**.

2. In SQL Server Configuration Manager, click **SQL Server Services**.
3. In the details pane, right-click the name of the SQL Server instance for which you want to change the service startup account, and then click **Properties**.
4. In the **SQL Server <***instancename***> Properties** dialog box, click the **Log On** tab, and select a **Log on as** account type.
5. After selecting the new service startup account, click **OK**.

   A message box asks whether you want to restart the SQL Server service.
6. Click **Yes**, and then close SQL Server Configuration Manager.

See Also

Start, Stop, Pause, Resume, Restart the Database Engine, SQL Server Agent, or SQL Server Browser Service
Configure WMI to Show Server Status in SQL Server Tools

SCM Services - Configure Server Startup Options

3/24/2017 • 2 min to read • Edit Online

This topic describes how to to configure startup options that will be used every time the Database Engine starts in SQL Server 2016 by using SQL Server Configuration Manager. For a list of startup options, see Database Engine Service Startup Options.

Before You Begin

Limitations and Restrictions

SQL Server Configuration Manager writes startup parameters to the registry. They take effect upon the next startup of the Database Engine.

On a cluster, changes must be made on the active server when SQL Server is online, and will take effect when the Database Engine is restarted. The registry update of the startup options on the other node will occur upon the next failover.

Security

Permissions

Configuring server startup options is restricted to users who can change the related entries in the registry. This includes the following users.

- Members of the local administrators group.
- The domain account that is used by SQL Server, if the Database Engine is configured to run under a domain account.

Using SQL Server Configuration Manager

To configure startup options

1. Click the **Start** button, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

   > **NOTE**
   > Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager does not appear as an application in newer versions of Windows.
   >
   > - **Windows 10**:
   >   To open SQL Server Configuration Manager, on the **Start Page**, type SQLServerManager13.msc (for SQL Server 2016). For previous versions of SQL Server replace 13 with a smaller number. Clicking SQLServerManager13.msc opens the Configuration Manager. To pin the Configuration Manager to the Start Page or Task Bar, right-click SQLServerManager13.msc, and then click **Open file location**. In the Windows File Explorer, right-click SQLServerManager13.msc, and then click **Pin to Start** or **Pin to taskbar**.
   >   - **Windows 8**:
   >   To open SQL Server Configuration Manager, in the **Search** charm, under **Apps**, type **SQLServerManager<version>.msc** such as **SQLServerManager13.msc**, and then press **Enter**.

2. In SQL Server Configuration Manager, click **SQL Server Services**.
3. In the right pane, right-click **SQL Server ()**, and then click **Properties**.

4. On the **Startup Parameters** tab, in the **Specify a startup parameter** box, type the parameter, and then click **Add**.

   For example, to start in single-user mode, type **-m** in the **Specify a startup parameter** box and then click **Add**. (When you restart SQL Server in single-user mode, stop the SQL Server Agent. Otherwise, SQL Server Agent might connect first and prevent you from connecting as a second user.)

5. Click **OK**.

6. Restart the Database Engine.

   WARNING
   After you are finished using single-user mode, in the Startup Parameters box, select the **-m** parameter in the **Existing Parameters** box, and then click **Remove**. Restart the Database Engine to restore SQL Server to the typical multi-user mode.

See Also

Start SQL Server in Single-User Mode
Connect to SQL Server When System Administrators Are Locked Out
Start, Stop, or Pause the SQL Server Agent Service

SCM Services - Change the Password of the Accounts Used

3/29/2017 • 3 min to read • Edit Online

This topic describes how to change the password of the accounts used by the Database Engine and the SQL Server Agent in SQL Server 2016 by using SQL Server Configuration Manager. The SQL Server Database Engine and SQL Server Agent run on a computer as a service using credentials that are initially provided during setup. If the instance of SQL Server is running under a domain account and the password for that account is changed, the password used by SQL Server must be updated to the new password. If the password is not updated, SQL Server may lose access to some domain resources and if SQL Server stops, the service will not restart until the password is updated.

To change SQL Server Authentication passwords, see Password Expired.

Before You Begin

SQL Server Configuration Manager is the tool designed and authorized to change the settings of the SQL Server services. Changing a SQL Server service by using the Windows Service Control Manager (**services.msc**) application does not always change all of the necessary settings and might prevent the service from functioning properly. However, in a clustered environment, after changing the password on the active node by using SQL Server Configuration Manager, you must change the password on the passive node by using the Service Control Manager.

Security

Permissions

You must be an administrator of the computer to change the password used by a service.

Using SQL Server Configuration Manager

To change the password used by the SQL Server (Database Engine) service

1. Click the **Start** button, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

   > **NOTE**
   >
   > Because SQL Server Configuration Manager is a snap-in for the Microsoft Management Console program and not a stand-alone program, SQL Server Configuration Manager does not appear as an application in newer versions of Windows.
   >
   > - **Windows 10**:
   >   To open SQL Server Configuration Manager, on the **Start Page**, type SQLServerManager13.msc (for SQL Server 2016). For previous versions of SQL Server replace 13 with a smaller number. Clicking SQLServerManager13.msc opens the Configuration Manager. To pin the Configuration Manager to the Start Page or Task Bar, right-click SQLServerManager13.msc, and then click **Open file location**. In the Windows File Explorer, right-click SQLServerManager13.msc, and then click **Pin to Start** or **Pin to taskbar**.
   >   - **Windows 8**:
   >   To open SQL Server Configuration Manager, in the **Search** charm, under **Apps**, type **SQLServerManager<version>.msc** such as **SQLServerManager13.msc**, and then press **Enter**.

2. In SQL Server Configuration Manager, click **SQL Server Services**.

3. In the details pane, right-click **SQL Server (**<instancename>**)**, and then click **Properties**.

4. In the **SQL Server (**<instancename>**) Properties** dialog box, on the Log On tab, for the account listed in the **Account Name** box, type the new password in the **Password** and **Confirm Password** boxes, and then click **OK**.

   The password takes effect immediately, without restarting SQL Server.

To change the password used by the SQL Server Agent service

1. Click the **Start** button, point to **All Programs**, point to **Microsoft SQL Server 2016**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

2. In SQL Server Configuration Manager, click **SQL Server Services**.

3. In the details pane, right-click **SQL Server Agent (**<instancename>**)**, and then click **Properties**.

4. In the **SQL Server Agent (**<instancename>**) Properties** dialog box, on the Log On tab, for the account listed in the **Account Name** box, type the new password in the **Password** and **Confirm Password** boxes, and then click **OK**.

   On a stand-alone instance of SQL Server, the password takes effect immediately, without restarting SQL Server. On a clustered instance, SQL Server might take the SQL Server resource offline, and require a restart.

See Also

Managing Services How-to Topics (SQL Server Configuration Manager)

SCM Services - Configure SQL Server Error Logs

3/24/2017 • 1 min to read • Edit Online

This topic describes how to view or modify the way SQL Server error logs are recycled.

To open the Configure SQL Server Error Logs dialog box

1. In Object Explorer, expand the instance of SQL Server, expand **Management**, right-click **SQL Server Logs**, and then click **Configure**.
2. In the **Configure SQL Server Error Logs** dialog box, choose from the following options.

   Limit the number of the error log files before they are recycled
   Check to limit the number of error logs created before they are recycled. A new error log is created each time an instance of SQL Server is started. SQL Server retains backups of the previous six logs, unless you check this option, and specify a different maximum number of error log files below.

   Maximum number of error log files
   Specify the maximum number of error log files created before they are recycled. The default is 6, which is the number of previous backup logs SQL Server retains before recycling them.

Change Server Authentication Mode

3/24/2017 • 2 min to read • Edit Online

This topic describes how to change the server authentication mode in SQL Server 2016 by using SQL Server Management Studio or Transact-SQL. During installation, SQL Server Database Engine is set to either **Windows Authentication mode** or **SQL Server and Windows Authentication mode**. After installation, you can change the authentication mode at any time.

If **Windows Authentication mode** is selected during installation, the sa login is disabled and a password is assigned by setup. If you later change authentication mode to **SQL Server and Windows Authentication mode**, the sa login remains disabled. To use the sa login, use the ALTER LOGIN statement to enable the sa login and assign a new password. The sa login can only connect to the server by using SQL Server Authentication.

In This Topic

- **Before you begin:**

  Security

- **To change server authentication mode, using:**

  SQL Server Management Studio

  Transact-SQL

Before You Begin

Security

The sa account is a well-known SQL Server account and it is often targeted by malicious users. Do not enable the sa account unless your application requires it. It is very important that you use a strong password for the sa login.

Using SQL Server Management Studio

To change security authentication mode

1. In SQL Server Management Studio Object Explorer, right-click the server, and then click **Properties**.
2. On the **Security** page, under **Server authentication**, select the new server authentication mode, and then click **OK**.
3. In the SQL Server Management Studio dialog box, click **OK** to acknowledge the requirement to restart SQL Server.
4. In Object Explorer, right-click your server, and then click **Restart**. If SQL Server Agent is running, it must also be restarted.

To enable the sa login

1. In Object Explorer, expand **Security**, expand Logins, right-click **sa**, and then click **Properties**.
2. On the **General** page, you might have to create and confirm a password for the login.
3. On the **Status** page, in the **Login** section, click **Enabled**, and then click **OK**.

Using Transact-SQL

To enable the sa login

1. In Object Explorer, connect to an instance of Database Engine.

2. On the Standard bar, click **New Query**.

3. Copy and paste the following example into the query window and click **Execute**. The following example enables the sa login and sets a new password.

```
ALTER LOGIN sa ENABLE ;
GO
ALTER LOGIN sa WITH PASSWORD = '<enterStrongPasswordHere>' ;
GO
```

See Also

Strong Passwords
Security Considerations for a SQL Server Installation
ALTER LOGIN (Transact-SQL)
Connect to SQL Server When System Administrators Are Locked Out

SQL Writer Service

3/24/2017 • 2 min to read • Edit Online

The SQL Writer Service provides added functionality for backup and restore of SQL Server through the Volume Shadow Copy Service framework.

The SQL Writer Service is installed automatically. It must be running when the Volume Shadow Copy Service (VSS) application requests a backup or restore. To configure the service, use the Microsoft Windows Services applet. The SQL Writer Service installs on all operating systems.

Purpose

When running, Database Engine locks and has exclusive access to the data files. When the SQL Writer Service is not running, backup programs running in Windows do not have access to the data files, and backups must be performed using SQL Server backup.

Use the SQL Writer Service to permit Windows backup programs to copy SQL Server data files while SQL Server is running.

Volume Shadow Copy Service

The VSS is a set of COM APIs that implements a framework to allow volume backups to be performed while applications on a system continue to write to the volumes. The VSS provides a consistent interface that allows coordination between user applications that update data on disk (writers) and those that back up applications (requestors).

The VSS captures and copies stable images for backup on running systems, particularly servers, without unduly degrading the performance and stability of the services they provide. For more information on the VSS, see your Windows documentation.

Virtual Backup Device Interface (VDI)

SQL Server provides an API called Virtual Backup Device Interface (VDI) that enables independent software vendors to integrate SQL Server into their products for providing support for backup and restore operations. These APIs are engineered to provide maximum reliability and performance, and support the full range of SQL Server backup and restore functionality, including the full range of hot and snapshot backup capabilities.

Permissions

The SQL Writer service must run under the **Local System** account. The SQL Writer service uses the **NT Service\SQLWriter** login to connect to SQL Server. Using the **NT Service\SQLWriter** login allows the SQL Writer process to run at a lower privilege level in an account designated as **no login**, which limits vulnerability. If the SQL Writer service is disabled, then any utility which in relies on VSS snapshots, such as System Center Data Protection Manager, as well as some other 3rd-party products, would be broken, or worse, at risk of taking backups of databases which were not consistent. If neither SQL Server, the system it runs on, nor the host system (in the event of a virtual machine), need to use anything besides Transact-SQL backup, then the SQL Writer service can be safely disabled and the login removed. Note that the SQL Writer service may be invoked by a system or volume level backup, whether the backup is directly snapshot-based or not. Some system backup products use VSS to avoid being blocked by open or locked files. The SQL Writer service needs elevated permissions in SQL Server because in the course of its activities it briefly freezes all I/O for the instance of SQL Server.

Features

SQL Writer supports:

- Full database backup and restore including full-text catalogs
- Differential backup and restore
- Restore with move
- Database rename
- Copy-only backup
- Auto-recovery of database snapshot

  SQL Writer does not support:

- Log backups
- File and filegroup backup
- Page restore

Broadcast a Shutdown Message (Command Prompt)

3/24/2017 • 1 min to read • Edit Online

This topic describes how to broadcast a shutdown message in SQL Server 2016 by using the **net send** command. In the message, include the time the instance of SQL Server will be stopped so that users can finish their tasks.

To broadcast a shutdown message

1. From a command prompt, enter:

   net send /users "message"

   The **/users** option specifies that the message be sent to all users connected to the server

> **NOTE**
>
> The **net send** command requires the messenger service to be running on both the sending and the receiving computers. The messenger service is disabled by default on Windows Server 2003. For information about **net send**, see the Windows documentation.

On your network, it may be more appropriate to contact users by e-mail or the telephone. To determine which users are currently connected to SQL Server, use the Activity Monitor. For information on the Activity Monitor, see Activity Monitor and Open Activity Monitor (SQL Server Management Studio).

See Also

Start, Stop, Pause, Resume, Restart the Database Engine, SQL Server Agent, or SQL Server Browser Service

Log In to an Instance of SQL Server (Command Prompt)

3/24/2017 • 1 min to read • Edit Online

This topic describes how to test connectivity to an instance of SQL Server, use the **sqlcmd** utility.

To log in to the default instance of SQL Server

1. From a command prompt, enter the following command to connect by using Windows Authentication:

```
sqlcmd [ /E ] [ /S servername ]
```

To log in to a named instance of SQL Server

1. From a command prompt, enter the following command to connect by using Windows Authentication:

```
sqlcmd [ /E ] /S servername\instancename
```

See Also

sqlcmd Utility
osql Utility

Server Network Configuration

3/24/2017 • 3 min to read • Edit Online

Server network configuration tasks include enabling protocols, modifying the port or pipe used by a protocol, configuring encryption, configuring the SQL Server Browser service, exposing or hiding the SQL Server Database Engine on the network, and registering the Server Principal Name. Most of the time, you do not have to change the server network configuration. Only reconfigure the server network protocols if special network requirements.

Network configuration for SQL Server is done using SQL Server Configuration Manager. For earlier versions of SQL Server, use the Server Network Utility that ships with those products.

Protocols

Use SQL Server Configuration Manager to enable or disable the protocols used by SQL Server, and to configure the options available for the protocols. More than one protocol can be enabled. You must enable all protocols that you want clients to use. All protocols have equal access to the server. For information about which protocols you should use, see Enable or Disable a Server Network Protocol and Default SQL Server Network Protocol Configuration.

Changing a Port

You can configure the TCP/IP protocol to listen on a designated port. By default, the default instance of the Database Engine listens on TCP port 1433. Named instances of the Database Engine and SQL Server Compact are configured for dynamic ports. This means they select an available port when the SQL Server service is started. The SQL Server Browser service helps clients identify the port when they connect.

When configured for dynamic ports, the port used by SQL Server may change each time it is started. When connecting to SQL Server through a firewall, you must open the port used by SQL Server. Configure SQL Server to use a specific port, so you can configure the firewall to allow communication to the server. For more information, see Configure a Server to Listen on a Specific TCP Port (SQL Server Configuration Manager).

Changing a Named Pipe

You can configure the named pipe protocol to listen on a designated named pipe. By default, the default instance of SQL Server Database Engine listens on pipe \\.\pipe\sql\query for the default instance and \\.\pipe\MSSQL$<*instancename*>\sql\query for a named instance. The Database Engine can only listen on one named pipe, but you can change the pipe to another name if you wish. The SQL Server Browser service helps clients identify the pipe when they connect. For more information, see Configure a Server to Listen on an Alternate Pipe (SQL Server Configuration Manager).

Force Encryption

The Database Engine can be configured to require encryption when communicating with client applications. For more information, see Enable Encrypted Connections to the Database Engine (SQL Server Configuration Manager).

Extended Protection for Authentication

Support for Extended Protection for Authentication by using channel binding and service binding is available for operating systems that support Extended Protection. For more information, see Connect to the Database Engine Using Extended Protection.

Authenticating by Using Kerberos

SQL Server supports Kerberos authentication. For more information, see Register a Service Principal Name for Kerberos Connections and Microsoft Kerberos Configuration Manager for SQL Server.

Registering a Server Principal Name (SPN)

The Kerberos authentication service uses an SPN to authenticate a service. For more information, see Register a Service Principal Name for Kerberos Connections.

SPNs may also be used to make client authentication more secure when connecting with NTLM. For more information, see Connect to the Database Engine Using Extended Protection.

SQL Server Browser Service

The SQL Server Browser service runs on the server, and helps client computers to find instances of SQL Server. The SQL Server Browser service does not need to be configured, but must be running under some connection scenarios. For more information about SQL Server Browser, see SQL Server Browser Service (Database Engine and SSAS)

Hiding SQL Server

When running, SQL Server Browser responds to queries, with the name, version, and connection information for each installed instance. For SQL Server, the **HideInstance** flag, indicates that SQL Server Browser should not respond with information about this server instance. Client applications can still connect, but they must know the required connection information. For more information, see Hide an Instance of SQL Server Database Engine.

Related Content

Client Network Configuration

Manage the Database Engine Services

Connect to SQL Server When System Administrators Are Locked Out

3/24/2017 • 5 min to read • Edit Online

This topic describes how you can regain access to the SQL Server Database Engine as a system administrator. A system administrator can lose access to an instance of SQL Server because of one of the following reasons:

- All logins that are members of the sysadmin fixed server role have been removed by mistake.
- All Windows Groups that are members of the sysadmin fixed server role have been removed by mistake.
- The logins that are members of the sysadmin fixed server role are for individuals who have left the company or who are not available.
- The sa account is disabled or no one knows the password.

  One way in which you can regain access is to reinstall SQL Server and attach all the databases to the new instance. This solution is time-consuming; and, to recover the logins, it might require restoring the master database from a backup. If the backup of the master database is older, it might not have all the information. If the backup of the master database is more recent, it might have the same logins as the previous instance; therefore, administrators will still be locked out.

Resolution

Start the instance of SQL Server in single-user mode by using either the **-m** or **-f** options. Any member of the computer's local Administrators group can then connect to the instance of SQL Server as a member of the sysadmin fixed server role.

NOTE
When you start an instance of SQL Server in single-user mode, first stop the SQL Server Agent service. Otherwise, SQL Server Agent might connect first and prevent you from connecting as a second user.

When you use the **-m** option with **sqlcmd** or SQL Server Management Studio, you can limit the connections to a specified client application. For example, **-m"sqlcmd"** limits connections to a single connection and that connection must identify itself as the **sqlcmd** client program. Use this option when you are starting SQL Server in single-user mode and an unknown client application is taking the only available connection. To connect through the Query Editor in Management Studio, use **-m"Microsoft SQL Server Management Studio - Query"**.

IMPORTANT
Do not use this option as a security feature. The client application provides the client application name, and can provide a false name as part of the connection string.

For step-by-step instructions about how to start SQL Server in single-user mode, see Configure Server Startup Options (SQL Server Configuration Manager).

Step-By-Step Instructions

The following instructions describe the process for connecting to SQL Server 2016 running on Windows 8 or higher. Slight adjustments for earlier versions of SQL Server or Windows are provided. These instructions must be performed while logged in to Windows as a member of the local administrators group, and they assume that SQL Server Management Studio is installed on the computer.

1. From the Start page, start SQL Server Management Studio. On the **View** menu, select **Registered Servers**. (If your server is not already registered, right-click **Local Server Groups**, point to **Tasks**, and then click **Register Local Servers**.)

2. In the Registered Servers area, right-click your server, and then click **SQL Server Configuration Manager**. This should ask for permission to run as administrator, and then open the Configuration Manager program.

3. Close Management Studio.

4. In SQL Server Configuration Manager, in the left pane, select **SQL Server Services**. In the right-pane, find your instance of SQL Server. (The default instance of SQL Server includes **(MSSQLSERVER)** after the computer name. Named instances appear in upper case with the same name that they have in Registered Servers.) Right-click the instance of SQL Server, and then click **Properties**.

5. On the **Startup Parameters** tab, in the **Specify a startup parameter** box, type `-m` and then click **Add**. (That's a dash then lower case letter m.)

   > **NOTE**
   >
   > For some earlier versions of SQL Server there is no **Startup Parameters** tab. In that case, on the **Advanced** tab, double-click **Startup Parameters**. The parameters open up in a very small window. Be careful not to change any of the existing parameters. At the very end, add a new parameter `;-m` and then click **OK**. (That's a semi-colon then a dash then lower case letter m.)

6. Click **OK**, and after the message to restart, right-click your server name, and then click **Restart**.

7. After SQL Server has restarted your server will be in single-user mode. Make sure that that SQL Server Agent is not running. If started, it will take your only connection.

8. On the Windows 8 start screen, right-click the icon for Management Studio. At the bottom of the screen, select **Run as administrator**. (This will pass your administrator credentials to SSMS.)

   > **NOTE**
   >
   > For earlier versions of Windows, the **Run as administrator** option appears as a sub-menu.

   In some configurations, SSMS will attempt to make several connections. Multiple connections will fail because SQL Server is in single-user mode. You can select one of the following actions to perform. Do one of the following.

   a. Connect with Object Explorer using Windows Authentication (which includes your Administrator credentials). Expand **Security**, expand **Logins**, and double-click your own login. On the **Server Roles** page, select **sysadmin**, and then click **OK**.

   b. Instead of connecting with Object Explorer, connect with a Query Window using Windows Authentication (which includes your Administrator credentials). (You can only connect this way if you did not connect with Object Explorer.) Execute code such as the following to add a new Windows Authentication login that is a member of the **sysadmin** fixed server role. The following example adds a domain user named `CONTOSO\PatK`.

```
CREATE LOGIN [CONTOSO\PatK] FROM WINDOWS;
ALTER SERVER ROLE sysadmin ADD MEMBER [CONTOSO\PatK];
```

c. If your SQL Server is running in mixed authentication mode, connect with a Query Window using Windows Authentication (which includes your Administrator credentials). Execute code such as the following to create a new SQL Server Authentication login that is a member of the **sysadmin** fixed server role.

```
CREATE LOGIN TempLogin WITH PASSWORD = '************';
ALTER SERVER ROLE sysadmin ADD MEMBER TempLogin;
```

WARNING

Replace ************ with a strong password.

d. If your SQL Server is running in mixed authentication mode and you want to reset the password of the **sa** account, connect with a Query Window using Windows Authentication (which includes your Administrator credentials). Change the password of the **sa** account with the following syntax.

```
ALTER LOGIN sa WITH PASSWORD = '************';
```

WARNING

Replace ************ with a strong password.

9. The following steps now change SQL Server back to multi-user mode. Close SSMS.

10. In SQL Server Configuration Manager, in the left pane, select **SQL Server Services**. In the right-pane, right-click the instance of SQL Server, and then click **Properties**.

11. On the **Startup Parameters** tab, in the **Existing parameters** box, select `-m` and then click **Remove**.

NOTE

For some earlier versions of SQL Server there is no **Startup Parameters** tab. In that case, on the **Advanced** tab, double-click **Startup Parameters**. The parameters open up in a very small window. Remove the `;-m` which you added earlier, and then click **OK**.

12. Right-click your server name, and then click **Restart**.

    Now you should be able to connect normally with one of the accounts which is now a member of the **sysadmin** fixed server role.

See Also

Start SQL Server in Single-User Mode
Database Engine Service Startup Options

Default SQL Server Network Protocol Configuration

3/24/2017 • 2 min to read • Edit Online

To enhance security, SQL Server disables network connectivity for some new installations. Network connectivity using TCP/IP is not disabled if you are using the Enterprise, Standard, or Workgroup edition or if a previous installation of SQL Server is present. For all installations, shared memory protocol is enabled to allow local connections to the server. The SQL Server Browser service might be stopped, depending on installation conditions and installation options.

Use the SQL Server Network Configuration node of SQL Server Configuration Manager to configure the network protocols after installation. Use the SQL Server Services node of SQL Server Configuration Manager to configure the SQL Server Browser service to start automatically. For more information, see Enable or Disable a Server Network Protocol.

Default Configuration

The following table describes the configuration after installation.

EDITION	NEW INSTALLATION VS. PREVIOUS INSTALLATION IS PRESENT	SHARED MEMORY	TCP/IP	NAMED PIPES
Enterprise	New installation	Enabled	Enabled	Disabled for network connections.
Standard	New installation	Enabled	Enabled	Disabled for network connections.
Web	New installation	Enabled	Enabled	Disabled for network connections.
Developer	New installation	Enabled	Disabled	Disabled for network connections.
Evaluation	New installation	Enabled	Disabled	Disabled for network connections.
SQL Server Express	New installation	Enabled	Disabled	Disabled for network connections.
All editions	Previous installation is present but is not being upgraded.	Same as new installation	Same as new installation	Same as new installation
All editions	Upgrade	Enabled	Settings from the previous installation are preserved.	Settings from the previous installation are preserved.

NOTE

If the instance is running on a SQL Server failover cluster, it will listen on those ports on each IP address selected for SQL Server during SQL Server setup.

NOTE

When you are installing SQL Server with command-prompt arguments, you can specify the protocols to enable by using the `TCPENABLED` and `NPENABLED` parameters. For more information, see Install SQL Server from the Command Prompt.

Creating a Connection String

See the following topics for samples of connection strings:

- Creating a Valid Connection String Using Shared Memory Protocol
- Creating a Valid Connection String Using TCP IP

SQL Server Browser Settings

The SQL Server Browser service can be configured to start automatically during setup. The default is to start automatically under the following conditions:

- When upgrading an installation.
- When installing side-by-side with another instance of SQL Server.
- When installing on a cluster.
- When installing a named instance of the Database Engine including all instances of SQL Server Express.
- When installing a named instance of Analysis Services.

See Also

Hardware and Software Requirements for Installing SQL Server 2016

Surface Area Configuration

Enable or Disable a Server Network Protocol

3/24/2017 • 3 min to read • Edit Online

All network protocols are installed by SQL Server Setup, but may or may not be enabled. This topic describes how to enable or disable a server network protocol in SQL Server 2016 by using SQL Server Configuration Manager or PowerShell. The Database Engine must be stopped and restarted for the change to take effect.

IMPORTANT
During setup of SQL Server Express a login is added for the BUILTIN\Users group. This allows all authenticated users of the computer to access the instance of SQL Server Express as a member of the public role. The BUILTIN\Users login can be safely removed to restrict Database Engine access to computer users who have individual logins or are members of other Windows groups with logins.

WARNING
SQL Server and Microsoft data providers for SQL Server support TLS 1.0 and SSL 3.0. If you enforce a different protocol (such as TLS 1.1 or TLS 1.2) by making changes in the operating system SChannel layer, your connections to SQL Server might fail.

In This Topic

- **To enable or disable a server network protocol using:**

  SQL Server Configuration Manager

  PowerShell

Using SQL Server Configuration Manager

To enable a server network protocol

1. In SQL Server Configuration Manager, in the console pane, expand **SQL Server Network Configuration**.
2. In the console pane, click **Protocols for** <*instance name*>.
3. In the details pane, right-click the protocol you want to change, and then click **Enable** or **Disable**.
4. In the console pane, click **SQL Server Services**.
5. In the details pane, right-click **SQL Server (**<*instance name*>**)**, and then click **Restart**, to stop and restart the SQL Server service.

Using SQL Server PowerShell

To Enable a Server Network Protocol Using PowerShell

1. Using administrator permissions open a command prompt.
2. Start Windows PowerShell from the taskbar, or click Start, then All Programs, then Accessories, then Windows PowerShell, then Windows PowerShell.
3. Import the **sqlps** module by entering **Import-Module “sqlps”**
4. Execute the following statements to enable both the TCP and named pipes protocols. Replace `<computer_name>` with the name of the computer that is running SQL Server. If you are configuring a named

instance, replace `MSSQLSERVER` with the instance name.

To disable protocols, set the `IsEnabled` properties to `$false`.

```
$smo = 'Microsoft.SqlServer.Management.Smo.'
$wmi = new-object ($smo + 'Wmi.ManagedComputer').

# List the object properties, including the instance names.
$Wmi

# Enable the TCP protocol on the default instance.
$uri = "ManagedComputer[@Name='<computer_name>']/
ServerInstance[@Name='MSSQLSERVER']/ServerProtocol[@Name='Tcp']"
$Tcp = $wmi.GetSmoObject($uri)
$Tcp.IsEnabled = $true
$Tcp.Alter()
$Tcp

# Enable the named pipes protocol for the default instance.
$uri = "ManagedComputer[@Name='<computer_name>']/
ServerInstance[@Name='MSSQLSERVER']/ServerProtocol[@Name='Np']"
$Np = $wmi.GetSmoObject($uri)
$Np.IsEnabled = $true
$Np.Alter()
$Np
```

To configure the protocols for the local computer

- When the script is run locally and configures the local computer, SQL Server PowerShell can make the script more flexible by dynamically determining the local computer name. To retrieve the local computer name, replace the line setting the `$uri` variable with the following line.

```
$uri = "ManagedComputer[@Name='" + (get-item env:\computername).Value +
"']/ServerInstance[@Name='MSSQLSERVER']/ServerProtocol[@Name='Tcp']"
```

To restart the Database Engine by using SQL Server PowerShell

- After you enable or disable protocols, you must stop and restart the Database Engine for the change to take effect. Execute the following statements to stop and start the default instance by using SQL Server PowerShell. To stop and start a named instance replace `'MSSQLSERVER'` with `'MSSQL$<instance_name>'`.

```
# Get a reference to the ManagedComputer class.
CD SQLSERVER:\SQL\<computer_name>
$Wmi = (get-item .).ManagedComputer
# Get a reference to the default instance of the Database Engine.
$DfltInstance = $Wmi.Services['MSSQLSERVER']
# Display the state of the service.
$DfltInstance
# Stop the service.
$DfltInstance.Stop();
# Wait until the service has time to stop.
# Refresh the cache.
$DfltInstance.Refresh();
# Display the state of the service.
$DfltInstance
# Start the service again.
$DfltInstance.Start();
# Wait until the service has time to start.
# Refresh the cache and display the state of the service.
$DfltInstance.Refresh(); $DfltInstance
```

Configure a Server to Listen on a Specific TCP Port

3/29/2017 • 2 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to configure an instance of the SQL Server Database Engine to listen on a specific fixed port by using the SQL Server Configuration Manager. If enabled, the default instance of the SQL Server Database Engine listens on TCP port 1433. Named instances of the Database Engine and SQL Server Compact are configured for dynamic ports. This means they select an available port when the SQL Server service is started. When you are connecting to a named instance through a firewall, configure the Database Engine to listen on a specific port, so that the appropriate port can be opened in the firewall.

For more information about the default Windows firewall settings, and a description of the TCP ports that affect the Database Engine, Analysis Services, Reporting Services, and Integration Services, see Configure the Windows Firewall to Allow SQL Server Access.

TIP

When selecting a port number, consult http://www.iana.org/assignments/port-numbers for a list of port numbers that are assigned to specific applications. Select an unassigned port number. For more information, see The default dynamic port range for TCP/IP has changed in Windows Vista and in Windows Server 2008.

WARNING

The Database Engine begins listening on a new port when restarted. However the SQL Server Browser service monitors the registry and reports the new port number as soon as the configuration is changed, even though the Database Engine might not be using it. Restart the Database Engine to ensure consistency and avoid connection failures.

In This Topic

- **To configure a server to listen on a specific TCP port, using:**

  SQL Server Configuration Manager

Using SQL Server Configuration Manager

To assign a TCP/IP port number to the SQL Server Database Engine

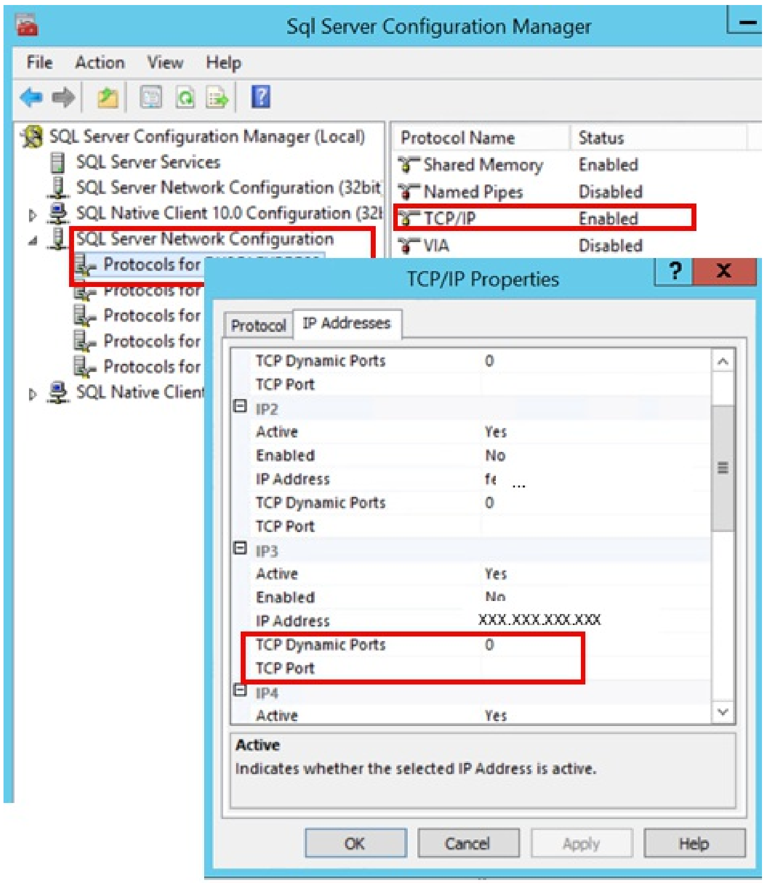
1. In SQL Server Configuration Manager, in the console pane, expand **SQL Server Network Configuration**, expand **Protocols for <instance name>**, and then double-click **TCP/IP**.

   NOTE

   If you are having trouble opening SQL Server Configuration Manager, see SQL Server Configuration Manager.

2. In the **TCP/IP Properties** dialog box, on the **IP Addresses** tab, several IP addresses appear in the format **IP1**, **IP2**, up to **IPAll**. One of these is for the IP address of the loopback adapter, 127.0.0.1. Additional IP addresses appear for each IP Address on the computer. (You will probably see both IP version 4 and IP version 6 addresses.) Right-click each address, and then click **Properties** to identify the IP address that you want to configure.

3. If the **TCP Dynamic Ports** dialog box contains **0**, indicating the Database Engine is listening on dynamic ports, delete the 0.


Sql Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit
SQL Native Client 10.0 Configuration (32
SQL Server Network Configuration
Protocols for
Protocols for
Protocols for
Protocols for
Protocols for
SQL Native Client
TCP/IP Properties
Protocol
IP Addresses
Active
Indicates whether the selected IP Address is active.
OK
Cancel
Apply
Help

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Disabled
TCP/IP	Enabled
VIA	Disabled

TCP Dynamic Ports	0
TCP Port	
IP2	
Active	Yes
Enabled	No
IP Address	fe ...
TCP Dynamic Ports	0
TCP Port	
IP3	
Active	Yes
Enabled	No
IP Address	XXX.XXX.XXX.XXX
TCP Dynamic Ports	0
TCP Port	
IP4	
Active	Yes

4. In the **IP*n* Properties** area box, in the **TCP Port** box, type the port number you want this IP address to listen on, and then click **OK**.

5. In the console pane, click **SQL Server Services**.

6. In the details pane, right-click **SQL Server (**<instance name>**)** and then click **Restart**, to stop and restart SQL Server.

   After you have configured SQL Server to listen on a specific port, there are three ways to connect to a specific port with a client application:

- Run the SQL Server Browser service on the server to connect to the Database Engine instance by name.
- Create an alias on the client, specifying the port number.
- Program the client to connect using a custom connection string.

See Also

Create or Delete a Server Alias for Use by a Client (SQL Server Configuration Manager)
SQL Server Browser Service

Configure a Server to Listen on an Alternate Pipe

3/29/2017 • 1 min to read • Edit Online

This topic describes how to configure a server to listen on an alternate pipe in SQL Server 2016 by using SQL Server Configuration Manager. By default, the default instance of SQL Server Database Engine listens on named pipe \\.\pipe\sql\query. Named instances of SQL Server Database Engine and SQL Server Compact listen on other pipes.

There are three ways to connect to a specific named pipe with a client application:

- Run the SQL Server Browser service on the server.
- Create an alias on the client, specifying the named pipe.
- Program the client to connect using a custom connection string.

Using SQL Server Configuration Manager

To configure the named pipe used by the SQL Server Database Engine

1. In SQL Server Configuration Manager, in the console pane, expand **SQL Server Network Configuration**, and then click expand **Protocols for** <*instance name*>.
2. In the details pane, right-click **Named Pipes**, and then click **Properties**.
3. On the **Protocol** tab, in the **Pipe Name** box, type the pipe you want the Database Engine to listen on, and then click **OK**.
4. In the console pane, click **SQL Server Services**.
5. In the details pane, right-click **SQL Server (**<instance name>**)** and then click **Restart**, to stop and restart SQL Server.

   When SQL Server is listening on an alternate pipe, there are three ways to connect to a specific named pipe with a client application:

- Run the SQL Server Browser service on the server.
- Create an alias on the client, specifying the named pipe.
- Program the client to connect using a custom connection string.

See Also

Create or Delete a Server Alias for Use by a Client (SQL Server Configuration Manager)
Server Network Configuration

Enable Encrypted Connections to the Database Engine

3/29/2017 • 3 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2016) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to enable encrypted connections for an instance of the SQL Server Database Engine by specifying a certificate for the Database Engine using SQL Server Configuration Manager. The server computer must have a certificate provisioned, and the client machine must be set up to trust the certificate's root authority. Provisioning is the process of installing a certificate by importing it into Windows.

The certificate must be issued for **Server Authentication**. The name of the certificate must be the fully qualified domain name (FQDN) of the computer.

Certificates are stored locally for the users on the computer. To install a certificate for use by SQL Server, you must be running SQL Server Configuration Manager under the same user account as the SQL Server service unless the service is running as LocalSystem, NetworkService, or LocalService, in which case you may use an administrative account.

The client must be able to verify the ownership of the certificate used by the server. If the client has the public key certificate of the certification authority that signed the server certificate, no further configuration is necessary. Microsoft Windows includes the public key certificates of many certification authorities. If the server certificate was signed by a public or private certification authority for which the client does not have the public key certificate, you must install the public key certificate of the certification authority that signed the server certificate.

NOTE

To use encryption with a failover cluster, you must install the server certificate with the fully qualified DNS name of the virtual server on all nodes in the failover cluster. For example, if you have a two-node cluster, with nodes named test1.<*your company*>.com and test2.<*your company*>.com, and you have a virtual server named virtsql, you need to install a certificate for virtsql.<*your company*>.com on both nodes. You can set the value of the **ForceEncryption**option to **Yes**.

NOTE

When creating encrypted connections for an Azure Search indexer to SQL Server on an Azure VM, see Configure a connection from an Azure Search indexer to SQL Server on an Azure VM.

To provision (install) a certificate on the server

1. On the **Start** menu, click **Run**, and in the **Open** box, type **MMC** and click **OK**.
2. In the MMC console, on the **File** menu, click **Add/Remove Snap-in**.
3. In the **Add/Remove Snap-in** dialog box, click **Add**.
4. In the **Add Standalone Snap-in** dialog box, click **Certificates**, click **Add**.
5. In the **Certificates snap-in** dialog box, click **Computer account**, and then click **Finish**.
6. In the **Add Standalone Snap-in** dialog box, click **Close.**

7. In the **Add/Remove Snap-in** dialog box, click **OK**.

8. In the **Certificates** snap-in, expand **Certificates**, expand **Personal**, and then right-click **Certificates**, point to **All Tasks**, and then click **Import**.

9. Right-click the imported certificate, point to **All Tasks**, and then click **Manage Private Keys**. In the **Security** dialog box, add read permission for the user account used by the SQL Server service account.

10. Complete the **Certificate Import Wizard**, to add a certificate to the computer, and close the MMC console. For more information about adding a certificate to a computer, see your Windows documentation.

To export the server certificate

1. From the **Certificates** snap-in, locate the certificate in the **Certificates** / **Personal** folder, right-click the **Certificate**, point to **All Tasks**, and then click **Export**.

2. Complete the **Certificate Export Wizard**, storing the certificate file in a convenient location.

To configure the server to accept encrypted connections

1. In **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for** <*server instance*>, and then select**Properties**.

2. In the **Protocols for**<*instance name*> **Properties** dialog box, on the **Certificate** tab, select the desired certificate from the drop down for the **Certificate** box, and then click **OK**.

3. On the **Flags** tab, in the **ForceEncryption** box, select **Yes**, and then click **OK** to close the dialog box.

4. Restart the SQL Server service.

To configure the client to request encrypted connections

1. Copy either the original certificate or the exported certificate file to the client computer.

2. On the client computer, use the **Certificates** snap-in to install either the root certificate or the exported certificate file.

3. In the console pane, right-click **SQL Server Native Client Configuration**, and then click **Properties**.

4. On the **Flags** page, in the **Force protocol encryption** box, click **Yes**.

To encrypt a connection from SQL Server Management Studio

1. On the Object Explorer toolbar, click **Connect**, and then click **Database Engine**.

2. In the **Connect to Server** dialog box, complete the connection information, and then click **Options**.

3. On the **Connection Properties** tab, click **Encrypt connection**.

See Also

TLS 1.2 support for Microsoft SQL Server

Connect to SQL Server Through a Proxy Server (SQL Server Configuration Manager)

3/29/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This topic describes how to connect to SQL Server through a proxy server in SQL Server 2016 by using SQL Server Configuration Manager. To listen remotely by way of Remote WinSock (RWS), define the local address table (LAT) for the proxy server so that the listening node address is outside the range of LAT entries.

Using SQL Server Configuration Manager

To enable connections to SQL Server through Microsoft Proxy Server

1. Follow the steps in Configure a Server to Listen on a Specific TCP Port (SQL Server Configuration Manager) to determine which TCP/IP ports are used by the Database Engine, or to configure the Database Engine to use a desired port.

2. In your proxy server define the local address table (LAT) for the proxy server so that the listening node address is outside the range of LAT entries. For more information, see your proxy server documentation.

NOTE

This topic applies to on-premises SQL Server. For connection issues related to SQL Database, see Troubleshoot connection issues to Azure SQL Database.

Configure a Windows Firewall for Database Engine Access

3/24/2017 • 4 min to read • Edit Online

This topic describes how to configure a Windows firewall for Database Engine access in SQL Server 2016 by using SQL Server Configuration Manager. Firewall systems help prevent unauthorized access to computer resources. To access an instance of the SQL Server Database Engine through a firewall, you must configure the firewall on the computer running SQL Server to allow access.

For more information about the default Windows firewall settings, and a description of the TCP ports that affect the Database Engine, Analysis Services, Reporting Services, and Integration Services, see Configure the Windows Firewall to Allow SQL Server Access. There are many firewall systems available. For information specific to your system, see the firewall documentation.

The principal steps to allow access are:

1. Configure the Database Engine to use a specific TCP/IP port. The default instance of the Database Engine uses port 1433, but that can be changed. The port used by the Database Engine is listed in the SQL Server error log. Instances of SQL Server Express, SQL Server Compact, and named instances of the Database Engine use dynamic ports. To configure these instances to use a specific port, see Configure a Server to Listen on a Specific TCP Port (SQL Server Configuration Manager).

2. Configure the firewall to allow access to that port for authorized users or computers.

> **NOTE**
> The SQL Server Browser service lets users connect to instances of the Database Engine that are not listening on port 1433, without knowing the port number. To use SQL Server Browser, you must open UDP port 1434. To promote the most secure environment, leave the SQL Server Browser service stopped, and configure clients to connect using the port number.

> **NOTE**
> By default, Microsoft Windows enables the Windows Firewall, which closes port 1433 to prevent Internet computers from connecting to a default instance of SQL Server on your computer. Connections to the default instance using TCP/IP are not possible unless you reopen port 1433. The basic steps to configure the Windows firewall are provided in the following procedures. For more information, see the Windows documentation.

As an alternative to configuring SQL Server to listen on a fixed port and opening the port, you can list the SQL Server executable (Sqlservr.exe) as an exception to the blocked programs. Use this method when you want to continue to use dynamic ports. Only one instance of SQL Server can be accessed in this way.

In This Topic

- **Before you begin:**

  Security

- **To configure a Widows Firewall for Database Engine access, using:**

  SQL Server Configuration Manager

Before You Begin

Security

Opening ports in your firewall can leave your server exposed to malicious attacks. Make sure that you understand firewall systems before you open ports. For more information, see Security Considerations for a SQL Server Installation

Using SQL Server Configuration Manager

Applies to Windows Vista, Windows 7, and Windows Server 2008

The following procedures configure the Windows Firewall by using the Windows Firewall with Advanced Security Microsoft Management Console (MMC) snap-in. The Windows Firewall with Advanced Security only configures the current profile. For more information about the Windows Firewall with Advanced Security, see Configure the Windows Firewall to Allow SQL Server Access

To open a port in the Windows firewall for TCP access

1. On the **Start** menu, click **Run**, type **WF.msc**, and then click **OK**.
2. In the **Windows Firewall with Advanced Security**, in the left pane, right-click **Inbound Rules**, and then click **New Rule** in the action pane.
3. In the **Rule Type** dialog box, select **Port**, and then click **Next**.
4. In the **Protocol and Ports** dialog box, select **TCP**. Select **Specific local ports**, and then type the port number of the instance of the Database Engine, such as **1433** for the default instance. Click **Next**.
5. In the **Action** dialog box, select **Allow the connection**, and then click **Next**.
6. In the **Profile** dialog box, select any profiles that describe the computer connection environment when you want to connect to the Database Engine, and then click **Next**.
7. In the **Name** dialog box, type a name and description for this rule, and then click **Finish**.

To open access to SQL Server when using dynamic ports

1. On the **Start** menu, click **Run**, type **WF.msc**, and then click **OK**.
2. In the **Windows Firewall with Advanced Security**, in the left pane, right-click **Inbound Rules**, and then click **New Rule** in the action pane.
3. In the **Rule Type** dialog box, select **Program**, and then click **Next**.
4. In the **Program** dialog box, select **This program path**. Click **Browse**, and navigate to the instance of SQL Server that you want to access through the firewall, and then click **Open**. By default, SQL Server is at **C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Binn\Sqlservr.exe**. Click **Next**.
5. In the **Action** dialog box, select **Allow the connection**, and then click **Next**.
6. In the **Profile** dialog box, select any profiles that describe the computer connection environment when you want to connect to the Database Engine, and then click **Next**.
7. In the **Name** dialog box, type a name and description for this rule, and then click **Finish**.

See Also

How to: Configure Firewall Settings (Azure SQL Database)

Hide an Instance of SQL Server Database Engine

3/24/2017 • 2 min to read • Edit Online

This topic describes how to hide an instance of the Database Engine in SQL Server 2016 by using SQL Server Configuration Manager. SQL Server uses the SQL Server Browser service to enumerate instances of the Database Engine installed on the computer. This enables client applications to browse for a server, and helps clients distinguish between multiple instances of the Database Engine on the same computer. You can use the following procedure to prevent the SQL Server Browser service from exposing an instance of the Database Engine to client computers that try to locate the instance by using the **Browse** button.

Using SQL Server Configuration Manager

To hide an instance of the SQL Server Database Engine

1. In **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for** <*server instance*>, and then select **Properties**.

2. On the **Flags** tab, in the **HideInstance** box, select **Yes**, and then click **OK** to close the dialog box. The change takes effect immediately for new connections.

Remarks

If you hide a named instance, you will need to provide the port number in the connection string to connect to the hidden instance, even if the browser service is running. We recommend that you use a static port instead of a dynamic port for the named hidden instance.
For more information, see Configure a Server to Listen on a Specific TCP Port (SQL Server Configuration Manager).

Clustering

If you hide a clustered named instance, cluster service may not be able to connect to SQL Server. This will cause the cluster instance's **IsAlive** check to fail and SQL Server will go offline. We recommend that you create an alias in all the nodes of the clustered instance to reflect the static port that you configured for the instance.
For more information, see Create or Delete a Server Alias for Use by a Client (SQL Server Configuration Manager).

If you hide a clustered named instance, cluster service may not be able to connect to SQL Server if the **LastConnect** registry key (**HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer\Client\SNI11.0\LastConnect**) has a different port than the port that SQL Server is listening on. If the cluster service is unable to make a connection to the SQL Server, you might see an error similar to the following:
Event ID: 1001: Event Name: Failover clustering resource deadlock.

See Also

Server Network Configuration
Description of SQL Virtual Server client connections
How to assign a static port to a SQL Server named instance - and avoid a common pitfall

Configure the Database Engine to Listen on Multiple TCP Ports

3/24/2017 • 3 min to read • Edit Online

This topic describes how to configure the Database Engine to listen on multiple TCP ports in SQL Server 2016 by using SQL Server Configuration Manager. When TCP/IP is enabled for SQL Server, the Database Engine will listen for incoming connections on a connection point consisting of an IP address and TCP port number.The following procedures create a tabular data stream (TDS) endpoint, so that SQL Server will listen on an additional TCP port.

Possible reasons to create a second TDS endpoint include:

- Increase security by configuring the firewall to restrict access to the default endpoint to local client computers on a specific subnet. Maintain Internet access to SQL Server for your support team by creating a new endpoint that the firewall exposes to the Internet, and restricting connection rights to this endpoint to your server support team.
- Affinitizing connections to specific processors when using Non-Uniform Memory Access (NUMA).

  Configuring a TDS endpoint consists of the following steps, which can be done in any order:
- Create the TDS endpoint for the TCP port, and restore access to the default endpoint if appropriate.
- Grant access to the endpoint to the desired server principals.
- Specify the TCP port number for the selected IP address.

  For more information about the default Windows firewall settings, and a description of the TCP ports that affect the Database Engine, Analysis Services, Reporting Services, and Integration Services, see Configure the Windows Firewall to Allow SQL Server Access.

To create a TDS endpoint

- Issue the following statement to create an endpoint named **CustomConnection** for port 1500 for all available TCP addresses on the server.

```
USE master;
GO
CREATE ENDPOINT [CustomConnection]
STATE = STARTED
AS TCP
   (LISTENER_PORT = 1500, LISTENER_IP =ALL)
FOR TSQL() ;
GO
```

  When you create a new Transact-SQL endpoint, connect permissions for **public** are revoked for the default TDS endpoint. If access to the **public** group is needed for the default endpoint, reapply this permission by using the `GRANT CONNECT ON ENDPOINT::[TSQL Default TCP] to [public];` statement.

To grant access to the endpoint

- Issue the following statement to grant access to the **CustomConnection** endpoint to the SQLSupport group in the corp domain.

```
GRANT CONNECT ON ENDPOINT::[CustomConnection] to [corp\SQLSupport] ;
GO
```

To configure the SQL Server Database Engine to listen on an additional TCP port

1. In SQL Server Configuration Manager, expand **SQL Server Network Configuration**, and then click **Protocols for**.
2. Expand **Protocols for**, and then click **TCP/IP**.
3. In the right pane, right-click each disabled IP address that you want to enable, and then click **Enable**.
4. Right-click **IPAll**, and then click **Properties**.
5. In the **TCP Port** box, type the ports that you want the Database Engine to listen on, separated by commas. In our example, if the default port 1433 is listed, type **,1500** so the box reads **1433,1500**, and then click **OK**.

   > **NOTE**
   >
   > If you are not enabling the port on all IP addresses, configure the additional port in the property box for only for the desired address. Then, in the console pane, right-click **TCP/IP**, click **Properties**, and in the **Listen All** box, select **No**.

6. In the left pane, click **SQL Server Services**.
7. In the right pane, right-click **SQL Server**, and then click **Restart**.

   When the Database Engine restarts, the Error log will list the ports on which SQL Server is listening.

To connect to the new endpoint

- Issue the following statement to connect to the **CustomConnection** endpoint of the default instance of SQL Server on the server named ACCT, using a trusted connection, and assuming the user is a member of the [corp\SQLSupport] group.

  ```
  sqlcmd -SACCT,1500
  ```

See Also

CREATE ENDPOINT (Transact-SQL)
DROP ENDPOINT (Transact-SQL)
GRANT Endpoint Permissions (Transact-SQL)
Map TCP IP Ports to NUMA Nodes (SQL Server)

Determine Whether the Database Engine Is Installed and Started

3/24/2017 • 1 min to read • Edit Online

A successful installation of the SQL Server Database Engine installs files to the file system, creates entries in the registry, and installs several tools. This topic describes how to determine whether the Database Engine is installed and started in SQL Server 2016 by using SQL Server Configuration Manager.

Using SQL Server Configuration Manager

How to view and start the Database Engine by using SQL Server Configuration Manager

1. Click **Start**, point to **All Programs**, point to **Microsoft SQL Server**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.

   If you do not have these entries on the **Start** menu, SQL Server is not correctly installed. Run Setup to install the SQL Server Database Engine.

2. In **SQL Server Configuration Manager**, on the left pane, click **SQL Server Services**. The right pane lists several services that are related to SQL Server. If the Database Engine is installed, the Database Engine service is listed as **SQL Server (MSSQLSERVER)** if it is the default instance; or **SQL Server (**<*instance_name*>**)**, if the Database Engine is installed as a named instance. Unless the instance name is changed, SQL Server Express installs as a named instance with the name **SQLEXPRESS**. A green triangle icon indicates that the Database Engine is running. A red square icon indicates that the Database Engine is stopped.

3. To start the Database Engine, in the right pane, right-click the Database Engine, and then click **Start**.

NOTE

During setup, the user can select a location in which to install the program files and the database files. If the user accepts the default location, the files are installed to C:\Program Files\Microsoft SQL Server\130\ and C:\Program Files\Microsoft SQL Server\MSSQL.*x*, where *x* is a number.

Connect to the Database Engine Using Extended Protection

3/24/2017 • 6 min to read • Edit Online

SQL Server supports **Extended Protection** beginning with SQL Server 2008 R2. **Extended Protection for Authentication** is a feature of the network components implemented by the operating system. **Extended Protection** is supported in Windows 7 and Windows Server 2008 R2. **Extended Protection** is included in service packs for older Microsoft operating systems. SQL Server is more secure when connections are made using **Extended Protection**.

IMPORTANT

Windows does not enable **Extended Protection** by default. For information about how to enable **Extended Protection** in Windows, see Extended Protection for Authentication.

Description of Extended Protection

Extended Protection uses service binding and channel binding to help prevent an authentication relay attack. In an authentication relay attack, a client that can perform NTLM authentication (for example, Windows Explorer, Microsoft Outlook, a .NET SqlClient application, etc.), connects to an attacker (for example, a malicious CIFS file server). The attacker uses the client's credentials to masquerade as the client and authenticate to a service (for example, an instance of the Database Engine service).

There are two variations of this attack:

- In a luring attack, the client is lured to voluntarily connect to the attacker.
- In a spoofing attack, the client intends to connect to a valid service, but is unaware that one or both of DNS and IP routing are poisoned to redirect the connection to the attacker instead.

  SQL Server supports service binding and channel binding to help reduce these attacks on SQL Server instances.

Service Binding

Service binding addresses luring attacks by requiring a client to send a signed service principal name (SPN) of the SQL Server service that the client intends to connect to. As part of the authentication response, the service validates that the SPN received in the packet matches its own SPN. If a client is lured to connect to an attacker, the client will include the signed SPN of the attacker. The attacker cannot relay the packet to authenticate to the real SQL Server service as the client, because it would include the SPN of the attacker. Service binding incurs a one-time, negligible cost, but it does not address spoofing attacks. Service Binding occurs when a client application does not use encryption to connect to the SQL Server.

Channel Binding

Channel binding establishes a secure channel (Schannel) between a client and an instance of the SQL Server service. The service verifies the authenticity of the client by comparing the client's channel binding token (CBT) specific to that channel, with its own CBT. Channel binding addresses both luring and spoofing attacks. However, it incurs a larger runtime cost, because it requires Transport Layer Security (TLS) encryption of all the session traffic. Channel Binding occurs when a client application uses encryption to connect to the SQL Server, regardless of whether encryption is enforced by the client or by the server.

WARNING

SQL Server and Microsoft data providers for SQL Server support TLS 1.0 and SSL 3.0. If you enforce a different protocol (such as TLS 1.1 or TLS 1.2) by making changes in the operating system SChannel layer, your connections to SQL Server might fail.

Operating System Support

The following links provide more information about how Windows supports **Extended Protection**:

- Integrated Windows Authentication with Extended Protection
- Microsoft Security Advisory (973811), Extended Protection for Authentication

Settings

There are three SQL Server connection settings that affect service binding and channel binding. The settings can be configured by using the SQL Server Configuration Manager, or by using WMI, and can by viewed by using the **Server Protocol Settings** facet of Policy Based Management.

- **Force Encryption**

  Possible values are **On** and **Off**. To use channel binding, **Force Encryption** must be set to **On**, and all clients will be forced to encrypt. If it is **Off**, only service binding is guaranteed. **Force Encryption** is on the **Protocols for MSSQLSERVER Properties (Flags Tab)** in SQL Server Configuration Manager.

- **Extended Protection**

  Possible values are **Off**, **Allowed**, and **Required**. The **Extended Protection** variable lets users configure the **Extended Protection** level for each SQL Server instance. **Extended Protection** is on the **Protocols for MSSQLSERVER Properties (Advanced Tab)** in SQL Server Configuration Manager.

  - When set to **Off**, **Extended Protection** is disabled. The instance of SQL Server will accept connections from any client regardless of whether the client is protected or not. **Off** is compatible with older and unpatched operating systems, but is less secure. Use this setting when you know that the client operating systems do not support extended protection.

  - When set to **Allowed**, **Extended Protection** is required for connections from operating systems that support **Extended Protection**. **Extended Protection** is ignored for connections from operating systems that do not support **Extended Protection**. Connections from unprotected client applications that are running on protected client operating systems are rejected. This setting is more secure than **Off**, but is not the most secure setting. Use this setting in mixed environments, where some operating systems support **Extended Protection** and some do not.

  - When set to **Required**, only connections from protected applications on protected operating systems are accepted. This setting is the most secure but connections from operating systems or applications that do not support **Extended Protection** will not be able to connect to SQL Server.

- **Accepted NTLM SPNs**

  The **Accepted NTLM SPNs** variable is needed when a server is known by more than one SPN. When a client attempts to connect to the server by using a valid SPN that the server does not know, service binding will fail. To avoid this problem, users can specify several SPNs that represent the server by using **Accepted NTLM SPNs**. **Accepted NTLM SPNs** is a series of SPNs separated my semicolons. For example, to allow the SPNs **MSSQLSvc/ HostName1.Contoso.com** and **MSSQLSvc/ HostName2.Contoso.com**, type **MSSQLSvc/HostName1.Contoso.com;MSSQLSvc/HostName2.Contoso.com** in the **Accepted NTLM SPNs** box. The variable has a maximum length of 2,048 characters. **Accepted NTLM SPNs** is on the **Protocols for MSSQLSERVER Properties (Advanced Tab)** in SQL Server Configuration Manager.

Enabling Extended Protection for the Database Engine

To use **Extended Protection**, both the server and the client must have an operating system on that supports **Extended Protection**, and **Extended Protection** must be enabled on the operating system. For more information about how to enable **Extended Protection** for the operating system, see Extended Protection for Authentication.

SQL Server supports **Extended Protection** beginning with SQL Server 2008 R2. **Extended Protection** for some earlier versions of SQL Server will be made available in future updates. After enabling **Extended Protection** on the server computer, use the following steps to enable **Extended Protection**:

1. On the **Start** menu, choose **All Programs**, point to **Microsoft SQL Server** and then click **SQL Server Configuration Manager**.

2. Expand **SQL Server Network Configuration**, and then right-click **Protocols for** <InstanceName>, and then click **Properties**.

3. For both channel binding and service binding, on the **Advanced** tab, set **Extended Protection** to the appropriate setting.

4. Optionally, when a server is known by more than one SPN, on the **Advanced** tab configure the **Accepted NTLM SPNs** field as described in the "Settings" section.

5. For channel binding, on the **Flags** tab, set **Force Encryption** to **On**.

6. Restart the Database Engine service.

Configuring Other SQL Server Components

For more information about how to configure Reporting Services, see Extended Protection for Authentication with Reporting Services.

When using IIS to access Analysis Services data using an HTTP or HTTPs connection, Analysis Services can take advantage of Extended Protection provided by IIS. For more information about how to configure IIS to use Extended Protection, see Configure Extended Protection in IIS 7.5.

See Also

Server Network Configuration
Client Network Configuration
Extended Protection for Authentication Overview
Integrated Windows Authentication with Extended Protection

Register a Service Principal Name for Kerberos Connections

3/24/2017 • 8 min to read • Edit Online

To use Kerberos authentication with SQL Server requires both the following conditions to be true:

- The client and server computers must be part of the same Windows domain, or in trusted domains.
- A Service Principal Name (SPN) must be registered with Active Directory, which assumes the role of the Key Distribution Center in a Windows domain. The SPN, after it is registered, maps to the Windows account that started the SQL Server instance service. If the SPN registration has not been performed or fails, the Windows security layer cannot determine the account associated with the SPN, and Kerberos authentication will not be used.

  NOTE

  If the server cannot automatically register the SPN, the SPN must be registered manually. See Manual SPN Registration.

  You can verify that a connection is using Kerberos by querying the sys.dm_exec_connections dynamic management view. Run the following query and check the value of the auth_scheme column, which will be "KERBEROS" if Kerberos is enabled.

```
SELECT auth_scheme FROM sys.dm_exec_connections WHERE session_id = @@spid ;
```

TIP

Microsoft Kerberos Configuration Manager for SQL Server is a diagnostic tool that helps troubleshoot Kerberos related connectivity issues with SQL Server. For more information, see Microsoft Kerberos Configuration Manager for SQL Server.

The Role of the SPN in Authentication

When an application opens a connection and uses Windows Authentication, SQL Server Native Client passes the SQL Server computer name, instance name and, optionally, an SPN. If the connection passes an SPN it is used without any changes.

If the connection does not pass an SPN, a default SPN is constructed based on the protocol used, server name, and the instance name.

In both of the preceding scenarios, the SPN is sent to the Key Distribution Center to obtain a security token for authenticating the connection. If a security token cannot be obtained, authentication uses NTLM.

A service principal name (SPN) is the name by which a client uniquely identifies an instance of a service. The Kerberos authentication service can use an SPN to authenticate a service. When a client wants to connect to a service, it locates an instance of the service, composes an SPN for that instance, connects to the service, and presents the SPN for the service to authenticate.

NOTE

The information that is provided in this topic also applies to SQL Server configurations that use clustering.

Windows Authentication is the preferred method for users to authenticate to SQL Server. Clients that use Windows Authentication are authenticated by either using NTLM or Kerberos. In an Active Directory environment, Kerberos authentication is always attempted first. Kerberos authentication is not available for SQL Server 2005 clients using named pipes.

Permissions

When the Database Engine service starts, it attempts to register the Service Principal Name (SPN). If the account starting SQL Server doesn't have permission to register a SPN in Active Directory Domain Services, this call will fail and a warning message will be logged in the Application event log as well as the SQL Server error log. To register the SPN, the Database Engine must be running under a built-in account, such as Local System (not recommended), or NETWORK SERVICE, or an account that has permission to register an SPN, such as a domain administrator account. When SQL Server is running on the Windows 7 or Windows Server 2008 R2 operating system, you can run SQL Server using a virtual account or a managed service account (MSA). Both virtual accounts and MSA's can register an SPN. If SQL Server is not running under one of these accounts, the SPN is not registered at startup and the domain administrator must register the SPN manually.

NOTE

When the Windows domain is configured to run at less than the Windows Server 2008 R2 Windows Server 2008 R2 functional level, then the Managed Service Account will not have the necessary permissions to register the SPNs for the SQL Server Database Engine service. If Kerberos authentication is required, the Domain Administrator should manually register the SQL Server SPNs on the Managed Service Account.

The KB article, How to use Kerberos authentication in SQL Server, contains information about how to grant read or write permission to an SPN for an account that is not a Domain Administrator.

Additional information is available at How to Implement Kerberos Constrained Delegation with SQL Server 2008

SPN Formats

Beginning with SQL Server 2008, the SPN format is changed in order to support Kerberos authentication on TCP/IP, named pipes, and shared memory. The supported SPN formats for named and default instances are as follows.

Named instance

- *MSSQLSvc/FQDN*:[*port*|*instancename*], where:
  - *MSSQLSvc* is the service that is being registered.
  - *FQDN* is the fully qualified domain name of the server.
  - *port* is the TCP port number.
  - *instancename* is the name of the SQL Server instance.

  Default instance

- *MSSQLSvc/FQDN:port*|*MSSQLSvc/FQDN*, where:
  - *MSSQLSvc* is the service that is being registered.

- *FQDN* is the fully qualified domain name of the server.
- *port* is the TCP port number.

The new SPN format does not require a port number. This means that a multiple-port server or a protocol that does not use port numbers can use Kerberos authentication.

> **NOTE**
> In the case of a TCP/IP connection, where the TCP port is included in the SPN, SQL Server must enable the TCP protocol for a user to connect by using Kerberos authentication.

MSSQLSvc/*fqdn:port*	The provider-generated, default SPN when TCP is used. *port* is a TCP port number.
MSSQLSvc/*fqdn*	The provider-generated, default SPN for a default instance when a protocol other than TCP is used. *fqdn* is a fully-qualified domain name.
MSSQLSvc/*fqdn/InstanceName*	The provider-generated, default SPN for a named instance when a protocol other than TCP is used. *InstanceName* is the name of an instance of SQL Server.

Automatic SPN Registration

When an instance of the SQL Server Database Engine starts, SQL Server tries to register the SPN for the SQL Server service. When the instance is stopped, SQL Server tries to unregister the SPN. For a TCP/IP connection the SPN is registered in the format *MSSQLSvc/<FQDN>*:*<tcpport>*.Both named instances and the default instance are registered as *MSSQLSvc*, relying on the *<tcpport>* value to differentiate the instances.

For other connections that support Kerberos the SPN is registered in the format *MSSQLSvc/<FQDN>/<instancename>* for a named instance. The format for registering the default instance is *MSSQLSvc/<FQDN>*.

Manual intervention might be required to register or unregister the SPN if the service account lacks the permissions that are required for these actions.

Manual SPN Registration

To register the SPN manually, the administrator must use the Setspn.exe tool that is provided with the Microsoft Windows Server 2003 Support Tools. For more information, see the Windows Server 2003 Service Pack 1 Support Tools KB article.

Setspn.exe is a command line tool that enables you to read, modify, and delete the Service Principal Names (SPN) directory property. This tool also enables you to view the current SPNs, reset the account's default SPNs, and add or delete supplemental SPNs.

The following example illustrates the syntax used to register manually register an SPN for a TCP/IP connection.

```
setspn -A MSSQLSvc/myhost.redmond.microsoft.com:1433 accountname
```

Note If an SPN already exists, it must be deleted before it can be reregistered. You do this by using the `setspn` command together with the `-D` switch. The following examples illustrate how to manually register a new instance-based SPN. For a default instance, use:

```
setspn -A MSSQLSvc/myhost.redmond.microsoft.com accountname
```

For a named instance, use:

```
setspn -A MSSQLSvc/myhost.redmond.microsoft.com/instancename accountname
```

Client Connections

User-specified SPNs are supported in client drivers. However, if an SPN is not provided, it will be generated automatically based on the type of a client connection. For a TCP connection, an SPN in the format *MSSQLSvc/FQDN*:[*port*] is used for both the named and default instances.

For named pipes and shared memory connections, an SPN in the format *MSSQLSvc/FQDN*:*instancename* is used for a named instance and *MSSQLSvc/FQDN* is used for the default instance.

Using a service account as an SPN

Service accounts can be used as an SPN. They are specified through the connection attribute for the Kerberos authentication and take the following formats:

- **username@domain** or **domain\username** for a domain user account
- **machine$@domain** or **host\FQDN** for a computer domain account such as Local System or NETWORK SERVICES.

  To determine the authentication method of a connection, execute the following query.

```
SELECT net_transport, auth_scheme
FROM sys.dm_exec_connections
WHERE session_id = @@SPID;
```

Authentication Defaults

The following table describes the authentication defaults that are used based on SPN registration scenarios.

SCENARIO	AUTHENTICATION METHOD
The SPN maps to the correct domain account, virtual account, MSA, or built-in account. For example, Local System or NETWORK SERVICE.	Local connections use NTLM, remote connections use Kerberos.
The SPN is the correct domain account, virtual account, MSA, or built-in account.	Local connections use NTLM, remote connections use Kerberos.
The SPN maps to an incorrect domain account, virtual account, MSA, or built-in account	Authentication fails.
The SPN lookup fails or does not map to a correct domain account, virtual account, MSA, or built-in account, or is not a correct domain account, virtual account, MSA, or built-in account.	Local and remote connections use NTLM.

NOTE

'Correct' means that the account mapped by the registered SPN is the account that the SQL Server service is running under.

Comments

The Dedicated Administrator Connection (DAC) uses an instance name based SPN. Kerberos authentication can be used with a DAC if that SPN is registered successfully. As an alternative a user can specify the account name as an SPN.

If SPN registration fails during startup, this failure is recorded in the SQL Server error log, and startup continues.

If SPN de-registration fails during shutdown, this failure is recorded in the SQL Server error log, and shutdown continues.

See Also

Service Principal Name (SPN) Support in Client Connections
Service Principal Names (SPNs) in Client Connections (OLE DB)
Service Principal Names (SPNs) in Client Connections (ODBC)
SQL Server Native Client Features
Manage Kerberos Authentication Issues in a Reporting Services Environment

Client Network Configuration

3/24/2017 • 3 min to read • Edit Online

Client software enables client computers to connect to an instance of Microsoft SQL Server on a network. A "client" is a front-end application that uses the services provided by a server such as the SQL Server Database Engine. The computer that hosts this application is referred to as the *client computer*.

At the simplest level, a SQL Server client can reside on the same machine as an instance of SQL Server. Typically, however, a client connects to one or more remote servers over a network. The client/server architecture of SQL Server allows it to seamlessly manage multiple clients and servers on a network. The default client configurations suffice in most situations.

SQL Server clients can include applications of various types, such as:

- OLE DB consumers

  These applications use the SQL Server Native Client OLE DB provider to connect to an instance of SQL Server. The OLE DB provider mediates between SQL Server and client applications that consume SQL Server data as OLE DB rowsets. The **sqlcmd** command prompt utility and SQL Server Management Studio, are examples of OLE DB applications.

- ODBC applications

  These applications include client utilities installed with previous versions of SQL Server, such as the **osql** command prompt utility, as well as other applications that use the SQL Server Native Client ODBC driver to connect to an instance of SQL Server.

- DB-Library clients

  These applications include the SQL Server **isql** command prompt utility and clients written to DB-Library. SQL Server support for client applications using DB-Library is limited to Microsoft SQL Server 7.0 features.

NOTE

Although the SQL Server Database Engine still supports connections from existing applications using the DB-Library and Embedded SQL APIs, it does not include the files or documentation needed to do programming work on applications that use these APIs. A future version of the SQL Server Database Engine will drop support for connections from DB-Library or Embedded SQL applications. Do not use DB-Library or Embedded SQL to develop new applications. Remove any dependencies on either DB-Library or Embedded SQL when modifying existing applications. Instead of these APIs, use the SQLClient namespace or an API such as OLE DB or ODBC. SQL Server does not include the DB-Library DLL required to run these applications. To run DB-Library or Embedded SQL applications you must have available the DB-Library DLL from SQL Server version 6.5, SQL Server 7.0, or SQL Server 2000.

Regardless of the type of application, managing a client consists mainly of configuring its connection with the server components of SQL Server. Depending on the requirements of your site, client management can range from little more than entering the name of the server computer to building a library of custom configuration entries to accommodate a diverse multiserver environment.

The SQL Server Native Client DLL contains the network libraries and is installed by the setup program. The network protocols are not enabled during setup for new installations of SQL Server. Upgraded installations enable the previously enabled protocols. The underlying network protocols are installed as part of Windows Setup (or through Networks in Control Panel). The following tools are used to manage SQL Server clients:

- SQL Server Configuration Manager

  Both client and server network components are managed with SQL Server Configuration Manager, which combines the SQL Server Network Utility, SQL Server Client Network Utility, and Service Manager of previous versions. SQL Server Configuration Manager is a Microsoft Management Console (MMC) snap-in. It also appears as a node in the Windows Computer Manager snap-in. Individual network libraries can be enabled, disabled, configured, and prioritized using SQL Server Configuration Manager.

- Setup

  Run SQL Server setup to install the network components on a client computer. Individual network libraries can be enabled or disabled during setup when Setup is started from the command prompt.

- ODBC Data Source Administrator

  The ODBC Data Source Administrator lets you create and modify ODBC data sources on computers running the Microsoft Windows operating system.

In This Section

Configure Client Protocols

Create or Delete a Server Alias for Use by a Client (SQL Server Configuration Manager)

Logging In to SQL Server

Open the ODBC Data Source Administrator

Check the ODBC SQL Server Driver Version (Windows)

Related Content

Server Network Configuration

Manage the Database Engine Services

Configure Client Protocols

3/24/2017 • 2 min to read • Edit Online

This topic describes how to configure client protocols used by client applications in SQL Server 2016 by using SQL Server Configuration Manager. Microsoft SQL Server supports client communication with the TCP/IP network protocol and the named pipes protocol. The shared memory protocol is also available if the client is connecting to an instance of the Database Engine on the same computer. There are three common methods of selecting the protocol.

- Configure all client applications to use the same network protocol by setting the protocol order in SQL Server Configuration Manager.
- Configure a single client application to use a different network protocol by creating an alias. For more information, see Create or Delete a Server Alias for Use by a Client (SQL Server Configuration Manager).
- Some client applications, such as sqlcmd.exe, can specify the protocol as part of the connection string. For more information, see Connect to the Database Engine With sqlcmd.

Using SQL Server Configuration Manager

To enable or disable a client protocol

1. In SQL Server Configuration Manager, expand **SQL Server Native Client Configuration**, right-click **Client Protocols**, and then click **Properties**.
2. Click a protocol in the **Disabled Protocols** box, and then click **Enable**, to enable a protocol.
3. Click a protocol in the **Enabled Protocols** box, and then click **Disable**, to disable a protocol.

To change the default protocol or the protocol order for client computers

1. In SQL Server Configuration Manager, expand **SQL Server Native Client Configuration**, right-click **Client Protocols**, and then click **Properties**.
2. In the **Enabled Protocols** box, click **Move Up** or **Move Down**, to change the order in which protocols are tried, when attempting to connect to SQL Server. The top protocol in the **Enabled Protocols** box is the default protocol.

   IMPORTANT
   SQL Server Configuration Manager creates registry entries for the server alias configurations and default client network library. However, the application does not install either the SQL Server client network libraries or the network protocols. The SQL Server client network libraries are installed during SQL Server Setup; the network protocols are installed as part of Microsoft Windows Setup (or through **Networks** in **Control Panel**). A particular network protocol may not be available as part of Windows Setup. For more information about installing these network protocols, see the vendor documentation.

To configure a client to use TCP/IP

1. In SQL Server Configuration Manager, expand **SQL Server Native Client Configuration**, right-click **Client Protocols**, and then click **Properties**.
2. In the **Enabled Protocols** box, click the up and down arrows to change the order in which protocols are tried, when attempting to connect to SQL Server. The top protocol in the **Enabled Protocols** box is the default protocol.

The shared memory protocol is enabled separately by checking the **Enabled Shared Memory Protocol** box.

See Also

Configure the remote login timeout Server Configuration Option

Create or Delete a Server Alias for Use by a Client

3/29/2017 • 1 min to read • Edit Online

This topic describes how to create or delete a server alias in SQL Server 2016 by using SQL Server Configuration Manager. An alias is an alternate name that can be used to make a connection. The alias encapsulates the required elements of a connection string, and exposes them with a name chosen by the user. Aliases can be used with any client application. By creating server aliases, your client computer can connect to multiple servers using different network protocols, without having to specify the protocol and connection details for each one. In addition, you can also have different network protocols enabled all the time, even if you only need to use them occasionally. If you have configured the server to listen on a non-default port number or named pipe, and you have disabled the SQL Server Browser service, create an alias that specifies the new port number or named pipe.

Using SQL Server Configuration Manager

To create an alias

1. In SQL Server Configuration Manager, expand **SQL Server Native Client Configuration**, right-click **Aliases**, and then click **New Alias**.
2. In the **Alias Name** box, type the name of the alias. Client applications use this name when they connect.
3. In the **Server** box, type the name or IP address of a server. For a named instance append the instance name.
4. In the **Protocol** box, select the protocol used for this alias. Selecting a protocol, changes the title of the optional properties box to Port No, Pipe Name, or Connection String.

   The connection strings described in SQL Server Configuration Manager Help can be useful for programmers who create their own connection strings. To access this information, in the **New Alias** dialog box, press F1, or click **Help**.

> **NOTE**
> If a configured alias is connecting to the wrong server or instance, disable and then reenable the associated network protocol. Doing this clears any cached connection information and allows the client to connect correctly.

To delete an alias

1. In SQL Server Configuration Manager, expand **SQL Server Native Client Configuration**, and then click **Aliases**.
2. In the details pane, right-click the alias that you want to delete, and then click **Delete**.

Logging In to SQL Server

3/24/2017 • 4 min to read • Edit Online

You can log in to an instance of Microsoft SQL Server by using any of the graphical administration tools or from a command prompt.

When you log in to an instance of SQL Server by using a graphical administration tool such as SQL Server Management Studio, you are prompted to supply the server name, a SQL Server login, and a password, if necessary. If you log in to SQL Server using Windows Authentication, you do not have to provide a SQL Server login each time you access an instance of SQL Server. Instead, SQL Server uses your Microsoft Windows account to log you in automatically. If SQL Server is running in mixed mode authentication (SQL Server and Windows Authentication Mode), and you choose to log in using SQL Server Authentication, you must provide a SQL Server login and password. When possible, use Windows Authentication.

NOTE

If you selected a case-sensitive collation when you installed SQL Server, your SQL Server login is also case sensitive.

Format for Specifying the Name of SQL Server

When connecting to an instance of the Database Engine you must specify the name of the instance of SQL Server. If the instance of SQL Server is the default instance (an unnamed instance), then specify the name of the computer where SQL Server is installed, or the IP address of the computer. If the instance of SQL Server is a named instance (such as SQLEXPRESS), then specify the name of the computer where SQL Server is installed, or the IP address of the computer, and add a slash and the instance name.

The following examples connect to an instance of SQL Server running on a computer named APPHOST. When specifying a named instance, the examples use an instance name SQLEXPRESS.

Examples:

TYPE OF INSTANCE	ENTRY FOR THE SERVER NAME
Connection to a default instance using the default protocol. (This is the recommended entry for a default instance.)	APPHOST
Connection to a named instance using the default protocol. (This is the recommended entry for a named instance.)	APPHOST\SQLEXPRESS
Connection to a default instance on the same computer using a period to indicate that the instance is running on the local computer.	.
Connection to a named instance on the same computer using a period to indicate that the instance is running on the local computer.	.\SQLEXPRESS
Connection to a default instance on the same computer using localhost to indicate that the instance is running on the local computer.	localhost

TYPE OF INSTANCE	ENTRY FOR THE SERVER NAME
Connection to a named instance on the same computer using localhost to indicate that the instance is running on the local computer.	localhost\SQLEXPRESS
Connection to a default instance on the same computer using (local) to indicate that the instance is running on the local computer.	(local)
Connection to a named instance on the same computer using (local) to indicate that the instance is running on the local computer.	(local)\SQLEXPRESS
Connection to a default instance on the same computer forcing a shared memory connection.	lpc:APPHOST
Connection to a named instance on the same computer forcing a shared memory connection.	lpc:APPHOST\SQLEXPRESS
Connection to a default instance listening on TCP address 192.168.17.28 using an IP address.	192.168.17.28
Connection to a named instance listening on TCP address 192.168.17.28 using an IP address.	192.168.17.28\SQLEXPRESS
Connection to a default instance that is not listening on the default TCP port, by specifying the port that is being used, in this case 2828. (This is not necessary if the Database Engine is listening on the default port (1433).)	APPHOST,2828
Connection to a named instance on a designated TCP port, in this case 2828. (This is often necessary if the SQL Server Browser service is not running on the host computer.)	APPHOST,2828
Connection to a default instance that is not listening on the default TCP port, by specifying both the IP address and the TCP port that is being used, in this case 2828.	192.168.17.28,2828
Connection to a named instance by specifying both the IP address and the TCP port that is being used, in this case 2828.	192.168.17.28,2828
Connecting to default instance by name, forcing a TCP connection.	tcp:APPHOST
Connecting to named instance by name, forcing a TCP connection.	tcp:APPHOST\SQLEXPRESS
Connecting to a default instance by specifying a named pipe name.	\\APPHOST\pipe\unit\app
Connecting to a named instance by specifying a named pipe name.	\\APPHOST\pipe\MSSQL$SQLEXPRESS\SQL\query
Connecting to default instance by name, forcing a named pipes connection.	np:APPHOST

TYPE OF INSTANCE	ENTRY FOR THE SERVER NAME
Connecting to named instance by name, forcing a named pipes connection.	np:APPHOST\SQLEXPRESS

Verifying your Connection Protocol

When connected to the Database Engine, the following query will return the protocol used for the current connection, along with the authentication method (NTLM or Kerberos), and will indicate if the connection is encrypted.

```
SELECT net_transport, auth_scheme, encrypt_option
FROM sys.dm_exec_connections
WHERE session_id = @@SPID;
```

Related Tasks

Log In to an Instance of SQL Server (Command Prompt)

The following resources can help you troubleshoot a connection problem.

- How to Troubleshoot Connecting to the SQL Server Database Engine
- Steps to troubleshoot SQL connectivity issues

Related Content

Choose an Authentication Mode

Use the sqlcmd Utility

Creating a Login

Open the ODBC Data Source Administrator

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✓ Azure SQL Database ✓ Azure SQL Data Warehouse ✓ Parallel Data Warehouse

This topic describes how to open the ODBC Data Source Administrator. The ODBC Data Source Administrator is a Windows component. Use the ODBC Data Source Administrator to create and manage ODBC data sources.

To open the ODBC Data Source Administrator in Windows 10

1. On the **Start** page, type **ODBC Data Sources**. The *ODBC Data Sources Destop App* should appear as a choice.

To open the ODBC Data Source Administrator in Windows 7

1. On the **Start** menu, click **Control Panel**.
2. In **Control Panel**, click **Administrative Tools**.
3. In **Administrative Tools**, click **Data Sources (ODBC)**.

To open the ODBC Data Source Administrator in Windows Server 2008

1. On the **Start** menu, point to **Administrative Tools**, and then click **Data Sources (ODBC)**.

> **NOTE**
>
> For connections to Azure Active Directory Authentication for SQL Database install the latest driver, such as ODBC Driver 13.1 for SQL Server.

See Also

Check the ODBC SQL Server Driver Version (Windows)

Check the ODBC SQL Server Driver Version (Windows)

3/24/2017 • 1 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✓ Azure SQL Database ✓ Azure SQL Data Warehouse ✓ Parallel Data Warehouse

Your computer may contain a variety of ODBC drivers, from Microsoft and from other companies. This topic describes how to use the Windows **ODBC Data Source Administrator** to check the version of the installed ODBC drivers.

To check the ODBC SQL Server driver version (32-bit ODBC)

- In the **ODBC Data Source Administrator**, click the **Drivers** tab.

  Information for the Microsoft SQL Server entry is displayed in the **Version** column.

> **NOTE**
>
> For connections to Azure Active Directory Authentication for SQL Database install the latest driver, such as ODBC Driver 13.1 for SQL Server.

See Also

Open the ODBC Data Source Administrator

Troubleshoot Connecting to the SQL Server Database Engine

3/24/2017 • 15 min to read • Edit Online

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ⊗ Azure SQL Database ⊗ Azure SQL Data Warehouse ⊗ Parallel Data Warehouse

This is an exhaustive list of troubleshooting techniques to use when you cannot connect to the SQL Server Database Engine. These steps are not in the order of the most likely problems which you probably already tried. These steps are in order of the most basic problems to more complex problems. These steps assume that you are connecting to SQL Server from another computer by using the TCP/IP protocol, which is the most common situation. These steps are written for SQL Server 2016 with both the SQL Server and the client applications running Windows 10, however the steps generally apply to other versions of SQL Server and other operating systems with only slight modifications.

These instructions are particularly useful when troubleshooting the "**Connect to Server**" error, which can be Error Number: 11001 (or 53), Severity: 20, State: 0, and error messages such as:

- "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. "
- "(provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server) (Microsoft SQL Server, Error: 53)" or "(provider: TCP Provider, error: 0 - No such host is known.) (Microsoft SQL Server, Error: 11001)"

This error usually means that the SQL Server computer can't be found or that the TCP port number is either not known, or is not the correct port number, or is blocked by a firewall.

TIP

An interactive troubleshooting page is available from Microsoft Customer Support Services at Solving Connectivity errors to SQL Server.

Not included

- This topic does not include information about SSPI errors. For SSPI errors, see How to troubleshoot the "Cannot generate SSPI context" error message.
- This topic does not include information about Kerberos errors. For help, see Microsoft Kerberos Configuration Manager for SQL Server.
- This topic does not include information about SQL Azure Connectivity. For help, see Troubleshooting connectivity issues with Microsoft Azure SQL Database.

Gathering Information about the Instance of SQL Server

First you must gather basic information about the database engine.

1. Confirm the instance of the SQL Server Database Engine is installed and running.
   a. Logon to the computer hosting the instance of SQL Server.
   b. Start SQL Server Configuration Manager. (Configuration Manager is automatically installed on the computer when SQL Server is installed. Instructions on starting Configuration Manager vary slightly by

version of SQL Server and Windows. For help starting Configuration Manager, see SQL Server Configuration Manager.)

   c. Using Configuration Manager, in the left pane select **SQL Server Services**. In the right-pane confirm that the instance of the Database Engine is present and running. An instance named **SQL Server (MSSQLSERVER)** is a default (unnamed) instance. There can only be one default instance. Other (named) instances will have their names listed between the parentheses. SQL Server Express uses the name **SQL Server (SQLEXPRESS)** as the instance name unless someone named it something else during installation. Make a note of the name of the instance that you are trying to connect to. Also, confirm that the instance is running, by looking for the green arrow. If the instance has a red square, right-click the instance and then click **Start**. It should turn green.

      a. If you are attempting to connect to a named instance, make sure the SQL Server Browser service is running.

2. Get the IP Address of the computer.

   a. On the Start menu, click **Run**. In the **Run** window type **cmd**, and then click **OK**.
   b. In the command prompt window, type **ipconfig** and then press enter. Make a note of the **IPv4** Address and the **IPv6** Address. (SQL Server can connect using the older IP version 4 protocol or the newer IP version 6 protocol. Your network could allow either or both. Most people start by troubleshooting the **IPv4** address. It's shorter and easier to type.)

3. Get the TCP port number used by SQL Server. In most cases you are connecting to the Database Engine from another computer using the TCP protocol.

   a. Using SQL Server Management Studio on the computer running SQL Server, connect to the instance of SQL Server. In Object Explorer, expand **Management**, expand **SQL Server Logs**, and then double-click the current log.
   b. In the Log Viewer, click the **Filter** button on the toolbar. In the **Message contains text** box, type **server is listening on**, click **Apply filter**, and then click **OK**.
   c. A message similar to **Server is listening on ['any' <ipv4> 1433]** should be listed. This message indicates that this instance of SQL Server is listening on all the IP addresses on this computer (for IP version 4) and is listening to TCP port 1433. (TCP port 1433 is usually the port used by the Database Engine. Only one instance of SQL Server can use a port, so if there is more than one instance of SQL Server installed, some instances must use other port numbers.) Make a note of the port number used by the instance of SQL Server that you are trying to connect to.

> **NOTE**
>
> IP address 127.0.0.1 is probably listed. It is called the loopback adapter address and can only be connected to from processes on the same computer. It can be useful for troubleshooting, but you can't use it to connect from another computer.

Enable Protocols

In some installations of SQL Server, connecting to the Database Engine from another computer is not enabled unless an administrator uses Configuration Manager to enable it. To enable connections from another computer:

1. Open SQL Server Configuration Manager, as described earlier.
2. Using Configuration Manager, in the left pane expand **SQL Server Network Configuration**, and then select the instance of SQL Server that you want to connect to. The right-pane lists the connection protocols available. Shared Memory is normally enabled. It can only be used from the same computer, so most installations leave Shared Memory enabled. To connect to SQL Server from another computer you will normally use TCP/IP. If TCP/IP is not enabled, right-click **TCP/IP**, and then click **Enable**.
3. If you changed the enabled setting for any protocol you must restart the Database Engine. In the left pane select

SQL Server Services. In the right-pane, right-click the instance of the Database Engine, and then click **Restart**.

Testing TCP/IP Connectivity

Connecting to SQL Server by using TCP/IP requires that Windows can establish the connection. Use the `ping` tool to test TCP.

1. On the Start menu, click **Run**. In the **Run** window type **cmd**, and then click **OK**.
2. In the command prompt window, type `ping` and then the IP address of the computer that is running SQL Server. For example, `ping 192.168.1.101` using an IPv4 address, or `ping fe80::d51d:5ab5:6f09:8f48%11` using an IPv6 address. (You must replace the numbers after ping with the IP addresses on your computer which you gathered earlier.)
3. If your network is properly configured you will receive a response such as **Reply from <IP address>** followed by some additional information. If you receive an error such as **Destination host unreachable.** or **Request timed out.** then TCP/IP is not correctly configured. (Check that the IP address was correct and was correctly typed.) Errors at this point could indicate a problem with the client computer, the server computer, or something about the network such as a router. The internet has many resources for troubleshooting TCP/IP. A resonable place to start, is this article from 2006, How to Troubleshoot Basic TCP/IP Problems.
4. Next, if the ping test succeeded using the IP address, test that the computer name can be resolved to the TCP/IP address. On the client computer, in the command prompt window, type `ping` and then the computer name of the computer that is running SQL Server. For example, `ping newofficepc`
5. If you could ping the ipaddress, but noww receive an error such as **Destination host unreachable.** or **Request timed out.** you might have old (stale) name resolution information cached on the client computer. Type `ipconfig /flushdns` to clear the DNS (Dynamic Name Resolution) cache. Then ping the computer by name again. With the DNS cache empty, the client computer will check for the newest information about the IP address for the server computer.
6. If your network is properly configured you will receive a response such as **Reply from <IP address>** followed by some additional information. If you can successfully ping the server computer by IP address but receive an error such as **Destination host unreachable.** or **Request timed out.** when pinging by computer name, then name resolution is not correctly configured. (For more information, see the 2006 article previously referenced, How to Troubleshoot Basic TCP/IP Problems.) Successful name resolution is not required to connect to SQL Server, but if the computer name cannot be resolved to an IP address, then connections must be made specifying the IP address. This is not ideal, but name resolution can be fixed later.

Testing a Local Connection

Before troubleshooting a connection problem from another computer, first test your ability to connect from a client application installed on the computer that is running SQL Server. (This will keep firewall issues out of the way.) This procedure uses SQL Server Management Studio. If you do not have Management Studio installed, see Download SQL Server Management Studio (SSMS). (If you are not able to install Management Studio, you can test the connection using the `sqlcmd.exe` utility which is installed with the Database Engine. For information about `sqlcmd.exe`, see sqlcmd Utility.)

1. Logon to the computer where SQL Server is installed, using a login that has permission to access SQL Server. (During installation, SQL Server requires at least one login to be specified as a SQL Server Administrator. If you do not know an administrator, see Connect to SQL Server When System Administrators Are Locked Out.)
2. On the Start page, type **SQL Server Management Studio**, or on older versions of Windows on the Start menu, point to **All Programs**, point to **Microsoft SQL Server**, and then click **SQL Server Management Studio**.
3. In the **Connect to Server** dialog box, in the **Server** type box, select **Database Engine**. In the **Authentication** box, select **Windows Authentication**. In the **Server name** box, type one of the following:

CONNECTING TO:	TYPE:	EXAMPLE:
Default instance	The computer name	ACCNT27
Named Instance	The computer name\instance name	ACCNT27\PAYROLL

NOTE

When connecting to a SQL Server from a client application on the same computer, the shared memory protocol is used. Shared memory is a type of local named pipe, so sometimes errors regarding pipes are encountered.

If you receive an error at this point, you will have to resolve it before proceeding. There are many possible things that could be a problem. Your login might not be authorized to connect. Your default database might be missing.

NOTE

Some error messages passed to the client intentionally do not give enough information to troubleshoot the problem. This is a security feature to avoid providing an attacker with information about SQL Server. To view the complete information about the error, look in the SQL Server error log. The details are provided there. If you are receiving error **18456 Login failed for user**, Books Online topic MSSQLSERVER_18456 contains additional information about error codes. And Aaron Bertrand's blog has a very extensive list of error codes at Troubleshooting Error 18456. You can view the error log with SSMS (if you can connect), in the Management section of the Object Explorer. Otherwise, you can view the error log with the Windows Notepad program. The default location varies with your version and can be changed during setup. The default location for SQL Server 2016 is `C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\ERRORLOG`.

1. If you can connect using shared memory, test connecting using TCP. You can force a TCP connection by specifying **tcp:** before the name. For example:

CONNECTING TO:	TYPE:	EXAMPLE:
Default instance	tcp: The computer name	tcp:ACCNT27
Named Instance	tcp: The computer name/instance name	tcp:ACCNT27\PAYROLL

If you can connect with shared memory but not TCP, then you must fix the TCP problem. The most likely issue is that TCP is not enabled. To enable TCP, See the **Enable Protocols** steps above.

1. If your goal is to connect with an account other than an administrator account, once you can connect as an administrator, try the connection again using the Windows Authentication login or the SQL Server Authentication login that the client application will be using.

Opening a Port in the Firewall

Beginning many years ago with Windows XP Service Pack 2, the Windows firewall is turned on and will block connections from another computer. To connect using TCP/IP from another computer, on the SQL Server computer you must configure the firewall to allow connections to the TCP port used by the Database Engine. As mentioned earlier, the default instance is usually listening on TCP port 1433. If you have named instances or if you changed the default, the SQL Server TCP port may be listening on another port. See the begining section on gathering information to determine your port.
If you are connecting to a named instance or a port other than TCP port 1433, you must also open the UDP port 1434 for the SQL Server Browser service. For step by step instruction on opening a port in the Windows firewall, see Configure a Windows Firewall for Database Engine Access.

Testing the Connection

Once you can connect using TCP on the same computer, it's time to try connecting from the client computer. You could theoretically use any client application, but to avoid additional complexity, install the SQL Server Management tools on the client and make the attempt using SQL Server Management Studio.

1. On the client computer, using SQL Server Management Studio, attempt to connect using the IP Address and the TCP port number in the format IP address comma port number. For example, **192.168.1.101,1433** If this doesn't work, then you probably have one of the following problems:

   - **Ping** of the IP address doesn't work, indicating a general TCP configuration problem. Go back to the section **Testing TCP/IP Connectivity**.
   - SQL Server is not listening on the TCP protocol. Go back to the section **Enable Protocols**.
   - SQL Server is listening on a port other than the port you specified. Go back to the section **Gathering Information about the Instance of SQL Server**.
   - The SQL Server TCP port is being blocked by the firewall. Go back to the section **Opening a Port in the Firewall**.
2. Once you can connect using the IP address and port number, attempt to connect using the IP address without a port number. For a default instance, just use the IP address. For a named instance, use the IP address and the instance name in the format IP address backslash instance name, for example **192.168.1.101\PAYROLL** If this doesn't work, then you probably have one of the following problems:

   - If you are connecting to the default instance, it might be listening on a port other than TCP port 1433, and the client isn't attempting to connect to the correct port number.
   - If you are connecting to a named instance, the port number is not being returned to the client.

Both of these problems are related to the SQL Server Browser service, which provides the port number to the client. The solutions are:

- Start the SQL Server Browser service. Go back to the section **Gathering Information about the Instance of SQL Server**, section 1.d.
- The SQL Server Browser service is being blocked by the firewall. Open UDP port 1434 in the firewall. Go back to the section **Opening a Port in the Firewall**. (Make sure you are opening a UDP port, not a TCP port. Those are different things.)
- The UDP port 1434 information is being blocked by a router. UDP communication (user datagram protocol) is not designed to pass through routers. This keeps the network from getting filled with low priority traffic. You might be able to configure your router to forward UDP traffic, or you can decide to always provide the port number when you connect.
- If the client computer is using Windows 7 or Windows Server 2008, (or a more recent operating system,) the UDP traffic might be dropped by the client operating system because the response from the server is returned from a different IP address than was queried. This is a security feature blocking "loose source mapping." For more information, see the **Multiple Server IP Addresses** section of the Books Online topic Troubleshooting: Timeout Expired. This is an article from SQL Server 2008 R2, but the principals still apply. You might be able to configure the client to use the correct IP address, or you can decide to always provide the port number when you connect.

1. Once you can connect using the IP address (or IP address and instance name for a named instance), attempt to connect using the computer name (or computer name and instance name for a named instance). Put `tcp:` in front of the computer name to force a TCP/IP connection. For example, for the default instance on a computer named `ACCNT27`, use `tcp:ACCNT27` For a named instance called `PAYROLL`, on that computer use `tcp:ACCNT27\PAYROLL` If you can connect using the IP address but not using the computer name, then you have a name resolution problem. Go back to the section **Testing TCP/IP Connectivity**, section 4.

2. Once you can connect using the computer name forcing TCP, attempt connecting using the computer name

but not forcing TCP. For example, for a default instance use just the computer name such as `CCNT27` For a named instance use the computer name and instance name like `ACCNT27\PAYROLL` If you could connect while forcing TCP, but not without forcing TCP, then the client is probably using another protocol (such as named pipes).

a. On the client computer, using SQL Server Configuration Manager, in the left-pane expand **SQL Native Client** *version* **Configuration**, and then select **Client Protocols**.
b. On the right-pane, Make sure TCP/IP is enabled. If TCP/IP is disabled, right-click **TCP/IP** and then click **Enable**.
c. Make sure that the protocol order for TCP/IP is a smaller number that the named pipes (or VIA on older versions) protocols. Generally you should leave Shared Memory as order 1 and TCP/IP as order 2. Shared memory is only used when the client and SQL Server are running on the same computer. All enabled protocols are tried in order until one succeeds, except that shared memory is skipped when the connection is not to the same computer.

Diagnostic Connection for Database Administrators

3/24/2017 • 7 min to read • Edit Online

SQL Server provides a special diagnostic connection for administrators when standard connections to the server are not possible. This diagnostic connection allows an administrator to access SQL Server to execute diagnostic queries and troubleshoot problems even when SQL Server is not 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 may not be successful.

Applies to: SQL Server (SQL Server 2008 through current version), SQL Database.

Connecting with DAC

By default, the connection is only allowed from a client running on the server. Network connections are not permitted unless they are 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 Use sqlcmd with Scripting Variables. You can also connect prefixing **admin:**to the instance name in the format **sqlcmd -Sadmin:**. You can also initiate a DAC from a SQL Server Management Studio Query Editor by connecting to **admin:**<*instance_name*>.

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 does not 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:

  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. Do not use the DAC to run resource-intensive queries (for example. a complex join on large table) or queries that may block. This helps prevent the DAC from compounding any existing server problems. To avoid potential blocking scenarios, if you have to run queries that may 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 CNTRL-C but it is not guaranteed. In that case, your only option may 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 does not 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. Do not run 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 always succeed; then the only option may be to restart SQL Server. The following are some general guidelines:

    - Verify that the SPID was actually 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. Note that this may take considerable amount of time and may 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 do not 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. Note that it may not be enough to kill a single session: you may have to kill multiple ones.

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 requiring a restart of 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 is not 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 is not 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–Stcp: *<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:

sqlcmd–S127.0.0.1,**1434**

TIP

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.

Example

In this example, an administrator notices that server `URAN123` is not responding and wants to diagnose the problem. To do this, the user activates the `sqlcmd` command prompt utility and connects to server `URAN123` using `-A` to indicate the DAC.

```
sqlcmd -S URAN123 -U sa -P <xxx> –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:

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

Related Tasks

Related Content

Use sqlcmd with Scripting Variables

sqlcmd Utility

SELECT (Transact-SQL)

sp_who (Transact-SQL)

sp_lock (Transact-SQL)

KILL (Transact-SQL)

DBCC CHECKALLOC (Transact-SQL)

DBCC CHECKDB (Transact-SQL)

DBCC OPENTRAN (Transact-SQL)

DBCC INPUTBUFFER (Transact-SQL)

Server Configuration Options (SQL Server)

Transaction Related Dynamic Management Views and Functions (Transact-SQL)

Trace Flags (Transact-SQL)

SQL Server 2016 Express LocalDB

3/29/2017 • 8 min to read • Edit Online

Microsoft SQL Server 2016 Express **LocalDB** is a feature of SQL Server Express targeted to developers. It is available on SQL Server 2016 Express with Advanced Services.

LocalDB installation copies a minimal set of files necessary to start the SQL Server Database Engine. Once LocalDB is installed, you can initiate a connection using a special connection string. When connecting, the necessary SQL Server infrastructure is automatically created and started, enabling the application to use the database without complex configuration tasks. Developer Tools can provide developers with a SQL Server Database Engine that lets them write and test Transact-SQL code without having to manage a full server instance of SQL Server.

Try it out!

- To download and install SQL Server 2016 Express, go to **SQL Server downloads**. LocalDB is a feature you select during installation, and is available when you download the media. If you download the media, either choose **Express Advanced** or the **LocalDB** package.
- Have an Azure account? Then go **here** to spin up a Virtual Machine with SQL Server 2016 already installed.

Install LocalDB

Install **LocalDB** through the installation wizard or by using the SqlLocalDB.msi program. **LocalDB** is an option when installing SQL Server 2016 Express.

Select **LocalDB** on the **Feature Selection/Shared Features** page during installation. There can be only one installation of the **LocalDB** binary files for each major SQL Server Database Engine version. Multiple Database Engine processes can be started and will all use the same binaries. An instance of the SQL Server Database Engine started as the **LocalDB** has the same limitations as SQL Server Express

An instance of SQL Server Express **LocalDB** is managed by using the **SqlLocalDB.exe** utility. SQL Server Express **LocalDB** should be used in place of the SQL Server Express user instance feature which is deprecated.

Description

The **LocalDB** setup program uses the SqlLocalDB.msi program to install the necessary files on the computer. Once installed, **LocalDB** is an instance of SQL Server Express that can create and open SQL Server databases. The system database files for the database are stored in the users' local AppData path which is normally hidden. For example **C:\Users\\AppData\Local\Microsoft\Microsoft SQL Server Local DB\Instances\LocalDBApp1\**. User database files are stored where the user designates, typically somewhere in the **C:\Users\\Documents** folder.

For more information about including **LocalDB** in an application, see the Visual Studio documentation Local Data Overview, Walkthrough: Creating a SQL Server LocalDB Database, and Walkthrough: Connecting to Data in a SQL Server LocalDB Database (Windows Forms).

For more information about the **LocalDB** API, see SQL Server Express LocalDB Instance API Reference and LocalDBStartInstance Function.

The SqlLocalDb utility can create new instances of **LocalDB**, start and stop an instance of **LocalDB**, and includes options to help you manage **LocalDB**. For more information about the SqlLocalDb utility, see SqlLocalDB Utility.

The instance collation for **LocalDB** is set to SQL_Latin1_General_CP1_CI_AS and cannot be changed. Database-level, column-level, and expression-level collations are supported normally. Contained databases follow the metadata and tempdb collations rules defined by Contained Database Collations.

Restrictions

LocalDB cannot be a merge replication subscriber.

LocalDB does not support FILESTREAM.

LocalDB only allows local queues for Service Broker.

An instance of **LocalDB** owned by the built-in accounts such as NT AUTHORITY\SYSTEM can have manageability issues due to windows file system redirection; Instead use a normal windows account as the owner.

Automatic and Named Instances

LocalDB supports two kinds of instances: Automatic instances and named instances.

- Automatic instances of **LocalDB** are public. They are created and managed automatically for the user and can be used by any application. One automatic instance of **LocalDB** exists for every version of **LocalDB** installed on the user's computer. Automatic instances of **LocalDB** provide seamless instance management. There is no need to create the instance; it just works. This allows for easy application installation and migration to a different computer. If the target machine has the specified version of **LocalDB** installed, the automatic instance of **LocalDB** for that version is available on the target machine as well. Automatic instances of **LocalDB** have a special pattern for the instance name that belongs to a reserved namespace. This prevents name conflicts with named instances of **LocalDB**. The name for the automatic instance is **MSSQLLocalDB**.

- Named instances of **LocalDB** are private. They are owned by a single application that is responsible for creating and managing the instance. Named instances provide isolation from other instances and can improve performance by reducing resource contention with other database users. Named instances must be created explicitly by the user through the **LocalDB** management API or implicitly via the app.config file for a managed application (although managed application may also use the API, if desired). Each named instance of **LocalDB** has an associated **LocalDB** version that points to the respective set of **LocalDB** binaries. The instance name of a **LocalDB** is **sysname** data type and can have up to 128 characters. (This differs from regular named instances of SQL Server, which limits names to regular NetBIOS names of 16 ASCII chars.) The name of an instance of **LocalDB** can contain any Unicode characters that are legal within a filename. A named instance that uses an automatic instance name becomes an automatic instance.

  Different users of a computer can have instances with the same name. Each instance is a different processes running as a different user.

Shared Instances of LocalDB

To support scenarios where multiple users of the computer need to connect to a single instance of **LocalDB**, **LocalDB** supports instance sharing. An instance owner can choose to allow the other users on the computer to connect to his instance. Both automatic and named instances of **LocalDB** can be shared. To share an instance of **LocalDB** a user selects a shared name (alias) for it. Because the shared name is visible to all users of the computer, this shared name must be unique on the computer. The shared name for an instance of **LocalDB** has the same format as the named instance of **LocalDB**.

Only an administrator on the computer can create a shared instance of **LocalDB**. A shared instance of **LocalDB** can be unshared by an administrator or by the owner of the shared instance of **LocalDB**. To share and unshared an instance of **LocalDB**, use the `LocalDBShareInstance` and `LocalDBUnShareInstance` methods of the **LocalDB** API, or the share and unshared options of the SqlLocalDb utility.

Starting LocalDB and Connecting to LocalDB

Connecting to the Automatic Instance

The easiest way to use **LocalDB** is to connect to the automatic instance owned by the current user by using the connection string **"Server=(localdb)\MSSQLLocalDB;Integrated Security=true"**. To connect to a specific database by using the file name, connect using a connection string similar to **"Server=(LocalDB)\MSSQLLocalDB; Integrated Security=true ;AttachDbFileName=D:\Data\MyDB1.mdf"**.

> **NOTE**
>
> The first time a user on a computer tries to connect to **LocalDB**, the automatic instance must be both created and started. The extra time for the instance to be created can cause the connection attempt to fail with a timeout message. When this happens, wait a few seconds to let the creation process complete, and then connect again.

Creating and Connecting to a Named Instances

In addition to the automatic instance, **LocalDB** also supports named instances. Use the SqlLocalDB.exe program to create, start, and stop an named instance of **LocalDB**. For more information about SqlLocalDB.exe, see SqlLocalDB Utility.

```
REM Create an instance of LocalDB
"C:\Program Files\Microsoft SQL Server\130\Tools\Binn\SqlLocalDB.exe" create LocalDBApp1
REM Start the instance of LocalDB
"C:\Program Files\Microsoft SQL Server\130\Tools\Binn\SqlLocalDB.exe" start LocalDBApp1
REM Gather information about the instance of LocalDB
"C:\Program Files\Microsoft SQL Server\130\Tools\Binn\SqlLocalDB.exe" info LocalDBApp1
```

The last line above, returns information similar to the following.

Name	"LocalDBApp1"
Version	<Current Version>
Shared name	""
Owner	"<Your Windows User>"
Auto create	No
State	running
Last start time	<Date and Time>
Instance pipe name	np:\\.\pipe\LOCALDB#F365A78E\tsql\query

> **NOTE**
>
> If your application uses a version of .NET before 4.0.2 you must connect directly to the named pipe of the **LocalDB**. The Instance pipe name value is the named pipe that the instance of **LocalDB** is listening on. The portion of the Instance pipe name after LOCALDB# will change each time the instance of **LocalDB** is started. To connect to the instance of **LocalDB** by using SQL Server Management Studio, type the Instance pipe name in the **Server name** box of the **Connect to Database Engine** dialog box. From your custom program you can establish connection to the instance of **LocalDB** using a connection string similar to `SqlConnection conn = new SqlConnection(@"Server=np:\\.\pipe\LOCALDB#F365A78E\tsql\query");`

Connecting to a Shared Instance of LocalDB

To connect to a shared instance of **LocalDB** add **.** (dot + backslash) to the connection string to reference the namespace reserved for shared instances. For example, to connect to a shared instance of **LocalDB** named `AppData` use a connection string such as `(localdb)\.\AppData` as part of the connection string. A user connecting to a shared instance of **LocalDB** that they do not own must have a Windows Authentication or SQL Server Authentication login.

Troubleshooting

For information about troubleshooting **LocalDB**, see Troubleshooting SQL Server 2012 Express LocalDB.

Permissions

An instance of SQL Server 2016 Express**LocalDB** is an instance created by a user for their use. Any user on the computer can create a database using an instance of **LocalDB**, storing files under their user profile and running the process under their credentials. By default, access to the instance of **LocalDB** is limited to its owner. The data contained in the **LocalDB** is protected by file system access to the database files. If user database files are stored in a shared location, the database can be opened by anyone with file system access to that location by using an instance of **LocalDB** that they own. If the database files are in a protected location, such as the users data folder, only that user, and any administrators with access to that folder, can open the database. The **LocalDB** files can only be opened by one instance of **LocalDB** at a time.

NOTE

LocalDB always runs under the users security context; that is, **LocalDB** never runs with credentials from the local Administrator's group. This means that all database files used by a **LocalDB** instance must be accessible using the owning user's Windows account, without considering membership in the local Administrators group.

See Also

SqlLocalDB Utility

Remote Servers

3/24/2017 • 4 min to read • Edit Online

Remote servers are supported in SQL Server for backward compatibility only. New applications should use linked servers instead. For more information, see Linked Servers (Database Engine).

A remote server configuration allows for a client connected to one instance of SQL Server to execute a stored procedure on another instance of SQL Server without establishing a separate connection. Instead, the server to which the client is connected accepts the client request and sends the request to the remote server on behalf of the client. The remote server processes the request and returns any results to the original server. This server in turn passes those results to the client. When you set up a remote server configuration, you should also consider how to establish security.

If you want to set up a server configuration to execute stored procedures on another server and do not have existing remote server configurations, use linked servers instead of remote servers. Both stored procedures and distributed queries are allowed against linked servers; however, only stored procedures are allowed against remote servers.

Remote Server Details

Remote servers are set up in pairs. To set up a pair of remote servers, configure both servers to recognize each other as remote servers.

Most of the time, you should not have to set configuration options for remote servers. SQL Server Set sets the defaults on both the local and remote computers to allow for remote server connections.

For remote server access to work, the **remote access** configuration option must be set to 1 on both the local and remote computers. (This is the default setting.) **remote access** controls logins from remote servers. You can reset this configuration option by using either the Transact-SQL **sp_configure** stored procedure or SQL Server Management Studio. To set the option in SQL Server Management Studio, on the **Server Properties Connections** page, use **Allow remote connections to this server**. To reach the **Server Properties Connections** page, in Object Explorer, right-click the server name, and then click **Properties**. On the **Server Properties** page, click the **Connections** page.

From the local server, you can disable a remote server configuration to prevent access to that local server by users on the remote server with which it is paired.

Security for Remote Servers

To enable remote procedure calls (RPC) against a remote server, you must set up login mappings on the remote server and possibly on the local server that is running an instance of SQL Server. RPC is disabled by default in SQL Server. This configuration enhances the security of your server by reducing its attackable surface area. Before using RPC you must enable this feature. For more information see sp_configure (Transact-SQL).

Setting Up the Remote Server

Remote login mappings must be set up on the remote server. Using these mappings, the remote server maps the incoming login for an RPC connection from a specified server to a local login. Remote login mappings can be set up by using the **sp_addremotelogin** stored procedure on the remote server.

NOTE

The **trusted** option of **sp_remoteoption** is not supported in SQL Server.

Setting Up the Local Server

For SQL Server authenticated local logins, you do not have to set up a login mapping on the local server. SQL Server uses the local login and password to connect to the remote server. For Windows authenticated logins, set up a local login mapping on a local server that defines what login and password are used by an instance of SQL Server when it makes an RPC connection to a remote server.

For logins created by Windows Authentication, you must create a mapping to a login name and password by using the **sp_addlinkedservlogin** stored procedure. This login name and password must match the incoming login and password expected by the remote server, as created by **sp_addremotelogin**.

NOTE

When possible, use Windows Authentication.

Remote Server Security Example

Consider these SQL Server installations: **serverSend** and **serverReceive**. **serverReceive** is configured to map an incoming login from **serverSend**, called **Sales_Mary**, to a SQL Server authenticated login in **serverReceive**, called **Alice**. Another incoming login from **serverSend**, called **Joe**, is mapped to a SQL Server authenticated login in **serverReceive**, called **Joe**.

The following Transact-SQL code example configures `serverSend` to perform RPCs against `serverReceive`.

```
--Create remote server entry for RPCs
--from serverSend in serverReceive.
EXEC sp_addserver 'serverSend';
GO

--Create remote login mapping for login 'Sales_Mary' from serverSend
--to Alice.
EXEC sp_addremotelogin 'serverSend', 'Alice', 'Sales_Mary';
GO
--Create remote login mapping for login Joe from serverReceive
--to same login.
--Assumes same password for Joe in both servers.
EXEC sp_addremotelogin 'serverSend', 'Joe', 'Joe';
GO
```

On `serverSend`, a local login mapping is created for a Windows authenticated login `Sales\Mary` to a login `Sales_Mary`. No local mapping is required for `Joe`, because the default is to use the same login name and password, and `serverReceive` has a mapping for `Joe`.

```
--Create a remote server entry for RPCs from serverReceive.
EXEC sp_addserver 'serverReceive';
GO
--Create a local login mapping for the Windows authenticated login.
--Sales\Mary to Sales_Mary. The password should match the
--password for the login Sales_Mary in serverReceive.
EXEC sp_addlinkedsrvlogin 'serverReceive', false, 'Sales\Mary',
   'Sales_Mary', '430[fj%dk';
GO
```

Viewing Local or Remote Server Properties

You can use the **xp_msver** extended stored procedure to review server attributes for local or remote servers. These attributes include the version number of SQL Server, the type and number of processors in the computer, and the version of the operating system. From the local server, you can view databases, files, logins, and tools for a remote server. For more information, see xp_msver (Transact-SQL).

Related Tasks

Linked Servers (Database Engine)

Related Content

sp_configure (Transact-SQL)

Configure the remote access Server Configuration Option

RECONFIGURE (Transact-SQL)

SQL Server Monitor Overview

3/24/2017 • 1 min to read • Edit Online

SQL Server Monitor does not perform monitoring functions, but it hosts modules that do. The SQL Server Monitor modules include Replication Monitor and Database Mirroring Monitor.

To use one of these modules, select that module on the **Go** menu. The currently selected module owns the content of the navigation and detail panes, user interaction in the detail panes, and the queries for content and status.

NOTE

For more information about these monitors, see Monitoring Replication and Monitoring Database Mirroring (SQL Server).

Permissions

- Replication Monitor

  To monitor replication, you must be a member of the **sysadmin** fixed server role at the Distributor or a member of the **replmonitor** fixed database role in the distribution database. A system administrator can add any user to the **replmonitor** role, which allows that user to view replication activity in Replication Monitor; however, the user cannot administer replication.

- Database Mirroring Monitor

  To monitor database mirroring, you must be a member of either the **sysadmin** fixed server role or the **dbm_monitor** fixed database role on the server instance. If you are a member of **sysadmin** or **dbm_monitor** on only one of the partner server instances, the monitor can connect only to that partner; the monitor cannot retrieve information from the other partner. For more information, see Database Mirroring Monitor Overview.

Menu Options

SQL Server Monitor has a menu that contains commands that pertain to SQL Server Monitor. This menu may also contain commands from the selected module.

The following menu options pertain to SQL Server Monitor.

File
This menu contains the **Exit** command.

Action
Contains the context menu of the node selected in the navigation tree.

Go
Contains a list of monitoring components:

- Database Mirroring
- Replication

  To use SQL Server Management Studio to monitor database mirroring

- Start Database Mirroring Monitor (SQL Server Management Studio)

See Also

Monitoring Database Mirroring (SQL Server)

SQL Server Service Broker

3/24/2017 • 1 min to read • Edit Online

SQL Server Service Broker provides native support for messaging and queuing applications in the SQL Server Database Engine. This makes it easier for developers to create sophisticated applications that use the Database Engine components to communicate between disparate databases. Developers can use Service Broker to easily build distributed and reliable applications.

Application developers who use Service Broker can distribute data workloads across several databases without programming complex communication and messaging internals. This reduces development and test work because Service Broker handles the communication paths in the context of a conversation. It also improves performance. For example, front-end databases supporting Web sites can record information and send process intensive tasks to queue in back-end databases. Service Broker ensures that all tasks are managed in the context of transactions to assure reliability and technical consistency.

Where is the documentation for Service Broker?

The reference documentation for Service Broker is included in the SQL Server 2016 documentation. This reference documentation includes the following sections:

- Data Definition Language (DDL) Statements (Transact-SQL) for CREATE, ALTER, and DROP statements
- Service Broker Statements
- Service Broker Catalog Views (Transact-SQL)
- Service Broker Related Dynamic Management Views (Transact-SQL)
- ssbdiagnose Utility (Service Broker)

  See the previously published documentation for Service Broker concepts and for development and management tasks. This documentation is not reproduced in the SQL Server 2016 documentation due to the small number of changes in Service Broker in SQL Server 2016.

What's new in Service Broker

No significant changes are introduced in SQL Server 2016. The following changes were introduced in SQL Server 2012.

Messages can be sent to multiple target services (multicast)

The syntax of the SEND (Transact-SQL) statement has been extended to enable multicast by supporting multiple conversation handles.

Queues expose the message enqueued time

Queues have a new column, **message_enqueue_time**, that shows how long a message has been in the queue.

Poison message handling can be disabled

The CREATE QUEUE (Transact-SQL) and ALTER QUEUE (Transact-SQL) statements now have the ability to enable or disable poison message handling by adding the clause, `POISON_MESSAGE_HANDLING (STATUS = ON | OFF)`. The catalog view **sys.service_queues** now has the column **is_poison_message_handling_enabled** to indicate whether poison message is enabled or disabled.

Always On support in Service Broker

For more information, see Service Broker with Always On Availability Groups (SQL Server).

Buffer Pool Extension

3/24/2017 • 6 min to read • Edit Online

Introduced in SQL Server 2014, the buffer pool extension provides the seamless integration of a nonvolatile random access memory (that is, solid-state drive) extension to the Database Engine buffer pool to significantly improve I/O throughput. The buffer pool extension is not available in every SQL Server edition. For more information, see Features Supported by the Editions of SQL Server 2016.

Benefits of the Buffer Pool Extension

The primary purpose of a SQL Server database is to store and retrieve data, so intensive disk I/O is a core characteristic of the Database Engine. Because disk I/O operations can consume many resources and take a relatively long time to finish, SQL Server focuses on making I/O highly efficient. The buffer pool serves as a primary memory allocation source of SQL Server. Buffer management is a key component in achieving this efficiency. The buffer management component consists of two mechanisms: the buffer manager to access and update database pages, and the buffer pool), to reduce database file I/O.

Data and index pages are read from disk into the buffer pool and modified pages (also known as dirty pages) are written back to disk. Memory pressure on the server and database checkpoints cause hot (active) dirty pages in the buffer cache to be evicted from the cache and written to mechanical disks and then read back into the cache. These I/O operations are typically small random reads and writes on the order of 4 to 16 KB of data. Small random I/O patterns incur frequent seeks, competing for the mechanical disk arm, increasing I/O latency, and reducing aggregate I/O throughput of the system.

The typical approach to resolving these I/O bottlenecks is to add more DRAM, or alternatively, added high-performance SAS spindles. While these options are helpful, they have significant drawbacks: DRAM is more expensive than data storage drives and adding spindles increases capital expenditure in hardware acquisition and increases operational costs by increased power consumption and increased probability of component failure.

The buffer pool extension feature extends the buffer pool cache with nonvolatile storage (usually SSD). Because of this extension, the buffer pool can accommodate a larger database working set, which forces the paging of I/Os between RAM and the SSDs. This effectively offloads small random I/Os from mechanical disks to SSDs. Because of the lower latency and better random I/O performance of SSDs, the buffer pool extension significantly improves I/O throughput.

The following list describes the benefits of the buffer pool extension feature.

- Increased random I/O throughput
- Reduced I/O latency
- Increased transaction throughput
- Improved read performance with a larger hybrid buffer pool
- A caching architecture that can take advantage of present and future low-cost memory drives

Concepts

The following terms are applicable to the buffer pool extension feature.

Solid-state drive (SSD)
Solid-state drives store data in memory (RAM) in a persistent manner. For more information, see this definition.

Buffer

In SQL Server, A buffer is an 8-KB page in memory, the same size as a data or index page. Thus, the buffer cache is divided into 8-KB pages. A page remains in the buffer cache until the buffer manager needs the buffer area to read in more data. Data is written back to disk only if it is modified. These in-memory modified pages are known as dirty pages. A page is clean when it is equivalent to its database image on disk. Data in the buffer cache can be modified multiple times before being written back to disk.

Buffer pool
Also called buffer cache. The buffer pool is a global resource shared by all databases for their cached data pages. The maximum and minimum size of the buffer pool cache is determined during startup or when the instance of SQL server is dynamically reconfigured by using sp_configure. This size determines the maximum number of pages that can be cached in the buffer pool at any time in the running instance.

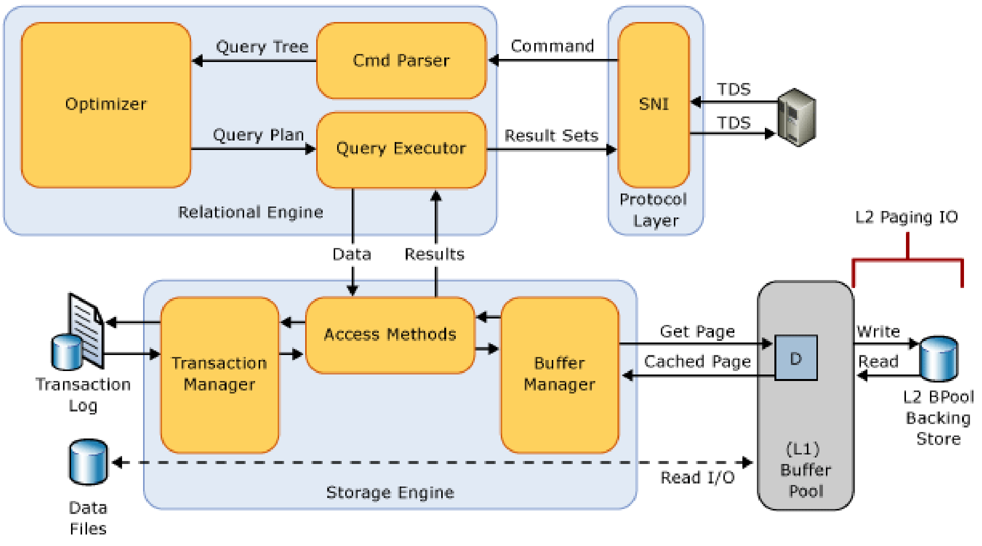
The maximum memory that can be comitted by Buffer Pool Extension can be limited by the other applications running on the machine in case those create significant memory pressure.

Checkpoint
A checkpoint creates a known good point from which the Database Engine can start applying changes contained in the transaction log during recovery after an unexpected shutdown or crash. A checkpoint writes the dirty pages and transaction log information from memory to disk and, also, records information about the transaction log. For more information, see Database Checkpoints (SQL Server).

Buffer Pool Extension Details

SSD storage is used as an extension to the memory subsystem rather than the disk storage subsystem. That is, the buffer pool extension file allows the buffer pool manager to use both DRAM and NAND-Flash memory to maintain a much larger buffer pool of lukewarm pages in nonvolatile random access memory backed by SSDs. This creates a multilevel caching hierarchy with level 1 (L1) as the DRAM and level 2 (L2) as the buffer pool extension file on the SSD. Only clean pages are written to the L2 cache, which helps maintain data safety. The buffer manager handles the movement of clean pages between the L1 and L2 caches.

The following illustration provides a high-level architectural overview of the buffer pool relative to other SQL Server components.



When enabled, the buffer pool extension specifies the size and file path of the buffer pool caching file on the SSD. This file is a contiguous extent of storage on the SSD and is statically configured during startup of the instance of SQL Server. Alterations to the file configuration parameters can only be done when the buffer pool extension feature is disabled. When the buffer pool extension is disabled, all related configuration settings are removed from the registry. The buffer pool extension file is deleted upon shutdown of the instance of SQL Server.

Best Practices

We recommend that you follow these best practices.

- After enabling Buffer Pool Extension for the first time it is recommended to restart the SQL Server instance to get the maximum performance benefits.
- The buffer pool extension size can be up to 32 times the value of max_server_memory. We recommend a ratio between the size of the physical memory (max_server_memory) and the size of the buffer pool extension of 1:16 or less. A lower ratio in the range of 1:4 to 1:8 may be optimal. For information about setting the max_server_memory option, see Server Memory Server Configuration Options.
- Test the buffer pool extension thoroughly before implementing in a production environment. Once in production, avoid making configuration changes to the file or turning the feature off. These activities may have a negative impact on server performance because the buffer pool is significantly reduced in size when the feature is disabled. When disabled, the memory used to support the feature is not reclaimed until the instance of SQL Server is restarted. However, if the feature is re-enabled, the memory will be reused without restarting the instance.

Return Information about the Buffer Pool Extension

You can use the following dynamic management views to display the configuration of the buffer pool extension and return information about the data pages in the extension.

- sys.dm_os_buffer_pool_extension_configuration (Transact-SQL)
- sys.dm_os_buffer_descriptors (Transact-SQL)

  Performance counters are available in the SQL Server, Buffer Manager Object to track the data pages in the buffer pool extension file. For more information, see buffer pool extension performance counters.

  The following Xevents are available.

XEVENT	DESCRIPTION	PARAMETERS
sqlserver.buffer_pool_extension_pages_written	Fires when a page or group of pages are evicted from the buffer pool and written to the buffer pool extension file.	*number_page* *first_page_id* *first_page_offset* *initiator_numa_node_id*
sqlserver.buffer_pool_extension_pages_read	Fires when a page is read from the buffer pool extension file to the buffer pool.	*number_page* *first_page_id* *first_page_offset* *initiator_numa_node_id*
sqlserver.buffer_pool_extension_pages_evicted	Fires when a page is evicted from the buffer pool extension file.	*number_page* *first_page_id* *first_page_offset* *initiator_numa_node_id*

XEVENT	DESCRIPTION	PARAMETERS
sqlserver.buffer_pool_eviction_thresholds_recalculated	Fires when the eviction threshold is calculated.	*warm_threshold* *cold_threshold* *pages_bypassed_eviction* *eviction_bypass_reason* *eviction_bypass_reason_description*

Related Tasks

Task Description	**Topic**
Enable and configure the buffer pool extension.	ALTER SERVER CONFIGURATION (Transact-SQL)
Modify the buffer pool extension configuration	ALTER SERVER CONFIGURATION (Transact-SQL)
View the buffer pool extension configuration	sys.dm_os_buffer_pool_extension_configuration (Transact-SQL)
Monitor the buffer pool extension	sys.dm_os_buffer_descriptors (Transact-SQL) Performance counters