Sequelize for Microsoft SQL Server

VERSION COMPATIBILITY

See Releases to see which versions of SQL Server are supported.

To use Sequelize with Microsoft SQL Server, you need to install the `@sequelize/mssql` dialect package:

npm Yarn pnpm

```
npm i @sequelize/mssql
```

Then use the `MsSqlDialect` class as the dialect option in your Sequelize instance:

```
import { Sequelize } from '@sequelize/core';
import { MsSqlDialect } from '@sequelize/mssql';

const sequelize = new Sequelize({
  dialect: MsSqlDialect,
  server: 'localhost',
  port: 1433,
  database: 'database',
  authentication: {
    type: 'default',
    options: {
      userName: 'username',
      password: 'password',
```

```
    },
  },
});
```

Connection Options

Connection Options are used to configure a connection to the database.

The simplest way to use them is at the root of the configuration object. These options can also be used in the `replication` option to customize the connection for each replica, and can be modified by the `beforeConnect` hook on a connection-by-connection basis.

The following options are passed as-is to the `tedious` package that Sequelize uses to connect to SQL Server. Please refer to the Tedious documentation for more information about what each of these options do.

For convenience, here is an edited copy of the documentation that only includes the options that are accepted by Sequelize:

Option	Description
`server`	Hostname to connect to.
`localAddress`	Network interface (ip address) to use when connecting to SQL Server.
`database`	Database to connect to.
`port`	Port to connect to (default: 1433). Mutually exclusive with `instanceName`.
`instanceName`	The instance name to connect to. The SQL Server Browser service must be running on the database server, and UDP port 1434 on the database server must be reachable. Mutually exclusive with `port`.

Option	Description
`authentication`	The authentication options. Please see which sub-options can be used on the Tedious documentation
`abortTransactionOnError`	A boolean determining whether to rollback a transaction automatically if any error is encountered during the given transaction's execution. This sets the value for SET XACT_ABORT during the initial SQL phase of a connection (documentation).
`appName`	Application name used for identifying a specific application in profiling, logging or tracing tools of SQL Server. (default: Tedious)
`cancelTimeout`	The number of milliseconds before the cancel (abort) of a request is considered failed (default: 5000).
`connectionRetryInterval`	Number of milliseconds before retrying to establish connection, in case of transient failure. (default: 500)
`connectTimeout`	The number of milliseconds before the attempt to connect is considered failed (default: 15000).
`connectionIsolationLevel`	The default isolation level for new connections. All out-of-transaction queries are executed with this setting. The isolation levels are available from the `TEDIOUS_ISOLATION_LEVEL` export. (default: `READ_COMMITED`).
`cryptoCredentialsDetails`	When `encrypt` is set to true, an object may be supplied that will be used as the secureContext field when creating a `TLSSocket`. The available options are listed under `tls.createSecureContext`.
`datefirst`	An integer representing the first day of the week. (default: 7)

Option	Description
`dateFormat`	A string representing the date format. (default: `mdy`)
`debug`	See `options.debug` in the Tedious documentation
`enableAnsiNull`	Controls the way null values should be used during comparison operation. (default: true)
`enableAnsiPadding`	Controls if padding should be applied for values shorter than the size of defined column. (default: true)
`enableAnsiWarnings`	If true, SQL Server will follow ISO standard behavior during various error conditions. For details, see documentation. (default: true)
`enableArithAbort`	Ends a query when an overflow or divide-by-zero error occurs during query execution. See documentation for more details. (default: true)
`enableConcatNullYieldsNull`	If true, concatenating a null value with a string results in a NULL value. (default: true)
`enableCursorCloseOnCommit`	If true, cursors will be closed when a transaction is committed or rolled back. (default: null)
`enableImplicitTransactions`	Sets the connection to either implicit or autocommit transaction mode. (default: false)
`enableNumericRoundabort`	If false, error is not generated during loss of precession. (default: false)
`encrypt`	A string value set to `'strict'` enables the TDS 8.0 protocol. Otherwise, encrypt can be set to a boolean value which determines whether or not the connection will be encrypted under the TDS 7.x protocol. (default: true)

Option	Description
`fallbackToDefaultDb`	By default, if the database requested by options.database cannot be accessed, the connection will fail with an error. However, if this is set to true, then the user's default database will be used instead (Default: false).
`language`	Specifies the language environment for the session. The session language determines the datetime formats and system messages. (default: us_english).
`maxRetriesOnTransientErrors`	The maximum number of connection retries for transient errors. (default: 3).
`multiSubnetFailover`	Sets the `MultiSubnetFailover = True` parameter, which can help minimize the client recovery latency when failovers occur. (default: false).
`packetSize`	The size of TDS packets (subject to negotiation with the server). Should be a power of 2. (default: 4096).
`readOnlyIntent`	A boolean, determining whether the connection will request read only access from a SQL Server Availability Group. For more information, see here. (default: false).
`requestTimeout`	The number of milliseconds before a request is considered failed, or 0 for no timeout (default: 15000).
`tdsVersion`	The version of TDS to use. If server doesn't support specified version, negotiated version is used instead. The versions are available from the `TDS_VERSION` export. (default: 7_4).
`textsize`	Specifies the size of varchar(max), nvarchar(max), varbinary(max), text, ntext, and image data returned by a SELECT statement. (default: 2147483647) (Textsize is set by a numeric value.)

Option	Description
`trustServerCertificate`	If "true", the SQL Server SSL certificate is automatically trusted when the communication layer is encrypted using SSL. If "false", the SQL Server validates the server SSL certificate. If the server certificate validation fails, the driver raises an error and terminates the connection. Make sure the value passed to serverName exactly matches the Common Name (CN) or DNS name in the Subject Alternate Name in the server certificate for an SSL connection to succeed. (default: true).

Domain Account

In order to connect with a domain account, use the following format.

```
const sequelize = new Sequelize({
  dialect: MsSqlDialect,
  instanceName: 'SQLEXPRESS',
  authentication: {
    type: 'ntlm',
    options: {
      domain: 'yourDomain',
      userName: 'username',
      password: 'password',
    },
  },
});
```