bcp utility

Article • 06/21/2024 • 37 contributors

Feedback

In this article

Applies to: ✓ SQL Server ✓ Azure SQL Database ✓ Azure SQL Managed Instance ✓ Azure Synapse Analytics ✓ Analytics Platform System (PDW)

The bulk copy program utility (**bcp**) bulk copies data between an instance of Microsoft SQL Server and a data file in a user-specified format.

For using **bcp** on Linux, see Install the SQL Server command-line tools sqlcmd and bcp on Linux. For detailed information about using **bcp** with Azure Synapse Analytics, see Load data with bcp.

The **bcp** utility can be used to import large numbers of new rows into SQL Server tables or to export data out of tables into data files. Except when used with the `queryout` option, the utility requires no knowledge of Transact-SQL. To import data into a table, you must either use a format file created for that table or understand the structure of the table and the types of data that are valid for its columns.

> **Note**
>
> If you use **bcp** to back up your data, create a format file to record the data format. **bcp** data files **don't include** any schema or format information, so if a table or view is dropped and you don't have a format file, you might be unable to import the data.

For the syntax conventions that are used for the **bcp** syntax, see Transact-SQL syntax conventions.

Download the latest version of the bcp utility

Download PDF

The command-line tools are General Availability (GA), however they're being released with the installer package for SQL Server 2019 (15.x) and later versions.

Windows

- Download ODBC Driver for SQL Server
- Download Microsoft Command Line Utilities 15 for SQL Server (x64)
- Download Microsoft Command Line Utilities 15 for SQL Server (x86)

Linux and macOS

See Install the SQL Server command-line tools sqlcmd and bcp on Linux for instructions to install **sqlcmd** and **bcp** on Linux and macOS.

Version information

- Release number: 15.0.4298.1
- Build number: 15.0.4298.1
- Release date: April 7, 2023

bcp supports Microsoft Entra authentication, including multifactor authentication (MFA) support for Azure SQL Database and Azure Synapse Analytics.

> **Note**
>
> Microsoft Entra ID was previously known as Azure Active Directory (Azure AD).

System requirements

- Windows 8, Windows 8.1, Windows 10, Windows 11
- Windows Server 2016, Windows Server 2019, Windows Server 2022

This component requires the latest Microsoft ODBC Driver 17 for SQL Server.

To check the **bcp** version, execute `bcp -v` command, and confirm that 15.0.4298.1 or later is in use.

> **Note**
>
> **sqlcmd** and **bcp** are also available on Linux. For more information, see Install the SQL Server command-line tools sqlcmd and bcp on Linux.

Syntax

Console

```
bcp [database_name.] schema.{table_name | view_name | "query"}
    {in data_file | out data_file | queryout data_file | format nul}

    [-a packet_size]
    [-b batch_size]
    [-c]
    [-C { ACP | OEM | RAW | code_page } ]
    [-d database_name]
    [-D]
    [-e err_file]
    [-E]
    [-f format_file]
    [-F first_row]
    [-G Azure Active Directory Authentication]
    [-h"hint [,...n]"]
    [-i input_file]
    [-k]
    [-K application_intent]
    [-l login_timeout]
    [-L last_row]
    [-m max_errors]
    [-n]
    [-N]
    [-o output_file]
    [-P password]
    [-q]
    [-r row_term]
    [-R]
    [-S [server_name[\instance_name]]]
    [-t field_term]
    [-T]
    [-U login_id]
    [-v]
    [-V (80 | 90 | 100 | 110 | 120 | 130 | 140 | 150 | 160 ) ]
    [-w]
    [-x]
```

Command-line options

database_name

The name of the database in which the specified table or view resides. If not specified, this is the default database for the user.

You can also explicitly specify the database name with `-d`.

schema

The name of the owner of the table or view. *schema* is optional if the user performing the operation owns the specified table or view. If *schema* isn't specified and the user performing the operation doesn't own the specified table or view, SQL Server returns an error message, and the operation is canceled.

table_name

The name of the destination table when importing data into SQL Server (`in`), and the source table when exporting data from SQL Server (`out`).

view_name

The name of the destination view when copying data into SQL Server (`in`), and the source view when copying data from SQL Server (`out`). Only views in which all columns refer to the same table can be used as destination views. For more information on the restrictions for copying data into views, see INSERT.

"query"

A Transact-SQL query that returns a result set. If the query returns multiple result sets, only the first result set is copied to the data file; subsequent result sets are ignored. Use double quotation marks around the query and single quotation marks around anything embedded in the query. `queryout` must also be specified when bulk copying data from a query.

The query can reference a stored procedure as long as all tables referenced inside the stored procedure exist before executing the **bcp** statement. For example, if the stored procedure generates a temp table, the **bcp** statement fails because the temp table is available only at run time and not at statement execution time. In this case, consider inserting the results of the stored procedure into a table and then use **bcp** to copy the data from the table into a data file.

in

Copies from a file into the database table or view. Specifies the direction of the bulk copy.

out

Copies from the database table or view to a file. Specifies the direction of the bulk copy.

If you specify an existing file, the file is overwritten. When extracting data, the **bcp** utility represents an empty string as a null and a null string as an empty string.

data_file

The full path of the data file. When data is bulk imported into SQL Server, the data file contains the data to be copied into the specified table or view. When data is bulk exported from SQL Server, the data file contains the data copied from the table or view. The path can have from 1 through 255 characters. The data file can contain a maximum of 2^63 - 1 rows.

queryout

Copies from a query and must be specified only when bulk copying data from a query.

format

Creates a format file based on the option specified (`-n`, `-c`, `-w`, or `-N`) and the table or view delimiters. When bulk copying data, the **bcp** command can refer to a format file, which saves you from reentering format information interactively. The `format` option requires the `-f` option; creating an XML format file, also requires the `-x` option. For more information, see Create a Format File (SQL Server). You must specify `nul` as the value (`format nul`).

-a packet_size

Specifies the number of bytes, per network packet, sent to and from the server. A server configuration option can be set by using SQL Server Management Studio (or the `sp_configure` system stored procedure). However, the server configuration option can be overridden on an individual basis by using this option. *packet_size* can be from 4,096 bytes to 65,535 bytes; the default is `4096`.

Increased packet size can enhance performance of bulk-copy operations. If a larger packet is requested but can't be granted, the default is used. The performance statistics generated by the **bcp** utility show the packet size used.

-b batch_size

Specifies the number of rows per batch of imported data. Each batch is imported and logged as a separate transaction that imports the whole batch before being committed. By default, all the rows in the data file are imported as one batch. To distribute the rows among multiple batches, specify a *batch_size* that is smaller than the number of rows in the data file. If the transaction for any batch fails, only insertions from the current batch are rolled back. Batches already imported by committed transactions are unaffected by a later failure.

Don't use this option with the `-h "ROWS_PER_BATCH=<bb>"` option.

-c

Performs the operation using a character data type. This option doesn't prompt for each field; it uses **char** as the storage type, without prefixes and with `\t` (tab character) as the field separator and `\r\n` (newline character) as the row terminator. `-c` isn't compatible with `-w`.

For more information, see Use character format to import or export data (SQL Server).

-C { ACP | OEM | RAW | *code_page* }

Specifies the code page of the data in the data file. *code_page* is relevant only if the data contains **char**, **varchar**, or **text** columns with character values greater than 127 or less than 32.

ⓘ **Note**

We recommend specifying a collation name for each column in a format file, except when you want the 65001 option to have priority over the collation/code page specification.

Expand table

Code page value	Description
ACP	ANSI/Microsoft Windows (ISO 1252).
OEM	Default code page used by the client. This is the default code page used if `-C` isn't specified.
RAW	No conversion from one code page to another occurs. This is the fastest option because no conversion occurs.
code_page	Specific code page number; for example, 850. Versions before version 13 (SQL Server 2016 (13.x)) don't support code page 65001 (UTF-8 encoding). Versions beginning with 13 can import UTF-8 encoding to earlier versions of SQL Server.

-d database_name

Specifies the database to connect to. By default, **bcp** connects to the user's default database. If `-d <database_name>` and a three part name (database_name.schema.table, passed as the first parameter to **bcp**) are specified, an error occurs because you can't specify the database name twice. If *database_name* begins with a hyphen (`-`) or a forward slash (`/`), don't add a space between `-d` and the database name.

-D

Causes the value passed to the `bcp -S` option to be interpreted as a data source name (DSN). A DSN can be used to embed driver options to simplify command lines, enforce driver options that aren't otherwise accessible from the command line such as MultiSubnetFailover, or to help protect sensitive credentials from being discoverable as command line arguments. For more information, see *DSN Support in sqlcmd and bcp* in Connecting with sqlcmd.

-e err_file

Specifies the full path of an error file used to store any rows that the **bcp** utility can't transfer from the file to the database. Error messages from the **bcp** command go to the workstation of the user. If this option isn't used, an error file isn't created.

If *err_file* begins with a hyphen (`-`) or a forward slash (`/`), don't include a space between `-e` and the *err_file* value.

-E

Specifies that identity value or values in the imported data file are to be used for the identity column. If `-E` isn't given, the identity values for this column in the data file being imported are ignored, and SQL Server automatically assigns unique values based on the seed and increment values specified during table creation. For more information, see DBCC CHECKIDENT.

If the data file doesn't contain values for the identity column in the table or view, use a format file to specify that the identity column in the table or view should be skipped when importing data; SQL Server automatically assigns unique values for the column.

The `-E` option has a special permissions requirement. For more information, see "Remarks" later in this article.

-f format_file

Specifies the full path of a format file. The meaning of this option depends on the environment in which it's used, as follows:

- If `-f` is used with the `format` option, the specified *format_file* is created for the specified table or view. To create an XML format file, also specify the `-x` option. For more information, see Create a Format File (SQL Server).
- If used with the `in` or `out` option, `-f` requires an existing format file.

> **Note**
>
> Using a format file in with the `in` or `out` option is optional. In the absence of the `-f` option, if `-n`, `-c`, `-w`, or `-N` isn't specified, the command prompts for format information and lets you save your responses in a format file (whose default file name is `bcp.fmt`).

If *format_file* begins with a hyphen (`-`) or a forward slash (`/`), don't include a space between `-f` and the *format_file* value.

-F first_row

Specifies the number of the first row to export from a table or import from a data file. This parameter requires a value greater than (`>`) 0 but less than (`<`) or equal to (`=`) the total number rows. In the absence of this parameter, the default is the first row of the file.

first_row can be a positive integer with a value up to 2^63-1. `-F` *first_row* is 1-based.

-G

Applies to: Azure SQL Database and Azure Synapse Analytics only.

This switch is used by the client when connecting to Azure SQL Database or Azure Synapse Analytics to specify that the user be authenticated with Microsoft Entra ID. The -G switch requires version 14.0.3008.27 or later versions. To determine your version, execute `bcp -v`. For more information, see Use Microsoft Entra authentication with SQL Database or Azure Synapse Analytics.

> **Important**

Microsoft Entra interactive authentication isn't currently supported on Linux or macOS. Microsoft Entra integrated authentication requires Microsoft ODBC Driver 17 for SQL Server version 17.6.1 and later versions, and a properly configured Kerberos environment.

Tip

To check if your version of **bcp** includes support for Microsoft Entra authentication, type `bcp --help` and verify that you see `-G` in the list of available arguments.

- **Microsoft Entra username and password**

  When you want to use a Microsoft Entra username and password, you can provide the `-G` option and also use the username and password by providing the `-U` and `-P` options.

  The following example exports data using Microsoft Entra username and password credentials. The example exports table `bcptest` from database `testdb` from Azure server `aadserver.database.windows.net` and stores the data in file `c:\last\data1.dat`:

  Windows Command Prompt Copy

  ```
  bcp bcptest out "c:\last\data1.dat" -c -S aadserver.database.windows.net -d
  ```

  The following example imports data using the credentials of a Microsoft Entra user. The example imports data from file `c:\last\data1.dat` into table `bcptest` for database `testdb` on Azure server `aadserver.database.windows.net` using a Microsoft Entra username and password:

  Windows Command Prompt Copy

  ```
  bcp bcptest in "c:\last\data1.dat" -c -S aadserver.database.windows.net -d
  ```

- **Microsoft Entra integrated**

  For Microsoft Entra integrated authentication, provide the `-G` option without a username or password. This configuration requires that the current Windows user account (the account the **bcp** command is running under) is federated with Microsoft Entra ID:

  The following example exports data using Microsoft Entra integrated authentication. The example exports table `bcptest` from database `testdb` on the logical server `aadserver.database.windows.net` and stores the data in file `c:\last\data2.dat`, using Windows credentials federated with Microsoft Entra ID:

  Windows Command Prompt Copy

  ```
  bcp bcptest out "c:\last\data2.dat" -S aadserver.database.windows.net -d te
  ```

  The following example imports data using Microsoft Entra integrated authentication. The example imports data from file table `c:\last\data2.dat` into table `bcptest` in the database `testdb` on the logical server `aadserver.database.windows.net`, using Windows credentials federated with Microsoft Entra ID:

  Windows Command Prompt Copy

  ```
  bcp bcptest in "c:\last\data2.dat" -S aadserver.database.windows.net -d tes
  ```

- **Microsoft Entra Managed Service Identity**

  Important

bcp is tightly coupled with the driver. The major versions of both **bcp** and the driver a DSN is created with must be the same. To determine your version, execute `bcp -v`.

Windows Linux

Exporting data via **bcp** using a Managed Service Identity on Windows requires a DSN be configured.

To configure a DSN on a machine running Windows:

1. Press the Windows key on your keyboard
2. Type `ODBC` and select the appropriate version of the **ODBC Data Source Administrator**
3. Select either the **User DSN** or **System DSN** tab
4. Select **Add** and follow the prompts
5. When asked for an authentication type select **Azure Managed Service Identity authentication**
6. If you have a User Assigned Managed Identity, paste the `Object (principal) ID` of the identity into the **Login ID** box at the bottom of the authentication tab
7. Continue following the prompts to configure your DSN

For a full walkthrough including screenshots, see Creating and editing DSNs in the UI.

Once the DSN is configured, **bcp** can then be called using `-D` flag to indicate the value passed for `-S` is a DSN.

Windows Command Prompt Copy

```
bcp bcptest out "c:\last\data1.dat" -c -D -S myDSN -d testdb
```

- **Microsoft Entra ID access token**

  Users of **bcp** (17.8 and later versions) can also authenticate with a token. The following examples use PowerShell on Linux to retrieve an access token.

  This example retrieves an access token and places it into a file to export data using a system-assigned managed identity.

  Bash Copy

```
Connect-AzAccount -Identity
$access_token | cut -f 1 | tr -d '\n' | iconv -f ascii -t UTF-16LE > /tmp/t
bcp bcptest out data2.dat -S aadserver.database.windows.net -d testdb -G -P
```

  This example passes a `Client ID` to the `-AccountId` parameter of `Connect-AzAccount` to retrieve an access token and place it into a token file. The token is then used to export data using the specified User Assigned Managed Identity.

  Bash Copy

```
Connect-AzAccount -Identity -AccountId 'client_id_of_user_assigned_managed_
$access_token | cut -f 1 | tr -d '\n' | iconv -f ascii -t UTF-16LE > /tmp/t
bcp bcptest out data2.dat -S aadserver.database.windows.net -d testdb -G -P
```

- **Microsoft Entra interactive**

  Microsoft Entra interactive authentication, available for all Azure SQL and SQL Server 2022+, allows you to use an interactive dialog to authenticate, which also supports multifactor authentication.

Microsoft Entra interactive authentication requires **bcp** version 15.0.1000.34 or later as well as ODBC version 17.2 or later.

To enable interactive authentication, provide the `-G` option with user name (`-U`) only, and no password.

The following example exports data using Microsoft Entra interactive authentication, which includes specifying the username of a Microsoft Entra account.

Interactive mode requires a password to be manually entered, or for accounts with multifactor authentication enabled, complete your configured MFA authentication method.

Windows Command Prompt Copy

```
bcp bcptest out "c:\last\data1.dat" -c -S aadserver.database.windows.net -d
```

If using a Microsoft Entra user that's a Windows account from a federated domain, the username entered in the command line must contain its domain (for example, `joe@contoso.com`):

Windows Command Prompt Copy

```
bcp bcptest out "c:\last\data1.dat" -c -S aadserver.database.windows.net -d
```

If guest users exist in a specific Microsoft Entra tenant and are part of a group that exists in SQL Database that has database permissions to execute the **bcp** command, their guest user alias is used (for example, `keith0@adventure-works.com`).

-h "hints [, ... *n*]"

Specifies the hint or hints to be used during a bulk import of data into a table or view.

- **ORDER (*column* [ASC | DESC] [, ...*n*])**

  The sort order of the data in the data file. Bulk import performance is improved if the data being imported is sorted according to the clustered index on the table, if any. If the data file is sorted in a different order, that is, other than the order of a clustered index key, or if there's no clustered index on the table, the ORDER clause is ignored. The column names supplied must be valid column names in the destination table. By default, **bcp** assumes the data file is unordered. For optimized bulk import, SQL Server also validates that the imported data is sorted.

- **ROWS_PER_BATCH = *bb***

  Number of rows of data per batch (as *bb*). Used when `-b` isn't specified, resulting in the entire data file being sent to the server as a single transaction. The server optimizes the bulkload according to the value *bb*. By default, ROWS_PER_BATCH is unknown.

- **KILOBYTES_PER_BATCH = *cc***

  Approximate number of kilobytes of data per batch (as *cc*). By default, KILOBYTES_PER_BATCH is unknown.

- **TABLOCK**

  Specifies that a bulk update table-level lock is acquired for the duration of the bulkload operation; otherwise, a row-level lock is acquired. This hint significantly improves performance because holding a lock for the duration of the bulk-copy operation reduces lock contention on the table. A table can be loaded concurrently from multiple clients if the table has no indexes and TABLOCK is specified. By default, locking behavior is determined by the table option **table lock on bulkload**.

> ⓘ Note
>
> If the target table is clustered columnstore index, TABLOCK hint isn't required for loading by multiple concurrent clients because each concurrent thread is assigned a separate rowgroup within the index and loads data into it. Please refer to columnstore index conceptual articles for details,

- CHECK_CONSTRAINTS

  Specifies that all constraints on the target table or view must be checked during the bulk-import operation. Without the CHECK_CONSTRAINTS hint, any CHECK, and FOREIGN KEY constraints are ignored, and after the operation the constraint on the table is marked as not-trusted.

  > ⓘ Note
  >
  > UNIQUE, PRIMARY KEY, and NOT NULL constraints are always enforced.

  At some point, you need to check the constraints on the entire table. If the table was nonempty before the bulk import operation, the cost of revalidating the constraint can exceed the cost of applying CHECK constraints to the incremental data. Therefore, we recommend that normally you enable constraint checking during an incremental bulk import.

  A situation in which you might want constraints disabled (the default behavior) is if the input data contains rows that violate constraints. With CHECK constraints disabled, you can import the data and then use Transact-SQL statements to remove data that isn't valid.

  > ⓘ Note
  >
  > **bcp** now enforces data validation and data checks that can cause scripts to fail if they're executed on invalid data in a data file.

  > ⓘ Note
  >
  > The `-m` *max_errors* switch doesn't apply to constraint checking.

- FIRE_TRIGGERS

  When specified with the *in* argument, any insert triggers defined on the destination table run during the bulk-copy operation. If FIRE_TRIGGERS isn't specified, no insert triggers will run. FIRE_TRIGGERS is ignored for the `out`, `queryout`, and `format` arguments.

-i input_file

Specifies the name of a response file, containing the responses to the command prompt questions for each data field when a bulk copy is being performed using interactive mode (`-n`, `-c`, `-w`, or `-N` not specified).

If *input_file* begins with a hyphen (`-`) or a forward slash (`/`), don't include a space between `-i` and the *input_file* value.

-k

Specifies that empty columns should retain a null value during the operation, rather than have any default values for the columns inserted. For more information, see Keep nulls or default values during bulk import (SQL Server).

-K application_intent

Declares the application workload type when connecting to a server. The only value that is possible is `ReadOnly`. If `-K` isn't specified, the **bcp** utility doesn't support connectivity to a secondary replica in an Always On availability group. For more information, see Offload read-only workload to secondary replica of an Always On availability group.

-l login_timeout

Specifies a login timeout. The `-l` option specifies the number of seconds before a login to SQL Server times out when you try to connect to a server. The default login timeout is 15 seconds. The login timeout must be a number between 0 and 65534. If the value supplied isn't numeric or doesn't fall into that range, **bcp** generates an error message. A value of 0 specifies an infinite timeout.

-L last_row

Specifies the number of the last row to export from a table or import from a data file. This parameter requires a value greater than (`>`) 0 but less than (`<`) or equal to (`=`) the number of the last row. In the absence of this parameter, the default is the last row of the file.

last_row can be a positive integer with a value up to 2^63-1.

-m max_errors

Specifies the maximum number of syntax errors that can occur before the **bcp** operation is canceled. A syntax error implies a data conversion error to the target data type. The *max_errors* total excludes any errors that can be detected only at the server, such as constraint violations.

A row that can't be copied by the **bcp** utility is ignored and is counted as one error. If this option isn't included, the default is 10.

Note

The `-m` option also doesn't apply to converting the **money** or **bigint** data types.

-n

Performs the bulk-copy operation using the native (database) data types of the data. This option doesn't prompt for each field; it uses the native values.

For more information, see Use native format to import or export data (SQL Server).

-N

Performs the bulk-copy operation using the native (database) data types of the data for noncharacter data, and Unicode characters for character data. This option offers a higher performance alternative to the `-w` option, and is intended for transferring data from one instance of SQL Server to another using a data file. It doesn't prompt for each field. Use this option when you're transferring data that contains ANSI extended characters and you want to take advantage of the performance of native mode.

For more information, see Use Unicode Native Format to Import or Export Data (SQL Server).

If you export and then import data to the same table schema by using **bcp** with `-N`, you might see a truncation warning if there's a fixed length, non-Unicode character column (for example, **char(10)**).

The warning can be ignored. One way to resolve this warning is to use `-n` instead of `-N`.

-o output_file

Specifies the name of a file that receives output redirected from the command prompt.

If *output_file* begins with a hyphen (`-`) or a forward slash (`/`), don't include a space between `-o` and the *output_file* value.

-P password

Specifies the password for the login ID. If this option isn't used, the **bcp** command prompts for a password. If this option is used at the end of the command prompt without a password, **bcp** uses the default password (NULL).

Important

Do not use a blank password. Use a strong password.

To mask your password, don't specify the `-P` option along with the `-U` option. Instead, after specifying **bcp** along with the `-U` option and other switches (don't specify `-P`), press ENTER, and the command will prompt you for a password. This method ensures that your password is masked when it's entered.

If *password* begins with a hyphen (`-`) or a forward slash (`/`), don't add a space between `-P` and the *password* value.

-q

Executes the SET QUOTED_IDENTIFIER ON statement in the connection between the **bcp** utility and an instance of SQL Server. Use this option to specify a database, owner, table, or view name that contains a space or a single quotation mark. Enclose the entire three-part table or view name in quotation marks ("").

To specify a database name that contains a space or single quotation mark, you must use the `-q` option.

`-q` doesn't apply to values passed to `-d`.

For more information, see Remarks, later in this article.

-r row_term

Specifies the row terminator. The default is **\n** (newline character). Use this parameter to override the default row terminator. For more information, see Specify Field and Row Terminators (SQL Server).

If you specify the row terminator in hexadecimal notation in a **bcp** command, the value is truncated at `0x00`. For example, if you specify `0x410041`, `0x41` is used.

If *row_term* begins with a hyphen (`-`) or a forward slash (`/`), don't include a space between `-r` and the *row_term* value.

-R

Specifies that currency, date, and time data is bulk copied into SQL Server using the regional format defined for the locale setting of the client computer. By default, regional settings are ignored.

-S server_name [\*instance_name*]

Specifies the instance of SQL Server to which to connect. If no server is specified, the **bcp** utility connects to the default instance of SQL Server on the local computer. This option is required when a **bcp** command is run from a remote computer on the network or a local named instance. To connect to the default instance of SQL Server on a server, specify only *server_name*. To connect to a named instance of SQL Server, specify *server_name***\***instance_name*.

-t field_term

Specifies the field terminator. The default is **\t** (tab character). Use this parameter to override the default field terminator. For more information, see Specify Field and Row Terminators (SQL Server).

If you specify the field terminator in hexadecimal notation in a **bcp** command, the value is truncated at `0x00`. For example, if you specify `0x410041`, `0x41` is used.

If *field_term* begins with a hyphen (`-`) or a forward slash (`/`), don't include a space between `-t` and the *field_term* value.

-T

Specifies that the **bcp** utility connects to SQL Server with a trusted connection using integrated security. The security credentials of the network user, *login_id*, and *password* aren't required. If `-T` isn't specified, you need to specify `-U` and `-P` to successfully connect.

Important

When the **bcp** utility is connecting to SQL Server with a trusted connection using integrated security, use the `-T` option (trusted connection) instead of the *username* and *password* combination. When the **bcp** utility is connecting to SQL Database or Azure Synapse Analytics, using Windows authentication or Microsoft Entra authentication isn't supported. Use the `-U` and `-P` options.

-U login_id

Specifies the login ID used to connect to SQL Server.

-v

Reports the **bcp** utility version number and copyright.

-V (80 | 90 | 100 | 110 | 120 | 130 | 140 | 150 | 160)

Performs the bulk-copy operation using data types from an earlier version of SQL Server. This option doesn't prompt for each field; it uses the default values.

- `80` = SQL Server 2000 (8.x)
- `90` = SQL Server 2005 (9.x)
- `100` = SQL Server 2008 (10.0.x) and SQL Server 2008 R2 (10.50.x)
- `110` = SQL Server 2012 (11.x)
- `120` = SQL Server 2014 (12.x)
- `130` = SQL Server 2016 (13.x)
- `140` = SQL Server 2017 (14.x)
- `150` = SQL Server 2019 (15.x)

- `160` = SQL Server 2022 (16.x)

For example, to generate data for types not supported by SQL Server 2000 (8.x), but were introduced in later versions of SQL Server, use the `-V80` option.

For more information, see Import native and character format data from earlier versions of SQL Server.

-w

Performs the bulk copy operation using Unicode characters. This option doesn't prompt for each field; it uses **nchar** as the storage type, no prefixes, **\t** (tab character) as the field separator, and **\n** (newline character) as the row terminator. `-w` isn't compatible with `-c`.

For more information, see Use unicode character format to import or export data (SQL Server).

-x

This option is used with the `format` and `-f` *format_file* options, and generates an XML-based format file instead of the default non-XML format file. The `-x` doesn't work when importing or exporting data. It generates an error if used without both `format` and `-f` *format_file*.

Remarks

- The **bcp** 13.0 client is installed when you install Microsoft SQL Server 2019 (15.x) tools. If tools are installed for multiple versions of SQL Server, depending on the order of values of the PATH environment variable, you might be using the earlier **bcp** client instead of the **bcp** 13.0 client. This environment variable defines the set of directories used by Windows to search for executable files. To discover which version you're using, run the `bcp -v` command at the Windows Command Prompt. For information about how to set the command path in the PATH environment variable, see Environment Variables or search for Environment Variables in Windows Help.

  To make sure the newest version of the **bcp** utility is running, you need to remove any older versions of the **bcp** utility.

  To determine where all versions of the **bcp** utility are installed, type in the command prompt:

  Windows Command Prompt

  ```
  where bcp.exe
  ```

- The **bcp** utility can also be downloaded separately from the Microsoft SQL Server 2016 Feature Pack. Select either `ENU\x64\MsSqlCmdLnUtils.msi` or `ENU\x86\MsSqlCmdLnUtils.msi`.

- XML format files are only supported when SQL Server tools are installed together with SQL Server Native Client.

- For information about where to find or how to run the **bcp** utility and about the command prompt utilities syntax conventions, see SQL Command Prompt Utilities (Database Engine).

- For information on preparing data for bulk import or export operations, see Prepare data for bulk export or import.

- For information about when row-insert operations that are performed by bulk import are logged in the transaction log, see Prerequisites for minimal logging in bulk import.

- Using additional special characters

  The characters `<`, `>`, `|`, `&`, and `^` are special command shell characters, and they must be preceded by the escape character (`^`), or enclosed in quotation marks when used in String (for example, `"StringContaining&Symbol"`). If you use quotation marks to enclose a string that contains one of the special characters, the quotation marks are set as part of the environment variable value.

Native data file support

In SQL Server, the **bcp** utility supports native data files compatible with SQL Server versions starting with SQL Server 2000 (8.x) and later.

Computed columns and timestamp columns

Values in the data file being imported for computed or **timestamp** columns are ignored, and SQL Server automatically assigns values. If the data file doesn't contain values for the computed or **timestamp** columns in the table, use a format file to specify that the computed or **timestamp** columns in the table should be skipped when importing data; SQL Server automatically assigns values for the column.

Computed and **timestamp** columns are bulk copied from SQL Server to a data file as usual.

Specify identifiers that contain spaces or quotation marks

SQL Server identifiers can include characters such as embedded spaces and quotation marks. Such identifiers must be treated as follows:

- When you specify an identifier or file name that includes a space or quotation mark at the command prompt, enclose the identifier in quotation marks ("").

  For example, the following `bcp out` command creates a data file named `Currency Types.dat`:

  Windows Command Prompt

  ```
  bcp AdventureWorks2022.Sales.Currency out "Currency Types.dat" -T -c
  ```

- To specify a database name that contains a space or quotation mark, you must use the `-q` option.

- For owner, table, or view names that contain embedded spaces or quotation marks, you can either:

  - Specify the `-q` option, or

  - Enclose the owner, table, or view name in brackets (`[]`) inside the quotation marks.

Data validation

bcp now enforces data validation and data checks that can cause scripts to fail if they're executed on invalid data in a data file. For example, **bcp** now verifies that:

- The native representations of float or real data types are valid.

- Unicode data has an even-byte length.

Forms of invalid data that could be bulk imported in earlier versions of SQL Server can fail to load now; whereas, in earlier versions, the failure didn't occur until a client tried to access the invalid data. The added validation minimizes surprises when querying the data after bulkload.

Bulk exporting or importing SQLXML documents

To bulk export or import SQLXML data, use one of the following data types in your format file.

Expand table

Data type	Effect
SQLCHAR or SQLVARYCHAR	The data is sent in the client code page or in the code page implied by the collation). The effect is the same as specifying the `-c` switch without specifying a format file.
SQLNCHAR or SQLNVARCHAR	The data is sent as Unicode. The effect is the same as specifying the `-w` switch without specifying a format file.
SQLBINARY or SQLVARYBIN	The data is sent without any conversion.

Permissions

A `bcp out` operation requires SELECT permission on the source table.

A `bcp in` operation minimally requires SELECT/INSERT permissions on the target table. In addition, ALTER TABLE permission is required if any of the following conditions are true:

- Constraints exist and the CHECK_CONSTRAINTS hint isn't specified.

  Note

  Disabling constraints is the default behavior. To enable constraints explicitly, use the `-h` option with the CHECK_CONSTRAINTS hint.

- Triggers exist and the FIRE_TRIGGER hint isn't specified.

  Note

  By default, triggers aren't fired. To fire triggers explicitly, use the `-h` option with the FIRE_TRIGGERS hint.

- You use the `-E` option to import identity values from a data file.

Note

Requiring ALTER TABLE permission on the target table was new in SQL Server 2005 (9.x). This new requirement can cause **bcp** scripts that don't enforce triggers and constraint checks to fail if the user account lacks ALTER table permissions for the target table.

Character mode (`-c`) and native mode (`-n`) best practices

This section has recommendations for character mode (`-c`) and native mode (`-n`).

- (Administrator/User) When possible, use native format (`-n`) to avoid the separator issue. Use the native format to export and import using SQL Server. Export data from SQL Server using the `-c` or `-w` option if the data will be imported to a non-SQL Server database.
- (Administrator) Verify data when using BCP OUT. For example, when you use BCP OUT, BCP IN, and then BCP OUT verify that the data is properly exported and the terminator values aren't used as part of some data value. Consider overriding the default terminators (using `-t` and `-r` options) with random hexadecimal values to avoid conflicts between terminator values and data values.
- (User) Use a long and unique terminator (any sequence of bytes or characters) to minimize the possibility of a conflict with the actual string value. This can be done by using the `-t` and `-r` options.

Examples

The examples in this section make use of the `WideWorldImporters` sample database for SQL Server 2016 (13.x) and later versions, Azure SQL Database, and Azure SQL Managed Instance. `WideWorldImporters` can be downloaded from https://github.com/Microsoft/sql-server-samples/releases/tag/wide-world-importers-v1.0. See RESTORE Statements for the syntax to restore the sample database.

Example test conditions

Except where specified otherwise, the examples assume that you use Windows Authentication and have a trusted connection to the server instance on which you're running the **bcp** command. A directory named `D:\BCP` is used in many of the examples.

The following script creates an empty copy of the `WideWorldImporters.Warehouse.StockItemTransactions` table and then adds a primary key constraint. Run the following T-SQL script in SQL Server Management Studio (SSMS)

SQL Copy

```sql
USE WideWorldImporters;
GO

SET NOCOUNT ON;

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Warehouse.StockItemTransac
BEGIN
    SELECT * INTO WideWorldImporters.Warehouse.StockItemTransactions_bcp
    FROM WideWorldImporters.Warehouse.StockItemTransactions
    WHERE 1 = 2;

    ALTER TABLE Warehouse.StockItemTransactions_bcp
    ADD CONSTRAINT PK_Warehouse_StockItemTransactions_bcp PRIMARY KEY NONCLUSTEF
    (StockItemTransactionID ASC);
END
```

You can truncate the `StockItemTransactions_bcp` table as needed:

SQL Copy

```sql
TRUNCATE TABLE WideWorldImporters.Warehouse.StockItemTransactions_bcp;
```

A. Identify bcp utility version

At a command prompt, enter the following command:

Windows Command Prompt

```
bcp -v
```

B. Copy table rows into a data file (with a trusted connection)

The following examples illustrate the `out` option on the `WideWorldImporters.Warehouse.StockItemTransactions` table.

- **Basic**

  This example creates a data file named `StockItemTransactions_character.bcp` and copies the table data into it using **character** format.

  At a command prompt, enter the following command:

  Windows Command Prompt

  ```
  bcp WideWorldImporters.Warehouse.StockItemTransactions out D:\BCP\StockItem
  ```

- **Expanded**

  This example creates a data file named `StockItemTransactions_native.bcp` and copies the table data into it using the **native** format. The example also: specifies the maximum number of syntax errors, an error file, and an output file.

  At a command prompt, enter the following command:

  Windows Command Prompt

  ```
  bcp WideWorldImporters.Warehouse.StockItemTransactions OUT D:\BCP\StockItem
  ```

Review `Error_out.log` and `Output_out.log`. `Error_out.log` should be blank. Compare the file sizes between `StockItemTransactions_character.bcp` and `StockItemTransactions_native.bcp`.

C. Copy table rows into a data file (with mixed-mode authentication)

The following example illustrates the `out` option on the `WideWorldImporters.Warehouse.StockItemTransactions` table. This example creates a data file named `StockItemTransactions_character.bcp` and copies the table data into it using **character** format.

The example assumes that you use mixed-mode authentication, and you must use the `-U` switch to specify your login ID. Also, unless you're connecting to the default instance of SQL Server on the local computer, use the `-S` switch to specify the system name and, optionally, an instance name.

At a command prompt, enter the following command: (The system prompts you for your password.)

Windows Command Prompt

```
bcp WideWorldImporters.Warehouse.StockItemTransactions out D:\BCP\StockItemTrans
```

D. Copy data from a file to a table

The following examples illustrate the `in` option on the `WideWorldImporters.Warehouse.StockItemTransactions_bcp` table using files created previously.

- **Basic**

  This example uses the `StockItemTransactions_character.bcp` data file previously created.

  At a command prompt, enter the following command:

  Windows Command Prompt Copy

  ```
  bcp WideWorldImporters.Warehouse.StockItemTransactions_bcp IN D:\BCP\StockI
  ```

- **Expanded**

  This example uses the `StockItemTransactions_native.bcp` data file previously created. The example also: use the hint `TABLOCK`, specifies the batch size, the maximum number of syntax errors, an error file, and an output file.

  At a command prompt, enter the following command:

  Windows Command Prompt Copy

  ```
  bcp WideWorldImporters.Warehouse.StockItemTransactions_bcp IN D:\BCP\StockI
  ```

  Review `Error_in.log` and `Output_in.log`.

E. Copy a specific column into a data file

To copy a specific column, you can use the `queryout` option. The following example copies only the `StockItemTransactionID` column of the `Warehouse.StockItemTransactions` table into a data file.

At a command prompt, enter the following command:

Windows Command Prompt Copy

```
bcp "SELECT StockItemTransactionID FROM WideWorldImporters.Warehouse.StockItemTr
```

F. Copy a specific row into a data file

To copy a specific row, you can use the `queryout` option. The following example copies only the row for the person named `Amy Trefl` from the `WideWorldImporters.Application.People` table into a data file `Amy_Trefl_c.bcp`.

Note

The `-d` switch is used identify the database.

At a command prompt, enter the following command:

Windows Command Prompt Copy

```
bcp "SELECT * from Application.People WHERE FullName = 'Amy Trefl'" queryout D:\
```

G. Copy data from a query to a data file

To copy the result set from a Transact-SQL statement to a data file, use the `queryout` option. The following example copies the names from the `WideWorldImporters.Application.People` table, ordered by full name, into the `People.txt` data file.

> **Note**
>
> The `-t` switch is used to create a comma-delimited file.

At a command prompt, enter the following command:

Windows Command Prompt

```
bcp "SELECT FullName, PreferredName FROM WideWorldImporters.Application.People O
```

H. Create format files

The following example creates three different format files for the `Warehouse.StockItemTransactions` table in the `WideWorldImporters` database. Review the contents of each created file.

At a command prompt, enter the following commands:

Windows Command Prompt

```
REM non-XML character format
bcp WideWorldImporters.Warehouse.StockItemTransactions format nul -f D:\BCP\Stoc

REM non-XML native format
bcp WideWorldImporters.Warehouse.StockItemTransactions format nul -f D:\BCP\Stoc

REM XML character format
bcp WideWorldImporters.Warehouse.StockItemTransactions format nul -f D:\BCP\Stoc
```

> **Note**
>
> To use the `-x` switch, you must be using a **bcp** 9.0 client. For information about how to use the **bcp** 9.0 client, see "Remarks."

For more information, see Use Non-XML format files (SQL Server) and XML Format Files (SQL Server).

I. Use a format file to bulk import with bcp

To use a previously created format file when importing data into an instance of SQL Server, use the `-f` switch with the `in` option. For example, the following command bulk copies the contents of a data file, `StockItemTransactions_character.bcp`, into a copy of the `Warehouse.StockItemTransactions_bcp` table by using the previously created format file, `StockItemTransactions_c.xml`.

> **Note**
>
> The `-L` switch is used to import only the first 100 records.

At a command prompt, enter the following command:

Windows Command Prompt

```
bcp WideWorldImporters.Warehouse.StockItemTransactions_bcp in D:\BCP\StockItemTr
```

ⓘ Note

Format files are useful when the data file fields are different from the table columns; for example, in their number, ordering, or data types. For more information, see Format files to import or export data (SQL Server).

J. Specify a code page

The following partial code example shows **bcp** import while specifying a code page 65001:

Windows Command Prompt

```
bcp MyTable in "D:\data.csv" -T -c -C 65001 -t , ...
```

K. Example output file using a custom field and row terminators

This example shows two sample files, generated by **bcp** using custom field and row terminators.

1. Create a table `dbo.T1` in the `tempdb` database, with two columns, `ID` and `Name`.

   SQL

   ```
   USE tempdb;
   GO

   CREATE TABLE dbo.T1 (ID INT, [Name] NVARCHAR(20));
   GO

   INSERT INTO dbo.T1 VALUES (1, N'Natalia');
   INSERT INTO dbo.T1 VALUES (2, N'Mark');
   INSERT INTO dbo.T1 VALUES (3, N'Randolph');
   GO
   ```

2. Generate an output file from the example table `dbo.T1`, using a custom field terminator.

   In this example, the server name is `MYSERVER`, and the custom field terminator is specified by `-t ,`.

   Windows Command Prompt

   ```
   bcp dbo.T1 out T1.txt -T -S MYSERVER -d tempdb -w -t ,
   ```

   Here's the result set.

   Output

   ```
   1,Natalia
   2,Mark
   3,Randolph
   ```

3. Generate an output file from the example table `dbo.T1`, using a custom field terminator and custom row terminator.

   In this example, the server name is `MYSERVER`, the custom field terminator is specified by `-t`, `, and the custom row terminator is specified by `-r :`.

Windows Command Prompt

```
bcp dbo.T1 out T1.txt -T -S MYSERVER -d tempdb -w -t , -r :
```

Here's the result set.

Output

```
1,Natalia:2,Mark:3,Randolph:
```

Note

The row terminator is always added, even to the last record. The field terminator, however, isn't added to the last field.

Additional examples

The following articles contain examples of using **bcp**:

- Data Formats for Bulk Import or Bulk Export (SQL Server)
  - Use native format to import or export data (SQL Server)
  - Use character format to import or export data (SQL Server)
  - Use Unicode Native Format to Import or Export Data (SQL Server)
  - Use unicode character format to import or export data (SQL Server)
- Specify Field and Row Terminators (SQL Server)
- Keep nulls or default values during bulk import (SQL Server)
- Keep identity values when bulk importing data (SQL Server)
- Format Files for Importing or Exporting Data (SQL Server)
  - Create a Format File (SQL Server)
  - Use a format file to bulk import data (SQL Server)
  - Use a Format File to Skip a Table Column (SQL Server)
  - Use a format file to skip a data field (SQL Server)
  - Use a format file to map table columns to data-file fields (SQL Server)
- Examples of bulk import and export of XML documents (SQL Server)

Considerations and limitations

- The **bcp** utility has a limitation that the error message shows only 512-byte characters. Only the first 512 bytes of the error message are displayed.

Related content

- Prepare data for bulk export or import
- BULK INSERT (Transact-SQL)
- OPENROWSET (Transact-SQL)
- SET QUOTED_IDENTIFIER (Transact-SQL)
- sp_configure (Transact-SQL)
- sp_tableoption (Transact-SQL)
- Format files to import or export data (SQL Server)

Get help

- Ideas for SQL: Have suggestions for improving SQL Server?
- Microsoft Q & A (SQL Server)
- DBA Stack Exchange (tag sql-server): Ask SQL Server questions
- Stack Overflow (tag sql-server): Answers to SQL development questions
- Reddit: General discussion about SQL Server
- Microsoft SQL Server License Terms and Information
- Support options for business users
- Contact Microsoft
- Additional SQL Server help and feedback

Contribute to SQL documentation

Did you know that you can edit SQL content yourself? If you do so, not only do you help improve our documentation, but you also get credited as a contributor to the page.

For more information, see How to contribute to SQL Server documentation

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A