The **dbexport** and **dbimport** utilities import and export a database and its schema to disk or tape.

The **dbexport** utility unloads an entire database into text files and creates a schema file. You can unload the database and its schema file either to disk or tape. If you prefer, you can unload the schema file to disk and unload the data to tape. You can use the schema file with the **dbimport** utility to re-create the database schema in another IBM Informix environment, and you can edit the schema file to modify the database that **dbimport** creates.

The **dbimport** utility creates a database and loads it with data from text files on tape or disk. The input files consist of a schema file that is used to re-create the database and data files that contain the database data. Normally, you generate the input files with the **dbexport** utility, but you can use any properly formatted input files.

The **dbexport** command unloads a database into text files that you can later import into another database. The command also creates a schema file.

Read syntax diagram[Skip visual syntax diagram](https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_116.htm?view=kc#d26590e89)

>>-dbexport----------------------------------------------------->

    .---------------------------------------------------.

    V |

>--+---+-----------------------------------------------+-+--*database*-+-><

   | +- -c ---------------------------------------+ |

   | +- -d-------------------------------------------+ |

   | +- -no-data-tables--*tablenames*------------------+ |

   | +- -no-data-tables-accessmethods--*accessmethods*-+ |

   | +- -nw------------------------------------------+ |

   | +- -q-------------------------------------------+ |

   | | (1) | |

   | +-| Destination Options |-----------------------+ |

   | +- -ss------------------------------------------+ |

   | +- -si------------------------------------------+ |

   | '- -X-------------------------------------------' |

   '-+-----------+---------------------------------------------------'

    +- -V-------+

    '- -version-'

| **Element** | **Purpose** | **Key Considerations** |
| --- | --- | --- |
| **-c** | Makes **dbexport** complete exporting unless a fatal error occurs |  |
| **-d** | Makes **dbexport** export simple-large-object descriptors only, not simple-large-object data |  |
| **-q** | Suppresses the display of error messages, warnings, and generated SQL data-definition statements | None. |
| **-ss** | Generates database server-specific information for all tables in the specified database |  |
| **-si** | Excludes the generation of index storage clauses for non-fragmented tables  The **-si** option is available only with the **-ss** option. |  |
| **-X** | Recognizes HEX binary data in character fields | None. |
| -no-data-tables | Prevents data from being exported for the specified tables. Only the definitions of the specified tables are exported. | Accepts a comma-separated list of names of tables for which data will not be exported.  **Default behavior** Only the definition of the **tsinstanceTable** table is exported, not the data. The data and definitions of all other tables are exported. |
| -no-data-tables-accessmethods | Prevents data from being unloaded using the specified access methods. | Accepts a comma-separated list of names of access methods. Tables using those access methods are not unloaded.  **Default value:**  ts\_rts\_vtam, ts\_vtam  Tables using ts\_rts\_vtam and ts\_vtam access methods are not unloaded. |
| **-nw** | Generates the SQL for creating a database without the specification of an owner | None. |
| **-V** | Displays the software version number and the serial number | None. |
| **-version** | Extends the **-V** option to display additional information about the build operating system, build number, and build date | None. |
| *database* | Specifies the name of the database that you want to export | **Additional information**: If your locale is set to use multibyte characters, you can use multibyte characters for the database name. |

You must have DBA privileges or log in as user **informix** to export a database.

**Global Language Support** When the environment variables are set correctly, as described in the *IBM Informix GLS User's Guide*, **dbexport** can handle foreign characters in data and export the data from GLS databases.

You can set the IFX\_UNLOAD\_EILSEQ\_MODE environment variable to enable **dbexport** to use character data that is invalid for the locale specified in the environment.

You can use delimited identifiers with the **dbexport** utility. The utility detects database objects that are keywords, mixed case, or have special characters, and the utility places double quotes around them.

In addition to the data files and the schema file, **dbexport** creates a file of messages named dbexport.out in the current directory. This file contains error messages, warnings, and a display of the SQL data definition statements that it generates. The same material is also written to standard output unless you specify the **-q** option.

During export, the database is locked in exclusive mode. If **dbexport** cannot obtain an exclusive lock, it displays a diagnostic message and exits.

**Tip** The **dbexport** utility can create files larger than 2 GB. To support such large files, make sure your operating system file-size limits are set sufficiently high. For example, on UNIX, set **ulimit** to unlimited.

**Example**

The following command exports the table definitions but no data for all the tables in the customer database.

dbexport stores\_demo -no-data-tables -no-data-tables-accessmethods=customer

**Example**

The following command generates the schema and data for the customer database without the specification of an owner:

dbexport stores\_demo -nw

The **dbexport** utility supports disk and tape destination options.

Read syntax diagram[Skip visual syntax diagram](https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_120.htm?view=kc#d28392e53)

Destination options

|--+---------------------------------------------------------------+--|

   +- -o--*directory*------------------------------------------------+

   '- -t--*device*-- -b--*blocksize*-- -s--*tapesize*--+---------------+-'

    '- -f--*pathname*-'

| **Element** | **Purpose** | **Key Considerations** |
| --- | --- | --- |
| -b *blocksize* | Specifies, in kilobytes, the block size of the tape device. | None. |
| -f *pathname* | Specifies the name of the path where you want the schema file stored, if you are storing the data files on tape. | The path name can be a complete path name or a file name. If only a file name is given, the file is stored in the current directory.  If you do not use the -f option, the SQL source code is written to the tape. |
| -o *directory* | Specifies the directory on disk in which **dbexport** creates the *database*.exp directory.  This directory holds the data files and the schema file that **dbexport** creates for the database. | The specified directory must exist. |
| -s *tapesize* | Specifies, in kilobytes, the amount of data that you can store on the tape. | To write to the end of the tape, set the value to 0.  If you do not specify 0, the maximum size is 2 097 151 KB. |
| -t *device* | Specifies the path name of the tape device where you want the text files and, possibly, the schema file stored. | You cannot specify a remote tape device. |

When you write to disk, **dbexport** creates a subdirectory, *database*.exp, in the directory that the -o option specifies. The **dbexport** utility creates a file with the .unl extension for each table in the database. The schema file is written to the file *database*.sql. The .unl and .sql files are in the *database*.exp directory.

If you do not specify a destination for the data and schema files, the subdirectory *database*.exp is placed in the current working directory.

When you write the data files to tape, you can use the -f option to store the schema file to disk. You are not required to name the schema file *database*.sql. You can give it any name.

**UNIX/Linux Only**

For database servers on UNIX or Linux, the command is:

dbexport //finland/reports

The following command exports the database **stores\_demo** to tape with a block size of 16 KB and a tape capacity of 24 000 KB. The command also writes the schema file to /tmp/stores\_demo.imp.

dbexport -t /dev/rmt0 -b 16 -s 24000 -f /tmp/stores\_demo.imp

stores\_demo

The following command exports the same **stores\_demo** database to the directory named /work/exports/stores\_demo.exp. The resulting schema file is /work/exports/stores\_demo.exp/stores\_demo.sql.

dbexport -o /home/informix/labs/work/exports stores\_demo

**Windows Only**

For Windows, the following command exports the database **stores\_demo** to tape with a block size of 16 KB and a tape capacity of 24 000 KB. The schema file is written to C:\temp\stores\_demo.imp.

dbexport -t \\.\TAPE2 -b 16 -s 24000 -f

C:\temp\stores\_demo.imp stores\_demo

The following command exports the same **stores\_demo** database to the directory named D:\work\exports\stores\_demo.exp. The resulting schema file is D:\work\exports\stores\_demo.exp\stores\_demo.sql.

dbexport -o D:\work\exports stores\_demo

**dbimport:**

The **dbimport** command imports previously exported data into another database.

Read syntax diagram[Skip visual syntax diagram](https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_123.htm?view=kc#d29439e94)

    .----------.

    V | (1)

>>-dbimport----+------+-+--| Input-File Location |-------------->

    +- -c--+

    +- -D--+

    +- -nv-+

    +- -q--+

    '- -X--'

    (2)

>--| Create Options |------+-----------+--*database*-------------><

    +- -V-------+

    '- -version-'

| **Element** | **Purpose** | **Key Considerations** |
| --- | --- | --- |
| **-c** | Completes importing data even when certain nonfatal errors occur |  |
| -D | Specifies a default extent size of 16 KB for the first and subsequent extents during the import operation, if the extent sizes are not specified in the CREATE TABLE statement. | This option is ignored if the extent sizes are specified in the CREATE TABLE statement.  Default values help to ensure that enough space is available in the dbspace that is designated for the import operation. This option prevents the automatic calculation of extent sizes during the import operation, and is useful especially in the following situations:   * When importing tables that contain columns with large maximum row sizes, such as LVARCHAR columns. * When importing data after the **dbexport** command was run without the **-ss** option. The **-ss** option specifies server-specific information about extent sizes. |
| **-nv** | While the **dbimport -nv** command is running, tables with foreign-key constraints that ALTER TABLE ADD CONSTRAINT creates in enabled or filtering mode are not checked for violations, as if you had also specified NOVALIDATE | By bypassing the checking of referential constraints, this option can reduce migration time for very large tables that already conform to their foreign-key constraints. The NOVALIDATE mode does not persist after the ALTER TABLE ADD CONSTRAINT statement has completed. |
| **-q** | Suppresses the display of error messages, warnings, and generated SQL data-definition statements | None. |
| **-V** | Displays the software version number and the serial number | None. |
| **-version** | Extends the **-V** option to display additional information about the build operating system, build number, and build date | None. |
| -**X** | Recognizes HEX binary data in character fields | None. |
| *database* | Declares the name of the database to create |  |

The **dbimport** utility can use files from the following location options:

* All input files are on disk.
* All input files are on tape.
* The schema file is on disk, and the data files are on tape.

The **dbimport** utility supports the following tasks for an imported Informix database server:

* Specify the dbspace where the database will reside
* Create an ANSI-compliant database with unbuffered logging
* Create a database that supports explicit transactions (with buffered or unbuffered logging)
* Create an unlogged database
* Create a database with the NLS case-insensitive property for NCHAR and NVARCHAR strings.
* Process all ALTER TABLE ADD CONSTRAINT and SET CONSTRAINTS statements in the .sql file of the exported database that define enabled or filtering referential constraints so that any foreign-key constraints that are not specified as DISABLED are in ENABLED NOVALIDATE or in FILTERING NOVALIDATE mode.

The user who runs the **dbimport** utility is granted the DBA privilege on the newly created database. The **dbimport** process locks each table as it is being loaded and unlocks the table when the loading is complete.

The input-file location specifies the location of the database.exp directory, which contains the files that the **dbimport** utility imports.

If you do not specify an input-file location, **dbimport** searches for data files in the directory database.exp under the current directory and for the schema file in database.exp/database.sql.

Read syntax diagram[Skip visual syntax diagram](https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_126.htm?view=kc#d30871e77)

dbimport input-file location

|--+-------------------------------------------------------------------+--|

   +- -i--directory----------------------------------------------------+

   '- -t--device--+-------------------------------+--+---------------+-'

    '- -b--blocksize-- -s--tapesize-' '- -f--pathname-'

| **Element** | **Purpose** | **Key Considerations** |
| --- | --- | --- |
| -b blocksize | Specifies, in kilobytes, the block size of the tape device | If you are importing from tape, you must use the same block size that you used to export the database.  If you do not use the -b option, the default block size is 1. |
| -f pathname | Specifies where **dbimport** can find the schema file to use as input to create the database when the data files are read from tape | If you use the -f option to export a database, you typically use the same path name that you specified in the **dbexport** command. If you specify only a file name, **dbimport** looks for the file in the .exp subdirectory of your current directory.  If you do not use the -f option, the SQL source code is written to the tape. |
| -i directory | Specifies the complete path name on disk of the database.exp directory, which holds the input data files and schema file that **dbimport** uses to create and load the new database. The directory name must be the same as the database name. | This directory must be the same directory that you specified with the **dbexport -o** option. If you change the directory name, you also rename your database. |
| -s tapesize | Specifies, in kilobytes, the amount of data that you can store on the tape | To read to the end of the tape, specify a tape size of 0.  If you are importing from tape, you must use the same tape size that you used to export the database. The maximum size is 2 097 151 KB.  If you do not use the -s option, the default value is 0 (read to the end of the tape). |
| -t device | Specifies the path name of the tape device that holds the input files | You cannot specify a remote tape device. |

## Examples showing input file location on UNIX or Linux

To import the **stores\_demo** database from a tape with a block size of 16 KB and a capacity of 24 000 KB, issue this command:

dbimport -c -t /dev/rmt0 -b 16 -s 24000 -f

/tmp/stores\_demo.imp stores\_demo

The schema file is read from /tmp/stores\_demo.imp.

To import the **stores\_demo** database from the stores\_demo.exp directory under the /work/exports directory, issue this command:

dbimport -c -i /work/exports stores\_demo

The schema file is assumed to be /work/exports/stores\_demo.exp/stores\_demo.sql.

## Examples showing input file location on Windows

To import the **stores\_demo** database from a tape with a block size of 16 KB and a capacity of 24 000 KB, issue this command:

dbimport -c -t \\.\TAPEDRIVE -b 16 -s 24000 -f

C:\temp\stores\_demo.imp stores\_demo

The schema file is read from C:\temp\stores\_demo.imp.

To import the **stores\_demo** database from the stores\_demo.exp directory under the D:\work\exports directory, issue this command:

dbimport -c -i D:\work\exports stores\_demo

The schema file is assumed to be D:\work\exports\stores\_demo.exp\stores\_demo.sql.

The **dbimport** utility supports options for creating a database, specifying a dbspace for that database, defining logging options, and optionally specifying ANSI/ISO-compliance or NLS case-insensitivity (or both) as properties of the database.

Read syntax diagram[Skip visual syntax diagram](https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.mig.doc/ids_mig_127.htm?view=kc#d31650e52)

Create options

|--+--------------+--+-----------------------+--+------+--------|

   '- -d--dbspace-' '- -l--+-+----------+-+-' '- -ci-'

    | '-buffered-' |

    '- -ansi-------'

| **Element** | **Purpose** | **Key Considerations** |
| --- | --- | --- |
| **-ansi** | Creates an ANSI/ISO-compliant database in which the ANSI/ISO rules for transaction logging are enabled. Otherwise, the database uses explicit transactions by default. | If you omit the **-ansi** option, the database uses explicit transactions.  **Additional Information:** For more information about ANSI/ISO-compliant databases, see the IBM® Informix Guide to SQL: Reference. |
| **-ci** | Specifies the NLS case-insensitive property. Otherwise, the database is case-sensitive by default. | **Additional Information:** See the IBM Informix Guide to SQL: Syntax and IBM Informix Guide to SQL: Reference descriptions of the NLS case-insensitive property. |
| **-d** **dbspace** | Specifies the dbspace where the database is created. . | If this is omitted, the default location is the **root** dbspace |
| **-l** | Establishes unbuffered transaction logging for the imported database. If the **-l** flag is omitted, the database is unlogged, |  |
| **-l buffered** | Establishes buffered transaction logging for the imported database. If **-l** is included but **buffered** is omitted, the database uses unbuffered logging. |  |

If you created a table or index fragment containing partitions in Informix® Version 10.00 or a later version of the Informix database server, you must use syntax containing the partition name when importing a database that contains multiple partitions within a single dbspace. See the IBM Informix Guide to SQL: Syntax for syntax details.

## Example showing dbimport create options (UNIX or Linux)

To import the **stores\_demo** database from the **/usr/informix/port/stores\_demo.exp** directory, issue this command:

dbimport -c stores\_demo -i /usr/informix/port -l -ansi

The new database is ANSI/ISO-compliant.

The next example similarly imports the **stores\_demo** database from the **/usr/informix/port/stores\_demo.exp** directory. The imported database uses buffered transaction logging and explicit transactions. The **-ci** flag specifies case insensitivity in queries and in other operations on columns and character strings of the NCHAR and NVARCHAR data types:

dbimport -c stores\_demo -i /usr/informix/port -l buffered -ci

The **-ansi** and **-ci** options for database properties are not mutually exclusive. You can specify an ANSI/ISO-compliant database that is also NLS case-insensitive, as in the following example of the **dbimport** command:

dbimport -c stores\_demo -i /usr/informix/port -l -ansi -ci

## Example showing dbimport create options (Windows)

To import the **stores\_demo** database from the **C:\USER\informix\port\stores\_demo.exp** directory, issue this command:

dbimport -c stores\_demo -i C:\USER\informix\port -l -ansi

The imported database is ANSI/ISO-compliant and is case-sensitive for all built-in character data types.