




Examples of bulk import and export of XML documents (SQL Server)

Article • 06/22/2023

Applies to:  SQL Server 2016 (13.x) and later  [Azure SQL Database](#)  [Azure SQL Managed Instance](#)

You can bulk import XML documents into a SQL Server database, or bulk export them from a SQL Server database. This article provides examples of both.

To bulk import data from a data file into a SQL Server table or non-partitioned view, you can use the following options:

- **bcp** utility

You can also use the **bcp** utility to export data from anywhere in a SQL Server database that a `SELECT` statement works, including partitioned views.

- `BULK INSERT`
- `INSERT ... SELECT * FROM OPENROWSET(BULK...)`

For more information, see the following articles:

- [Import and export bulk data using bcp \(SQL Server\)](#)
- [Use BULK INSERT or OPENROWSET\(BULK...\) to import data to SQL Server](#)
- [How to import XML into SQL Server with the XML Bulk Load component](#)
- [XML schema collections \(SQL Server\)](#)

Examples

- [A. Bulk importing XML data as a binary byte stream](#)
- [B. Bulk importing XML data in an existing row](#)
- [C. Bulk importing XML data from a file that contains a DTD](#)
- [D. Specifying the field terminator explicitly using a format file](#)
- [E. Bulk exporting XML data](#)

Bulk importing XML data as a binary byte stream

When you bulk import XML data from a file that contains an encoding declaration that you want to apply, specify the `SINGLE_BLOB` option in the `OPENROWSET(BULK...)` clause. The `SINGLE_BLOB` option ensures that the XML parser in SQL Server imports the data according to the encoding scheme specified in the XML declaration.

Sample table

To test example A, create sample table `T`.

SQL

```
USE tempdb;
GO

CREATE TABLE T (
    IntCol INT IDENTITY(1,1),
    XmlCol XML
);
GO
```

Sample data file

Before you can run example A, you must create a UTF-8 encoded file (`C:\SampleFolder\SampleData3.txt`) that contains the following sample instance that specifies the UTF-8 encoding scheme.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <ProductDescription ProductModelID="5">
    <Summary>Some Text</Summary>
  </ProductDescription>
</Root>
```

Example A

This example uses the `SINGLE_BLOB` option in an `INSERT ... SELECT * FROM OPENROWSET(BULK...)` statement to import data from a file named `SampleData3.txt` and insert an XML instance in the single-column table, sample table `T`.

SQL

```
INSERT INTO T (XmlCol)
SELECT *
FROM OPENROWSET(
    BULK 'C:\SampleFolder\SampleData3.txt',
    SINGLE_BLOB)
AS x;
```

You can also explicitly specify the column names as follows:

SQL

```
INSERT INTO T (
    XmlCol
)
SELECT
    x.BulkColumn
FROM OPENROWSET(
    BULK 'C:\SampleFolder\SampleData3.txt',
    SINGLE_BLOB)
AS x;
```

Remarks

By using `SINGLE_BLOB` in this case, you can avoid a mismatch between the encoding of the XML document (as specified by the XML encoding declaration) and the string codepage implied by the server.

If you use `NCLOB` or `CLOB` data types and run into a codepage or encoding conflict, you must do one of the following:

- Remove the XML declaration to successfully import the contents of the XML data file.
- Specify a code page in the `CODEPAGE` option of the query that matches the encoding scheme that is used in the XML declaration.
- Match, or resolve, the database collation settings with a non-Unicode XML encoding scheme.

[\[Top\]](#)

Bulk importing XML data in an existing row

This example uses the `OPENROWSET` bulk rowset provider to add an XML instance to an

existing row or rows in sample table `T`.

Note

To run this example, you must first complete the test script provided in example A. That example creates the `tempdb.dbo.T` table and bulk imports data from `SampleData3.txt`.

Sample data file

Example B uses a modified version of the `SampleData3.txt` sample data file from the preceding example. To run this example, modify the content of this file as follows:

XML

```
<Root>
  <ProductDescription ProductModelID="10">
    <Summary>Some New Text</Summary>
  </ProductDescription>
</Root>
```

Example B

SQL

```
-- Query before update shows initial state of XmlCol values.
SELECT * FROM T;

UPDATE T
SET XmlCol = (
  SELECT *
  FROM OPENROWSET(
    BULK 'C:\SampleFolder\SampleData3.txt',
    SINGLE_BLOB
  ) AS x
)
WHERE IntCol = 1;
GO
```

[\[Top\]](#)

Bulk importing XML data from a file that contains a DTD

Important

We recommended that you don't enable support for Document Type Definitions (DTDs) if it is not required in your XML environment. Turning on DTD support increases the attackable surface area of your server, and may expose it to a denial-of-service attack. If you must enable DTD support, you can reduce this security risk by processing only trusted XML documents.

You may get the following error, when you use **bcp** to import XML data from a file that contains a DTD:

Output

```
SQLState = 42000, NativeError = 6359

Error = [Microsoft][SQL Server Native Client][SQL Server]Parsing XML with
internal subset DTDs not allowed.
Use CONVERT with style option 2 to enable limited internal subset DTD
support.

BCP copy %s failed
```

To work around this problem, you can import XML data from a data file that contains a DTD by using the `OPENROWSET(BULK...)` function and then specifying the `CONVERT` option in the `SELECT` clause of the command. The basic syntax for the command is:

```
INSERT ... SELECT CONVERT(...) FROM OPENROWSET(BULK...)
```

Sample data file

Before you can test this bulk import example, create a file (`C:\SampleFolder\Dtddfile.xml`) that contains the following sample instance:

XML

```
<!DOCTYPE DOC [<!ATTLIST elem1 attr1 CDATA "defVal1">]>
<elem1>January</elem1>
```

Sample table

Example C uses the `T1` sample table that is created by the following `CREATE TABLE`

statement:

SQL

```
USE tempdb;  
GO  
CREATE TABLE T1(XmlCol XML);  
GO
```

Example C

This example uses `OPENROWSET(BULK...)` and specifies the `CONVERT` option in the `SELECT` clause to import the XML data from `Dtdfile.xml` into sample table `T1`.

SQL

```
INSERT INTO T1  
SELECT CONVERT(XML, BulkColumn, 2)  
FROM OPENROWSET(  
    BULK 'C:\SampleFolder\Dtdfile.xml',  
    SINGLE_BLOB  
    ) AS [rowsetresults];
```

After the `INSERT` statement executes, the DTD is stripped from the XML and stored in the `T1` table.

[\[Top\]](#)

Specify the field terminator explicitly using a format file

The following example shows how to bulk import the following XML document, `Xmltable.dat`.

Sample data file

The document in `xmltable.dat` contains two XML values, one for each row. The first XML value is encoded with UTF-16, and the second value is encoded with UTF-8.

The contents of this data file are shown in the following hex dump:

Output

```
FF FE 3C 00 3F 00 78 00-6D 00 6C 00 20 00 76 00  *..\<?.x.m.l. .v.*
```

```

65 00 72 00 73 00 69 00-6F 00 6E 00 3D 00 22 00 *e.r.s.i.o.n.=.".*
31 00 2E 00 30 00 22 00-20 00 65 00 6E 00 63 00 *1...0." .e.n.c.*
6F 00 64 00 69 00 6E 00-67 00 3D 00 22 00 75 00 *o.d.i.n.g.=."u.*
74 00 66 00 2D 00 31 00-36 00 22 00 3F 00 3E 00 *t.f.-.1.6."?.>.*
3C 00 72 00 6F 00 6F 00-74 00 3E 00 A2 4F 9C 76 *\

```

Sample table

When you bulk import or export an XML document, you should use a [field terminator](#) that can't possibly appear in any of the documents; for example, a series of four nulls (\0) followed by the letter z: \0\0\0\0z.

This example shows how to use this field terminator for the `xTable` sample table. To create this sample table, use the following `CREATE TABLE` statement:

SQL

```

USE tempdb;
GO
CREATE TABLE xTable (xCol XML);
GO

```

Sample format file

The field terminator must be specified in the format file. Example D uses a non-XML format file named `xmltable.fmt` that contains the following output:

Output

```

9.0
1
1      SQLBINARY      0      0      "\0\0\0\0z"      1      xCol
""

```

You can use this format file to bulk import XML documents into the `xTable` table by using a `bcp` command or a `BULK INSERT` or `INSERT ... SELECT * FROM OPENROWSET(BULK...)` statement.

Example D

This example uses the `xmltable.fmt` format file in a `BULK INSERT` statement to import the contents of an XML data file named `xmltable.dat`.

SQL

```
BULK INSERT xTable
FROM 'C:\SampleFolder\xmltable.dat'
WITH (FORMATFILE = 'C:\SampleFolder\xmltable.fmt');
GO
```

[\[Top\]](#)

Bulk exporting XML data

The following example uses `bcp` to bulk export XML data from the table that is created in the preceding example by using the same XML format file. In the following `bcp` command, `<server_name>` and `<instance_name>` represent placeholders that must be replaced with appropriate values:

Windows Command Prompt

```
bcp bulktest..xTable out a-wn.out -N -T -S<server_name>\<instance_name>
```

Note

SQL Server does not save the XML encoding when XML data is persisted in the database. Therefore, the original encoding of XML fields is not available when XML data is exported. SQL Server uses UTF-16 encoding when exporting XML data.

See also

- [INSERT \(Transact-SQL\)](#)

- [SELECT Clause \(Transact-SQL\)](#)
- [bcp utility](#)
- [Bulk Import and Export of Data \(SQL Server\)](#)
- [BULK INSERT \(Transact-SQL\)](#)
- [OPENROWSET \(Transact-SQL\)](#)