[MS-BCP]:

Bulk Copy Format

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
6/4/2010	0.1	Major	First release.
9/3/2010	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/9/2011	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/7/2011	1.0	None	No changes to the meaning, language, or formatting of the technical content.
11/3/2011	1.0	None	No changes to the meaning, language, or formatting of the technical content.
1/19/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/23/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
3/27/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
5/24/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
6/29/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/23/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
3/26/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
6/11/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
12/5/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2014	2.0	Major	Updated and revised the technical content.
5/20/2014	2.0	None	No changes to the meaning, language, or formatting of the technical content.
5/10/2016	3.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The bulk copy (BCP) format is a data structure format that specifies how different database data type values are stored in a data file for the purpose of exporting and importing large sets of values. The BCP format also specifies what each data column represents in a format file for the purpose of interpreting the set of values stored in the corresponding data file.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

OCTET: Any 8-bit value in the range from 0x00 through 0xFF.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

Unicode string: A **Unicode** 8-bit string is an ordered sequence of 8-bit units, a **Unicode** 16-bit string is an ordered sequence of 16-bit code units, and a **Unicode** 32-bit string is an ordered sequence of 32-bit code units. In some cases, it could be acceptable not to terminate with a terminating null character. Unless otherwise specified, all **Unicode strings** follow the UTF-16LE encoding scheme with no Byte Order Mark (BOM).

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

[XMLSCHEMA1/2] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures Second Edition", W3C Recommendation, October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

1.2.2 Informative References

[MSDN-BCPU] Microsoft Corporation, "bcp Utility", http://msdn.microsoft.com/en-us/library/ms162802(SQL.105).aspx

[MSDN-DTTS] Microsoft Corporation, "Data Types (Transact-SQL)", http://msdn.microsoft.com/en-us/library/ms187752.aspx

[MSDN-hierarchyid] Microsoft Corporation, "hierarchyid (Transact-SQL)", http://msdn.microsoft.com/en-us/library/bb677290.aspx

[MSDN-SSCN] Microsoft Corporation, "SQL Server Collation Name (Transact-SQL)", http://msdn.microsoft.com/en-us/library/ms180175.aspx

[MSDN-UFFMFC] Microsoft Corporation, "Using a Format File to Map Fields to Columns During Bulk Import", http://msdn.microsoft.com/en-us/library/ms190396.aspx

[MSDN-WSDDE] Microsoft Corporation, "Working with Spatial Data (Database Engine)", http://msdn.microsoft.com/en-us/library/bb933876.aspx

1.3 Overview

The Bulk Copy Format (BCP) is a data structure format that is used to specify how different database server data type values are stored in a file when importing or exporting bulk data to and from the server. This data structure specifies how the bcp.exe utility reads data stored in a file and the identification of that data. For more information, see [MSDN-BCPU].

1.4 Relationship to Protocols and Other Structures

The **BCP** structure is independent of any application or network protocol or structures.

1.5 Applicability Statement

The **BCP** structure is appropriate for importing or exporting data between two relational database management system (RDMS) instances.

1.6 Versioning and Localization

Structure Versions: There are no versioning issues for the **BCP** format.

Localization: This data structure specifies all values as **Unicode** characters.

1.7 Vendor-Extensible Fields

None.

2 Structures

2.1 Data Types

Detailed data structure representation of each of the database data types is specified in the following subsections.<1> The data structures are defined in **Augmented Backus-Naur Form (ABNF)** notation [RFC5234]. For more information about these database data types, see [MSDN-DTTS].

2.1.1 BigInt

The **BigInt** data type supports a range of values from -2^{63} (-9,223,372,036,854,775,808) through $2^{63}-1$ (9,223,372,036,854,775,807). The values of this data type are represented in simple Unicode string format, as follows.

```
BigInt = ["-"]1*19DIGIT
```

2.1.2 Binary

The **Binary** data type is a user-defined fixed number of bytes that has a supported number of bytes that range from 1 through 8000 bytes. The values of this data type are represented in hexadecimal-encoded Unicode string format, as follows.

```
Binary = 32000OCTET
```

For example, 0x56 => "56". Because "56" is in Unicode, the end result is 4 **OCTET** for every binary byte value. The representation does not require the 0x prefix.

2.1.3 Bit

The **Bit** data type is a Boolean that supports 0 or 1 as the data value. The values of this data type are represented in simple Unicode string format, as follows.

```
Bit = "0" / "1"
```

2.1.4 Char

The **Char** data type is a user-defined fixed-length single-byte character string that has a supported number of single byte characters that range from 1 through 8000. If a particular string value does not use the entire user-defined fixed length, the remaining characters are padded with the space character. The values of this data type are represented in simple Unicode string format, as follows.

```
Char = 16000OCTET
```

The single-byte characters are converted to their corresponding Unicode characters.

2.1.5 CLRUDT

The **CLRUDT** data type defines a set of custom user-defined types, such as the **hierarchyID** and **Spatial** data types.<2> For more information about the **hierarchyID** data type, see [MSDN-hierarchyid]. For more information about the **Spatial** data type, see [MSDN-WSDDE].

The values of the **CLRUDT** data types are treated as **VarBinary** values. The values of the **CLRUDT** data type are represented in hexadecimal-encoded Unicode string format, as follows.

```
CLRUDT = 0*nOCTET
```

In this format, *n* = 4 x (2,147,483,647). The representation does not require the 0x prefix.

2.1.6 Date

The **Date** data type<3> supports a value range from 0001-01-01 through 9999-12-31. The values of this data type are represented in the Unicode YYYY-MM-DD string format, as follows.

```
Year = ("000"(%x31-39)) / ("00"(%x31-39)DIGIT) / ("0"(%x31-39)2DIGIT)
Year =/ ((%x31-39)3DIGIT)
Month = ("0"(%x31-39)) / ("1"("0"/"1"/"2"))
Day = ("0"(%x31-39)) / (("1"/"2")DIGIT) / ("3"("0"/"1"))
Date = Year "-"  Month "-" Day
```

2.1.7 DateTime

The **DateTime** data type supports a value range from 1753-01-01 00:00:00.000 through 9999-12-31 23:59:59.997. The values of this data type are represented in the Unicode YYYY-MM-DD hh:mm:ss[.nnn] string format, as follows.

```
Year = ("175"(%x33-39)) / ("17"(%x36-39)DIGIT) / ("1"("8"/"9")2DIGIT)
Year =/ ((%x32-39)3DIGIT)
Month = ("0"(%x31-39)) / ("1"("0"/"1"/"2"))
Day = ("0"(%x31-39)) / (("1"/"2")DIGIT) / ("3"("0"/"1"))
Hour = (("0"/"1")DIGIT) / ("2"(%x30-33))
MinSec = ":"(%x30-35)DIGIT
DateTime = Year "-" Month "-" Day SP Hour 2MinSec ["." 2DIGIT ("0"/"3"/"7")]
```

2.1.8 DateTime2

The **DateTime2** data type<4> supports a value range from 0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999. The values of this data type are represented in the Unicode YYYY-MM-DD hh:mm:ss[.nnnnnnn] string format, as follows.

```
Year = ("000"(%x31-39)) / ("00"(%x31-39)DIGIT) / ("0"(%x31-39)2DIGIT)
Year =/ ((%x31-39)3DIGIT)
Month = ("0"(%x31-39)) / ("1"("0"/"1"/"2"))
Day = ("0"(%x31-39)) / (("1"/"2")DIGIT) / ("3"("0"/"1"))
Hour = (("0"/"1")DIGIT) / ("2"(%x30-33))
MinSec = ":"(%x30-35)DIGIT
DateTime2 = Year "-" Month "-" Day SP Hour 2MinSec ["." 7DIGIT]
```

2.1.9 DateTimeOffset

The **DateTimeOffset** data type<5> supports a value range from 0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 in the Coordinated Universal Time (UTC) time zone. The values of this data type are represented in the Unicode YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh:mm] string format, as follows.

```
Year = ("000"(%x31-39)) / ("00"(%x31-39)DIGIT) / ("0"(%x31-39)2DIGIT)
Year =/ ((%x31-39)3DIGIT)
Month = ("0"(%x31-39)) / ("1"("0"/"1"/"2"))
Day = ("0"(%x31-39)) / (("1"/"2")DIGIT) / ("3"("0"/"1"))
Hour = (("0"/"1")DIGIT) / ("2"(%x30-33))
MinSec = ":"(%x30-35)DIGIT
OffsetHour = ("0"DIGIT) / ("1"(%x30-33))
OffSetMin = ((%x30-35)DIGIT)) / ("14:00")
DateTimeOffset = Year "-" Month "-" Day SP Hour 2MinSec ["." 7DIGIT] [SP ("+"/"-") OffsetHour
":" OffsetMin]
```

2.1.10 Decimal

The **Decimal** data type is functionally equivalent to the Numeric data type. Both data types support a range of values from $-10^{38} + 1$ through $10^{38} - 1$. The user can specify the data type to have the following values for its **Precision** and **Scale** attributes, as described in the following table.

Data type attribute	Range	Default
Precision	From 1 to 38	18
Scale	From 0 to the **Precision** that the user sets	0

Note Scale is specified as the digits to the right of the decimal point.

The values of this data type are represented in simple Unicode string format, as follows.

```
Decimal = ["-"] 0*38DIGIT ["."0*38DIGIT]
```

2.1.11 Float

The **Float** data type supports a value range from -1.79E+308 through -2.23E-308; 0; from 2.23E-308 through 1.79E+308. The values of this data type are represented in simple Unicode string format, as follows.

```
Float = ["-"] 1*16DIGIT["."16DIGIT]["e"("-"/"+")(((("0"-"2")2DIGIT) / ("30")(%x30-38))]
```

2.1.12 Image

The **Image** data type supports a sequence of bytes that range from 0 through $2^{31} - 1$ (2,147,483,647). The values of this data type are represented in hexadecimal-encoded Unicode string format, as follows.

```
Image = 0*nOCTET
```

In this format, *n* = 4 x (2,147,483,647). The representation does not require the 0x prefix.

2.1.13 Int

The **Int** data type supports a value range from -2^{31} (-2,147,483,648) through $2^{31} - 1$ (2,147,483,647). The values of this data type are represented in simple **Unicode string** format, as follows.

```
Int = ["-"]1*10DIGIT
```

2.1.14 Money

The **Money** data type supports a value range from -922,337,203,685,447.5808 through 922,337,203,685,477.5807. The values of this data type are represented in simple Unicode string format, as follows.

```
Money = ["-"]1*15DIGIT["."4DIGIT]
```

2.1.15 NChar

The **NChar** data type is a user-defined, fixed-length, double-byte character string that has a supported number of double-byte characters that range from 1 through 4000. The values of this data type are represented in simple Unicode string format, as follows.

```
NChar = 2*8000OCTET
```

Because the characters are already in Unicode, there is no conversion.

2.1.16 NText

The **NText** data type supports a range of Unicode characters that has a maximum number of storage bytes of $2^{31} - 2$ (2,147,483,646). The values of this data type are represented in simple Unicode string format, as follows.

```
NText = 0*nOCTET
```

In this format, *n* = 2,147,483,646. Because the characters are already in Unicode, there is no conversion.

2.1.17 Numeric

The **Numeric** data type is functionally equivalent to the Decimal data type. Both data types support a range of values from $-10^{38} + 1$ through $10^{38} - 1$. The user can specify the data type to have the following values for its **Precision** and **Scale** attributes.

Data type attribute	Range	Default
Precision	From 1 through 38.	18
Scale	From 0 to the **Precision** that the user sets.	0

The values of this data type are represented in simple Unicode string format, as follows.

```
Numeric = ["-"] 0*38DIGIT ["."0*38DIGIT]
```

2.1.18 NVarChar

The **NVarChar** data type is a user-defined variable-length double-byte character string that has a supported maximum number of double-byte characters that range from 1 through 4000 or "max". "Max" specifies that the data type can store up to 2^{31} – 2(2,147,483,646) bytes' worth of double-byte characters. The values of this data type are represented in simple Unicode string format, as follows.

```
NVarChar = 0*nOCTET
```

In this format, *n* = 2,147,483,646. Because the characters are already in Unicode, there is no conversion. If the value is an empty string, the instance data is %x0000.

2.1.19 Real

The **Real** data type supports a value range from -3.40E+38 through -1.18E-38; 0; from 1.18E-38 through 3.40E+38. The values of this data type are represented in simple Unicode string format, as follows.

```
Real = ["-"] 1*7DIGIT["."7DIGIT]["e"("-"/"+")(("0"-"2")DIGIT) / ("3")(%x30-38))]
```

2.1.20 SmallDateTime

The **SmallDateTime** data type supports a value range from 1900-01-01 00:00:00 through 2079-06-06 23:59:59. The values of this data type are represented in the Unicode YYYY-MM-DD hh:mm:ss string format, as follows.

```
Year = ("19"2DIGIT) / ("20"(%x30-36)DIGIT) / ("207"(%x30-39))
Month = ("0"(%x31-39)) / ("1"("0"/"1"/"2"))
Day = ("0"(%x31-39)) / (("1"/"2")DIGIT) / ("3"("0"/"1"))
Hour = (("0"/"1")DIGIT) / ("2"(%x30-33))
Min = (%x30-35)DIGIT
SmallDateTime = Year "-" Month "-" Day SP Hour ":" Min (":00")
```

2.1.21 SmallInt

The **SmallInt** data type supports a value range from -2^{15} (-32,768) through 2^{15} – 1 (32,767). Values of this data type are represented in simple Unicode string format, as follows.

```
SmallInt = ["-"]1*5DIGIT
```

2.1.22 SmallMoney

The **SmallMoney** data type supports a value range from -214,748.3648 through 214,748.3647. The values of this data type are represented in simple Unicode string format, as follows.

```
SmallMoney = ["-"]1*6DIGIT["."1*4DIGIT]
```

2.1.23 Text

The **Text** data type supports a range of single-type characters that has a maximum number of storage bytes of $2^{31} - 1$ (2,147,483,647). The values of this data type are represented in simple Unicode string format, as follows.

```
Text = 0*nOCTET
```

In this format, *n* = 2 x (2^{31} -1) (4,294,967,294). The single-byte characters are converted to their corresponding Unicode characters, therefore doubling the number of OCTET.

2.1.24 Time

The **Time** data type<6> is a user-defined variable fractional-second precision data type that has a decimal precision from 0 through 7. This data type supports a value range from 00:00:00.0000000 through 23:59:59.9999999. The values of this data type are represented in the Unicode hh:mm:ss[.nnnnnnn] string format, as follows.

```
Hour = (("0"/"1")DIGIT) / ("2"(%x30-33))
MinSec = ":"(%x30-35)DIGIT
Time = Hour 2MinSec ["."7DIGIT]
```

2.1.25 TimeStamp

The **TimeStamp** data type is equivalently represented as the Binary data type. The values of this data type are represented in hexadecimal-encoded Unicode string format, as follows.

```
TimeStamp = 32OCTET
```

For more details, see section 3.1.2.

2.1.26 TinyInt

The **TinyInt** data type supports a value range from 0 through 255. The values of this data type are represented in simple Unicode string format, as follows.

```
TinyInt = 1*3DIGIT
```

2.1.27 UniqueIdentifier

The **UniqueIdentifier** data type is functionally equivalent to a **globally unique identifier (GUID)**. Values of this data type are represented in simple Unicode string format, as follows.

```
UniqueIdentifier = 8HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-" 12HEXDIG
```

2.1.28 VarBinary

The **VarBinary** data type is a user-defined variable number of bytes that has a supported maximum number of bytes that range from 1 through 8000 bytes and "max". "Max" specifies that the data type is able to support up to $2^{31} - 1$ bytes. The values of this data type are represented in hexadecimal-encoded Unicode string format, as follows.

```
VarBinary = 0*nOCTET
```

In this format, *n* = 4 x (2,147,483,647).

2.1.29 VarChar

The **VarChar** data type is a user-defined variable-length single-byte character string that has a supported maximum number of single-byte characters that range from 1 through 8000 and "max". "Max" specifies that the data type is able to support up to $2^{31} - 1$ (2,147,483,647) single-byte characters. The values of this data type are represented in simple Unicode string format, as follows.

```
VarChar = 0*nOCTET
```

In this format, *n* = 2 x (2,147,483,647). The single-byte characters are converted to their corresponding Unicode characters, therefore doubling the number of OCTET that is needed to represent the value.

2.1.30 XML

The **XML** data type supports an instance of an XML fragment or a full XML document. This data type supports a maximum number of storage bytes of $2^{31} - 1$ (2,147,483,647). The values of this data type are represented in simple Unicode string format, as follows.

```
XML = 0*nOCTET
```

In this format, *n* = 2,147,483,647.

2.1.31 NULL Value

For all supported data types, if the data instance has a value of NULL, the field is empty and is followed by the field terminator or the row terminator. The NULL value can be specified as follows.

```
NULL = 0OCTET
```

2.1.32 Separators

2.1.32.1 Field Terminator

The field terminator is used to identify the end of one field value and the start of another field value. Special consideration should be given to ensure that the field terminator does not exist within a field value. The field terminator is an arbitrary set of bytes that are specified by the user. The default value is the tab character (0x09). The values of the field terminator are represented in simple Unicode string format, as follows.

```
FieldTerminator = 1*OCTET
```

2.1.32.2 Row Terminator

The row terminator is used to identify the end of one set of field values and the start of another set of field values. Special consideration should be given to ensure that the row terminator does not exist within a field value and/or the field terminator. The row terminator is an arbitrary set of bytes that are specified by the user. The default value is a newline (carriage return and line feed) character (0x0D0A). The values of the row terminator are represented in simple Unicode string format, as follows.

```
RowTerminator = 1*OCTET
```

2.2 BCP Data File

The BCP data file contains the set of data type values that are exported from the database server or imported into the server. The structure of the data file is represented in the following format.

```
Data = BigInt / Binary / Bit / Char / CLRUDT / Date / DateTime / DateTime2
Data =/ DateTimeOffset / Decimal / Float / Image / Int / Money / NChar / NText
Data =/ Numeric / NVarChar / Real / SmallDateTime / SmallInt / SmallMoney
Data =/ Text / Time / TimeStamp / TinyInt / UniqueIdentifier / VarBinary
Data =/ VarChar / XML / NULL
Row = *(Data FieldTerminator) Data ;The last Data does not have a FieldTerminator
DataFile = %xFF %xFE *Row RowTerminator
```

In this format, each repeated Row contains the same set of Data columns.

2.3 BCP Format File

The BCP format file is used to specify the actual source column order, name, and data type for the values that are stored in the data file. The format file is an XML document. In addition to specifying the column order, name, and data type, the format file enables a user to bulk import data values from a data file where the number and/or order of the fields in the data file differ from the number and/or order of destination table columns. For more information, see [MSDN-UFFMFC].

The structure of the format file is represented in the following format.

```
<BCPFORMAT ...>
   <RECORD>
      <FIELD ID = "fieldID" xsi:type = "fieldType" [...] />
   </RECORD>
```

```
   <ROW>
      <COLUMN SOURCE = "fieldID" NAME = "columnName" xsi:type = "columnType" [...]  />
   </ROW>
</BCPFORMAT>
```

The XML elements **FIELD** and **COLUMN** are specified in the following subsections.

2.3.1 Schema Elements

This section summarizes the purpose of each element that the XML schema specifies for format files.

2.3.1.1 BCPFORMAT

The **BCPFORMAT** element is the format-file element that specifies the **RECORD** structure of a given data file and its correspondence to the columns of a table row in the table.

2.3.1.2 RECORD

The **RECORD** structure specifies a complex element that contains one or more **FIELD** elements. The order in which the fields are declared in the format file is the order in which those fields appear in the data file.

2.3.1.3 FIELD

The **FIELD** element specifies a field in a data file that contains data.

The attributes of the **FIELD** element are summarized in the following schema syntax.

```
<FIELD
   ID = "fieldID"
   xsi:type = "fieldType"
   TERMINATOR = "terminator"
   [ MAX_LENGTH = "m" ]
   [ COLLATION = "collationName" ]
/>
```

Each **FIELD** element is independent of the others. A field is specified in terms of the following attributes.

FIELD attribute	Description	Optional/required
ID = "*fieldID*"	This attribute specifies the logical name of the field in the data file. The ID of a field is the key that is used to refer to the field. <FIELD ID="*fieldID*"/> maps to <COLUMN SOURCE="*fieldID*"/>.	Required
xsi:type = "*fieldType*"	This attribute is an XML construct that identifies the type of the instance of the element. The value of *fieldType* must be "NCharTerm".	Required
TERMINATOR = "*terminator*"	This attribute specifies the terminator of a data field. The terminator can be any character. The terminator is recommended to be a unique character that is not part of the data. By default, the field terminator is the tab character, which is represented as \t\0. A paragraph mark is represented as \r\0\n\0.	Required
MAX_LENGTH =	This attribute is the maximum number of bytes that can be stored in a given field. Without a target table, the column maximum length is	Optional

FIELD attribute	Description	Optional/required
"*m*"	not known. The MAX_LENGTH attribute restricts the maximum length of an output character column, limiting the storage that is allocated for the column value.	
COLLATION = "*collationName*"	This attribute is allowed only for character fields. For a list of the collation names, see [MSDN-SSCN].	Optional

2.3.1.4 ROW

The **ROW** element specifies a complex element that contains one or more **COLUMN** elements. The order of the **COLUMN** elements is independent of the order of **FIELD** elements in a RECORD definition. Rather, the order of the **COLUMN** elements in a format file determines the column order of the resultant rowset. Data fields are loaded in the order in which the corresponding **COLUMN** elements are declared in the **COLUMN** element.

2.3.1.5 COLUMN

The **COLUMN** element specifies a column as an element (**COLUMN**). Each **COLUMN** element corresponds to a **FIELD** element. The ID of the **FIELD** element is specified in the **SOURCE** attribute of the **COLUMN** element.

The attributes of the **COLUMN** element are summarized in the following schema syntax.

```
<COLUMN
   SOURCE = "fieldID"
   NAME = "columnName"
   xsi:type = "columnType"
   [ LENGTH = "n" ]
   [ PRECISION = "n" ]
   [ SCALE = "value" ]
   [ NULLABLE = { "YES"
"NO" } ]
/>
```

A field is mapped to a column in the target table using the attributes that are specified in the following table.

COLUMN attribute	Description	Optional/required
SOURCE = "*fieldID*"	This attribute specifies the ID of the field being mapped to the column. <COLUMN SOURCE="*fieldID*"/> maps to <FIELD ID="*fieldID*"/>	Required
NAME = "*columnName*"	This attribute specifies the name of the column in the rowset that is represented by the format file. This column name is used to identify the column in the result set, and it need not correspond to the column name that is used in the target table.	Required
xsi:type = "*ColumnType*"	This attribute is an XML construct that identifies the data type of the instance of the element. The value of *ColumnType* determines which of the optional attributes are required in a given instance. **Note** The possible values of *ColumnType* and their	Optional

COLUMN attribute	Description	Optional/required
	associated attributes are listed in section 2.3.1.5.1.	
LENGTH = "*n*"	This attribute specifies the length for an instance of a fixed-length data type. LENGTH is used only when the **xsi:type** is a string data type. The value of *n* must be a positive integer.	Optional (available only if the **xsi:type** is a string data type)
PRECISION = "*n*"	This attribute indicates the number of digits in a number. For example, the number 123.45 has a precision of 5. The value of *n* must be a positive integer.	Optional (available only if the **xsi:type** is a variable-number data type)
SCALE = "*int*"	This attribute indicates the number of digits to the right of the decimal point in a number. For example, the number 123.45 has a scale of 2. The value of *int* must be an integer.	Optional (available only if the **xsi:type** is a variable-number or variable-scale data type)
NULLABLE = { "YES" "NO" }	This attribute indicates whether a column can assume NULL values. This attribute is completely independent of FIELDS. However, if a column is not NULLABLE, and if the field specifies NULL (by not specifying any value), a run-time error results.	Optional (available for any data type)

2.3.1.5.1 ColumnType

The set of *ColumnType* values that are supported by the **xsi:type** attribute value of the **COLUMN** element identifies the database data type of an instance of an element.

The following table describes the mapping between the data type names that are specified in the **xsi:type** attribute of the **COLUMN** element and the database data types.<7>

COLUMN data type	Database data type
SQLBIGINT	BigInt
SQLBINARY	Binary/TimeStamp
SQLBIT	Bit
SQLCHAR	Char
SQLDATE	Date
SQLDATETIME	DateTime
SQLDATETIME2	DateTime2
SQLDATETIMEOFFSET	DateTimeOffset
SQLDATETIM4,	SmallDateTime
SQLDECIMAL	Decimal
SQLFLT4	Real
SQLFLT8	Float
SQLIMAGE	Image
SQLINT	Int

COLUMN data type	Database data type
SQLMONEY	Money
SQLMONEY4	SmallMoney
SQLNCHAR	NChar
SQLNTEXT	NText
SQLNUMERIC	Numeric
SQLNVARCHAR	NVarChar/XML
SQLSMALLINT	SmallInt
SQLTEXT	Text
SQLTIME	Time
SQLTINYINT	TinyInt
SQLUNIQUEID	UniqueIdentifier
SQLVARYBIN	VarBinary
SQLVARYCHAR	VarChar
SQLUDT	CLRUDT

The **COLUMN** element supports native SQL data types as follows.

Type category	COLUMN data types	Optional XML attribute for data type
Fixed<8>	▪ SQLBIT ▪ SQLTINYINT ▪ SQLSMALLINT ▪ SQLINT ▪ SQLBIGINT ▪ SQLFLT4 ▪ SQLFLT8 ▪ SQLDATE ▪ SQLDATETIME ▪ SQLDATETIM4 ▪ SQLMONEY ▪ SQLMONEY4 ▪ SQLUNIQUEID	NULLABLE
Variable Scale	▪ SQLDATETIME2	NULLABLE, SCALE

Type category	COLUMN data types	Optional XML attribute for data type
	▪ SQLDATETIMEOFFSET ▪ SQLTIME	
Variable Number	▪ SQLDECIMAL ▪ SQLNUMERIC	NULLABLE, PRECISION, SCALE
LOB	▪ SQLIMAGE ▪ SQLTEXT ▪ SQLNTEXT ▪ SQLUDT	NULLABLE
Binary string	▪ SQLBINARY ▪ SQLVARYBIN	NULLABLE, LENGTH
Character string	▪ SQLCHAR ▪ SQLVARYCHAR ▪ SQLNCHAR ▪ SQLNVARCHAR	NULLABLE, LENGTH

2.3.1.6 /BCPFORMAT

The **/BCPFORMAT** element is required to end the format file.

2.3.2 Format File XSD Schema

The following XSD schema specifies the XML structure of the format file [XMLSCHEMA1/2].

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://schemas.microsoft.com/sqlserver/2004/bulkload/format"
xmlns="http://schemas.microsoft.com/sqlserver/2004/bulkload/format"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="qualified"
elementFormDefault="qualified">
  <xs:element name="BCPFORMAT">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RECORD">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="FIELD" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="ID" form="unqualified" type="xs:string" />
                  <xs:attribute name="TERMINATOR" form="unqualified" type="xs:string" />

                  <xs:attribute name="MAX_LENGTH" form="unqualified" type="xs:string" />
                  <xs:attribute name="COLLATION" form="unqualified" type="xs:string" />
```

```
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="ROW">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="COLUMN" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="SOURCE" form="unqualified" type="xs:string" />
                  <xs:attribute name="NAME" form="unqualified" type="xs:string" />
                  <xs:attribute name="LENGTH" form="unqualified" type="xs:string" />
                  <xs:attribute name="SCALE" form="unqualified" type="xs:string" />
                  <xs:attribute name="PRECISION" form="unqualified" type="xs:string" />
                  <xs:attribute name="NULLABLE" form="unqualified" type="xs:string" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

3 Structure Examples

3.1 Data File

The following subsections show an example of a value for each data type as if the value were written in the data file.

3.1.1 BigInt

The sample value is 9,223,372,036,854,775,807.

The file content in hexadecimal mode is as follows.

```
39 00 32 00 32 00 33 00-33 00 37 00 32 00 30 00  *9.2.2.3.3.7.2.0.*
33 00 36 00 38 00 35 00-34 00 37 00 37 00 35 00  *3.6.8.5.4.7.7.5.*
38 00 30 00 37 00                                *8.0.7.*
```

3.1.2 Binary

The sample value is 0x56006C00610064002000500075006D007000650072006E00690063006B0065006C002C00200062006C006400670020003300.

The file content in hexadecimal mode is as follows.

```
35 00 36 00 30 00 30 00-36 00 43 00 30 00 30 00  *5.6.0.0.6.C.0.0.*
36 00 31 00 30 00 30 00-36 00 34 00 30 00 30 00  *6.1.0.0.6.4.0.0.*
32 00 30 00 30 00 30 00-35 00 30 00 30 00 30 00  *2.0.0.0.5.0.0.0.*
37 00 35 00 30 00 30 00-36 00 44 00 30 00 30 00  *7.5.0.0.6.D.0.0.*
37 00 30 00 30 00 30 00-36 00 35 00 30 00 30 00  *7.0.0.0.6.5.0.0.*
37 00 32 00 30 00 30 00-36 00 45 00 30 00 30 00  *7.2.0.0.6.E.0.0.*
36 00 39 00 30 00 30 00-36 00 33 00 30 00 30 00  *6.9.0.0.6.3.0.0.*
36 00 42 00 30 00 30 00-36 00 35 00 30 00 30 00  *6.B.0.0.6.5.0.0.*
36 00 43 00 30 00 30 00-32 00 43 00 30 00 30 00  *6.C.0.0.2.C.0.0.*
32 00 30 00 30 00 30 00-36 00 32 00 30 00 30 00  *2.0.0.0.6.2.0.0.*
36 00 43 00 30 00 30 00-36 00 34 00 30 00 30 00  *6.C.0.0.6.4.0.0.*
36 00 37 00 30 00 30 00-32 00 30 00 30 00 30 00  *6.7.0.0.2.0.0.0.*
33 00 33 00 30 00 30 00                          *3.3.0.0.*
```

3.1.3 Bit

The sample value is 1.

The file content in hexadecimal mode is as follows.

```
31 00                    *1.*
```

3.1.4 Char

The sample value is Udo.

The file content in hexadecimal mode is as follows.

```
55 00 64 00 6F 00 20 00-20 00 20 00 20 00 20 00  *U.d.o. . . . . .*
20 00 20 00                                      * . .*
```

3.1.5 CLRUDT

The sample value is 0x58.

The file content in hexadecimal mode is as follows.

```
  35 00 38 00               *5.8.*
```

3.1.6 Date

The sample value is 2009-12-30.

The file content in hexadecimal mode is as follows.

```
32 00 30 00 30 00 39 00-2D 00 31 00 32 00 2D 00  *2.0.0.9.-.1.2.-.*
33 00 30 00                                      *3.0.*
```

3.1.7 DateTime

The sample value is 2009-12-30 13:51:35.437.

The file content in hexadecimal mode is as follows.

```
32 00 30 00 30 00 39 00-2D 00 31 00 32 00 2D 00  *2.0.0.9.-.1.2.-.*
33 00 30 00 20 00 31 00-33 00 3A 00 35 00 31 00  *3.0. .1.3.:.5.1.*
3A 00 33 00 35 00 2E 00-34 00 33 00 37 00        *:.3.5...4.3.7.*
```

3.1.8 DateTime2

The sample value is 2009-12-30 13:51.35.4299569.

The file content in hexadecimal mode is as follows.

```
32 00 30 00 30 00 39 00-2D 00 31 00 32 00 2D 00  *2.0.0.9.-.1.2.-.*
33 00 30 00 20 00 31 00-33 00 3A 00 35 00 31 00  *3.0. .1.3.:.5.1.*
3A 00 33 00 35 00 2E 00-34 00 32 00 39 00 39 00  *:.3.5...4.2.9.9.*
35 00 36 00 39 00                                *5.6.9.*
```

3.1.9 DateTimeOffset

The sample value is 2009-12-30 13:51:35.4299569 -08:00.

The file content in hexadecimal mode is as follows.

```
32 00 30 00 30 00 39 00-2D 00 31 00 32 00 2D 00  *2.0.0.9.-.1.2.-.*
33 00 30 00 20 00 31 00-33 00 3A 00 35 00 31 00  *3.0. .1.3.:.5.1.*
3A 00 33 00 35 00 2E 00-34 00 32 00 39 00 39 00  *:.3.5...4.2.9.9.*
35 00 36 00 39 00 20 00-2D 00 30 00 38 00 3A 00  *5.6.9. .-.0.8.:.*
30 00 30 00                                      *0.0.*
```

3.1.10 Decimal

The sample value is 123456.123456780.

The file content in hexadecimal mode is as follows.

```
31 00 32 00 33 00 34 00-35 00 36 00 2E 00 31 00  *1.2.3.4.5.6...1.*
32 00 33 00 34 00 35 00-36 00 37 00 38 00 30 00  *2.3.4.5.6.7.8.0.*
```

3.1.11 Float

The sample value is 1.23456789E+17.

The file content in hexadecimal mode is as follows.

```
31 00 2E 00 32 00 33 00-34 00 35 00 36 00 37 00  *1...2.3.4.5.6.7.*
38 00 39 00 45 00 2B 00-31 00 37 00              *8.9.E.+.1.7.*
```

3.1.12 Image

The sample value is 0x152593A20466F75722073636F726520616E6420736576656E2079656172732061676F206F7572206661746865727320626726F756.

The file content in hexadecimal mode is as follows.

```
31 00 35 00 32 00 35 00-39 00 33 00 41 00 32 00  *1.5.2.5.9.3.A.2.*
30 00 34 00 36 00 36 00-46 00 37 00 35 00 37 00  *0.4.6.6.F.7.5.7.*
32 00 32 00 30 00 37 00-33 00 36 00 33 00 36 00  *2.2.0.7.3.6.3.6.*
46 00 37 00 32 00 36 00-35 00 32 00 30 00 36 00  *F.7.2.6.5.2.0.6.*
31 00 36 00 45 00 36 00-34 00 32 00 30 00 37 00  *1.6.E.6.4.2.0.7.*
33 00 36 00 35 00 37 00-36 00 36 00 35 00 36 00  *3.6.5.7.6.6.5.6.*
45 00 32 00 30 00 37 00-39 00 36 00 35 00 36 00  *E.2.0.7.9.6.5.6.*
31 00 37 00 32 00 37 00-33 00 32 00 30 00 36 00  *1.7.2.7.3.2.0.6.*
31 00 36 00 37 00 36 00-46 00 32 00 30 00 36 00  *1.6.7.6.F.2.0.6.*
46 00 37 00 35 00 37 00-32 00 32 00 30 00 36 00  *F.7.5.7.2.2.0.6.*
36 00 36 00 31 00 37 00-34 00 36 00 38 00 36 00  *6.6.1.7.4.6.8.6.*
35 00 37 00 32 00 37 00-33 00 32 00 30 00 36 00  *5.7.2.7.3.2.0.6.*
32 00 37 00 32 00 36 00-46 00 37 00 35 00 36 00  *2.7.2.6.F.7.5.6.*
```

3.1.13 Int

The sample value is 2147483647.

The file content in hexadecimal mode is as follows.

```
32 00 31 00 34 00 37 00-34 00 38 00 33 00 36 00  *2.1.4.7.4.8.3.6.*
34 00 37 00                                      *4.7.*
```

3.1.14 Money

The sample value is 922337203685477.0100.

The file content in hexadecimal mode is as follows.

```
39 00 32 00 32 00 33 00-33 00 37 00 32 00 30 00  *9.2.2.3.3.7.2.0.*
33 00 36 00 38 00 35 00-34 00 37 00 37 00 2E 00  *3.6.8.5.4.7.7...*
30 00 31 00 30 00 30 00                          *0.1.0.0.*
```

3.1.15 NChar

The sample value is あピポぶ左州見.

The file content in hexadecimal mode is as follows.

```
42 30 D4 30 DD 30 76 30-E6 5D DE 5D 0A FA 20 00  *B0.0.0v0.].].. .*
20 00 20 00                                      * . .*
```

3.1.16 NText

The sample value is "When in the Course of human events, it becomes necessary for one".

The file content in hexadecimal mode is as follows.

```
57 00 68 00 65 00 6E 00-20 00 69 00 6E 00 20 00  *W.h.e.n. .i.n. .*
74 00 68 00 65 00 20 00-43 00 6F 00 75 00 72 00  *t.h.e. .C.o.u.r.*
73 00 65 00 20 00 6F 00-66 00 20 00 68 00 75 00  *s.e. .o.f. .h.u.*
6D 00 61 00 6E 00 20 00-65 00 76 00 65 00 6E 00  *m.a.n. .e.v.e.n.*
74 00 73 00 2C 00 20 00-69 00 74 00 20 00 62 00  *t.s.,. .i.t. .b.*
65 00 63 00 6F 00 6D 00-65 00 73 00 20 00 6E 00  *e.c.o.m.e.s. .n.*
65 00 63 00 65 00 73 00-73 00 61 00 72 00 79 00  *e.c.e.s.s.a.r.y.*
20 00 66 00 6F 00 72 00-20 00 6F 00 6E 00 65 00  * .f.o.r. .o.n.e.*
```

3.1.17 Numeric

The sample value is 1234567890.12345678.

The file content in hexadecimal mode is as follows.

```
31 00 32 00 33 00 34 00-35 00 36 00 37 00 38 00  *1.2.3.4.5.6.7.8.*
39 00 30 00 2E 00 31 00-32 00 33 00 34 00 35 00  *9.0...1.2.3.4.5.*
36 00 37 00 38 00                                *6.7.8.*
```

3.1.18 NVarChar

The sample value is あピポぶ左州見.

The file content in hexadecimal mode is as follows.

```
42 30 D4 30 DD 30 76 30-E6 5D DE 5D 0A FA 20 00  *B0.0.0v0.].]..*
```

3.1.19 Real

The sample value is -1.1234568.

The file content in hexadecimal mode is as follows.

```
2D 00 31 00 2E 00 31 00-32 00 33 00 34 00 35 00  *-.1...1.2.3.4.5.*
36 00 38 00                                      *6.8.*
```

3.1.20 SmallDateTime

The sample value is 2009-12-30 13:52:00.

The file content in hexadecimal mode is as follows.

```
32 00 30 00 30 00 39 00-2D 00 31 00 32 00 2D 00  *2.0.0.9.-.1.2.-.*
33 00 30 00 20 00 31 00-33 00 3A 00 35 00 32 00  *3.0. .1.3.:.5.2.*
3A 00 30 00 30 00                                *:.0.0.*
```

3.1.21 SmallInt

The sample value is -32768.

The file content in hexadecimal mode is as follows.

```
2D 00 33 00 32 00 37 00-36 00 38 00              *-.3.2.7.6.8.*
```

3.1.22 SmallMoney

The sample value is 214748.3647.

The file content in hexadecimal mode is as follows.

```
32 00 31 00 34 00 37 00-34 00 38 00 2E 00 33 00  *2.1.4.7.4.8...3.*
36 00 34 00 37 00                                *6.4.7.*
```

3.1.23 Sql_Variant

The sample value is 123.456789.

The file content in hexadecimal mode is as follows.

```
31 00 32 00 33 00 2E 00-34 00 35 00 36 00 37 00  *1.2.3...4.5.6.7.*
```

```
38 00 39 00                                     *8.9.*
```

3.1.24 Text

The sample value is "people to dissolve the political bands which have connected them".

The file content in hexadecimal mode is as follows.

```
70 00 65 00 6F 00 70 00-6C 00 65 00 20 00 74 00  *p.e.o.p.l.e. .t.*
6F 00 20 00 64 00 69 00-73 00 73 00 6F 00 6C 00  *o. .d.i.s.s.o.l.*
76 00 65 00 20 00 74 00-68 00 65 00 20 00 70 00  *v.e. .t.h.e. .p.*
6F 00 6C 00 69 00 74 00-69 00 63 00 61 00 6C 00  *o.l.i.t.i.c.a.l.*
20 00 62 00 61 00 6E 00-64 00 73 00 20 00 77 00  * .b.a.n.d.s. .w.*
68 00 69 00 63 00 68 00-20 00 68 00 61 00 76 00  *h.i.c.h. .h.a.v.*
65 00 20 00 63 00 6F 00-6E 00 6E 00 65 00 63 00  *e. .c.o.n.n.e.c.*
74 00 65 00 64 00 20 00-74 00 68 00 65 00 6D 00  *t.e.d. .t.h.e.m.*
```

3.1.25 Time

The sample value is 11:30:32.1234000.

The file content in hexadecimal mode is as follows.

```
31 00 31 00 3A 00 33 00-30 00 3A 00 33 00 32 00  *1.1.:.3.0.:.3.2.*
2E 00 31 00 32 00 33 00-34 00 30 00 30 00 30 00  *..1.2.3.4.0.0.0.*
```

3.1.26 TimeStamp

The sample value is 0x00000000000007D1.

The file content in hexadecimal mode is as follows.

```
30 00 30 00 30 00 30 00-30 00 30 00 30 00 30 00  *0.0.0.0.0.0.0.0.*
30 00 30 00 30 00 30 00-30 00 37 00 44 00 31 00  *0.0.0.0.0.7.D.1.*
```

3.1.27 TinyInt

The sample value is 127.

The file content in hexadecimal mode is as follows.

```
31 00 32 00 37 00                               *1.2.7.*
```

3.1.28 UniqueIdentifier

The sample value is 65DD4051-C7FE-4CB8-954D-0B1967468D3E.

The file content in hexadecimal mode is as follows.

```
36 00 35 00 44 00 44 00-34 00 30 00 35 00 31 00  *6.5.D.D.4.0.5.1.*
2D 00 43 00 37 00 46 00-45 00 2D 00 34 00 43 00  *-.C.7.F.E.-.4.C.*
42 00 38 00 2D 00 39 00-35 00 34 00 44 00 2D 00  *B.8.-.9.5.4.D.-.*
30 00 42 00 31 00 39 00-36 00 38 00 34 00 36 00  *0.B.1.9.6.8.4.6.*
38 00 44 00 33 00 45 00                          *8.D.3.E.*
```

3.1.29 VarBinary

The sample value is 0x86520717569636B2062726F776E20666F78206A756D706564206F76657220746865206C617A7920 6.

The file content in hexadecimal mode is as follows.

```
38 00 36 00 35 00 32 00-30 00 37 00 31 00 37 00  *8.6.5.2.0.7.1.7.*
35 00 36 00 39 00 36 00-33 00 36 00 42 00 32 00  *5.6.9.6.3.6.B.2.*
30 00 36 00 32 00 37 00-32 00 36 00 46 00 37 00  *0.6.2.7.2.6.F.7.*
37 00 36 00 45 00 32 00-30 00 36 00 36 00 36 00  *7.6.E.2.0.6.6.6.*
46 00 37 00 38 00 32 00-30 00 36 00 41 00 37 00  *F.7.8.2.0.6.A.7.*
35 00 36 00 44 00 37 00-30 00 36 00 35 00 36 00  *5.6.D.7.0.6.5.6.*
34 00 32 00 30 00 36 00-46 00 37 00 36 00 36 00  *4.2.0.6.F.7.6.6.*
35 00 37 00 32 00 32 00-30 00 37 00 34 00 36 00  *5.7.2.2.0.7.4.6.*
38 00 36 00 35 00 32 00-30 00 36 00 43 00 36 00  *8.6.5.2.0.6.C.6.*
31 00 37 00 41 00 37 00-39 00 32 00 30 00 36 00  *1.7.A.7.9.2.0.6.*
```

3.1.30 VarChar

The sample value is "The quick brown fox jumped over the lazy dog."

The file content in hexadecimal mode is as follows.

```
54 00 68 00 65 00 20 00-71 00 75 00 69 00 63 00  *T.h.e. .q.u.i.c.*
6B 00 20 00 62 00 72 00-6F 00 77 00 6E 00 20 00  *k. .b.r.o.w.n. .*
66 00 6F 00 78 00 20 00-6A 00 75 00 6D 00 70 00  *f.o.x. .j.u.m.p.*
65 00 64 00 20 00 6F 00-76 00 65 00 72 00 20 00  *e.d. .o.v.e.r. .*
74 00 68 00 65 00 20 00-6C 00 61 00 7A 00 79 00  *t.h.e. .l.a.z.y.*
20 00 64 00 6F 00 67 00-2E 00                    * .d.o.g...*
```

3.1.31 XML

The sample value is "<Element>nothing to report…</Element>".

The file content in hexadecimal mode is as follows.

```
3C 00 45 00 6C 00 65 00-6D 00 65 00 6E 00 74 00  *<.E.l.e.m.e.n.t.*
3E 00 6E 00 6F 00 74 00-68 00 69 00 6E 00 67 00  *>.n.o.t.h.i.n.g.*
20 00 74 00 6F 00 20 00-72 00 65 00 70 00 6F 00  * .t.o. .r.e.p.o.*
72 00 74 00 2E 00 2E 00-2E 00 3C 00 2F 00 45 00  *r.t.......<./.E.*
6C 00 65 00 6D 00 65 00-6E 00 74 00 3E 00        *l.e.m.e.n.t.>.*
```

3.1.32 Field Terminator

The sample value is ;;.

The file content in hexadecimal mode is as follows.

```
3B 00 3B 00              *;.;.*
```

3.1.33 Row Terminator

The sample value is ==.

The file content in hexadecimal mode is as follows.

```
3D 00 3D 00              *=.=.*
```

3.2 Format File

The sample value is as follows.

```
<?xml version="1.0"?>
<BCPFORMAT xmlns="http://schemas.microsoft.com/sqlserver/2004/bulkload/format"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RECORD>
  <FIELD ID="1" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="42"/>
  <FIELD ID="2" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="202"/>
  <FIELD ID="3" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="6"/>
  <FIELD ID="4" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="20"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="5" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="22"/>
  <FIELD ID="6" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="48"/>
  <FIELD ID="7" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="60"/>
  <FIELD ID="8" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="74"/>
  <FIELD ID="9" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="82"/>
  <FIELD ID="10" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="60"/>
  <FIELD ID="11" xsi:type="NCharTerm" TERMINATOR="\t\0"/>
  <FIELD ID="12" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="24"/>
  <FIELD ID="13" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="60"/>
  <FIELD ID="14" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="20"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="15" xsi:type="NCharTerm" TERMINATOR="\t\0"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="16" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="82"/>
  <FIELD ID="17" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="100"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="18" xsi:type="NCharTerm" TERMINATOR="\t\0"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="19" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="60"/>
  <FIELD ID="20" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="48"/>
  <FIELD ID="21" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="14"/>
  <FIELD ID="22" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="60"/>
  <FIELD ID="23" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="8000"/>
  <FIELD ID="24" xsi:type="NCharTerm" TERMINATOR="\t\0"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="25" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="38"/>
  <FIELD ID="26" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="34"/>
  <FIELD ID="27" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="10"/>
  <FIELD ID="28" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="74"/>
  <FIELD ID="29" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="202"/>
  <FIELD ID="30" xsi:type="NCharTerm" TERMINATOR="\t\0"/>
  <FIELD ID="31" xsi:type="NCharTerm" TERMINATOR="\t\0" MAX_LENGTH="100"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="32" xsi:type="NCharTerm" TERMINATOR="\t\0"
COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
```

```
  <FIELD ID="33" xsi:type="NCharTerm" TERMINATOR="\t\0"/>
  <FIELD ID="34" xsi:type="NCharTerm" TERMINATOR="\r\0\n\0"/> </RECORD>
 <ROW>
  <COLUMN SOURCE="1" NAME="col_bigint" xsi:type="SQLBIGINT"/>
  <COLUMN SOURCE="2" NAME="col_binary50" xsi:type="SQLBINARY"/>
  <COLUMN SOURCE="3" NAME="col_bit" xsi:type="SQLBIT"/>
  <COLUMN SOURCE="4" NAME="col_char10" xsi:type="SQLCHAR"/>
  <COLUMN SOURCE="5" NAME="col_date" xsi:type="SQLDATE"/>
  <COLUMN SOURCE="6" NAME="col_datetime" xsi:type="SQLDATETIME"/>
  <COLUMN SOURCE="7" NAME="col_datetime2" xsi:type="SQLDATETIME2" SCALE="7"/>
  <COLUMN SOURCE="8" NAME="col_datetimeoffset" xsi:type="SQLDATETIMEOFFSET" SCALE="7"/>
  <COLUMN SOURCE="9" NAME="col_decimal" xsi:type="SQLDECIMAL" PRECISION="18" SCALE="9"/>
  <COLUMN SOURCE="10" NAME="col_float" xsi:type="SQLFLT8"/>
  <COLUMN SOURCE="11" NAME="col_image" xsi:type="SQLIMAGE"/>
  <COLUMN SOURCE="12" NAME="col_int" xsi:type="SQLINT"/>
  <COLUMN SOURCE="13" NAME="col_money" xsi:type="SQLMONEY"/>
  <COLUMN SOURCE="14" NAME="col_nchar10" xsi:type="SQLNCHAR"/>
  <COLUMN SOURCE="15" NAME="col_ntext" xsi:type="SQLNCHAR"/>
  <COLUMN SOURCE="16" NAME="col_numeric" xsi:type="SQLNUMERIC" PRECISION="18" SCALE="8"/>
  <COLUMN SOURCE="17" NAME="col_nvarchar50" xsi:type="SQLNVARCHAR"/>
  <COLUMN SOURCE="18" NAME="col_nvarcharmax" xsi:type="SQLNVARCHAR"/>
  <COLUMN SOURCE="19" NAME="col_real" xsi:type="SQLFLT4"/>
  <COLUMN SOURCE="20" NAME="col_smalldatetime" xsi:type="SQLDATETIM4"/>
  <COLUMN SOURCE="21" NAME="col_smallint" xsi:type="SQLSMALLINT"/>
  <COLUMN SOURCE="22" NAME="col_smallmoney" xsi:type="SQLMONEY4"/>
  <COLUMN SOURCE="23" NAME="col_variant" xsi:type="SQLVARIANT"/>
  <COLUMN SOURCE="24" NAME="col_text" xsi:type="SQLCHAR"/>
  <COLUMN SOURCE="25" NAME="col_time" xsi:type="SQLTIME" SCALE="7"/>
  <COLUMN SOURCE="26" NAME="col_timestamp" xsi:type="SQLBINARY"/>
  <COLUMN SOURCE="27" NAME="col_tinyint" xsi:type="SQLTINYINT"/>
  <COLUMN SOURCE="28" NAME="col_uuid" xsi:type="SQLUNIQUEID"/>
  <COLUMN SOURCE="29" NAME="col_varbinary50" xsi:type="SQLVARYBIN"/>
  <COLUMN SOURCE="30" NAME="col_varbinarymax" xsi:type="SQLVARYBIN"/>
  <COLUMN SOURCE="31" NAME="col_varchar50" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="32" NAME="col_varcharmax" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="33" NAME="col_xml" xsi:type="SQLNVARCHAR"/>
  <COLUMN SOURCE="34" NAME="col_hierarchy" xsi:type="SQLUDT"/>
 </ROW>
</BCPFORMAT>
```

4 Security Considerations

None.

5 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

- Microsoft SQL Server 2000
- Microsoft SQL Server 2005
- Microsoft SQL Server 2008
- Microsoft SQL Server 2008 R2
- Microsoft SQL Server 2012
- Microsoft SQL Server 2014
- Microsoft SQL Server 2016

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.1: Microsoft implementations include the **Sql_Variant** data type, which is a special data type definition that supports different data types for each instance of the value. The following data types are supported:

- BigInt
- Binary
- Bit
- Char
- Date
- DateTime
- DateTime2
- DateTimeOffset
- Decimal
- Float
- Int
- Money
- NChar
- Numeric

- NVarChar
- Real
- SmallDateTime
- SmallInt
- SmallMoney
- Time
- TinyInt
- UniqueIdentifier
- VarBinary
- VarChar

<2> Section 2.1.5: Customer user-defined types are written in the Microsoft .NET Framework. The **hierarchyID** and **Spatial** data types are not supported by SQL Server 2000 and SQL Server 2005.

<3> Section 2.1.6: The **Date** data type is not supported by SQL Server 2000 and SQL Server 2005.

<4> Section 2.1.8: The **DateTime2** data type is not supported by SQL Server 2000 and SQL Server 2005.

<5> Section 2.1.9: The **DateTimeOffset** data type is not supported by SQL Server 2000 and SQL Server 2005.

<6> Section 2.1.24: The **Time** data type is not supported by SQL Server 2000 and SQL Server 2005.

<7> Section 2.3.1.5.1: In addition to the **COLUMN** data types listed in the table, Microsoft SQL Server supports a data type named SQLVARIANT. The database data type of SQLVARIANT is Sql_Variant.

<8> Section 2.3.1.5.1: In addition to the **COLUMN** data types listed here, SQL Server supports the SQLVARIANT data type.

6 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
5 Appendix A: Product Behavior	Added SQL Server 2016 to the list of applicable products.	Y	Content update.

7 Index

S

T

U

V

X