# BER Date-and-Time Format

**Version 1.0 - 20th May 2020**

## Overview

In this document, we provide a specification for the encoding forms of BDE date-and-time types.

The BDE BER codec provides 3 ways of encoding BDE date-and-time types.  The first is the one defined in the **ITU-T X.690**[1] specification itself, which provides the official definition for BER.  This specification requires one represent date-and-time values according to another standard: **ISO 8601**[2]. The second format is a machine-readable "compact-binary" form, and is a non-conforming extension to BER that is only supported  by the BDE BER codec.  This format provides a much more compact representation of date-and-time values, and represents time components with millisecond precision. The final format, the "extended-binary" form, is a recent extension to the compact-binary form which represents time components with microsecond precision, and is less compact than the compact-binary form, yet still more compact than the ISO 8601 form.  One must consider the tradeoffs involved between portability and compactness when deciding whether to use the standard **ISO 8601** form or a binary form.

## Table of Contents

[1] **ITU-T X.690** is the official specification for the ASN.1 Basic Encoding Rules, which define the BER format.  Its Wikipedia page is here, and its official specification is here.
[2] **ISO 8601** is an international standard for the textual rendering and interpretation of date and time values. It is convenient because it is both a human- and machine-readable format, and it can be represented using ASCII, which is UTF-8 compatible. It's Wikipedia page is here, and its official specification is here.

## Definitions

1. **BER integer**: A variable-length encoding of an integer value specified in **ITU-T X.690** section 8.3, "Encoding of an integer value".
2. **compact-binary date epoch**: A synonym for January 1st 2020 in the Proleptic Gregorian calendar.
3. **compact-binary datetime epoch**: A synonym for 12:00AM at the beginning of the day on the compact-binary date epoch.
4. **compact-binary epoch date component**: A representation of a date value as an integer number of days before or since the compact-binary date epoch. Days before the epoch are represented by negative integers, the epoch itself by the integer 0, and days after it by positive integers.
5. **compact-binary epoch datetime component**: A representation of a date-and-time value as an integer number of milliseconds before or since the compact-binary datetime epoch. Date-and-time values before the epoch are represented by negative integers, the epoch itself by the integer 0, and values after it by positive integers.

6. **compact-binary form**: An encoding form of a date-and-time type characterized by very efficient space utilization and mandated millisecond time precision (if applicable). The encoding of all date-and-time types has a compact-binary form.
7. **compact-binary time component**: A representation of a time value by an integer number of milliseconds since 12:00AM at the beginning of the day.
8. **date component**: An attribute of a date-and-time value that represents a date in the interval between January 1st, year 1 and December 31st, 9999 inclusive, in the Proleptic Gregorian calendar. The date-and-time types with date components are `bdlt::Date`, `bdlt::DateTz`, `bdlt::Datetime`, and `bdlt::DatetimeTz`.
9. **date-and-time type**: One of the 6 BDE types used to represent simple chronological values: `bdlt::Date`, `bdlt::DateTz`, `bdlt::Time`, `bdlt::TimeTz`, `bdlt::Datetime`, or `bdlt::DatetimeTz.`
10. **date-and-time value**: A value represented by an object of a date-and-time type. Depending on the object's type, the value may consist of a date component and/or time component, and may also include a time-zone component.
11. **day component**: A sub-component of a date component that represents the number of whole days since the start of the month in the date according to the Proleptic Gregorian calendar. Day components count from 1. For example, the first day of the month is represented by the integer 1, the second by 2, etc.
12. **encoding form**: One of typically several different ways of representing a value as a sequence of bits. Different encoding forms for a value may provide different degrees of fidelity of the original value when decoding.
13. **extended-binary form**: An encoding form of a date-and-time type characterized by efficient space utilization and mandated microsecond time precision. The date-and-time types with an extended binary encoding form are `bdlt::Time`, `bdlt::TimeTz`, `bdlt::Datetime`, and `bdlt::DatetimeTz`.
14. **hour, minute, second, millisecond, and microsecond components**: Sub-attributes of a time component that represent, respectively, the number of whole hours, and then minutes, seconds, milliseconds, and microseconds since 12:00AM at the beginning of the day.
15. **hour-offset component**: A sub-attribute of a time-zone component that represents the number of whole hours ahead of or behind UTC in the time-zone value.
16. **minute-offset component**: A sub-attribute of a time-zone component that represents the number of whole minutes to the hour ahead of or bind UTC in the time-zone value.
17. **month component**: A sub-attribute of a date component that represents the number of whole months since the start of the year in the date value according to the Proleptic Gregorian calendar. Month components count from 1. For example, January is represented by the integer 1, February by 2, etc.
18. **octet**: The term used by the BER specification to refer to a byte, which is an ordered sequence of 8 bits.

19. **offset-sign component**: A sub-attribute of a time-zone component that is said to be "positive" if the time-zone represents UTC or a time-zone ahead of UTC, and "negative" otherwise.
20. **time component**: An attribute of a date-and-time type whose value represents an intra-day time in the interval between 12:00AM at the beginning of the day, and 12:00AM at the end of the day. Time components have microsecond precision, but different encoding forms may provide less precision. The date-and-time types with time components are `bdlt::Time`, `bdlt::TimeTz`, `bdlt::Datetime`, and `bdlt::DatetimeTz`.
21. **time-zone component**: An attribute of a date-and-time value that represents an offset in minutes ahead of or behind UTC, in the interval between 23 hours and 59 minutes ahead of UTC and 23 hours and 59 minutes behind UTC, inclusive, with minute precision. Time-zone components are used to localize date-and-time values with respect to a time point in UTC.
22. **year component**: A sub-attribute of a date component that represents the largest number of whole years in the date according to the Proleptic Gregorian calendar. Year components count from 1. For example, the year 1 in the Proleptic Gregorian calendar is represented by the integer 1, year 2 by 2, etc.

## BER Encoding Review

### Top-Level Structure

BER represents information as sequences of octets. It represents all values using the same overall structure, which has the following form, in order:

| | | | |
|---|---|---|---|
| Identifier Octets | Length Octets | Contents Octets | End-of-Contents Octets |

For a description of the Identifier Octets, Length Octets, and (optional) End-of-Contents Octets, see the ITU-T X.690 specification. The formats described in this document provide a definition for the Contents Octets in the encoding of date-and-time values. Note that the BER decoder uses the value encoded in the length octets, which is referred to as the "length of the encoding" in the remainder of this document, as one of the values used to disambiguate different encoding forms for the same type.

## BDE Date-and-Time BER Encoding Overview

### Encoding Forms

The encodings of all date-and-time types have similar overall structure, which is helpful to keep in mind when trying to understand individual encoding forms. In general, the encodings are in one of the following forms (with the exception of `bdlt::Date` and `bdlt::DateTz`, which do not have extended-binary forms):

1. An **ISO 8601** form, which is a human-readable ASCII text string,
2. A compact-binary form, which represents time components with millisecond precision, and consists of, in order:
    a. If the type has a time-zone component, a time-zone offset in minutes since UTC, encoded as a variable-length BER integer, and
    b. An offset from a reference (or "epoch") time point, in milliseconds or days, whichever is appropriate for the type, encoded as a variable-length BER integer (which is sometimes sign-extended to provide padding),
3. An extended-binary form, which represents time values with microsecond precision, and consists of, in order:
    a. A 2-byte header that consists of, in order:
        i. A `1` bit
        ii. Two `0` bits,
        iii. If the type has a time-zone component, a `1` bit; and a `0` bit otherwise
        iv. If the type has a time-zone component, the time-zone offset in minutes since UTC, encoded as a 12-bit, 2's-complement, big-endian, signed integer; and 12 `0` bits otherwise,
    b. If the type has a date component, the date as a number of days since January 1st, year 1 in the Proleptic Gregorian calendar, encoded as a 3-byte, big-endian, unsigned integer,
    c. If the type has a time component, the time as a number of microseconds since 12:00AM at the beginning of the day, encoded as a 5-byte, big-endian, unsigned integer

### Notation

This document specifies the representations of encoded date-and-time types using the Extended Backus-Naur grammar notation defined by **ISO/IEC 14977**, with one notable deviation. In ISO Backus-Naur form, there is no difference in meaning between a literal surrounded by single (') quotes and the same literal surrounded by double (") quotes. In this document, a literal surrounded by single quotes denotes a sequence of bits, the length of which need not be a multiple of 8. The character set of these literals is restricted to `0` and `1`. A literal surrounded by double quotes denotes one or more octets corresponding to the **ASCII** encoding of the surrounded characters. The character set of such literals is the set of printable characters in the **ASCII** specification, which is **ISO/IEC 8859-1**.

The grammars for each date-and-time type share some nonterminals. For nonterminals that are not immediately found in the description of the grammar for a particular type, see the **Shared Grammar Nonterminals** section.

Note that, for any particular date-and-time type, some bit strings that satisfy the grammar for the type may not be valid representations of values of that type. Each section on the representation for a particular type specifies the encoding rules that define valid representations of a value.

## bdlt::Date Encoding

### Grammar

The contents octets of an encoded `bdlt::Date` value satisfy the `date` nonterminal of the following grammar:

```
                date ::= iso8601_date | compact_binary_date ;
        iso8601_date ::= yyyy , "-" , mm , "-" , dd ;
 compact_binary_date ::= ber_integer ;
```

### Encoding Rules

The encoding of a `bdlt::Date` value shall use 1 of 2 forms:

1. The **ISO 8601** `bdlt::Date` form, `iso8601_date`
2. The compact-binary `bdlt::Date` form, `compact_binary_date`

The form will be selected according to the following procedure:

1. If the `EncodeDateAndTimeTypesAsBinary` encoding option is `true`, the compact-binary `bdlt::Date` form is used, and
2. Otherwise, the **ISO 8601** form is used

### ISO 8601 Form

The **ISO 8601** `bdlt::Date` form is encoded by first converting the date component of the value to integer year, month, and day components according to the rules defined in the **Date Component to Year, Month, and Day Component Conversion** section.

The `yyyy` of the `iso8601_date` is the 4 digit representation of the year component. If there are less than 4 digits in the year component, the `yyyy` is prepended with 0 digits until it contains 4 digits. The `mm` of the `iso8601_date` is the 2 digit representation of the month component. If there is only 1 digit in the month component, a 0 digit is prepended to the `mm`. The dd of the `iso8601_date` is the 2 digit representation of the day component. If there is only 1 digit in the day component, a 0 digit is prepended to the dd. The **ISO 8601** `bdlt::Date` form has a fixed length of 10 octets.

### Compact-Binary Form

The compact-binary `bdlt::Date` form is encoded by first converting the date component to a compact-binary epoch date component according to the rules defined in the **Date Component to Compact-Binary Epoch Date Component Conversion** section.

The `ber_integer` of the `compact_binary_date` is the compact-binary epoch date component encoded as a variable-length BER integer.  The compact-binary `bdlt::Date` form has a variable length between 1 and 3 octets.

### Decoding Rules for Form Disambiguation

The form of the contents octets of an encoded `bdlt::Date` value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 3, decoding proceeds according to the compact-binary form, otherwise
2. Decoding proceeds according to the **ISO 8601** form

Decoding proceeds deterministically as the reverse of the encoding process for the determined form.

## `bdlt::DateTz` Encoding

### Grammar

The contents octets of an encoded `bdlt::DateTz` value satisfy the `datetz` nonterminal of the following grammar:

```
                datetz ::= iso8601_datetz

                         | compact_binary_datetz ;

          iso8601_datetz ::= yyyy , "-" , mm , "-" , dd

                         , iso860_tz_offset ;

compact_binary_datetz ::= compact_binary_tz_offset

                         , compact_binary_days_since_epoch ;

compact_binary_tz_offset

                      ::= 2_octet_twos_comp_integer ;

compact_binary_days_since_epoch

                      ::= ber_integer ;
```

### Encoding Rules

The encoding of a `bdlt::DateTz` value shall use one of two forms:

1. The **ISO 8601** `bdlt::DateTz` form, `iso8601_datetz`
2. The compact-binary `bdlt::DateTz` form, `compact_binary_datetz`

The form will be selected according to the following procedure:

3. If the `EncodeDateAndTimeTypesAsBinary` encoding option is `true`, the compact-binary `bdlt::DateTz` form is used, and
4. Otherwise, the **ISO 8601** `bdlt::DateTz` form is used

### ISO 8601 Form

The **ISO 8601** `bdlt::DateTz` form is encoded by first converting the date component of the value to year, month, and day components according to the rules defined in the **Date Component to Year, Month, and Day Component Conversion** section, and converting the time-zone component to offset-sign, hour-offset, and minute-offset components according to the rules defined in **Time-Zone Component to Offset-Sign, Hour-Offset, and Minute-Offset Components Conversion** section.

The `yyyy` of the `iso8601_datetz` is the 4-digit representation of the year component. If there are less than 4 digits in the year component, the `yyyy` is prepended with 0 digits until it contains 4 digits. The mm of the `iso8601_datetz` is the 2-digit representation of the month component. If there is only 1 digit in the month component, a 0 digit is prepended to the `mm`. The dd of the `iso8601_datetz` is the 2-digit representation of the day component. If there is only 1 digit in the day component, a 0 digit is prepended to the dd.

If the offset-sign component is negative, the first octet of the `iso8601_tz_offset` is `"-"`, and is `"+"` otherwise. The hh of the `iso8601_tz_offset` is the 2-digit representation of the hour-offset component. If there is only 1 digit in the hour-offset component, a 0 digit is prepended to the hh. The `mm` of the `iso8601_tz_offset` is the 2-digit representation of the minute-offset component. If there is only 1 digit in the minute-offset component, a 0 digit is prepended to the `mm`. The **ISO 8601** `bdlt::DateTz` form has a fixed length of 16 octets.

### Compact-Binary Form

The compact-binary `bdlt::DateTz` form is encoded by first converting the date component to a compact-binary epoch date component according to the rules defined in the **Date Component to Compact-Binary Epoch Date Component Conversion** section.

The `2_octet_twos_comp_integer` of the `compact_binary_tz_offset` is the time-zone component encoded as a 16-bit, 2's-complement, big-endian, signed integer. The `ber_integer` of the `compact_binary_days_since_epoch` is the compact-binary epoch date component encoded as a variable-length BER integer, but sign-extended to at least 2 octets if it would otherwise be 1 octet.  The compact-binary `bdlt::DateTz` form has a variable length between 4 and 5 octets.

### Decoding Rules for Form Disambiguation

The form of the contents octets of an encoded `bdlt::DateTz` value is determined according to the following procedure:

1. If the length of the encoding is less than 4, the encoding is rejected as invalid,
2. If the length of the encoding is 4 or 5, decoding proceeds according to the compact-binary form,
3. Otherwise, decoding proceeds according to the **ISO 8601** form

## `bdlt::Date` and `bdlt::DateTz` Disambiguation

The BDE BER decoder permits decoding an encoded `bdlt::Date` or `bdlt::DateTz` into an object of type `bdlb::Variant2<bdlt::Date, bdlt::DateTz>`, hereafter referred to as a `DateOrDateTz`.  To do so, the decoder must disambiguate an encoded `bdlt::Date` from an encoded `bdlt::DateTz`.  The type of such an encoded value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 3, decoding proceeds with a `bdlt::Date` selection, otherwise
2. If the length of the encoding is less than or equal to 5, decoding proceeds with a `bdlt::DateTz` selection, otherwise
3. If the length of the encoding is less than or equal to 10, decoding proceeds with a `bdlt::Date` selection, otherwise
4. Decoding proceeds with a `bdlt::DateTz` selection

## `bdlt::Time` Encoding

### Grammar

The contents octets of an encoded `bdlt::Time` value satisfy the time nonterminal of the following grammar:

```
time ::= iso8601_time

       | compact_binary_time

       | extended_binary_time ;
```

```
        iso8601_time ::= hh , ":" , mm ":" , ss
                       , iso8601_fractional_seconds ;
 compact_binary_time ::= ber_integer ;
extended_binary_time ::= '1' , '0' , '0' , '0'
                       , 12 * '0'
                       , 5_octet_twos_comp_integer ;
```

### Encoding Rules

The encoding of a `bdlt::Time` value shall use 1 of 3 forms:

1. The **ISO 8601** `bdlt::Time` form, `iso8601_time`
2. The compact-binary `bdlt::Time` form, `compact_binary_time`
3. The extended-binary `bdlt::Time` form, `extended_binary_time`

The form is selected according to the following procedure:

1. If all of the following are true, the extended-binary `bdlt::Time` form is used:
   a. The `EncodeDateAndTimeTypesAsBinary` encoding option is true,
   b. The `BdeVersionConformance` option is `35500` or greater,
   c. Either or both of the following are true:
      i. The `DatetimeFractionalSecondPrecision` option is `6`
      ii. The time component of the value to encode represents exactly 24 hours, otherwise
2. If the `EncodeDateAndTimeTypesAsBinary` encoding option is `true`, the compact-binary `bdlt::Time` form is used, otherwise
3. The ISO 8601 `bdlt::Time` form is used

### ISO 8601 Form

The **ISO 8601** `bdlt::Time` form is encoded by first converting the time component of the value to hour, minute, second, millisecond, and microsecond components according to the rules defined in the **Time Component to Hour, Minute, Second, Millisecond, and Microsecond Components Conversion** section.

The hh of the `iso8601_time` is the 2-digit representation of the hour component. If there is only 1 digit in the hour component, a 0 digit is prepended to the hh. The mm of the `iso8601_time` is the 2-digit representation of the minute component. If there is only 1 digit in the minute component, a 0 digit is prepended to the mm. If the `DatetimeFractionalSecondPrecision` option is 0, the `iso8601_fraction_seconds` of the `iso8601_time` is the empty string. Otherwise, the `iso8601_fractional_seconds` is a `"."` octet followed by the 1-to-6-digit representation of the combined millisecond and microsecond components, having the number of digits specified by the `DatetimeFractionalSecondPrecision` option.

### Compact-Binary Form

The compact-binary `bdlt::Time` form is encoded by first converting the time component of the value to a compact-binary time component according to the rules defined in the **Time Component to Compact-Binary Time Component Conversion** section.

The `ber_integer` of the `compact_binary_time` is the compact-binary time component encoded as a variable-length BER integer. The compact-binary `bdlt::Time` form has a variable length between 1 and 4 octets.

### Extended-Binary Form

In the extended-binary `bdlt::Time` form, the `5_octet_twos_comp_integer` of the `extended_binary_time` is the value of the time component encoded as a 40-bit, big-endian, 2's-complement unsigned integer number of microseconds since 12:00AM at the beginning of the day. The extended-binary `bdlt::Time` form has a fixed length of 7 octets.

### Decoding Rules for Form Disambiguation

The form of the contents octets of an encoded `bdlt::Time` value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 4, decoding proceeds according to the compact-binary form, otherwise
2. If the first 4 bits of the encoding are `'1000'`, decoding proceeds according to the extended-binary form, otherwise
3. Decoding proceeds according to the ISO 8601 form

Decoding proceeds deterministically as the reverse of the encoding process for the determined form.

## `bdlt::TimeTz` Encoding

### Grammar

The contents octets of an encoded `bdlt::TimeTz` value satisfy the timetz nonterminal of the following grammar:

```
timetz ::= iso8601_timetz
```

```
                          | compact_binary_timetz

                          | extended_binary_timetz ;

         iso8601_timetz ::= hh , ":" , mm , ":" , ss

                          , iso8601_fractional_seconds

                          , iso8601_tz_offset ;

  compact_binary_timetz ::= compact_binary_time

                          | compact_binary_tz_offset

                          , compact_binary_time ;

 extended_binary_timetz ::= '1' , '0' , '0' , '1'

                          , extended_binary_tz_offset

                          , 5_octet_twos_comp_integer ;
```

### Encoding Rules

The encoding of a `bdlt::TimeTz` value shall use 1 of 3 forms:

1. The **ISO 8601** `bdlt::TimeTz` form, `iso8601_timetz`
2. The compact-binary `bdlt::TimeTz` form, `compact_binary_timetz`
3. The extended-binary `bdlt::TimeTz` form, `extended_binary_timetz`

The form is selected according to the following procedure:

4. If all of the following are true, the extended-binary `bdlt::Time` form is used:
    a. The `EncodeDateAndTimeTypesAsBinary` encoding option is `true`,
    b. The `BdeVersionConformance` option is `35500` or greater,
    c. Either or both of the following are true:
        i. The `DatetimeFractionalSecondPrecision` option is 6
        ii. The time component of the value is 12:00AM at the end of the day, otherwise
5. If the `EncodeDateAndTimeTypesAsBinary` encoding option is `true`, the compact-binary `bdlt::TimeTz` form is used, otherwise
6. The **ISO 8601** `bdlt::TimeTz` form is used

### ISO 8601 Form

The **ISO 8601** `bdlt::TimeTz` form is encoded by first converting the time component of the value to hour, minute, second, millisecond, and microsecond components according to the defined in the

**Time Component to Hour, Minute, Second, Millisecond, and Microsecond Component Conversion** section, and converting the time-zone component to offset-sign, hour-offset, and minute-offset components according to the rules defined in the **Time-Zone Component to Offset-Sign, Hour-Offset, and Minute-Offset Component Conversion** section.

The hh of the `iso8601_timetz` is the 2-digit representation of the hour component. If there is only 1 digit on the hour component, a `0` digit is prepended to the hh. The mm of the `iso8601_timetz` is the 2-digit representation of the minute component. If there is only 1 digit in the minute component, a `0` digit is prepended to the mm. If the `DatetimeFractionalSecondPrecision` option is `0`, the `iso8601_fractional_seconds` of the `iso8601_time` is the empty string. Otherwise, the `iso8601_fractional_seconds` is a `"."` octet followed by the 1-to-6-digit representation of the combined millisecond and microsecond components, having the number of digits specified by the `DatetimeFractionalSecondPrecision` option.

If the offset-sign component is negative, the first octet of the `iso8601_tz_offset` is `"-"`, and is `"+"` otherwise. The hh of the `iso8601_tz_offset` is the 2-digit representation of the hour-offset component. If there is only 1 digit in the hour-offset component, a `0` digit is prepended to the hh. The mm of the `iso8601_tz_offset` is the 2-digit representation of the minute-offset component. If there is only 1 digit in the minute-offset component, a `0` digit is prepended to the mm. The **ISO 8601** `bdlt::TimeTz` form has a variable length between 13 and 19 octets.

### Compact-Binary Form

The compact-binary `bdlt::TimeTz` form is encoded by first converting the time component of the value to a compact-binary time component according to the rules defined in the **Time Component to Compact-Binary Time Component Conversion** section.

If the time-zone component of the value is 12:00AM at the beginning or the end of the day, the `compact_binary_timetz` uses the `compact_binary_time` alternative. The `ber_integer` of the `compact_binary_time` is the compact-binary time component encoded as a variable-length BER integer. When decoding this alternative, the time-zone component of the value is taken to be 12:00AM at the beginning of the day.

Otherwise, the `compact_binary_timetz` uses the alternative that is a sequence of a `compact_binary_tz_offset` and a `compact_binary_time`. The `2_octet_twos_comp_integer` of the `compact_binary_tz_offset` is the time-zone component encoded as a 16-bit 2's-complement, big-endian, signed integer. The `ber_integer` of the `compact_binary_time` is the compact-binary time component encoded as a variable-length BER integer.

### Extended-Binary Form

In the extended-binary `bdlt::TimeTz` form, the `extended_binary_tz_offset` of the `extended_binary_timetz` is the time-zone offset component encoded as a 12-bit, 2's-complement, big-endian signed integer. The `5_octet_twos_comp_integer` is the time component represented as a 40-bit, big-endian, 2's-complement unsigned integer number of microseconds since 12:00AM at the beginning of the day. The extended-binary `bdlt::TimeTz` form has a fixed length of 7 octets.

### Decoding Rules for Form Disambiguation

The form of the contents octets of an encoded `bdlt::TimeTz` value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 6, decoding proceeds according to the compact-binary form, otherwise
2. If the first 4 bits of the encoding are `'1001'`, decoding proceeds according to the extended-binary form, otherwise
3. Decoding proceeds according to the ISO 8601 form

## `bdlt::Time` and `bdlt::TimeTz` Disambiguation

The BDE BER decoder permits decoding an encoded `bdlt::Time` or `bdlt::TimeTz` into an object of type `bdlb::Variant2<bdlt::Time, bdlt::TimeTz>`, hereafter referred to as a `TimeOrTimeTz`. To do so, the decoder must disambiguate an encoded `bdlt::Time` from an encoded `bdlt::TimeTz`. The type of such an encoded value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 4, decoding proceeds with a `bdlt::Time` selection, otherwise
2. If the length of the encoding is less than or equal to 6, decoding proceeds with a `bdlt::TimeTz` selection, otherwise
3. If the length of the encoding is 7 and the first 4 bits of the encoding are `'1000'`, decoding proceeds with a `bdlt::Time` selection, otherwise
4. If the length of the encoding is 7 and the first 4 bits of the encoding are `'1001'`, decoding proceeds with a `bdlt::TimeTz` selection, otherwise
5. If the length of the encoding is less than or equal to 15, decoding proceeds with a `bdlt::Time` selection, otherwise
6. Decoding proceeds with a `bdlt::TimeTz` selection

## `bdlt::Datetime Encoding`

## Grammar

The contents octets of an encoded `bdlt::Datetime` value satisfy the `datetime` nonterminal of the following grammar:

```
                datetime ::= iso8601_datetime
                           | compact_binary_datetime
                           | extended_binary_datetime ;
        iso8601_datetime ::= yyyy , "-" , mm , "-" , dd , "T"
                           , hh   , ":" , mm , ":" , ss
                           , iso8601_fractional_seconds ;
 compact_binary_datetime ::= ber_integer ;
extended_binary_datetime ::= '1' , '0' , '0' , '0'
                           , 12 * '0'
                           , 3_octet_twos_comp_integer
                           , 5_octet_twos_comp_integer ;
```

## Encoding Rules

The encoding of a `bdlt::Datetime` value shall use 1 of 3 forms:

1. The ISO 8601 `bdlt::Datetime` form, `iso8601_datetime`
2. The compact-binary `bdlt::Datetime` form, `compact_binary_datetime`
3. The extended-binary `bdlt::Datetime` form, `extended_binary_datetime`

The form is selected according to the following procedure:

1. If all of the following are true, the extended-binary `bdlt::Datetime` form is used:
    a. The `EncodeDateAndTimeTypesAsBinary` encoding option is true,
    b. The `BdeVersionConformance` option is `35500` or greater,
    c. Either or both of the following are true:
        i. The `DatetimeFractionalSecondPrecision` option is 6
        ii. The time component of the value to encode represents exactly 24 hours, otherwise
2. If the `EncodeDateAndTimeTypesAsBinary` encoding option is `true`, the compact-binary `bdlt::Datetime` form is used, otherwise

3. The ISO 8601 `bdlt::Datetime` form is used

### ISO 8601 Form

The **ISO 8601** `bdlt::Datetime` form is encoded by first converting the date component of the value to year, month, and day components according to the rules defined in the **Date Component to Year, Month, and Day Component Conversion** section, and the time component of the value to hour, minute, second, millisecond, and microsecond components according to the rules defined in the **Time Component to Hour, Minute, Second, Millisecond, and Microsecond Component Conversion** section.

The `yyyy` of the `iso8601_datetime` is the 4 digit representation of the year component. If there are less than 4 digits in the year component, the `yyyy` is prepended with 0 digits until it contains 4 digits. The mm of the `iso8601_datetime` is the 2 digit representation of the month component. If there is only 1 digit in the month component, a 0 digit is prepended to the mm. The dd of the `iso8601_datetime` is the 2 digit representation of the day component. If there is only 1 digit in the day component, a 0 digit is prepended to the dd.

The hh of the `iso8601_datetime` is the 2-digit representation of the hour component. If there is only 1 digit in the hour component, a 0 digit is prepended to the hh. The mm of the `iso8601_datetime` is the 2-digit representation of the minute component. If there is only 1 digit in the minute component, a 0 digit is prepended to the mm. If the `DatetimeFractionalSecondPrecision` option is 0, the `iso8601_fraction_seconds` of the `iso8601_datetime` is the empty string. Otherwise, the `iso8601_fractional_seconds` is a `"."` octet followed by the 1-to-6-digit representation of the combined millisecond and microsecond components, having the number of digits specified by the `DatetimeFractionalSecondPrecision` option.

The **ISO 8601** `bdlt::Datetime` form has a variable length between 19 and 26 octets.

### Compact-Binary Form

The compact-binary `bdlt::Datetime` form is encoded by first converting the date and time components to a compact-binary epoch datetime component according to the rules defined in the **Date and Time Components to Compact-Binary Epoch Datetime Component Conversion** section.

The `ber_integer` of the `compact_binary_datetime` is the compact-binary epoch datetime component encoded as a variable-length BER integer. The compact-binary `bdlt::Datetime` form has a variable length between 1 and 6 octets.

### Extended-Binary Form

In the extended-binary `bdlt::Datetime` form, the `3_octet_twos_comp_integer` of the `extended_binary_datetime` is the value of the date component encoded as a 24-bit, 2's-complement unsigned integer number of days since January 1st year 1 in the Proleptic Gregorian calendar. The `5_octet_twos_comp_integer` of the `extended_binary_datetime` is the value of the time component encoded as a 40-bit, big-endian, 2's-complement unsigned integer number of microseconds since 12:00AM at the beginning of the day. The extended-binary `bdlt::Datetime` form has a fixed length of 10 octets.

### Decoding Rules for Form Disambiguation

The form of the contents octets of an encoded `bdlt::Datetime` value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 6, decoding proceeds according to the compact-binary form, otherwise
2. If the first 4 bits of the encoding are `'1000'`, decoding proceeds according to the extended-binary form, otherwise
3. Decoding proceeds according to the **ISO 8601** form

## `bdlt::DatetimeTz` Encoding

### Grammar

The contents octets of an encoded `bdlt::DatetimeTz` value satisfy the `datetimetz` nonterminal of the following grammar:

```
               datetimetz ::= iso8601_datetimetz

                            | compact_binary_datetimetz

                            | extended_binary_datetimetz ;

       iso8601_datetimetz ::= yyyy , "-" , mm , "-" , dd , "T"

                            , hh   , ":" , mm , ":" , ss

                            , iso8601_fractional_seconds

                            , iso8601_tz_offset ;

compact_binary_datetimetz ::= compact_binary_datetime

                            | compact_binary_tz_offset
```

```
                                  , compact_binary_datetime ;

extended_binary_datetimetz ::= '1' , '0' , '0' , '1'

                                  , extended_binary_tz_offset

                                  , 3_octet_twos_comp_integer

                                  , 5_octet_twos_comp_integer ;
```

### Encoding Rules

The encoding of a `bdlt::DatetimeTz` value shall use 1 of 3 forms:

1. The **ISO 8601** `bdlt::DatetimeTz` form, `iso8601_datetimetz`
2. The compact-binary `bdlt::DatetimeTz` form, `compact_binary_datetimetz`
3. The extended-binary `bdlt::DatetimeTz` form, `extended_binary_datetimetz`

The form is selected according to the following procedure:

1. If all of the following are true, the extended-binary `bdlt::DatetimeTz` form is used:
   a. The `EncodeDateAndTimeTypesAsBinary` encoding option is `true`,
   b. The `BdeVersionConformance` option is `35500` or greater,
   c. Either or both of the following are true:
      i. The `DatetimeFractionalSecondPrecision` option is 6
      ii. The time component of the value is 12:00AM at the end of the day, otherwise
2. If the `EncodeDateAndTimeTypesAsBinary` encoding option is `true`, the compact-binary `bdlt::DatetimeTz` form is used, otherwise
3. The **ISO 8601** `bdlt::DatetimeTz` form is used

### ISO 8601 Form

The **ISO 8601** `bdlt::TimeTz` form is encoded by first converting the time component of the value to hour, minute, second, millisecond, and microsecond components according to the defined in the **Time Component to Hour, Minute, Second, Millisecond, and Microsecond Component Conversion** section, and converting the time-zone component to offset-sign, hour-offset, and minute-offset components according to the rules defined in **Time-Zone Component to Offset-Sign, Hour-Offset, and Minute-Offset Component Conversion** section.

The `yyyy` of the `iso8601_datetimetz` is the 4 digit representation of the year component. If there are less than 4 digits in the year component, the `yyyy` is prepended with 0 digits until it contains 4 digits. The `mm` of the `iso8601_datetimetz` is the 2 digit representation of the month component. If there is only 1 digit in the month component, a 0 digit is prepended to the `mm`. The dd of the `iso8601_datetimetz` is the 2 digit representation of the day component. If there is only 1 digit in the day component, a 0 digit is prepended to the dd.

The hh of the `iso8601_datetimetz` is the 2-digit representation of the hour component. If there is only 1 digit in the hour component, a 0 digit is prepended to the hh. The mm of the `iso8601_datetimetz` is the 2-digit representation of the minute component. If there is only 1 digit in the minute component, a 0 digit is prepended to the mm. If the `DatetimeFractionalSecondPrecision` option is 0, the `iso8601_fraction_seconds` of the `iso8601_datetimetz` is the empty string. Otherwise, the `iso8601_fractional_seconds` is a `"."` octet followed by the 1-to-6-digit representation of the combined millisecond and microsecond components, having the number of digits specified by the `DatetimeFractionalSecondPrecision` option.

If the offset-sign component is negative, the first octet of the `iso8601_tz_offset` is `"-"`, and is `"+"` otherwise. The hh of the `iso8601_tz_offset` is the 2-digit representation of the hour-offset component. If there is only 1 digit in the hour-offset component, a `0` digit is prepended to the hh. The mm of the `iso8601_tz_offset` is the 2-digit representation of the minute-offset component. If there is only 1 digit in the minute-offset component, a `0` digit is prepended to the mm. The **ISO 8601** `bdlt::DatetimeTz` form has a variable length between 25 and 32 octets.

### Compact-Binary Form

The compact-binary `bdlt::DatetimeTz` form is encoded by first converting the date and time components of the value to a compact-binary epoch datetime component according to the rules defined in the **Date and Time Components to Compact-Binary Epoch Datetime Component Conversion** section.

If the time-zone component of the value is 12:00AM at the beginning or the end of the day, the `compact_binary_datetimetz` uses the `compact_binary_datetime` alternative. The `ber_integer` of the `compact_binary_datetime` is the compact-binary epoch datetime component encoded as a variable-length BER integer. When decoding this alternative, the time-zone component of the value is taken to be 12:00AM at the beginning of the day.

Otherwise, the `compact_binary_datetimetz` uses the alternative that is a sequence of a `compact_binary_tz_offset` and a `compact_binary_datetime`. The `2_octet_twos_comp_integer` of the `compact_binary_tz_offset` is the time-zone component encoded as a 16-bit 2's-complement, big-endian, signed integer. The `ber_integer` of the `compact_binary_datetime` is the compact-binary epoch datetime component encoded as a variable-length BER integer.

### Extended-Binary Form

In the extended-binary `bdlt::DatetimeTz` form, the `extended_binary_tz_offset` of the `extended_binary_datetimetz` is the time-zone offset component encoded as a 12-bit, 2's-complement, big-endian signed integer. The `3_octet_twos_comp_integer` of the `extended_binary_datetimetz` is the value of the date component encoded as a 24-bit, 2's-complement unsigned integer number of days since January 1st year 1 in the Proleptic Gregorian calendar. The `5_octet_twos_comp_integer` of the `extended_binary_datetimetz` is the value of the time component encoded as a 40-bit, big-endian, 2's-complement unsigned integer number of microseconds since 12:00AM at the beginning of the day. The extended-binary `bdlt::DatetimeTz` form has a fixed length of 10 octets.

### Decoding Rules for Form Disambiguation

The form of the contents octets of an encoded `bdlt::DatetimeTz` value is determined according to the following procedure:

1. if the length of the encoding is less than or equal to 9, decoding proceeds according to the compact-binary form, otherwise
2. If the first 4 bits of the encoding are `'1001'`, decoding proceeds according to the extended-binary form, otherwise
3. Decoding proceeds according to the **ISO 8601** form

Decoding proceeds deterministically as the reverse of the encoding process for the determined form.

## `bdlt::Datetime` and `bdlt::DatetimeTz` Disambiguation

The BDE BER decoder permits decoding an encoded `bdlt::Datetime` or `bdlt::DatetimeTz` into an object of type `bdlb::Variant2<bdlt::Datetime, bdlt::DatetimeTz>`, hereafter referred to as a `DatetimeOrDatetimeTz`. To do so, the decoder must disambiguate an encoded `bdlt::Datetime` from an encoded `bdlt::DatetimeTz`. The type of such an encoded value is determined according to the following procedure:

1. If the length of the encoding is less than or equal to 6, decoding proceeds with a `bdlt::Datetime` selection, otherwise
2. If the length of the encoding is less than or equal to 9, decoding proceeds with a `bdlt::DatetimeTz` selection, otherwise
3. If the first 4 bits of the encoding are '1000', decoding proceeds with a `bdlt::Datetime` selection, otherwise
4. If the first 4 bits of the encoding are '1001', decoding proceeds with a `bdlt::DatetimeTz` selection, otherwise

5. If the length of the encoding is less than or equal to 26, decoding proceeds with a `bdlt::Datetime` selection, otherwise
6. Decoding proceeds with a `bdlt::DatetimeTz` selection

## Date Component to Year, Month, and Day Component Conversion

A date component is converted to integer year, month, and day components according to the following rules:

1. The year component is the non-negative integer number of whole years in the date since year 1 in the Proleptic Gregorian calendar
2. The month component is the remaining non-negative integer number of whole months in the date component since January, counting from 1, such that January is represented as 1, February as 2, etc., according to the Proleptic Gregorian calendar
3. The day component is the remaining number of whole days in the date component since the start of the month component, such that the first day of the month is represented as 1, the second as 2, etc., according to the Proleptic Gregorian calendar

## Date Component to Compact-Binary Epoch Date Component Conversion

A date component is converted to a compact-binary epoch date component according to the following rules:

1. The compact-binary epoch date component is the date component converted to an integer number of days before or since the compact-binary date epoch, which is January 1st 2020 in the Proleptic Gregorian calendar
    a. Days before the epoch are represented by negative integers (For example, December 31st 2019 is represented as `-1`)
    b. The epoch is represented by the integer `0`
    c. Days after the epoch are represented by positive integers (For example, January 2nd 2020 is represented as `1`)

## Time Component to Hour, Minute, Second, Millisecond, and Microsecond Components Conversion

A time component is converted to hour, minute, second, millisecond, and microsecond components according to the following rules:

1. The hour component is the non-negative integer number of whole hours in the time component, then

2. The minute component is the remaining non-negative integer number of whole minutes in the time component, then
3. the second component is the remaining non-negative integer number of whole seconds in the time component, then
4. the millisecond component is the remaining non-negative integer number of whole milliseconds in the time component, then
5. the microsecond component is the  remaining non-negative integer number of microseconds in the time component

### Time Component to Compact-Binary Time Component Conversion

A time component is converted to a compact-binary time component according to the following rules:

1. the compact-binary time component is the time component converted to an integer number of milliseconds since 12:00AM at the beginning of the day

### Date and Time Components to Compact-Binary Epoch Datetime Component Conversion

A pair of date and time components are converted to a compact-binary epoch datetime component according to the following rules:

1.

### Time-Zone Component to Offset-Sign, Hour-Offset, and Minute-Offset Components Conversion

A time-zone component is converted to offset-sign, hour-offset, and minute-offset components according to the following rules:

1. The offset-sign component is positive if the time-zone component is UTC or a time-zone after UTC, and is negative otherwise
2. The hour-offset component is the non-negative integer number of whole hours in the time-zone component, then
3. The minute-offset component is the remaining non-negative integer number of minutes in the time-zone component

### Time-Zone Component to Time-Zone Offset Component Conversion

A time-zone component is converted to a time-zone offset component according to the following rules:

1. If the time-zone component is before UTC, the time-zone offset component is the negative integer number of minutes before UTC in the time-zone component, otherwise
2. If the time-zone component is UTC, the time-zone offset component is `0`, otherwise
3. If the time-zone component is after UTC, the time-zone offset component is the positive integer number of minutes after UTC in the time-zone component

## Shared Grammar Nonterminals

```
            yyyy ::= 4 * digit ;
              hh ::= 2 * digit ;
              mm ::= 2 * digit ;
              ss ::= 2 * digit ;
              dd ::= 2 * digit ;
iso8601_tz_offset
                 ::= "+" , hh , ":" , mm
                   | "-" , hh , ":" , mm ;
iso8601_fractional_seconds
                 ::= "." , ( 1 * digit | 2 * digit | 3 * digit
                           | 4 * digit | 5 * digit | 6 * digit ) ;
           digit ::= "0" | "1" | "2" | "3" | "4"
                   | "5" | "6" | "7" | "8" | "9" ;
             bit ::= '0' | '1' ;
           octet ::= 8 * bit ;
ber_integer ::= octet | octet , ber_integer ;
3_octet_twos_comp_integer
                 ::= 3 * octet ;
5_octet_twos_comp_integer
                 ::= 5 * octet ;
extended_binary_header
```

```
                ::= extended_binary_header_without_tz
                  | extended_binary_header_with_tz ;
extended_binary_header_without_tz
                ::= '1' , '0' , '0' , '0' ;
extended_binary_header_with_tz
                ::= '1' , '0' , '0' , '1' ;
extended_binary_tz_offset
                ::= 12 * bit ;
```