INTERNATIONAL STANDARD

ISO/IEC 8825-2

First edition
2002-12-15

AMENDMENT 2
2007-04-01

Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER) —

AMENDMENT 2: Time type support

Technologies de l'information — Règles de codage ASN.1: Spécification des règles de codage compact (PER) —

AMENDEMENT 2: Support de type temps

Reference number
ISO/IEC 8825-2:2002/Amd.2:2007(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 2 to ISO/IEC 8825-2:2002 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems,* in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.691:2002/Amd.2.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)

Amendment 2

Time type support

1) Contents

Update the Contents as follows:

2) New clause 9.3.11 *bis*

Insert clause 9.3.11 bis after clause 9.3.11:

9.3.11 *bis* Property setting constraints on the time type (or on the useful and defined time types) which are not extensible after the application of ITU-T Rec. X.680 | ISO/IEC 8824-1, 48.3 to 48.5, are PER-visible. Property setting constraints which are extensible are not PER-visible.

3) New clause 28 *bis*

Insert clause 28 bis before clause 29:

28 *bis* Encoding the time type, the useful time types, the defined time types and the additional time types

28 *bis*.1 General

28 *bis*.1.1 The encoding of the useful time types, the defined time types and the additional time types shall be determined by the property settings of the abstract values of these types. Property settings for the abstract values of the useful and defined time types are specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 34 *bis*.4 and Annex A *bis*, respectively. Property settings for the abstract values of additional time types are determined by the property settings of the parent type, restricted by any PER-visible constraints that apply (see 9.3.11 *bis*).

28 *bis*.1.2 If all the abstract values of the type to be encoded have one of the property settings listed in a row of column 2 of Table 2, then that type shall be encoded as if the type with its constraints (if any) had been replaced by the type specified in the corresponding row of column 3 of Table 2. Otherwise, it shall be encoded as specified in 28 *bis*.11.

NOTE – If a time property (for example **`Midnight`**) is not listed in Table 2 for a particular row, there is no constraint on its setting.

28 *bis*.1.3 For rows 24 to 32 to be applicable, all abstract values of the type are required to have the same value of ***`n`*** in **`F`*`n`***.

28 *bis*.1.4 The types specified in column 3 of Table 2 are defined (using the ASN.1 notation) in 28 *bis*.2 to 28 *bis*.10, and are assumed to be defined in an environment of **`AUTOMATIC TAGS`**.

NOTE 1 – The use of these type reference names in the specification of PER encodings does not make them available for use by an application designer in an ASN.1 specification, nor are they reserved words in such a specification. However, with the removal of **`-ENCODING`**, they correspond to the names of the useful time types or defined time types specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 34 *bis*.4 and Annex A *bis*.

NOTE 2 – All the useful and defined time types satisfy the conditions for one of the rows of Table 2, and hence have optimized encodings. Additional time types may satisfy the conditions for one of the rows, but are otherwise encoded as specified in 28 *bis*.11. The unconstrained **`TIME`** type is always encoded as specified in 28 *bis*.11.

Table 2 – Encoding of a time subtype with all abstract values having specified property settings

Row number	Property settings	ASN.1 type to be encoded
1	**`"Basic=Date Date=C Year=Basic"`** or **`"Basic=Date Date=C Year=Proleptic"`**	**`CENTURY-ENCODING`** (see 28 *bis*.2.1)
2	**`"Basic=Date Date=C Year=Negative"`** or **`"Basic=Date Date=C Year=L`*`n`*`"`** (for any *n*)	**`ANY-CENTURY-ENCODING`** (see 28 *bis*.2.2)
3	**`"Basic=Date Date=Y Year=Basic"`** or **`"Basic=Date Date=Y Year=Proleptic"`**	**`YEAR-ENCODING`** (see 28 *bis*.2.3)
4	**`"Basic=Date Date=Y Year=Negative"`** or **`"Basic=Date Date=Y Year=L`*`n`*`"`** (for any *n*)	**`ANY-YEAR-ENCODING`** (see 28 *bis*.2.4)
5	**`"Basic=Date Date=YM Year=Basic"`** or **`"Basic=Date Date=YM Year=Proleptic"`**	**`YEAR-MONTH-ENCODING`** (see 28 *bis*.2.5)
6	**`"Basic=Date Date=YM Year=Negative"`** or **`"Basic=Date Date=YM Year=L`*`n`*`"`** (for any *n*)	**`ANY-YEAR-MONTH-ENCODING`** (see 28 *bis*.2.6)
7	**`"Basic=Date Date=YMD Year=Basic"`** or **`"Basic=Date Date=YMD Year=Proleptic"`**	**`DATE-ENCODING`** (see 28 *bis*.2.7)
8	**`"Basic=Date Date=YMD Year=Negative"`** or **`"Basic=Date Date=YMD Year=L`*`n`*`"`** (for any *n*)	**`ANY-DATE-ENCODING`** (see 28 *bis*.2.8)

Table 2 – Encoding of a time subtype with all abstract values having specified property settings

Row number	Property settings	ASN.1 type to be encoded
9	`"Basic=Date Date=YD Year=Basic"` or `"Basic=Date Date=YD Year=Proleptic"`	`YEAR-DAY-ENCODING` (see 28 *bis*.2.9)
10	`"Basic=Date Date=YD Year=Negative"` or `"Basic=Date Date=YD Year=L`*n*`"` (for any *n*)	`ANY-YEAR-DAY-ENCODING` (see 28 *bis*.2.10)
11	`"Basic=Date Date=YW Year=Basic"` or `"Basic=Date Date=YW Year=Proleptic"`	`YEAR-WEEK-ENCODING` (see 28 *bis*.2.11)
12	`"Basic=Date Date=YW Year=Negative"` or `"Basic=Date Date=YW Year=L`*n*`"` (for any *n*)	`ANY-YEAR-WEEK-ENCODING` (see 28 *bis*.2.12)
13	`"Basic=Date Date=YWD Year=Basic"` or `"Basic=Date Date=YWD Year=Proleptic"`	`YEAR-WEEK-DAY-ENCODING` (see 28 *bis*.2.13)
14	`"Basic=Date Date=YWD Year=Negative"` or `"Basic=Date Date=YWD Year=L`*n*`"` (for any *n*)	`ANY-YEAR-WEEK-DAY-ENCODING` (see 28 *bis*.2.14)
15	`"Basic=Time Time=H Local-or-UTC=L"`	`HOURS-ENCODING` (see 28 *bis*.3.1)
16	`"Basic=Time Time=H Local-or-UTC=Z"`	`HOURS-UTC-ENCODING` (see 28 *bis*.3.2)
17	`"Basic=Time Time=H Local-or-UTC=LD"`	`HOURS-AND-DIFF-ENCODING` (see 28 *bis*.3.3)
18	`"Basic=Time Time=HM Local-or-UTC=L"`	`MINUTES-ENCODING` (see 28 *bis*.3.4)
19	`"Basic=Time Time=HM Local-or-UTC=Z"`	`MINUTES-UTC-ENCODING` (see 28 *bis*.3.5)
20	`"Basic=Time Time=HM Local-or-UTC=LD"`	`MINUTES-AND-DIFF-ENCODING` (see 28 *bis*.3.6)
21	`"Basic=Time Time=HMS Local-or-UTC=L"`	`TIME-OF-DAY-ENCODING` (see 28 *bis*.3.7)
22	`"Basic=Time Time=HMS Local-or-UTC=Z"`	`TIME-OF-DAY-UTC-ENCODING` (see 28 *bis*.3.8)
23	`"Basic=Time Time=HMS Local-or-UTC=LD"`	`TIME-OF-DAY-AND-DIFF-ENCODING` (see 28 *bis*.3.9)
24	`"Basic=Time Time=HFn Local-or-UTC=L"` (but see 28 *bis*.1.3)	`HOURS-AND-FRACTION-ENCODING` (see 28 *bis*.3.10)
25	`"Basic=Time Time=HFn Local-or-UTC=Z"` (but see 28 *bis*.1.3)	`HOURS-UTC-AND-FRACTION-ENCODING` (see 28 *bis*.3.11)
26	`"Basic=Time Time=HFn Local-or-UTC=LD"` (but see 28 *bis*.1.3)	`HOURS-AND-DIFF-AND-FRACTION-ENCODING` (see 28 *bis*.3.12)
27	`"Basic=Time Time=HMFn Local-or-UTC=L"` (but see 28 *bis*.1.3)	`MINUTES-AND-FRACTION-ENCODING` (see 28 *bis*.3.13)
28	`"Basic=Time Time=HMFn Local-or-UTC=Z"` (but see 28 *bis*.1.3)	`MINUTES-UTC-AND-FRACTION-ENCODING` (see 28 *bis*.3.14)
29	`"Basic=Time Time=HMFn Local-or-UTC=LD"` (but see 28 *bis*.1.3)	`MINUTES-AND-DIFF-AND-FRACTION-ENCODING` (see 28 *bis*.3.15)
30	`"Basic=Time Time=HMSFn Local-or-UTC=L"` (but see 28 *bis*.1.3)	`TIME-OF-DAY-AND-FRACTION-ENCODING` (see 28 *bis*.3.16)

Table 2 – Encoding of a time subtype with all abstract values having specified property settings

Row number	Property settings	ASN.1 type to be encoded
31	**"Basic=Time Time=HMSFn Local-or-UTC=Z"** (but see 28 *bis*.1.3)	**TIME-OF-DAY-UTC-AND-FRACTION-ENCODING** (see 28 *bis*.3.17)
32	**"Basic=Time Time=HMSFn Local-or-UTC=LD"** (but see 28 *bis*.1.3)	**TIME-OF-DAY-AND-DIFF-AND-FRACTION-ENCODING** (see 28 *bis*.3.18)
33	**"Basic=Date-Time"** All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for **"Basic=Date"** together with the same additional property settings specified in one of the rows 15 to 32 for **"Basic=Time"**.	**DATE-TIME-ENCODING {Date-Type, Time-Type}** (instantiated as specified in 28 *bis*.4.1)
34	**"Basic=Interval Interval-type=SE SE-point=Date"** All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for **"Basic=Date"**.	**START-END-DATE-INTERVAL-ENCODING {Date-Type}** (see 28 *bis*.5.1)
35	**"Basic=Interval Interval-type=SE SE-point=Time"** All abstract values are required to have the same additional property settings specified in one of rows 15 to 32 for **"Basic=Time"**.	**START-END-TIME-INTERVAL-ENCODING {Time-Type}** (see 28 *bis*.5.2)
36	**"Basic=Interval Interval-type=SE SE-point=Date-Time"** All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for "Basic=Date" together with the same additional property settings specified in one of rows 15 to 32 for **"Basic=Time"**.	**START-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type}** (see 28 *bis*.5.3)
37	**"Basic=Interval Interval-type=D"**	**DURATION-INTERVAL-ENCODING** (see 28 *bis*.6.1)
38	**"Basic=Interval Interval-type=SD SE-point=Date"** All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for **"Basic=Date"**.	**START-DATE-DURATION-INTERVAL-ENCODING {Date-Type}** (see 28 *bis*.7.1)
39	**"Basic=Interval Interval-type=SD SE-point=Time"** All abstract values are required to have the same additional property settings specified in one of rows 15 to 32 for **"Basic=Time"**.	**START-TIME-DURATION-INTERVAL-ENCODING {Time-Type}** (see 28 *bis*.7.2)
40	**"Basic=Interval Interval-type=SD SE-point=Date-Time"** All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for **"Basic=Date"** together with the same additional property settings specified in one of rows 15 to 32 for **"Basic=Time"**.	**START-DATE-TIME-DURATION-INTERVAL-ENCODING {Date-Type, Time-Type}** (see 28 *bis*.7.3)
41	**"Basic=Interval Interval-type=DE SE-point=Date"** All abstract values are required to have the same additional properties specified in one of rows 1 to 14 for **"Basic=Date"**.	**DURATION-END-DATE-INTERVAL-ENCODING {Date-Type}** (see 28 *bis*.7.4)

Table 2 – Encoding of a time subtype with all abstract values having specified property settings

Row number	Property settings	ASN.1 type to be encoded
`42`	`"Basic=Interval Interval-type=DE SE-point=Time"` All abstract values are required to have the same additional properties specified in one of rows 15 to 32 for `"Basic=Time"`.	`DURATION-END-TIME-INTERVAL-ENCODING {Time-Type}` (see 28 *bis*.7.5)
`43`	`"Basic=Interval Interval-type=DE SE-point=Date-Time"` All abstract values are required to have the same additional properties specified in one of rows 1 to 14 for `"Basic=Date"` together with the same additional property settings specified in one of rows 15 to 32 for `"Basic=Time"`.	`DURATION-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type}` (see 28 *bis*.7.6)
`44`	`"Basic=Rec-Interval Interval-type=SE SE-point=Date"` All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for `"Basic=Date"`.	`REC-START-END-DATE-INTERVAL-ENCODING {Date-Type}` (see 28 *bis*.8.1)
`45`	`"Basic=Rec-Interval Interval-type=SE SE-point=Time"` All abstract values are required to have the same additional property settings specified in one of rows 15 to 32 for `"Basic=Time"`.	`REC-START-END-TIME-INTERVAL-ENCODING {Time-Type}` (see 28 *bis*.8.2)
`46`	`"Basic=Rec-Interval Interval-type=SE SE-point=Date-Time"` All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for `"Basic=Date"` together with the same additional property settings specified in one of rows 15 to 32 for `"Basic=Time"`.	`REC-START-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type}` (see 28 *bis*.8.3)
`47`	`"Basic=Rec-Interval Interval-type=D"`	`REC-DURATION-INTERVAL-ENCODING` (see 28 *bis*.9.1)
`48`	`"Basic=Rec-Interval Interval-type=SD SE-point=Date"` All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for `"Basic=Date"`.	`REC-START-DATE-DURATION-INTERVAL-ENCODING {Date-Type}` (see 28 *bis*.10.1)
`49`	`"Basic=Rec-Interval Interval-type=SD SE-point=Time"` All abstract values are required to have the same additional property settings specified in one of rows 15 to 32 for `"Basic=Time"`.	`REC-START-TIME-DURATION-INTERVAL-ENCODING {Time-Type}` (see 28 *bis*.10.2)
`50`	`"Basic=Rec-Interval Interval-type=SD SE-point=Date-Time"` All abstract values are required to have the same additional property settings specified in one of rows 1 to 14 for `"Basic=Date"` together with the same additional property settings specified in one of rows 15 to 32 for `"Basic=Time"`.	`REC-START-DATE-TIME-DURATION-INTERVAL-ENCODING {Date-Type, Time-Type}` (see 28 *bis*.10.3)
`51`	`"Basic=Rec-Interval Interval-type=DE SE-point=Date"` All abstract values are required to have the same additional properties specified in one of rows 1 to 14 for `"Basic=Date"`.	`REC-DURATION-END-DATE-INTERVAL-ENCODING {Date-Type}` (see 28 *bis*.10.4)

Table 2 – Encoding of a time subtype with all abstract values having specified property settings

Row number	Property settings	ASN.1 type to be encoded
52	`"Basic=Rec-Interval Interval-type=DE SE-point=Time"` All abstract values are required to have the same additional properties specified in one of rows 15 to 32 for `"Basic=Time"`.	`REC-DURATION-END-TIME-INTERVAL-ENCODING {Time-Type}` (see 28 *bis*.10.5)
53	`"Basic=Rec-Interval Interval-type=DE SE-point=Date-Time"` All abstract values are required to have the same additional properties specified in one of rows 1 to 14 for `"Basic=Date"` together with the same additional property settings specified in one of rows 15 to 32 for `"Basic=Time"`.	`REC-DURATION-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type}` (see 28 *bis*.10.6)

28 *bis*.2 Encoding subtypes with the `"Basic=Date"` property setting

This subclause defines the ASN.1 types referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Date"` property setting.

28 *bis*.2.1 The `CENTURY-ENCODING` type is:

```
CENTURY-ENCODING ::= INTEGER(0..99) -- 7 bits
```

with the integer value set to the value specified by the first two digits of the year component of the abstract value.

28 *bis*.2.2 The `ANY-CENTURY-ENCODING` type is:

```
ANY-CENTURY-ENCODING ::= INTEGER(MIN..MAX)
```

with the integer value set to the value specified by the year component of the abstract value, ignoring the last two digits.

28 *bis*.2.3 The `YEAR-ENCODING` type is:

```
YEAR-ENCODING ::= CHOICE { -- 2 bits for choice determinant
     immediate       INTEGER (2005..2020),   -- 4 bits
     near-future     INTEGER (2021..2276),   -- 8 bits
     near-past       INTEGER (1749..2004),   -- 8 bits
     remainder       INTEGER (MIN..1748 | 2277..MAX)}
```

with the integer value set to the year component of the abstract value.

NOTE – This has been optimized to provide a 6-bit or a 10-bit encoding in common cases.

28 *bis*.2.4 The `ANY-YEAR-ENCODING` type is:

```
ANY-YEAR-ENCODING ::= INTEGER(MIN..MAX)
```

with the integer value set to the year component of the abstract value.

28 *bis*.2.5 The `YEAR-MONTH-ENCODING` type is:

```
YEAR-MONTH-ENCODING ::= SEQUENCE {
     year            YEAR-ENCODING,
     month           INTEGER (1..12) -- 4 bits -- }
```

with the `YEAR-ENCODING` set according to 28 *bis*.2.3 and the `month` integer value set to the month component of the abstract value.

NOTE – This has been optimized to provide a 10-bit or a 14-bit encoding in common cases.

28 *bis*.2.6 The `ANY-YEAR-MONTH-ENCODING` type is:

```
ANY-YEAR-MONTH-ENCODING ::= SEQUENCE {
     year            ANY-YEAR-ENCODING,
     month           INTEGER (1..12) }
```

with the `ANY-YEAR-ENCODING` set according to 28 *bis*.2.4 and the `month` integer value set to the month component of the abstract value.

28 *bis*.2.7 The `DATE-ENCODING` type is:

```
DATE-ENCODING ::= SEQUENCE {
     year            YEAR-ENCODING,
     month           INTEGER (1..12), -- 4 bits
     day             INTEGER (1..31) -- 5 bits -- }
```

with the `YEAR-ENCODING` set according to 28 *bis*.2.3, the `month` integer value set to the month component of the abstract value and the `day` integer value set to the day component of the abstract value.

NOTE – This has been optimized to provide a 15-bit or a 19-bit encoding in common cases.

28 *bis*.2.8 The `ANY-DATE-ENCODING` type is:

```
ANY-DATE-ENCODING ::= SEQUENCE {
     year            ANY-YEAR-ENCODING,
     month           INTEGER (1..12),
     day             INTEGER (1..31)}
```

with the `ANY-YEAR-ENCODING` set according to 28 *bis*.2.4, the `month` integer value set to the month component of the abstract value and the `day` integer value set to the day component of the abstract value.

28 *bis*.2.9 The `YEAR-DAY-ENCODING` type is:

```
YEAR-DAY-ENCODING ::= SEQUENCE {
     year            YEAR-ENCODING,
     day             INTEGER (1..366)}
```

with the `YEAR-ENCODING` set according to 28 *bis*.2.3 and the `day` integer value set to the day component of the abstract value.

28 *bis*.2.10 The `ANY-YEAR-DAY-ENCODING` type is:

```
ANY-YEAR-DAY-ENCODING ::= SEQUENCE {
     year            ANY-YEAR-ENCODING,
     day             INTEGER (1..366)}
```

with the `ANY-YEAR-ENCODING` set according to 28 *bis*.2.4 and the `day` integer value set to the day component of the abstract value.

28 *bis*.2.11 The `YEAR-WEEK-ENCODING` type is:

```
YEAR-WEEK-ENCODING ::= SEQUENCE {
     year            YEAR-ENCODING,
     week            INTEGER (1..53) -- 6 bits --}
```

with the `YEAR-ENCODING` set according to 28 *bis*.2.3 and the `week` integer value set to the week component of the abstract value.

NOTE – This has been optimized to provide a 12-bit or a 16-bit encoding in common cases.

28 *bis*.2.12 The `ANY-YEAR-WEEK-ENCODING` type is:

```
ANY-YEAR-WEEK-ENCODING ::= SEQUENCE {
     year            ANY-YEAR-ENCODING,
     week            INTEGER (1..53)}
```

with the `ANY-YEAR-ENCODING` set according to 28 *bis*.2.4 and the `week` integer value set to the week component of the abstract value.

28 *bis*.2.13 The `YEAR-WEEK-DAY-ENCODING` type is:

```
YEAR-WEEK-DAY-ENCODING ::= SEQUENCE {
     year            YEAR-ENCODING,
     week            INTEGER (1..53), -- 6 bits
     day             INTEGER (1..7) -- 3 bits -- }
```

with the `YEAR-ENCODING` set according to 28 *bis*.2.3, the `week` integer value set to the week component of the abstract value and the `day` integer value set to the day component of the abstract value.

NOTE – This has been optimized to provide a 15-bit or a 19-bit encoding in common cases.

28 *bis*.2.14 The **ANY-YEAR-WEEK-DAY-ENCODING** type is:

```
ANY-YEAR-WEEK-DAY-ENCODING ::= SEQUENCE {
     year            ANY-YEAR-ENCODING,
     week            INTEGER (1..53),
     day             INTEGER (1..7)}
```

with the **ANY-YEAR-ENCODING** set according to 28 *bis*.2.4, the **week** integer value set to the week component of the abstract value and the **day** integer value set to the day component of the abstract value.

28 *bis*.3 Encoding subtypes with the "Basic=Time" property setting

This subclause defines the ASN.1 types referenced in Table 2, column 3 for types where all the abstract values of the type have the **Basic=Time** property setting.

28 *bis*.3.1 The **HOURS-ENCODING** type is:

```
HOURS-ENCODING ::= INTEGER(0..24) -- 5 bits
```

with the integer value set to the hours component of the abstract value.

NOTE – This has been optimized to provide a 5-bit encoding.

28 *bis*.3.2 The **HOURS-UTC-ENCODING** type is:

```
HOURS-UTC-ENCODING ::= INTEGER(0..24) -- 5 bits
```

with the integer value set to the hours component of the abstract value.

NOTE – This has been optimized to provide a 5-bit encoding.

28 *bis*.3.3 The **HOURS-AND-DIFF-ENCODING** type is:

```
HOURS-AND-DIFF-ENCODING ::= SEQUENCE {
     local-hours     INTEGER (0..24),
     time-difference TIME-DIFFERENCE }
```

where:

```
TIME-DIFFERENCE ::= SEQUENCE {
     hours           INTEGER (-15..16),
     minutes         INTEGER (1..59) OPTIONAL }
```

with the **local-hours** integer value set to the hours component of the local time of the abstract value and the **time-difference** set to the hours and the minutes of the time-difference component of the abstract value. If the minutes component of the time-difference is zero, the **TIME-DIFFERENCE minutes** shall be omitted.

28 *bis*.3.4 The **MINUTES-ENCODING** type is:

```
MINUTES-ENCODING ::= SEQUENCE {
     hours           INTEGER (0..24), -- 5 bits
     minutes         INTEGER (0..59) -- 5 bits -- }
```

with the **hours** integer value set to the hours component of the abstract value and the **minutes** integer value set to the minutes component.

NOTE – This has been optimized to provide a 10-bit encoding.

28 *bis*.3.5 The **MINUTES-UTC-ENCODING** type is:

```
MINUTES-UTC-ENCODING ::= SEQUENCE {
     hours           INTEGER (0..24), -- 5 bits
     minutes         INTEGER (0..59) -- 5 bits -- }
```

with the **hours** integer value set to the hours component of the abstract value and the **minutes** integer value set to the minutes component.

NOTE – This has been optimized to provide a 10-bit encoding.

28 *bis*.3.6 The `MINUTES-AND-DIFF-ENCODING` type is:

```
MINUTES-AND-DIFF-ENCODING ::= SEQUENCE {
     local-time      SEQUENCE {
           hours     INTEGER (0..24),
           minutes   INTEGER (0..59) },
     time-difference TIME-DIFFERENCE }
```

with the `local-time` set to the hours and minutes component of the local time of the abstract value and the `time-difference` set to the hours and minutes of the time-difference component of the abstract value as specified in 28 *bis*.3.3.

28 *bis*.3.7 The `TIME-OF-DAY-ENCODING` type is:

```
TIME-OF-DAY-ENCODING ::= SEQUENCE {
    hours               INTEGER (0..24), -- 5 bits
    minutes             INTEGER (0..59), -- 5 bits
    seconds             INTEGER (0..60) -- 5 bits -- }
```

with the `hours` integer value set to the hours component of the abstract value, the `minutes` integer value set to the minutes component, and the `seconds` integer value set to the seconds component.

NOTE – This has been optimized to provide a 15-bit encoding.

28 *bis*.3.8 The `TIME-OF-DAY-UTC-ENCODING` type is:

```
TIME-OF-DAY-UTC-ENCODING ::= SEQUENCE {
     hours              INTEGER (0..24), -- 5 bits
     minutes            INTEGER (0..59), -- 5 bits
     seconds            INTEGER (0..60) -- 5 bits -- }
```

with the `hours` integer value set to the hours component of the abstract value, the `minutes` integer value set to the minutes component, and the `seconds` integer value set to the seconds component.

NOTE – This has been optimized to provide a 15-bit encoding.

28 *bis*.3.9 The `TIME-OF-DAY-AND-DIFF-ENCODING` type is:

```
TIME-OF-DAY-AND-DIFF-ENCODING ::= SEQUENCE {
     local-time      SEQUENCE {
           hours     INTEGER (0..24),
           minutes   INTEGER (0..59),
           seconds   INTEGER (0..60) },
     time-difference TIME-DIFFERENCE }
```

with the `local-time` set to the hours, minutes and seconds components of the local time of the abstract value and the `time-difference` set to the hours and minutes of the time-difference component of the abstract value as specified in 28 *bis*.3.3.

28 *bis*.3.10 The `HOURS-AND-FRACTION-ENCODING` type is:

```
HOURS-AND-FRACTION-ENCODING ::= SEQUENCE {
     hours              INTEGER (0..24), -- 5 bits
     fraction           INTEGER (0..999, ..., 1000..MAX)
                        -- 11 bits for up to three digits accuracy -- }
```

with the `hours` integer value set to the hours component of the abstract value and the `fraction` integer value set to the fractional hours multiplied by ten-to-the-power-N, where N is the specified number of digits in the fractional part.

NOTE – This has been optimized to provide a 16-bit encoding for up to 3-digit accuracy.

28 *bis*.3.11 The `HOURS-UTC-AND-FRACTION-ENCODING` type is:

```
HOURS-UTC-AND-FRACTION-ENCODING ::= SEQUENCE {
     hours              INTEGER (0..24), -- 5 bits
     fraction           INTEGER (0..999, ..., 1000..MAX)
                        -- 11 bits for up to three digits accuracy -- }
```

with the `hours` integer value set to the hours component of the abstract value and the `fraction` integer value set to the fractional hours multiplied by ten-to-the-power-N, where N is the specified number of digits in the fractional part.

NOTE – This has been optimized to provide a 16-bit encoding for up to 3-digit accuracy.

28 *bis*.3.12 The `HOURS-AND-DIFF-AND-FRACTION-ENCODING` type is:

```
HOURS-AND-DIFF-AND-FRACTION-ENCODING ::= SEQUENCE {
     local-hours     INTEGER (0..24), -- 5 bits
     fraction        INTEGER (0..999, ..., 1000..MAX)
                     -- 11 bits for up to three digits accuracy -- ,
     time-difference TIME-DIFFERENCE }
```

with the `local-hours` integer value set to the hours component of the local time of the abstract value, the `fraction` integer value set to the fractional hours multiplied by ten-to-the-power-N (where N is the specified number of digits in the fractional part) and the `time-difference` set to the hours and the minutes of the time-difference component of the abstract value as specified in 28 *bis*.3.3.

28 *bis*.3.13 The `MINUTES-AND-FRACTION-ENCODING` type is:

```
MINUTES-AND-FRACTION-ENCODING ::= SEQUENCE {
     hours           INTEGER (0..24), -- 5 bits
     minutes         INTEGER (0..59), -- 5 bits
     fraction        INTEGER (0..999, ..., 1000..MAX)
                     -- 11 bits for up to three digits accuracy -- }
```

with the `hours` integer value set to the hours component of the abstract value, the `minutes` integer value set to the minutes component and the `fraction` integer value set to the fractional hours multiplied by ten-to-the-power-N, where N is the specified number of digits in the fractional part.

NOTE – This has been optimized to provide a 21-bit encoding for up to 3-digit accuracy.

28 *bis*.3.14 The `MINUTES-UTC-AND-FRACTION-ENCODING` type is:

```
MINUTES-UTC-AND-FRACTION-ENCODING ::= SEQUENCE {
     hours           INTEGER (0..24), -- 5 bits
     minutes         INTEGER (0..59), -- 5 bits
     fraction        INTEGER (0..999, ..., 1000..MAX)
                     -- 11 bits for up to three digits accuracy -- }
```

with the `hours` integer value set to the hours component of the abstract value, the `minutes` integer value set to the minutes component and the `fraction` integer value set to the fractional hours multiplied by ten-to-the-power-N (where N is the specified number of digits in the fractional part).

NOTE – This has been optimized to provide a 21-bit encoding for up to 3-digit accuracy.

28 *bis*.3.15 The `MINUTES-AND-DIFF-AND-FRACTION-ENCODING` type is:

```
MINUTES-AND-DIFF-AND-FRACTION-ENCODING ::= SEQUENCE {
     local-time      SEQUENCE {
           hours     INTEGER (0..24),
           minutes   INTEGER (0..59),
           fraction  INTEGER (0..999, ..., 1000..MAX)},
     time-difference TIME-DIFFERENCE }
```

with the `local-time` set to the hours and minutes component of the local time of the abstract value, the `fraction` integer value set to the fractional minutes multiplied by ten-to-the-power-N (where N is the specified number of digits in the fractional part) and the `time-difference` set to the hours and minutes of the time-difference component of the abstract value as specified in 28 *bis*.3.3.

28 *bis*.3.16 The `TIME-OF-DAY-AND-FRACTION-ENCODING` type is:

```
TIME-OF-DAY-AND-FRACTION-ENCODING ::= SEQUENCE {
     hours           INTEGER (0..24), -- 5 bits
     minutes         INTEGER (0..59), -- 5 bits
     seconds         INTEGER (0..60), -- 5 bits --
     fraction        INTEGER (0..999, ..., 1000..MAX)
                     -- 11 bits for up to three digits accuracy -- }
```

with the `hours` integer value set to the hours component of the abstract value, the `minutes` integer value set to the minutes component, the `seconds` integer value set to the seconds component and `fraction` integer value set to the fractional seconds multiplied by ten-to-the-power-N, where N is the specified number of digits in the fractional part.

NOTE – This has been optimized to provide a 26-bit encoding.

28 *bis*.3.17 The `TIME-OF-DAY-UTC-AND-FRACTION-ENCODING` type is:

```
TIME-OF-DAY-UTC-AND-FRACTION-ENCODING ::= SEQUENCE {
     hours            INTEGER (0..24), -- 5 bits
     minutes          INTEGER (0..59), -- 5 bits
     seconds          INTEGER (0..60), -- 5 bits --
     fraction         INTEGER (0..999, ..., 1000..MAX)
                      -- 11 bits for up to three digits accuracy -- }
```

with the `hours` integer value set to the hours component of the abstract value, the `minutes` integer value set to the minutes component, the `seconds` integer value set to the seconds component and `fraction` integer value set to the fractional seconds multiplied by ten-to-the-power-N, where N is the specified number of digits in the fractional part.

NOTE – This has been optimized to provide a 26-bit encoding.

28 *bis*.3.18 The `TIME-OF-DAY-AND-DIFF-AND-FRACTION-ENCODING` type is:

```
TIME-OF-DAY-AND-DIFF-AND-FRACTION-ENCODING ::= SEQUENCE {
     local-time       SEQUENCE {
           hours      INTEGER (0..24),
           minutes    INTEGER (0..59),
           seconds    INTEGER (0..60),
           fraction   INTEGER (0..999, ..., 1000..MAX)},
     time-difference  TIME-DIFFERENCE }
```

with the `local-time` set to the hours, minutes, seconds and fractional part components of the local time of the abstract value and the `time-difference` set to the hours and minutes of the time-difference component of the abstract value as specified in 28 *bis*.3.3.

28 *bis*.4 Encoding subtypes with the `"Basic=Date-Time"` property setting

This subclause defines the ASN.1 type referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Date-Time"` property setting.

28 *bis*.4.1 The `DATE-TIME-ENCODING` type is:

```
DATE-TIME-ENCODING {Date-Type, Time-Type} ::= SEQUENCE {
     date             Date-Type,
     time             Time-Type}
```

28 *bis*.4.2 The encoding shall be the encoding of an instantiation of this type with the `Date-Type` and `Time-Type` actual parameters set to the types specified in Table 2 column 3 of the `"Basic=Date"` and `"Basic=Time"` rows (respectively) that specify the additional property settings of all the abstract values of the type.

NOTE – This has been optimized to provide a 32-bit encoding in common cases.

28 *bis*.5 Encoding subtypes with the `"Basic=Interval Interval-type=SE"` property setting

This subclause defines the ASN.1 types referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Interval Interval-type=SE"` property setting.

28 *bis*.5.1 The `START-END-DATE-INTERVAL-ENCODING` type is:

```
START-END-DATE-INTERVAL-ENCODING {Date-Type} ::= SEQUENCE {
     start            Date-Type,
     end              Date-Type}
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Date"` row that specifies the additional property settings of all the abstract values of the type. The `start` component shall be set to the start date and the `end` component shall be set to the end date of the interval.

28 *bis*.5.2 The `START-END-TIME-INTERVAL-ENCODING` type is:

```
START-END-TIME-INTERVAL-ENCODING {Time-Type} ::= SEQUENCE {
     start            Time-Type,
     end              Time-Type}
```

and the encoding shall be the encoding of an instantiation of this type with the `Time-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Time"` row that specifies the additional property settings of all the abstract values of the type. The `start` component shall be set to the start time and the `end` component shall be set to the end time of the interval.

28 *bis*.5.3 The **START-END-DATE-TIME-INTERVAL-ENCODING** type is:

```
START-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type} ::=
     SEQUENCE {
           start      DATE-TIME-ENCODING {Date-Type, Time-Type},
           end        DATE-TIME-ENCODING {Date-Type, Time-Type}}
```

and the encoding shall be the encoding of an instantiation of this type with the **Date-Type** and **Time-Type** actual parameters set to the types specified in Table 2 column 3 of the **"Basic=Date"** and **"Basic=Time"** rows (respectively) that specify the additional property settings of all the abstract values of the type. The **start** component shall be set (as specified in 28 *bis*.4) to the start date-time and the **end** component shall be set to the end date-time of the interval.

28 *bis*.6 Encoding subtypes with the "Basic=Interval Interval-type=D" property setting

This subclause defines the ASN.1 type referenced in Table 2, column 3 for types where all the abstract values of the type have the **"Basic=Interval Interval-type=D"** property setting.

28 *bis*.6.1 The **DURATION-INTERVAL-ENCODING** type is:

```
DURATION-INTERVAL-ENCODING ::= SEQUENCE { -- 8 bits for optionality
     years            INTEGER (0..31, ..., 32..MAX) OPTIONAL,
                          -- 5 bits for up to 31 years
     months           INTEGER (0..15, ..., 16..MAX) OPTIONAL,
                          -- 4 bits for up to 15 months
     weeks            INTEGER (0..63, ..., 64..MAX) OPTIONAL,
                          -- 6 bits for up to 63 weeks
     days             INTEGER (0..31, ..., 32..MAX) OPTIONAL,
                          -- 5 bits for up to 31 days
     hours            INTEGER (0..31, ..., 32..MAX) OPTIONAL,
                           -- 5 bits for up to 31 hours
     minutes          INTEGER (0..63, ..., 64..MAX) OPTIONAL,
                           -- 6 bits for up to 63 minutes
     seconds          INTEGER (0..63, ..., 64..MAX) OPTIONAL,
                          -- 6 bits for up to 63 seconds
     fractional-part      SEQUENCE {
                      number-of-digits      INTEGER(1..3, ..., 4..MAX),
                          -- 3 bits for up to three digits accuracy
                      fractional-value      INTEGER(1..999, ..., 1000..MAX)
                          -- 11 bits for up to three digits accuracy
                                          } OPTIONAL }
```

28 *bis*.6.2 The **weeks** component shall be present if, and only if, the **years**, **months**, **days**, **hours**, **minutes**, and **seconds** components are all absent.

NOTE – This reflects restrictions that are present for the use of time elements in the definition of the **DURATION** abstract value.

28 *bis*.6.3 (Canonicalization) If a time element component of the abstract value is zero, and does not have a fractional part, then the corresponding component of **DURATION-INTERVAL-ENCODING** shall be absent unless this time element is the least significant time element in the abstract value. If a time element of the abstract value has the value zero, and is the least significant time element in the abstract value, or has a fractional part, then the corresponding component shall be present in **DURATION-INTERVAL-ENCODING** with the value zero.

28 *bis*.6.4 The **fractional-part** of **DURATION-INTERVAL-ENCODING** shall be absent if there is no fractional part of any time element, otherwise it shall be set to the fractional part (of the least significant time element) as specified in 28 *bis*.6.5.

28 *bis*.6.5 The number of digits in the fractional part shall be placed in **number-of-digits**. If the number of digits is N, then the value of the fractional part shall be multiplied by ten-to-the-power-N and the resulting integer value placed in **fractional-value**.

NOTE 1 – Decoders can recover the original fractional part from these encodings, including any trailing zeros.

NOTE 2 – This encoding has been optimized for the cases where there are only a few non-zero time elements in the abstract value, and where the values of the time elements are small. Encodings of less than 16 bits occur in simple cases.

28 *bis*.7 Encoding subtypes with the `"Basic=Interval Interval-type=SD"` or `"Basic=Interval Interval-type=DE"` property setting

This subclause defines the ASN.1 types referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Interval Interval-type=SD"` or `"Basic=Interval Interval-type=DE"` property setting.

28 *bis*.7.1 The `START-DATE-DURATION-INTERVAL-ENCODING` type is:

```
START-DATE-DURATION-INTERVAL-ENCODING {Date-Type} ::= SEQUENCE {
     start            Date-Type,
     duration         DURATION-INTERVAL-ENCODING}
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Date"` row that specifies the additional property settings of all the abstract values of the type. The `start` component shall be set to the start date and the `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval.

28 *bis*.7.2 The `START-TIME-DURATION-INTERVAL-ENCODING` type is:

```
START-TIME-DURATION-INTERVAL-ENCODING {Time-Type} ::= SEQUENCE {
     start            Time-Type,
     duration         DURATION-INTERVAL-ENCODING }
```

and the encoding shall be the encoding of an instantiation of this type with the `Time-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Time"` row that specifies the additional property settings of all the abstract values of the type. The `start` component shall be set to the start time and the `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval.

28 *bis*.7.3 The `START-DATE-TIME-DURATION-INTERVAL-ENCODING` type is:

```
START-DATE-TIME-DURATION-INTERVAL-ENCODING {Date-Type, Time-Type} ::=
     SEQUENCE {
           start      DATE-TIME-ENCODING {Date-Type, Time-Type},
           duration   DURATION-INTERVAL-ENCODING }
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` and `Time-Type` actual parameters set to the types specified in Table 2 column 3 of the `"Basic=Date"` and `"Basic=Time"` rows (respectively) that specify the additional property settings of all the abstract values of the type. The `start` component shall be set (as specified in 28 *bis*.4) to the start date-time and the `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval.

28 *bis*.7.4 The `DURATION-END-DATE-INTERVAL-ENCODING` type is:

```
DURATION-END-DATE-INTERVAL-ENCODING {Date-Type} ::= SEQUENCE {
     duration         DURATION-INTERVAL-ENCODING,
     end              Date-Type }
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Date"` row that specifies the additional property settings of all the abstract values of the type. The `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval and the `end` component shall be set to the end date.

28 *bis*.7.5 The `DURATION-END-TIME-INTERVAL-ENCODING` type is:

```
DURATION-END-TIME-INTERVAL-ENCODING {Time-Type} ::= SEQUENCE {
     duration         DURATION-INTERVAL-ENCODING,
     end              Time-Type }
```

and the encoding shall be the encoding of an instantiation of this type with the `Time-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Time"` row that specifies the additional property settings of all the abstract values of the type. The `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval and the `end` component shall be set to the end time.

28 *bis*.7.6 The `DURATION-END-DATE-TIME-INTERVAL-ENCODING` type is:

```
DURATION-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type} ::= SEQUENCE {
     duration         DURATION-INTERVAL-ENCODING,
     end              DATE-TIME-ENCODING {Date-Type, Time-Type}}
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` and `Time-Type` actual parameters set to the types specified in Table 2 column 3 of the `"Basic=Date"` and `"Basic=Time"` rows (respectively) that specify the additional property settings of all the abstract values of the type. The `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval and the `end` component shall be set (as specified in 28 *bis*.4) to the end date-time.

28 *bis*.8 Encoding subtypes with the `"Basic=Rec-Interval Interval-type=SE"` property setting

This subclause defines the ASN.1 types referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Rec-Interval Interval-type=SE"` property setting.

28 *bis*.8.1 The `REC-START-END-DATE-INTERVAL-ENCODING` type is:

```
REC-START-END-DATE-INTERVAL-ENCODING {Date-Type} ::= SEQUENCE {
  recurrence          INTEGER OPTIONAL,
  start               Date-Type,
  end                 Date-Type}
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Date"` row that specifies the additional property settings of all the abstract values of the type. The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `start` component shall be set to the start date and the `end` component shall be set to the end date of the interval.

28 *bis*.8.2 The `REC-START-END-TIME-INTERVAL-ENCODING` type is:

```
REC-START-END-TIME-INTERVAL-ENCODING {Time-Type} ::= SEQUENCE {
  recurrence          INTEGER OPTIONAL,
  start               Time-Type,
  end                 Time-Type}
```

and the encoding shall be the encoding of an instantiation of this type with the `Time-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Time"` row that specifies the additional property settings of all the abstract values of the type. The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `start` component shall be set to the start time and the `end` component shall be set to the end time of the interval.

28 *bis*.8.3 The `REC-START-END-DATE-TIME-INTERVAL-ENCODING` type is:

```
REC-START-END-DATE-TIME-INTERVAL-ENCODING {Date-Type, Time-Type} ::=
SEQUENCE {
  recurrence          INTEGER OPTIONAL,
  start               DATE-TIME-ENCODING {Date-Type, Time-Type},
  end                 DATE-TIME-ENCODING {Date-Type, Time-Type}}
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` and `Time-Type` actual parameters set to the types specified in Table 2 column 3 of the `"Basic=Date"` and `"Basic=Time"` rows (respectively) that specify the additional property settings of all the abstract values of the type. The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `start` component shall be set (as specified in 28 *bis*.4) to the start date-time and the `end` component shall be set to the end date-time of the recurring interval.

28 *bis*.9 Encoding subtypes with the `"Basic=Rec-Interval Interval-type=D"` property setting

This subclause defines the ASN.1 type referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Rec-Interval Interval-type=D"` property setting.

28 *bis*.9.1 The `REC-DURATION-INTERVAL-ENCODING` type is:

```
REC-DURATION-INTERVAL-ENCODING ::= SEQUENCE {
     recurrence       INTEGER OPTIONAL,
     duration         DURATION-INTERVAL-ENCODING}
```

28 *bis*.9.2 The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the recurring interval.

28 *bis*.10 Encoding subtypes with the `"Basic=Rec-Interval Interval-type=SD"` or `"Basic=Rec-Interval Interval-type=DE"` property setting

This subclause defines the ASN.1 types referenced in Table 2, column 3 for types where all the abstract values of the type have the `"Basic=Rec-Interval Interval-type=SD"` or `"Basic=Rec-Interval Interval-type=DE"` property setting.

28 *bis*.10.1 The `REC-START-DATE-DURATION-INTERVAL-ENCODING` type is:

```
REC-START-DATE-DURATION-INTERVAL-ENCODING {Date-Type} ::= SEQUENCE {
     recurrence       INTEGER OPTIONAL,
     start            Date-Type,
     duration         DURATION-INTERVAL-ENCODING}
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Date"` row that specifies the additional property settings of all the abstract values of the type. The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `start` component shall be set to the start date and the `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval.

28 *bis*.10.2 The `REC-START-TIME-DURATION-INTERVAL-ENCODING` type is:

```
REC-START-TIME-DURATION-INTERVAL-ENCODING {Time-Type} ::= SEQUENCE {
     recurrence       INTEGER OPTIONAL,
     start            Time-Type,
     duration         DURATION-INTERVAL-ENCODING }
```

and the encoding shall be the encoding of an instantiation of this type with the `Time-Type` actual parameter set to the type specified in Table 2 column 3 of the `"Basic=Time"` row that specifies the additional property settings of all the abstract values of the type. The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `start` component shall be set to the start time and the `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the interval.

28 *bis*.10.3 The `REC-START-DATE-TIME-DURATION-INTERVAL-ENCODING` type is:

```
REC-START-DATE-TIME-DURATION-INTERVAL-ENCODING {Date-Type, Time-Type} ::=
SEQUENCE {
     recurrence       INTEGER OPTIONAL,
     start            DATE-TIME-ENCODING {Date-Type, Time-Type},
     duration         DURATION-INTERVAL-ENCODING }
```

and the encoding shall be the encoding of an instantiation of this type with the `Date-Type` and `Time-Type` actual parameters set to the types specified in Table 2 column 3 of the `"Basic=Date"` and `"Basic=Time"` rows (respectively) that specify the additional property settings of all the abstract values of the type. The `recurrence` component shall be absent for an unlimited number of recurrences in the abstract value, and shall otherwise be set to the number of recurrences. The `start` component shall be set (as specified in 28 *bis*.4) to the start date-time and the `duration` component shall be set (as specified in 28 *bis*.6) to the duration of the recurring interval.