国際電気通信連合

ITU-T X.680

(2021/02)

電気通信
標準化部門
ITUの

シリーズ X: データ ネットワーク、オープン システム
通信とセキュリティ

OSI ネットワーキングとシステムの側面 – Abstract Syntax Notation One (ASN.1)

情報技術 – 抽象構文表記 1 (ASN.1): 基本的な表記法の仕様

推奨 ITU-T X.680

ITU-T Xシリーズの推奨事項

データネットワーク、オープンシステム通信、セキュリティ

パブリックデータネットワーク	
サービスと設備	X.1 ～ X.19
インターフェース	X.20 ～ X.49
伝送、信号伝達、スイッチング	X.50 ～ X.89
ネットワークの側面	X.90 ～ X.149
メンテナンス	X.150 ～ X.179
行政上の取り決め	X.180 ～ X.199
オープンシステム相互接続	
型式と表記	X.200～X.209
サービス定義	X.210～X.219
接続モードプロトコル仕様	X.220–X.229
コネクションレスモードのプロトコル仕様	X.230–X.239
PICS プロフォーマ	X.240–X.259
プロトコルの識別	X.260–X.269
セキュリティプロトコル	X.270–X.279
レイヤー管理オブジェクト	X.280–X.289
適合性テスト	X.290–X.299
ネットワーク間の相互作用	
一般的な	X.300–X.349
衛星データ伝送システム	X.350–X.369
IPベースのネットワーク	X.370–X.379
メッセージ処理システム	X.400 ～ X.499
ディレクトリ	X.500～X.599
OSI ネットワークとシステムの側面	
ネットワーキン	X.600 ～ X.629
グ効率 サービ	X.630–X.639
ス品質 命名、アドレス	X.640–X.649
指定、および登録抽象構文表記 1 (ASN.1)	X.650 ～ X.679
	X.680 ～ X.699
OSI管理	
システム管理のフレームワークとアーキテクチャ	X.700～X.709
管理通信サービスとプロトコル	X.710 ～ X.719
管理情報の構成	X.720–X.729
管理機能とODMA機能	X.730 ～ X.799
安全	X.800 ～ X.849
OSI アプリケーション	
コミットメント、同時実行性、およびリカバリ トランザ	X.850–X.859
クション処理 リモート操作	X.860 ～ X.879
ASN.1 の汎用アプリケー	X.880 ～ X.889
ション OPEN DISTRIBUTED	X.890 ～ X.899
PROCESSING	X.900 ～ X.999
情報とネットワークのセキュリティ	X.1000 ～ X.1099
安全なアプリケーションとサービス (1)	X.1100 ～ X.1199
サイバースペースセキュリティ	X.1200 ～ X.1299
安全なアプリケーションとサービス (2)	X.1300–X.1499
サイバーセキュリティ情報交換	X.1500–X.1599
クラウドコンピューティングのセキュリティ	X.1600 ～ X.1699
量子通信	X.1700–X.1729
データセキュリティ	X.1750–X.1799
5Gセキュリティ	X.1800–X.1819

詳細については、ITU-T 勧告のリストを参照してください。

国際規格 ISO/IEC 8824-1
推奨 ITU-T X.680

情報技術 – 抽象構文表記 1 (ASN.1): 基本表記の仕様

まとめ

推奨 ITU-T X.680 | ISO/IEC 8824-1 は、情報データの構文を定義するための Abstract Syntax Notation One (ASN.1) と呼ばれる表記法を提供します。これは、多数の単純なデータ型を定義し、これらの型を参照し、これらの型の値を指定するための表記法を指定します。

ASN.1 表記は、情報の抽象構文を定義する必要があるときはいつでも、送信のために情報をエンコードする方法をまったく制約することなく適用できます。

歴史

推奨エディション		承認研究会		一意のID*
1.0	ITU-T X.680	1994-07-01	7	11.1002/1000/3040
1.1	ITU-T X.680 (1994) Amd. 1 ITU-T	1995-04-10	7	11.1002/1000/3041
1.2	X.680 (1994) Technical Cor. 1 ITU-T X.680	1995-11-21	7	11.1002/1000/3282
1.3	(1994) Technical Cor. 2 ITU-T X.680 (1994)	1997-12-12	7	11.1002/1000/4180
1.4	Amd. 1/Technical Cor.1 1997-12-12 ITU-T X.680 (1994) Amd. 2		7	11.1002/1000/4179
1.5		1997-12-12	7	11.1002/1000/4181
2.0	ITU-T X.680	1997-12-12	7	11.1002/1000/4449
2.1	ITU-T X.680 (1997) Technical Cor. 1 ITU-T	1999-06-18	7	11.1002/1000/4700
2.2	X.680 (1997) Amd. 1 ITU-T X.680	1999-06-18	7	11.1002/1000/4698
2.3	(1997) Amd. 2 ITU-T X.680 (1997)	1999-06-18	7	11.1002/1000/4699
2.4	Technical Cor. 2 ITU-T X.680 (1997) Technical	2000-03-31	7	11.1002/1000/5046
2.5	Cor. 3 ITU-T X.680 (1997) Technical Cor. 4	2001-02-02	7	11.1002/1000/5331
2.6	ITU-T X.680 (1997) Amd. 3 ITU-T X.680 (1997)	2001-03-15	7	11.1002/1000/5332
2.7	Amd. 4	2001-10-29	7	11.1002/1000/5562
2.8		2001-10-29	7	11.1002/1000/5563
3.0	ITU-T X.680	2002-07-14	17	11.1002/1000/6085
3.1	ITU-T X.680 (2002) Amd. 1 ITU-T	2003-10-29	17	11.1002/1000/7019
3.2	X.680 (2002) Amd. 2 ITU-T X.680	2004-08-29	17	11.1002/1000/7291
3.3	(2002) Technical Cor. 1 ITU-T X.680 (2002)	2005-05-14	17	11.1002/1000/8512
3.4	Amd. 3 ITU-T X.680 (2002) Amd. 4	2006-06-13	17	11.1002/1000/8836
3.5		2007-05-29	17	11.1002/1000/9105
4.0	ITU-T X.680	2008-11-13	17	11.1002/1000/9604
4.1	ITU-T X.680 (2008) Cor. 1 ITU-T	2011-10-14	17	11.1002/1000/11376
4.2	X.680 (2008) Cor. 2	2014-03-01	17	11.1002/1000/12144
5.0	ITU-T X.680	2015-08-13	17	11.1002/1000/12479
5.1	ITU-T X.680 (2015) Cor. 1 ITU-T	2017-05-14	17	11.1002/1000/13257
5.2	X.680 (2015) Cor. 2 ITU-T X.680	2017-10-14	17	11.1002/1000/13361
5.3	(2015) Cor. 3 ITU-T X.680 (2015)	2018-05-14	17	11.1002/1000/13598
5.4	Amd. 1	2018-05-14	17	11.1002/1000/13597
6.0	ITU-T X.680	2021-02-13	17	11.1002/1000/14468

* 推奨事項にアクセスするには、Web ブラウザのアドレス フィールドに URL http://handle.itu.int/ を入力し、続いて推奨事項の一意の ID。たとえば、 http://handle.itu.int/11.1002/1000/11830-en です。

序文

国際電気通信連合 (ITU) は、電気通信、情報通信技術 (ICT) の分野における国連の専門機関です。 ITU 電気通信標準化部門 (ITU-T) は、ITU の常設機関です。 ITU-T は、世界規模で電気通信を標準化することを目的として、技術的、運用上、および料金に関する問題を研究し、それらに関する勧告を発行する責任を負っています。

4 年ごとに開催される世界電気通信標準化総会 (WTSA) は、ITU-T 研究グループによる研究テーマを確立し、ITU-T 研究グループはこれらのテーマに関する勧告を作成します。

ITU-T 勧告の承認は、WTSA 決議 1 に定められた手順の対象となります。

ITU-T の範囲内にある情報技術の一部の分野では、必要な規格が ISO および IEC との協力のもとに作成されています。

注記

この勧告では、簡潔にするために「管理」という表現を使用して、電気通信管理と承認された運営機関の両方を示します。

この推奨事項への準拠は任意です。ただし、推奨には特定の必須規定が含まれる場合があり（相互運用性や適用性などを確保するため）、これらの必須規定がすべて満たされた場合に推奨への準拠が達成されます。 「しなければならない」という言葉、または「しなければならない」などの他の義務的な言葉や否定的な同等の言葉は、要件を表現するために使用されます。このような言葉の使用は、いかなる当事者にも勧告の遵守が要求されることを示唆するものではありません。

知的財産権

ITU は、この勧告の実践または実施には、主張されている知的財産権の使用が含まれる可能性があることに注意を喚起します。 ITU は、ITU 加盟国または勧告策定プロセス以外の者が主張するかどうかにかかわらず、主張された知的財産権の証拠、有効性、または適用可能性に関していかなる立場も取りません。

この勧告の承認日の時点で、ITU は、この勧告の実施に必要となる可能性のある、特許/ソフトウェア著作権によって保護されている知的財産に関する通知を受け取っていません。
ただし、実装者は、これが最新の情報を表していない可能性があることに注意し、ITU-T Web サイトhttp://www.itu.int/ITU-T/ipr/から入手可能な適切な ITU-T データベースを参照することを強くお勧めします。 。

コンテンツ

ページ

導入 viii

1 範囲 1

2 規範的参照 1

2.1 同一の推奨事項 |国際規格 1

2.2 追加の参考資料 2

3 定義 2

3.1 国際オブジェクト識別子ツリーの仕様 2

3.2 情報オブジェクトの仕様 2

3.3 制約の仕様 3

3.4 ASN.1仕様のパラメータ化 3

3.5組織を識別するための構造 3

3.6 ユニバーサル複数オクテット符号化文字セット (UCS) 3

3.7 日付と時刻の表現 3

3.8 追加の定義 4

4 略語 9

5 表記 9

5.1 一般的な 9

5.2 プロダクション 10

5.3 代替コレクション 10

5.4 ノンスペーシングインジケータ 10

5.5 制作例 10

5.6 レイアウト 10

5.7 再帰 11

5.8 許可された語彙項目のシーケンスへの参照 11

5.9 語彙項目への参照 11

5.10 簡略記法 11

5.11 値の参照と値の型付け 12

6 タイプ拡張の ASN.1 モデル 12

7 エンコード ルールの拡張性要件 12

8 タグ 13

9 エンコード手順 14

10 ASN.1 表記の使用 15

11 ASN.1 文字セット 15

12 ASN.1 語彙項目 16

12.1 一般規則 16

12.2 型参照 17

12.3 識別子 17

12.4 値の参照 17

12.5 モジュールリファレンス 17

12.6 コメント 17

12.7 空の語彙項目 18

12.8 数値 18

12.9 実数 18

12.10 バイナリ文字列 18

12.11 XML バイナリ文字列項目 18

12.12 16 進文字列 18

12.13 XML 16 進文字列項目 19

12.14 文字列 19

12.15 XML文字列項目 20

12.16 単純な文字列字句項目 22

12.17 時刻値の文字列 22
12.18 XML時刻値文字列項目 22
12.19 プロパティ名と設定名の語彙項目 22
12.20 代入語彙項目 22
12.21 範囲区切り記号 22
12.22 省略記号 22
12.23 左バージョンの括弧 23
12.24 右バージョンの括弧 23
12.25 エンコーディングのリファレンス 23
12.26 整数値の Unicode ラベル 23
12.27 非整数 Unicode ラベル 23
12.28 XML終了タグ開始項目 23
12.29 XML 単一タグの終了項目 23
12.30 XML ブール値 true 項目 23
12.31 XML ブール拡張 true 項目 24
12.32 XML ブール値 false 項目 24
12.33 XML ブール型拡張 false 項目 24
12.34 XML 実数非数値項目 24
12.35 XML 実数無限項目 24
12.36 ASN.1 タイプの XML タグ名 25
12.37 単一文字の語彙項目 26
12.38 予約語 26
13 モジュール定義 27
14 型と値の定義の参照 31
15 ASN.1 コンポーネントへの参照をサポートするための表記 32
16 型と値の割り当て 33
17 型と値の定義 35
18 ブール型の表記法 38
19 整数型の表記法 38
20 列挙型の表記法 39
21 実数型の表記 41
22 ビット文字列型の表記法 42
23 オクテット文字列型の表記法 44
24 Null 型の表記法 44
25 シーケンスタイプの表記 45
26 sequence-of 型の表記法 48
27 セットタイプの表記 50
28 タイプのセットの表記法 51
29 選択肢タイプの表記 52
30 選択タイプの表記 54
31 接頭辞付きタイプの表記法 54
31.1 一般 54
31.2 タグ付きタイプ 55
31.3 エンコーディングのプレフィックス付きタイプ 55
32 オブジェクト識別子のタイプの表記法 56
33 相対オブジェクト識別子タイプの表記法 58
34 OID 国際化リソース識別子タイプの表記法 59
35 相対 OID 国際化リソース識別子タイプの表記 60
36 エンベデッド pdv タイプの表記 60
37 外部タイプの表記 62

38 時間の種類 63
38.1 一般 63
38.2 時間プロパティと時間抽象値の設定 63
38.3 指定されたプロパティ設定を持つ時間抽象値の基本的な値の表記法と XML 値の表記法 67
38.4 便利な時間の種類 71
39 文字列の種類 72
40 文字列型の表記 73
41 制限文字列型の定義 73
42 文字、コレクション、プロパティ カテゴリ セットの命名 77
43 正規の文字順序 81
44 無制限の文字列型の定義 82
45 第 46 項から第 48 項で定義されるタイプの表記 83
46 一般化された時間 83
47 世界時 84
48 オブジェクト記述子のタイプ 85
49 制約付きタイプ 86
50 要素セット仕様 87
51 サブタイプ要素 89
51.1 一般 89
51.2 単一値 90
51.3 含まれるサブタイプ 90
51.4 値の範囲 90
51.5 サイズの制約 91
51.6 型制約 91
51.7 許可されるアルファベット 91
51.8 内部サブタイプ 92
51.9 パターン制約 93
51.10 プロパティ設定 93
51.11 継続時間の範囲 94
51.12 時点の範囲 95
51.13 再発範囲 95
52 拡張マーカー 95
53 例外識別子 97
54 エンコーディング制御セクション 98
A.1 定義 99
A.2 メタキャラクター 99
B.1 一般 103
B.2 ASN.1 定義の時間タイプ モジュール 103
C.1 値マッピングの概念の必要性 (チュートリアルの導入) 108
C.2 値のマッピング 110
C.3 同一の型定義 111
C.4 値マッピングの仕様 113
C.5 文字列型に対して定義された追加の値マッピング 113
C.6 特定の型と値の互換性要件 114
C.7 例 115
C.7.2 例1 例2 例3 例 115
C.7.3 4 例5 例6 115
C.7.4 115
C.7.5 115
C.7.6 115
C.7.7 116

D.1 この推奨事項で割り当てられた値 |国際標準..... 117
D.2 ASN.1 およびエンコード ルール標準におけるオブジェクト識別子..... 117
F.1 一般..... 120
F.2 オブジェクト識別子 (OBJECT IDENTIFIER) タイプ別の国際オブジェクト識別子ツリーの使
用..... 120
F.3 OID 国際化リソース識別子 (OID-IRI) タイプによる国際オブジェクト識別子ツリーの使
用..... 120
G.1 人事記録の例..... 121
G.1.1 人事記録の非公式な説明..... 121
G.1.2 レコード構造の ASN.1 説明..... 121
G.1.3 レコード値の ASN.1 説明..... 122
G.2 表記の使用に関するガイドライン..... 122
G.2.1 ブール値..... 123
G.2.2 整数..... 123
G.2.3 列挙型..... 123
G.2.4 実数..... 124
G.2.5 ビット列..... 124
G.2.6 オクテット文字列..... 126
G.2.7 UniversalString、BMPString、および UTF8String..... 126
G.2.8 文字列..... 127
G.2.9 ヌル..... 128
G.2.10 シーケンスとシーケンス..... 128
G.2.11 セットとセット..... 130
G.2.12 タグ付き..... 132
G.2.13 選択..... 133
G.2.14 選択タイプ..... 135
G.2.16 埋め込み pdv..... 136
G.2.17 外部..... 136
G.2.18 インスタンス..... 136
G.2.19 オブジェクト識別子..... 137
G.2.20 OID 国際化リソース識別子..... 137
G.2.21 相対オブジェクト識別子..... 137
G.3 値の表記とプロパティの設定 (TIME 型と有用な時間型)..... 137
G.3.1 日付..... 137
G.3.2 時刻..... 138
G.3.3 日付と時刻..... 138
G.3.4 時間間隔..... 139
G.3.5 定期的な間隔..... 140
G.4 抽象構文の識別..... 140
G.5 サブタイプ..... 141
H.1 ASN.1 での文字列サポート..... 145
H.2 UniversalString、UTF8String、および BMPString 型..... 145
H.3 ISO/IEC 10646 適合要件について..... 146
H.4 ISO/IEC 10646 準拠に関する ASN.1 ユーザーへの推奨事項..... 146
H.5 抽象構文のパラメータとしてサブセットを採用..... 147
H.6 CHARACTER STRING 型..... 147
I.1 概要..... 148
I.2 バージョン番号の意味..... 149
I.3 エンコード規則の要件..... 150
I.4 (拡張可能な) 制約の組み合わせ..... 150
I.4.1 モデル..... 150
I.4.2 制約の連続適用..... 150
I.4.3 集合演算の使用..... 151
I.4.4 含まれるサブタイプ表記の使用..... 152
J.1 時刻と日付の ASN.1 タイプのコレクション..... 153
J.2 ISO 8601 の主要な概念..... 153
J.3 TIME 型の抽象値..... 154
J.4 時間抽象値の時間プロパティ..... 155
J.5 値の表記..... 155

J.6 ASN.1 サブタイプ表記の使用 156
J.7 プロパティ設定のサブタイプ表記 156
K.1 全般 158
K.2 完全な文字列の分析 158
K.3 音程を含む文字列の解析 159
K.4 日付を含む文字列の解析 159
K.5 年を含む文字列の分析 160
K.6 世紀を含む文字列の解析 160
K.7 時刻を含む文字列の解析 160
K.8 単純な時刻を含む文字列の解析 161

導入

この推奨事項 |国際標準は、データ型と値の定義の標準表記法を示しています。
データタイプ(または略してタイプ) は、情報のカテゴリ (数値、テキスト、静止画像、またはビデオ情報など) です。データ値(または略して値) は、そのような型のインスタンスです。この推奨事項 |国際標準では、いくつかの基本的な型とそれに対応する値、およびそれらを組み合わせてより複雑な型と値を作成するためのルールが定義されています。

一部のプロトコル アーキテクチャでは、各メッセージは一連のオクテットのバイナリ値として指定されます。ただし、標準作成者は、バイナリ表現を気にせずに、メッセージを伝送するための非常に複雑なデータ型を定義する必要があります。これらのデータ型を指定するには、各値の表現を必ずしも決定しない表記法が必要です。 ASN.1 はそのような表記です。この表記法は、アプリケーション セマンティクス (転送構文と呼ばれる) を運ぶオクテットの値を決定するエンコード ルールと呼ばれる 1 つ以上のアルゴリズムの仕様によって補足されます。記録ITU-T X.690 | ISO/IEC 8825-1、Rec. ITU-T X.691 | ISO/IEC 8825-2 および Rec. ITU-T X.693 | ISO/IEC 8825-4 では、 Basic Encoding Rules (BER)、 Packed Encoding Rules (PER)、およびXML Encoding Rules (XER) と呼ばれる、標準化されたエンコーディング ルールの 3 つのファミリーを指定しています。

一部のユーザーは、ASN.1 を使用してレガシー プロトコルを再定義したいと考えていますが、既存のバイナリ表現を保持する必要があるため、標準化されたエンコード ルールを使用できません。他のユーザーは、ワイヤ上のビットの正確なレイアウト (転送構文) をより完全に制御したいと考えています。これらの要件は Rec によって対処されます。 ITU-T X.692 | ASN.1 のEncoding Control Notation (ECN)を指定する ISO/IEC 8825-3 。 ECN を使用すると、設計者は ASN.1 を使用してプロトコルの抽象構文を正式に指定できますが、その後 (必要に応じて) 付属の ECN 仕様 (標準化されたエンコーディング ルールを参照する場合がある) を記述することで、回線上のビットを完全または部分的に制御できます。エンコードの一部)。

抽象レベルで複雑な型を定義する非常に一般的な手法は、単純な型のすべての可能な値を定義し、これらの単純な型をさまざまな方法で組み合わせることで、少数の単純な型を定義することです。新しい型を定義する方法のいくつかは次のとおりです。

a) 既存の型の (順序付けられた) リストが与えられると、既存の型のそれぞれから 1 つずつ、値の (順序付けられた) シーケンスとして値を形成できます。この方法で取得されたすべての可能な値のコレクションは新しい型です (リスト内の既存の型がすべて異なる場合、このメカニズムを拡張して、リストから一部の値を省略できるようにすることができます)。

b) (個別の) 既存の型の順序付けされていないセットが与えられた場合、値は、既存の型のそれぞれから 1 つずつ、(順序付けされていない) 値のセットとして形成できます。この方法で取得されたすべての可能な順序付けされていない値のセットのコレクションは新しいタイプです (このメカニズムは、一部の値の省略を許可するように再度拡張できます)。

c) 単一の既存の型が与えられた場合、値は、既存の型のゼロ、1 つ以上の値の (順序付き) リストまたは (順序なし) セットとして形成できます。この方法で取得されたすべての可能なリストまたは値のセットのコレクションは新しいタイプです。

d) (個別の) タイプのリストが与えられると、それらのいずれかから値を選択できます。可能なすべての値のセット
このようにして得られたのは新しいタイプです。

e) 型が与えられると、何らかの構造または順序関係を使用して、新しい型をそのサブセットとして形成できます。
価値観の中では。

この方法で型を結合する場合の重要な点は、エンコーディング ルールが結合構造を認識し、基本型の値のコレクションの明確なエンコーディングを提供する必要があることです。したがって、すべての基本型は、この推奨事項で指定されている表記法を使用して定義されます。国際標準には、値の明確なエンコードを支援するタグが割り当てられています。

タグは主に機械での使用を目的としており、この推奨事項で定義されている人間による表記には必須ではありません。国際標準。ただし、特定の型が異なることを要求する必要がある場合、これは、それらが異なるタグを持つことを要求することによって表現されます。したがって、タグの割り当てはこの表記法を使用する際の重要な部分ですが、(1994 年以降) タグの自動割り当てを指定できるようになりました。

注 1 – この推奨事項の範囲内 |国際標準では、タグ値はすべての単純なタイプと構築メカニズムに割り当てられます。表記法の使用に課された制限により、値を明確に識別するために転送時にタグを使用できることが保証されます。

エンコード命令を型に割り当てて、その型のエンコードに影響を与えることもできます。これは、型定義の前に配置された型プレフィックスまたは型参照の使用によって、または ASN.1 モジュールの最後に配置されたエンコード制御セクションによって実行できます。型プレフィックスとエンコーディング制御セクションの一般的な構文は、この推奨事項で指定されています。国際標準であり、エンコード命令によって変更されるエンコード規則を識別するためのエンコード参照が含まれています。エンコード命令のセマンティクスと詳細な構文は、エンコード ルールで指定されています。エンコーディング参照によって識別される国際標準。

ASN.1 仕様は、最初は完全に定義された ASN.1 タイプのセットを使用して作成されます。ただし、後の段階で、これらの型を変更する必要が生じる場合があります (通常は、シーケンスまたはセット型に追加のコンポーネントを追加することによって)。古い型定義を使用する実装が、定義された方法で新しい型定義を使用する実装と相互作用できるような方法でこれを可能にする場合、エンコーディング ルールは適切なサポートを提供する必要があります。 ASN.1 表記は、さまざまなタイプでの拡張マーカーの組み込みをサポートしています。これは、この型が拡張シリーズと呼ばれる一連の関連型 (つまり、同じ初期型のバージョン) の 1 つであること、およびエンコード ルールが実装間の情報転送を可能にするために必要であるという設計者の意図をエンコード ルールに通知します。同じ拡張シリーズの一部として関連付けられている異なるタイプ。

条項 11 から 33 (両端を含む) は、ASN.1 でサポートされる単純型を定義し、単純型の参照と、それらを使用した新しい型の定義に使用される表記法を指定します。第 11 条から第 33 条では、ASN.1 を使用して定義された型の値を指定するために使用される表記法も指定します。 2 つの値の表記が提供されます。 1 つ目は基本 ASN.1 値表記法と呼ばれるもので、最初の導入以来 ASN.1 表記法の一部となっています。 2 つ目は XML ASN.1 値表記法と呼ばれ、Extensible Markup Language (XML) を使用した値表記法を提供します。

注 2 – XML 値表記は、XML を使用して ASN.1 値を表す手段を提供します。したがって、ASN.1 タイプ定義では、XML 要素の構造と内容も指定します。これにより、ASN.1 は XML 用の単純なスキーマ言語になります。

第 36 条から第 37 条 (両端を含む) は、ASN.1 タイプの完全なエンコーディングを内部に保持するために ASN.1 によってサポートされるタイプを定義します。

第 38 条と付録 B は、ISO 8601 のサポートを提供するタイプを定義します。

条項 39 から条項 44 まで (両端を含む) は、文字列タイプを定義します。

第 45 条から第 48 条 (両端を含む) は、一般的な有用性があると考えられる特定の型を定義しますが、追加のエンコード規則は必要ありません。

条項 49 から 51 (両端を含む) は、親タイプの値からサブタイプを定義できるようにする表記法を定義します。

第 52 条では、「バージョン 1」仕様で指定された ASN.1 タイプを「バージョン 2」で拡張される可能性が高いものとして識別し、後続のバージョンで行われた追加を個別にリストしてバージョン番号で識別できるようにする表記を定義しています。 。

第 53 条は、現在の標準化された定義で指定されている値の範囲外にある値のエンコードを受信した場合に、ASN.1 タイプ定義に意図したエラー処理の指示を含めることを許可する表記法を定義します。

付録 A は、この勧告の不可欠な部分を形成します。国際標準であり、ASN.1 正規表現を指定します。

付属書 B は、この勧告の不可欠な部分を形成します。 ISO 8601 の完全な機能を提供する一連の時間タイプの定義を含む ASN.1 モジュールを定義します。これらのタイプは、句で指定された有用な時間タイプであれば、アプリケーション設計者によってこの ASN.1 モジュールからインポートできます。 38 はこのアプリケーションには不十分です。

付録 C は、この勧告の不可欠な部分を形成します。国際標準であり、型と値の互換性に関するルールを指定します。

附属書 D は、この勧告の不可欠な部分を形成します。国際標準であり、ASN.1 シリーズの勧告 | で割り当てられたオブジェクト識別子とオブジェクト記述子の値を記録します。国際規格。

附属書 E は、この勧告の不可欠な部分を形成します。国際標準であり、現在定義されているエンコード参照と推奨事項を指定します。これらのエンコード参照を使用してエンコード命令のセマンティクスと詳細な構文を定義する国際標準。

附属書 F は、この勧告の不可欠な部分を形成するものではありません。国際標準であり、国際オブジェクト識別子ツリーのトップレベル アークの仕様と、IANA に「oid」スキームとして登録された IRI または URI として使用できる OID 国際化リソース識別子を形成するためのそのツリーの使用を参照します。 。

付属書 G は、この勧告の不可欠な部分を形成しません。国際標準であり、ASN.1 表記法の使用例とヒントを提供します。

附属書 H は、この勧告の不可欠な部分を形成しません。国際標準であり、ASN.1 文字列に関するチュートリアルを提供します。

付属書 I は、この勧告の不可欠な部分を形成しません。国際標準であり、タイプ拡張の ASN.1 モデルに関するチュートリアルを提供します。

附属書 J は、この勧告の不可欠な部分を形成しません。国際規格であり、ISO 8601 とTIMEタイプの入門チュートリアルを提供します。規範文の前にこれを読むことをお勧めします。

付属書 K は、この勧告の不可欠な部分を形成しません。国際標準であり、値表記のインスタンスから抽象値の時間プロパティを識別する方法に関する情報を提供します。

附属書 L は、この勧告の不可欠な部分を形成するものではありません。国際規格であり、第 5 項の表記を使用して ASN.1 の概要を提供します。

国際標準
ITU-T の勧告

情報技術 -
抽象構文表記 1 (ASN.1):
基本表記の仕様

1 範囲

この推奨事項 |国際標準では、データ型、値、およびデータ型の制約の定義に使用される抽象構文表記法 1 (ASN.1) と呼ばれる標準表記法が提供されています。

この推奨事項 |国際標準：

– 多くの単純な型とそのタグを定義し、これらの型を参照するための表記法を指定します。
これらのタイプの値を指定するため。

– より基本的な型から新しい型を構築するメカニズムを定義し、そのような型を定義してタグを割り当て、これらの
型の値を指定するための表記法を指定します。

– 使用する文字セットを（他の推奨事項や国際標準を参照して）定義します。
ASN.1内。

ASN.1 表記は、情報の抽象構文を定義する必要がある場合にはいつでも適用できます。

ASN.1 表記は、ASN.1 タイプのエンコード規則を定義する他の標準によって参照されます。

2 規範的参照

以下の勧告および国際規格には、本文中の参照を通じてこの勧告の規定を構成する規定が含まれています。国際標準。発行時点では、示されているエディションは有効です。すべての推奨事項と標準は改訂される可能性があり、当事者はこの推奨事項に基づいて契約を締結します。国際規格は、以下にリストされている推奨事項と規格の最新版を適用する可能性を調査することが推奨されます。 IEC および ISO のメンバーは、現在有効な国際規格の登録簿を維持しています。
ITU の電気通信標準化局は、現在有効な ITU-T 勧告のリストを管理しています。

注 – この推奨事項 |国際規格は ISO/IEC 10646:2003 および Unicode 標準バージョン 3.2.0:2002 に基づいています。これら 2 つの規格の新しいバージョンを使用して適用することはできません。

2.1 同一の推奨事項 |国際規格

– 勧告 ITU-T X.660 (2011) | ISO/IEC 9834-1:2012、情報技術 – オープン システム相互接続 – OSI 登録機関の運用手順: ASN.1 国際オブジェクト識別子ツリーの一般手順と上位アーク。

– 勧告 ITU-T X.681 (2021) | ISO/IEC 8824-2:2021、情報技術 – 要約
構文表記 1 (ASN.1): 情報オブジェクトの仕様。

– 勧告 ITU-T X.682 (2021) | ISO/IEC 8824-3:2021、情報技術 – 要約
構文表記 1 (ASN.1): 制約の仕様。

– 勧告 ITU-T X.683 (2021) | ISO/IEC 8824-4:2021、情報技術 – 要約
構文表記 1 (ASN.1): ASN.1 仕様のパラメータ化。

– 勧告 ITU-T X.690 (2021) | ISO/IEC 8825-1:2021、情報技術 – ASN.1 エンコーディング ルール: 基本エンコーディング ルール (BER)、標準エンコーディング ルール (CER)、および識別エンコーディング ルール (DER) の仕様。

– 勧告 ITU-T X.691 (2021) | ISO/IEC 8825-2:2021、情報技術 – ASN.1
エンコーディング ルール: Packed Encoding Rules (PER) の仕様。

– 勧告 ITU-T X.692 (2021) | ISO/IEC 8825-3:2021、情報技術 – ASN.1 エンコーディング ルール: エンコーディング制御表記法 (ECN) の仕様。

– 勧告 ITU-T X.693 (2021) | ISO/IEC 8825-4:2021、情報技術 – ASN.1 エンコーディング ルール: XML エンコーディング ルール (XER)。

– 勧告 ITU-T X.695 (2021) | ISO/IEC 8825-6:2021、情報技術 – ASN.1
エンコーディング ルール: PER エンコーディング命令の登録と適用。
注 – 上記の参照は、特定された推奨事項への参照として解釈されるものとします。国際規格とその発行されたすべての修正および技術的修正事項。

2.2 追加の参考文献

– 勧告 ITU-R TF.460-6 (2002)、標準周波数および時間信号の放射。

– CCITT 勧告 T.100 (1988)、インタラクティブ Videotex のための国際情報交換。

– 勧告 ITU-T T.101 (1994)、 Videotex サービスの国際インターワーキング。

–エスケープ シーケンスで使用されるコード化文字セットのISO国際登録。

– ISO/IEC 646:1991、情報技術 – 情報交換用の ISO 7 ビット コード化文字セット。

– ISO/IEC 2022:1994、情報技術 – 文字コードの構造と拡張技術。

– ISO/IEC 6523:1998、データ交換 – 組織と組織の識別のための構造
組織の部分。

– ISO/IEC 7350:1991、情報技術 – グラフィック文字のレパートリーの登録
ISO/IEC 10367。

– ISO 8601:2019、日付と時刻 – 情報交換の表現。

– ISO/IEC 10646:2003、情報技術 – ユニバーサル複数オクテット符号化文字セット (UCS)。

– Unicode 標準、バージョン 3.2.0:2002。 Unicode コンソーシアム。 (マサチューセッツ州レディング、アディソン•ウェスリー)
注 1 – 上記のリファレンスは、制御文字の名前を提供し、文字のカテゴリを指定するために含まれています。

– W3C XML 1.0:2008、拡張マークアップ言語 (XML) 1.0 (第 5 版)、W3C 勧告、著作権 ©2008 W3C、(MIT、ERCIM、慶応)、 http: //www.w3.org/TR/2008/REC -xml-20081126/。

注 – この推奨事項内の文書への参照 |国際規格は、単独の文書として、勧告または国際規格の地位を与えません。

3 定義

この勧告の目的のために |国際規格では、次の定義が適用されます。

3.1 国際オブジェクト識別子ツリーの仕様

この推奨事項 |国際規格では、Rec. で定義されている次の用語を使用します。 ITU-T X.660 | ISO/IEC 9834-1:

a) 整数値の Unicode ラベル。

b) 国際オブジェクト識別子ツリー。

c) OID 国際化リソース識別子。

d) 長い弧。

e) オブジェクト識別子。

f) 主な整数値。

g) 二次識別子。

h) Unicode ラベル。

3.2 情報オブジェクトの仕様

この推奨事項 |国際規格では、Rec. で定義されている次の用語を使用します。 ITU-T X.681 | ISO/IEC 8824-2:

a) 情報オブジェクト。

b) 情報オブジェクトクラス。

c) 情報オブジェクトセット。

d) 型のインスタンス。

e) オブジェクト クラスのフィールド タイプ。

3.3 制約仕様

この推奨事項 |国際規格では、Rec. で定義されている次の用語を使用します。 ITU-T X.682 | ISO/IEC 8824-3:

a) コンポーネント関係の制約。 b) テーブル制約。

3.4 ASN.1仕様のパラメータ化

この推奨事項 |国際規格では、Rec. で定義されている次の用語を使用します。 ITU-T X.683 | ISO/IEC 8824-4:

a) パラメータ化されたタイプ。 b) パラメータ化された値。

3.5 組織を識別するための構造

この推奨事項 |国際規格では、ISO/IEC 6523 で定義されている次の用語が使用されます。

a) 発行組織。 b) 組織コード。
c) 国際コード指定者。

3.6 ユニバーサル複数オクテットコード化文字セット (UCS)

この推奨事項 |国際規格では、ISO/IEC 10646 で定義されている次の用語が使用されます。

a) 基本多言語面 (BMP)。 b) セル。 c) 組み合わせ文字。 d) 図記号。 e) グループ。 f) 限定されたサブセット。 g) 平面。 h) 行。 i) 選択されたサブセット。

3.7 日付と時刻の表現

この推奨事項 |国際規格では、ISO 8601 で定義されている次の用語が使用されます。 a) 基本フォーマット。 b) 暦日。 c) 平年。 d) 期間。 e) 拡張フォーマット。 f) グレゴリオ暦。 g) インスタント。 h) うるう秒。 i) 閏年。 j) 現地時刻。 k) 序数の日付。 l) 繰り返しの時間間隔 m) 時間軸。 n) 時間間隔。 o) 時間。 p) 時間スケール。 q) UTC;

r) 週の日付。

3.8 追加の定義

3.8.1 抽象文字:テキストの編成、制御、または表現に使用される抽象的な値。
データ。

注 – 付録 H には、抽象文字という用語のより完全な説明が記載されています。

3.8.2 抽象値:定義が何らかのセマンティクスを伝達するために使用される型のみに基づいている値。
は、エンコーディングでの表現方法とは関係ありません。

注 – 抽象値の例は、整数型、ブール型、文字列型、または整数とブール値のシーケンス (または選択) である型の値です。

3.8.3 追加の時間タイプ:プロパティ設定サブタイプ表記を時間タイプ、または有用な時間タイプまたは定義された時間タイプに適用することによって、時間タイプ (3.8.83 を参照) のサブタイプとして定義されたタイプ。

3.8.4 ASN.1 文字セット: ASN.1 表記で使用される、条項 11 で指定された文字セット。

3.8.5 ASN.1 仕様: 1 つ以上の ASN.1 モジュールの集合。

3.8.6 関連タイプ:タイプの値とサブタイプ表記を定義するためにのみ使用されるタイプ。

注 – 関連タイプはこの推奨事項で定義されています。 ASN.1 でのタイプの定義方法とエンコード方法の間に大きな違いがある可能性があることを明確にする必要がある場合の国際標準。関連タイプはユーザー仕様には表示されません。

3.8.7 bitstring 型:識別値が 0、1 つ以上のビットの順序付けされたシーケンスである単純な型。

注 – 抽象値の埋め込みエンコーディングを運ぶ必要がある場合は、内容制約のないビット文字列 (またはオクテット文字列) タイプを使用します (Rec. ITU-T X.682 | ISO/IEC 8824-3 の条項を参照) 11) は廃止されました。それ以外の場合は、embedded-pdv タイプ (第 36 節を参照) を使用すると、より柔軟なメカニズムが提供され、抽象構文と埋め込まれた抽象値のエンコーディングのアナウンスが可能になります。

3.8.8 ブール型: 2 つの区別できる値を持つ単純な型。

3.8.9 文字プロパティ:文字レパートリーを定義するテーブル内のセルに関連付けられた情報のセット。

注 – 通常、情報には次の項目の一部またはすべてが含まれます。

a) グラフィックシンボル。

b) キャラクター名。

c) 特定の環境で使用される場合のキャラクターに関連付けられた機能の定義。

d) それが数字を表すかどうか。

e) (大文字/小文字) のみが異なる関連文字。

3.8.10 文字抽象構文:値が、指定された文字の集合からの 0 文字、1 文字以上の文字列のセットとして指定される抽象構文。

3.8.11 文字レパートリー:文字セット内の文字。そのような文字がどのようにエンコードされるかには影響を与えません。

3.8.12 文字列型:値が定義された文字セットの文字列である単純な型。

3.8.13 文字転送構文:文字抽象構文の任意の転送構文。

3.8.14 選択タイプ:個別タイプのリストを参照することによって定義されたタイプ。選択タイプの各値は、コンポーネント タイプの 1 つの値から導出されます。

3.8.15 コンポーネントタイプ: CHOICE、 SET、 SEQUENCE、 SET OF、またはSEQUENCE OFを定義するときに参照されるタイプの 1 つ。

3.8.16 制約:型に関連して、その型のサブタイプを定義するために使用できる表記法。

3.8.17 コンテンツ制約:コンテンツが指定された ASN.1 タイプのエンコーディングであること、またはコンテンツの生成および処理に指定されたプロシージャが使用されることを指定する、ビット文字列またはオクテット文字列タイプの制約。

3.8.18 制御文字:名前 (およびおそらくは特定の環境に関連して定義された機能) が与えられているが、グラフィック記号が割り当てられておらず、スペース文字ではない、一部の文字レパートリーに出現する文字。

注 – 水平タブ (9) および改行 (10) は、印刷環境で書式設定機能が割り当てられた制御文字の例です。 DATA LINK ESCAPE (16) は、通信環境で機能が割り当てられた制御文字の例です。

3.8.19 協定世界時 (UTC):標準周波数と標準信号の調整された配布の基礎を形成する、国際時間局 (国際時間局) によって維持される時間スケール。

注 1 – この定義のソースは Rec. ITU-R TF.460-5。 ITU-R は、協定世界時の頭字語を UTC とも定義しています。

注 2 – UTC とグリニッジ標準時 (GMT) は、最も実用的な目的で同じ時間を決定する 2 つの代替時間標準です。

3.8.20 デフォルトのエンコーディング参照 (モジュール用):モジュールヘッダーで指定され、エンコーディング参照を含まないすべての型プレフィックスで想定されるエンコーディング参照。

注 – デフォルトのエンコード参照がモジュールヘッダーで指定されていない場合、エンコード参照を含まないすべてのタイププレフィックスがタグを割り当てます。

3.8.21 定義された時間タイプ:時間タイプ (3.8.83 を参照) のサブタイプとして付録 B で定義されたタイプ。アプリケーションに必要な場合にアプリケーション設計者によってインポートされることを目的としています。

3.8.22 要素:管理型の値、または管理情報オブジェクト クラスの情報オブジェクト。それぞれ、同じタイプの他のすべての値または同じクラスの情報オブジェクトから区別できます。

3.8.23 要素セット:要素のセット。そのすべては支配型の値、または支配クラスの情報オブジェクトです。

注 – 管理クラスは Rec. で定義されます。 ITU-T X.681 | ISO/IEC 8824-2、3.4.7。

3.8.24embedded-pdv 型:値のセットが形式的にすべての可能な抽象構文の値のセットの和集合である型。この型は、ASN.1 仕様の外部で型を定義できる抽象値をそのプロトコル内で運ぶことを希望する ASN.1 仕様で使用できます。これは、伝送される抽象値の抽象構文 (型) の識別子と、その抽象値をエンコードするために使用されるエンコード規則の識別子を伝送します。

3.8.25 エンコーディング:一連のエンコーディング規則を抽象値に適用した結果得られるビットパターン。

3.8.26 エンコード制御セクション: ASN.1 モジュール内で定義または使用されるタイプにエンコード命令を割り当てることを可能にする ASN.1 モジュールの一部。

3.8.27 エンコード命令:タイププレフィックスまたはエンコード制御セクションを使用してタイプに関連付けることができ、1 つ以上の ASN.1 エンコード規則によってそのタイプのエンコードに影響を与える情報。

注 – エンコード命令は型の抽象値に影響を与えず、アプリケーションから見えることは期待されていません。

3.8.28 エンコーディング参照:タイププレフィックスまたはエンコーディング制御セクションのエンコーディング命令によって影響を受けるエンコーディング規則を識別する名前 (付録 E を参照)。

注 – エンコード参照TAG は、型プレフィックスがエンコード命令ではなくタグを割り当てることを指定するために使用できます (31.2 を参照)。

3.8.29 (ASN.1) エンコード規則: ASN.1 型の値の転送時の表現を指定する規則。エンコーディング ルールにより、型の知識があれば、表現から値を回復することもできます。

注 – エンコード規則を指定する目的では、組み込み型 (および値) の代替表記を提供できるさまざまな参照型 (および値) 表記は関係ありません。

3.8.30 列挙型:値に型表記の一部として個別の識別子が与えられる単純型。

3.8.31 拡張機能追加:拡張機能シリーズで追加された表記法の 1 つ。セット、シーケンス、選択タイプの場合、各拡張機能の追加は、単一の拡張機能追加グループまたは単一のコンポーネント タイプの追加です。列挙型の場合は、さらに 1 つの列挙型を追加します。制約の場合、サブタイプ要素を (1 つだけ) 追加することになります。

注 – 拡張機能の追加は、テキスト的に順序付けされ (拡張子マーカーに従う)、論理的に順序付けされます (列挙値が増加し、 CHOICE代替の場合はタグが増加します)。

3.8.32 拡張追加グループ:バージョン括弧内にグループ化されたセット、シーケンス、または選択タイプの 1 つ以上のコンポーネント。拡張追加グループは、ASN.1 モジュールの特定のバージョンに追加されたセット、シーケンス、または選択タイプのコンポーネントを明確に識別するために使用され、単純な整数でそのバージョンを識別できます。

3.8.33 拡張追加タイプ:拡張追加グループ内に含まれるタイプ、またはそれ自体が拡張追加である単一コンポーネント タイプ (このような場合、拡張追加グループには含まれません)。

3.8.34 拡張可能制約:外部レベルに拡張マーカーを持つサブタイプ制約、または拡張可能な値のセットによる集合算術の使用によって拡張可能なサブタイプ制約。

3.8.35 拡張挿入ポイント (または挿入ポイント):拡張追加が挿入される型定義内の位置。この位置は、型定義に 1 つの省略記号がある場合は拡張シリーズの直前の型の型表記の終わり、型定義に拡張マーカーのペアがある場合は 2 番目の省略記号の直前です。

注 – 任意の選択、シーケンス、またはセット タイプのコンポーネント内に存在できる挿入ポイントは最大 1 つです。

3.8.36 拡張マーカー:拡張シリーズの一部を形成するすべての型に含まれる構文フラグ (省略記号)。

3.8.37 拡張マーカーペア:間に拡張付加が挿入される拡張マーカーのペア。

3.8.38 拡張子関連:同じ拡張子ルートを持つ 2 つのタイプ。一方はもう一方に 0 個以上の拡張機能を追加することによって作成されます。

3.8.39 拡張ルート:拡張シリーズの最初のタイプである拡張可能なタイプ。拡張子マーカーと対応する "}" または ")" の間に、コメントと空白以外の追加表記のない拡張子マーカー、または 1 つのコンマ、コメント、および白以外の追加表記のない拡張子マーカーのペアが含まれます。 -拡張マーカー間のスペース。

注 – 拡張シリーズの最初のタイプになれるのは、拡張ルートのみです。

3.8.40 拡張シリーズ:一連の ASN.1 タイプで、拡張挿入ポイントにテキストを追加することによってシリーズ内の連続する各タイプが形成されるように順序付けることができます。

3.8.41 拡張可能型:拡張マーカーを持つ型、または拡張可能な制約が適用されている型。
注 – 拡張マーカーはテキストとして存在することも、EXTENSIBILITY-IMPLIED によって挿入されることもできます (13.4 を参照)。

3.8.42 外部参照:参照されているモジュール以外のモジュールで定義される、型参照、値参照、情報オブジェクト クラス参照、情報オブジェクト参照、または情報オブジェクト セット参照 (パラメーター化される場合がある)。 、参照される項目の前にモジュール名を付けることで参照されます。

例 – ModuleName.TypeReference

3.8.43 外部型: ASN.1 仕様の一部であり、その ASN.1 仕様の外部で定義できる型の値を運ぶ型。また、伝送される値のタイプの識別も伝送されます。

3.8.44 false:ブール型の識別値の 1 つ (「true」も参照)。

3.8.45 管理 (タイプ);ガバナー: ASN.1 構文の一部の解釈に影響を与える型定義または参照。ASN.1 構文のその部分が管理型の値を参照する必要があります。

3.8.46 同一型定義: ASN.1 "型" 生成 (第 17 条を参照) の 2 つのインスタンスは、附属書 C で指定された変換を実行した後、同一の語彙項目の同一の順序リストである場合、同一型定義として定義されます (第 12 項を参照)。

3.8.47 OID 国際化リソース識別子のタイプ:すべての OID 国際化リソース識別子のセット。
注 1 – これは、国際オブジェクト識別子ツリーのルートからノードに至る一連のアークを識別する一連の Unicode ラベルを値とする単純なタイプです。 ITU-T X.660 | ISO/IEC 9834シリーズ。
注 2 – Rec のルール。 ITU-T X.660 | ISO/IEC 9834-1 では、幅広い当局が独自に Unicode ラベルをツリーの円弧に関連付けることを許可しています。

3.8.48 整数型: (単一の値として) ゼロを含む、正と負の整数である区別される値を持つ単純な型。

注 – 特定のエンコード規則が整数の範囲を制限する場合、そのような制限は ASN.1 のユーザーに影響を与えないように選択されます。

3.8.49 字句項目: ASN.1 表記を形成する際に使用される、第 12 項で指定される、ASN.1 文字セットの文字の名前付きシーケンス。

3.8.50 モジュール:タイプ、値、値セット、情報オブジェクト クラス、情報オブジェクト、および情報オブジェクト セット (およびそれらのパラメーター化されたバリアント) に対する ASN.1 表記法の使用の 1 つ以上のインスタンス。 ASN.1 モジュール表記法 (第 13 節を参照)。

注 – 情報オブジェクトクラス（など）という用語は、Rec. で指定されています。 ITU-T X.681 | ISO/IEC 8824-2、パラメータ化は Rec. で指定されています。 ITU-T X.683 | ISO/IEC 8824-4。

3.8.51 null 型: null とも呼ばれる、単一の値で構成される単純な型。

3.8.52 オブジェクト:通信のインスタンスでの使用を識別するために名前を必要とする、明確に定義された情報、定義、または仕様。

注 – このようなオブジェクトは、Rec. で定義されている情報オブジェクトである可能性があります。 ITU-T X.681 | ISO/IEC 8824-2。

3.8.53 オブジェクト記述子タイプ:識別値が、オブジェクトの簡単な説明を提供する人間が判読できるテキストであるタイプ (3.8.52 を参照)。

注 – オブジェクト記述子の値は通常、単一のオブジェクトに関連付けられます。オブジェクト識別子の値のみがオブジェクトを明確に識別します。

3.8.54 オブジェクト識別子タイプ: Rec. で指定されているように、国際オブジェクト識別子ツリーのルートからノードに至る一連の弧を識別する一連の主整数値である単純なタイプ。 ITU-T X.660 | ISO/IEC 9834シリーズ。

注 1 – Rec のルール。 ITU-T X.660 | ISO/IEC 9834-1 では、広範囲の権威者が独立して主整数値をツリーの円弧に関連付けることを許可しています。

注 2 – オブジェクト識別子タイプの値表記 (およびそのタイプの XML エンコーディング) では、アークの 2 番目の識別子を含めることができます。

3.8.55 オクテット文字列型:識別値が 0、1 つ以上のオクテットの順序付けされたシーケンスであり、各オクテットが 8 ビットの順序付けられたシーケンスである単純なタイプ。

3.8.56 オープン システム相互接続:この勧告で使用される多くの用語を提供するコンピュータ通信のアーキテクチャ。国際規格の先頭に「OSI」という略語が付きます。

注 – このような用語の意味は、Rec. から入手できます。必要に応じて、ITU-T X.200 シリーズおよび同等の ISO/IEC 標準。この条件は、ASN.1 が OSI 環境で使用される場合にのみ適用されます。

3.8.57 オープン型表記法:複数の ASN.1 型の値のセットを示すために使用される ASN.1 表記法。

注 1 – 「オープンタイプ」という用語は、この勧告の本文では「オープンタイプ表記」と同義に使用されます。国際標準。

注 2 – すべての ASN.1 エンコード規則は、単一の ASN.1 タイプの値に対して明確なエンコードを提供します。これらは、通常仕様時に決定されない ASN.1 型の値を運ぶ「オープン型表記法」の明確なエンコーディングを必ずしも提供するわけではありません。そのフィールドの抽象値を明確に決定するには、「オープン型表記法」でエンコードされている値の型を知る必要があります。

注 3 – この勧告における唯一の表記 |オープンな型表記である国際標準は、Rec. で規定されている「ObjectClassFieldType」です。 ITU-T X.681 | ISO/IEC 8824-2、条項 14。「FieldName」は型フィールドまたは変数型値フィールドのいずれかを示します。

3.8.58 (サブタイプの) 親タイプ:サブタイプを定義するときに制約され、サブタイプの表記を制御するタイプ。

注 – 親タイプ自体が他のタイプのサブタイプである場合があります。

3.8.59 プロダクション: ASN.1 を指定するために使用される正式な表記法 (文法またはバッカス•ナウア形式、BNF とも呼ばれる) の一部。

3.8.60 実数型:識別値 (条項 21 で指定) に、 NOT-A-NUMBERなどの特殊な値とともに実数 (数値実数) のセットが含まれる単純な型。

3.8.61 (型の) 再帰定義:構造で使用されるすべての型が構造の定義前に定義されるように並べ替えることができない ASN.1 定義のセット。

注 – ASN.1 では再帰的定義が許可されています。表記法のユーザーは、使用される (結果の型の) 値が有限表現を持ち、その型に関連付けられた値セットに少なくとも 1 つが含まれていることを確認する責任があります。価値。

3.8.62 相対 OID 国際化リソース識別子タイプ:既知の OID 国際化リソース識別子に対する相対的な位置によってオブジェクトを識別する値。

3.8.63 相対オブジェクト識別子:既知のオブジェクト識別子に対する相対的な位置によってオブジェクトを識別する値。

3.8.64 相対オブジェクト識別子タイプ:値がすべての可能な相対オブジェクト識別子のセットである単純なタイプ。

3.8.65 制限付き文字列型:型仕様で指定された固定文字レパートリーから文字が取得される文字列型。

3.8.66 選択タイプ:選択タイプのコンポーネント タイプを参照することによって定義されたタイプ。その値は正確にそのコンポーネント タイプの値です。

3.8.67 シーケンス型:型の固定された順序付きリストを参照することによって定義される型 (その一部はオプションとして宣言される場合があります)。シーケンス タイプの各値は、各コンポーネント タイプから 1 つずつ、順序付けられた値のリストです。

注 – コンポーネントタイプがオプションであると宣言されている場合、シーケンスタイプの値にそのコンポーネントタイプの値が含まれる必要はありません。

3.8.68 型のシーケンス:単一のコンポーネント型を参照することによって定義された型。 sequence-of type の各値は、コンポーネント タイプの 0 個、1 つ以上の値の順序付きリストです。

3.8.69 (制約の) シリアル適用:すでに制約されている親タイプへの制約の適用。

3.8.70 集合算術:和集合の演算を使用した新しい値または情報オブジェクトの集合の形成。
50.2 で指定されている交差と差の集合 (EXCEPT の使用) 。

注 – 制約を逐次適用した結果は、「集合演算」という用語には含まれません。

3.8.71 設定 (時間プロパティの):特定の時間プロパティに関連付けることができる多数の値の 1 つ (3.8.82 および J.4.2 の注記を参照)。

注 – 特定の時間抽象値に適用される時間プロパティには、設定が 1 つだけあります (表 6 を参照)。

3.8.72 セット型:固定で順序付けされていない型のリストを参照することによって定義された型 (一部はオプションとして宣言される場合があります)。セット タイプの各値は、各コンポーネント タイプから 1 つずつ、順序付けされていない値のリストです。

注 – コンポーネントタイプがオプションであると宣言されている場合、セットタイプの値にそのコンポーネントタイプの値が含まれる必要はありません。

3.8.73 タイプのセット:単一のコンポーネントタイプを参照することによって定義されたタイプ。 set-of type の各値は、コンポーネント タイプの 0、1 つまたは複数の値の順序なしリストです。

3.8.74 単純型:値のセットを直接指定することによって定義される型。

3.8.75 スペース文字:文字列の印刷においてグラフィック文字とともに含めることを目的とした文字レパートリー内の文字ですが、物理的な表現では空のスペースによって表現されます。通常、これは制御文字とはみなされません (3.8.18 を参照)。

注 – 文字レパートリーには単一のスペース文字が存在する場合もあれば、さまざまな幅を持つ複数のスペース文字が存在する場合もあります。

3.8.76 サブタイプ (親タイプの):値が他のタイプ (親タイプ) の値のサブセット (または完全なセット) であるタイプ。

3.8.77 タグ:型の抽象値とは別の追加情報。すべての ASN.1 型に関連付けられ、型プレフィックスによって変更または拡張できます。

注 – タグ情報は、エンコーディングがあいまいでないことを保証するために、一部のエンコーディング ルールで使用されます。タグ情報は、型プレフィックスがない場合でも、すべての ASN.1 型に関連付けられているため、エンコード命令とは異なります。

3.8.78 タグ付き型:単一の既存の型とタグを参照することによって定義された型。新しい型は既存の型と同型ですが、既存の型とは異なります。

3.8.79 タグ付け:新しいタグをタイプに割り当て、既存の (デフォルトの) タグを置換または追加します。

3.8.80 時間抽象値:時間型の抽象値。

3.8.81 時間コンポーネント:時間抽象値の一部を指定する、その抽象値の定義の一部。

注 – 時間コンポーネントの例としては、日付コンポーネント (年コンポーネントを含む)、時刻コンポーネント、時差コンポーネントなどがあります。

3.8.82 (時間抽象値の) 時間プロパティ:時間抽象値を説明するために使用される多数の用語の 1 つ (3.8.80 を参照)。

注 – 時間抽象値を記述するために使用できる時間プロパティは、多くの場合、その抽象値の他の時間プロパティの設定に依存します。時間プロパティは表 6 の列 1 にリストされています。

3.8.83 time 型: ISO 8601 によって暗黙的に定義されたすべての抽象値をサポートする TIME 型。

3.8.84 転送構文:抽象構文内の抽象値を交換するために使用されるビット文字列のセット。通常、抽象構文にエンコード規則を適用することによって取得されます。

注 – 「転送構文」という用語は「エンコーディング」と同義です。

3.8.85 true:ブール型の識別値の 1 つ (「false」も参照)。

3.8.86 type:名前付きの値のセット。

3.8.87 型プレフィックス:エンコード命令またはタグを型に割り当てるために使用できる ASN.1 表記法の一部。

3.8.88 型参照名:あるコンテキスト内で型に一意に関連付けられた名前。

注 – この推奨事項で定義されているタイプには参照名が割り当てられています。国際標準;これらは ASN.1 内で広く利用可能です。他の参照名は、他の勧告 | で定義されています。国際標準であり、その勧告の文脈においてのみ適用されます。国際標準。

3.8.89 無制限文字列型:抽象値が文字抽象構文の値と、そのエンコーディングで使用される文字抽象構文および文字転送構文の識別子を含む型。

3.8.90 便利な時間型:アプリケーション設計者による直接使用を目的とした、時間型 (3.8.83 を参照) のサブタイプとして定義された組み込み型。

3.8.91 ユーザー (ASN.1): ASN.1 を使用して特定の情報の抽象構文を定義する個人または組織。

3.8.92 値マッピング: 2 つの型の値間の 1 対 1 の関係。これにより、一方の値への参照をもう一方の値への参照として使用できるようになります。これは、たとえば、サブタイプとデフォルト値の指定に使用できます (付録 C を参照)。

3.8.93 値参照名:あるコンテキスト内の値に一意に関連付けられた名前。

3.8.94 値セット:型の値のコレクション。意味的にはサブタイプと同等です。

3.8.95 バージョン括弧:拡張機能追加グループの開始と終了を示すために使用される、隣接する左右の括弧のペア(「[[」または「]]」)。オプションで、左括弧のペアの後に、拡張機能追加グループのバージョン番号を示す番号を続けることができます。

3.8.96 バージョン番号:バージョン ブラケットに関連付けることができる番号 (I.1.8 を参照)。

注 – バージョン番号は、拡張機能追加グループの一部ではない拡張機能追加や、選択、シーケンス、またはセット以外のタイプの拡張機能追加に追加することはできません。

3.8.97 ホワイトスペース:スペースやタブなど、印刷ページ上にスペースを生成する書式設定アクション。

4 略語

この勧告の目的のために |国際規格では、次の略語が適用されます。

ASN.1 抽象構文表記 1

ASN.1 の BER 基本エンコーディング規則

BMP基本多言語面

DCC データの国コード

DNIC データネットワーク識別コード

ASN.1 の ECN エンコーディング制御表記

ICD 国際コード指定者

IRI 国際化されたリソース識別子

OID オブジェクト識別子

OSI オープンシステム相互接続

ASN.1のPERパックエンコーディングルール

ROA認定運営会社

UCS ユニバーサル複数オクテット コード化文字セット

URI ユニバーサルリソース識別子

UTC 協定世界時

XML 拡張マークアップ言語

5 表記

5.1 一般的な

5.1.1 ASN.1 表記は、第 11 項で指定された ASN.1 文字セットの一連の文字で構成されます。

5.1.2 ASN.1 表記法の各使用には、字句項目にグループ化された ASN.1 文字セットの文字が含まれています。
第 12 条では、語彙項目を形成するすべての文字シーケンスを指定し、各項目に名前を付けます。

5.1.3 ASN.1 表記法は、ASN.1 表記法の有効なインスタンスを形成する語彙項目のシーケンスを指定して名前を付け、各シーケンスの ASN.1 セマンティクスを指定することによって、第 13 条 (およびそれに続く条項) で指定されます。

5.1.4 許可される語彙項目の順序を指定するには、この推奨事項 |国際標準
では、次の節で定義される正式な表記法を使用します。

5.2 プロダクション

5.2.1 すべての語彙項目には名前が付けられ (第 12 節を参照)、許可された語彙項目のシーケンスにも名前が付けられます。

5.2.2新しい (より複雑な) 許可された語彙項目のシーケンスがプロダクションによって定義されます。これは、字句項目の名前と字句項目の許可されたシーケンスの名前を使用し、新しい名前付きの字句項目の許可されたシーケンスを形成します。

5.2.3各プロダクションは、1 つまたは複数の行にある次の部分で順番に構成されます。

a) 新しく許可された語彙項目のシーケンスの名前。

b) キャラクター

::=

c) 5.3 で定義されている、文字で区切られた 1 つ以上の語彙項目の代替シーケンス

|

5.2.4語彙項目のシーケンスは、1 つ以上の代替案に存在する場合、新しい許可された語彙項目のシーケンスに存在します。新しい許可された語彙項目の順序は、この推奨事項で参照されています。上記 5.2.3 a) の名前による国際規格。

注 – 同じ語彙項目のシーケンスが複数の代替案に出現する場合、結果の表記法における意味上の曖昧さは、関連するテキストによって解決されます。

5.3 代替コレクション

5.3.1 プロダクション内の各代替 (5.2.3.c を参照) は、名前のリストによって指定されます。それぞれの名前は、
字句項目、または他のプロダクションによって定義および命名された、許可された字句項目のシーケンスの名前です。

5.3.2各選択肢によって定義される語彙項目の許可されたシーケンスは、名に関連付けられたシーケンス (または語彙項目) のいずれかを取得し、シーケンスのいずれか 1 つと組み合わせて (その後に) 取得することによって得られるすべてのシーケンスで構成されます。 2 番目の名前に関連付けられた (または語彙項目)、3 番目の名前に関連付けられたシーケンス (または語彙項目) のいずれかと組み合わせて (またはその後に)、姓 (または語彙項目) まで続きます。代替案で。

5.4 非間隔インジケーター

非スペース標識「&」 (アンパサンド) が生成シーケンス内のこれらの項目の間に挿入される場合、その前の字句項目とその後の字句項目は空白で区切られません。

注 – このインジケータは、XML 値の表記を記述するプロダクションでのみ使用されます。たとえば、語彙項目「<」の直後に XML タグ名を続けることを指定するために使用されます。

5.5 制作例

5.5.1 生産：

例Production bstring ::=

| hstring
| "{" 識別子リスト "}"

名前「ExampleProduction」を次の語彙項目のシーケンスに関連付けます。

a) 任意の「bstring」（語彙項目）。または

b) 任意の「hstring」（語彙項目）。または

c) 「IdentifierList」に関連付けられた語彙項目のシーケンス。先頭に「{」 、その後に「}」が続きます。

注 – 「{」と「}」は、単一文字{と}を含む語彙項目の名前です(12.37 を参照)。

5.5.2この例では、「IdentifierList」は、「ExampleProduction」を定義するプロダクションの前または後の、さらなるプロダクションによって定義されます。

5.6 レイアウト

この推奨事項で使用される各プロダクション | International Standard の前後には空行が続きます。
プロダクション内では空行は表示されません。生産は単一ラインで行われる場合もあれば、複数のラインにまたがる場合もあります。レイアウトは重要ではありません。

5.7 再帰

この推奨事項の作品 |国際標準は再帰的なことがよくあります。この場合、新しいシーケンスが生成されなくなるまで、プロダクションが継続的に再適用されます。

注 – 多くの場合、このような再適用により、許可される語彙項目のシーケンスが無限に増えます。セット内のシーケンスの一部またはすべてには、それ自体に無制限の数の語彙項目が含まれる場合があります。これはエラーではありません。

5.8 許可された語彙項目のシーケンスへの参照

この推奨事項 |国際標準では、プロダクション内の「::=」の前に表示される名前を参照することによって、許可された語彙項目のシーケンス (ASN.1 表記の一部) を参照します。名前は、プロダクションの一部として表示される場合を除き、自然言語テキストと区別するために引用符 (34) 文字 (") で囲まれています。

5.9 語彙項目への参照

この推奨事項 |国際標準では、語彙項目の名前を使用して語彙項目を参照します。名前が自然言語テキストに表示され、そのようなテキストと混同される可能性がある場合は、引用符 (34) 文字 (") で囲まれます。

5.10 省略表記

生成物をより簡潔で読みやすくするために、この勧告では許可される語彙項目のシーケンスの定義に次の省略表記が使用されます。国際規格および Rec.

ITU-T X.681 | ISO/IEC 8824-2、Rec. ITU-T X.682 | ISO/IEC 8824-3 および Rec. ITU-T X.683 | ISO/IEC 8824-4:

a) 2 つの名前、「A」と「B」に続くアスタリスク (*) は、「空の」語彙項目 (12.7 を参照)、または「A」に関連付けられた許可された語彙項目のシーケンスの 1 つ、または交互のシリーズを示します。 「A」に関連付けられた語彙項目のシーケンスの 1 つと「B」に関連付けられた語彙項目のシーケンスの 1 つで、両方とも「A」に関連付けられた語彙項目で始まり、終わります。したがって：

```
C ::= AB *
```

は以下と同等です:

```
C ::= D |空の
D ::= A | ABD
```

「D」は補助的な名前であり、作品の他の場所には登場しません。

例 – 「C ::= AB *」は、次の C の代替表現の短縮表記です。

```
空の
あ
ABA
アババ
アバババ
...
```

b) プラス記号 (+) は、「空」の語彙項目が除外されることを除いて、a) のアスタリスクと似ています。したがって：

```
E ::= AB +
```

は以下と同等です:

```
E ::= A |安倍
```

例 – 「E ::= AB +」は、次の E の代替表現の省略表記です。

```
あ
ABA
アババ
アバババ
...
```

c) 名前の後の疑問符 (?) は、「空の」語彙項目 (12.7 を参照) または許可された語彙項目のいずれかを示します。「A」に関連付けられた語彙項目のシーケンス。したがって：

```
F ::= A ?
```

は以下と同等です:

F ::= 空 |あ

注 – これらの省略表記は、生成シーケンスにおける語彙項目の並置よりも優先されます (5.2.2 を参照)。

5.11 値の参照と値の型指定

5.11.1 ASN.1 値割り当て表記により、指定された型の値に名前を付けることができます。この名前は、その値への参照が必要な場合にはどこでも使用できます。付録 C は、値のマッピングを説明および指定します。
ある型の値の値参照名で 2 番目の (類似した) 型の値を識別できるようにするメカニズム。
したがって、最初の値への参照は、2 番目のタイプの値への参照が必要な場合はどこでも使用できます。

5.11.2 ASN.1 標準の本文では、複数のタイプが関係する構成要素の合法性 (またはその他) を指定するために、通常の英語のテキストが使用されます。これらの合法性仕様では、通常、2 つ以上のタイプが「互換性」があることが必要です。たとえば、値参照の定義に使用される型は、値参照が使用される場合の管理型と「互換性」がある必要があります。規範的な付属書 C は、値マッピングの概念を使用して、特定の ASN.1 構成要素が合法であるかどうかについて正確に述べています。

6 型拡張の ASN.1 モデル

拡張可能な型をデコードするときに、デコーダーは以下を検出する可能性があります。

a) シーケンスまたはセットタイプに予期される拡張機能の追加が存在しない。または

b) シーケンスまたはセット型で定義されているもの (存在する場合) を超える任意の予期しない拡張追加の存在、選択型の未知の代替、列挙型の未知の列挙、または予期しない長さまたは値の存在制約が拡張可能な型。

形式的に言えば、拡張可能な型Xによって定義された抽象構文には、型 X の値だけでなく、X に拡張に関連するすべての型の値も含まれます。したがって、デコード プロセスは、次のいずれかの場合にエラーを通知することはありません。上記の状況 (a または b) が検出されます。それぞれの状況で実行されるアクションは、ASN.1 指定子によって決定されます。

注 – 多くの場合、予期しない追加拡張子の存在を無視し、存在しない予想される追加拡張子に対してデフォルト値または「欠落」インジケータを使用することが行われます。

拡張可能なタイプのデコーダによって検出された予期しない拡張機能の追加は、後続の送信で同じ転送構文が使用される場合に限り、そのタイプの後続のエンコード (送信者または第三者への送信用) に含めることができます。

7 エンコードルールの拡張性要件

注 – これらの要件は、標準化されたエンコード規則に適用されます。これらは、ECN を使用して定義されたエンコード ルールには適用されません (Rec. ITU-T X.692 | ISO/IEC 8825-3 を参照)。

7.1すべての ASN.1 エンコーディング ルールは、Xに拡張に関連する拡張可能な型Yを使用してデコードできるような方法で、拡張可能な型Xの値のエンコードを許可するものとします。さらに、エンコード ルールは、次のような値を許可するものとします。 Yを使用してデコードされて (Y を使用して) 再エンコードされ、 Y (したがってXも)に関連する拡張である3 番目の拡張可能なタイプZを使用してデコードされました。

注 – タイプX、 Y、およびZ は、拡張シリーズ内で任意の順序で表示されます。

拡張可能型Xの値がエンコードされ、(直接または拡張関連の型 Z を使用する中継アプリケーションを通じて) 別のアプリケーションに中継され、そのアプリケーションが Xに拡張に関連する拡張可能型Yを使用して値をデコードする場合、デコーダはタイプY は、以下で構成される抽象値を取得します。

a) 拡張ルート型の抽象値。

b) XとYの両方に存在する各拡張機能の抽象値。

c) XにはあるがYには存在しない各拡張機能追加 (存在する場合) の区切りエンコード。

c) のエンコーディングは、アプリケーションで必要な場合、後の Y の値のエンコーディングに含めることができます。そのエンコーディングは、Xの値の有効なエンコーディングでなければなりません。

チュートリアルの例:システム A がシーケンス型またはオプションの整数型の拡張機能を追加したセット型である拡張可能なルート型 (タイプ X) を使用し、システム B が拡張関連の型 (タイプ Y) を使用している場合には 2 つの拡張加算があり、それぞれがオプションの整数型である場合、B による最初の拡張加算の整数値を省略し 2 番目の拡張加算を含むYの値の送信を、A が最初の (のみ) の存在と混同してはなりません。知っているXの拡張子追加。さらに、A はXの値を次のように再エンコードできなければなりません。

最初の整数型に存在する値、続いてアプリケーション プロトコルで必要な場合、B から受信した 2 番目の整数値。

7.2すべての ASN.1 エンコード規則は、送信される値がエンコーダとデコーダによって共通に保持される拡張追加のセットに含まれる場合、列挙型と選択型の値のエンコードとデコードを指定するものとします。その後、デコードに成功します。それ以外の場合は、デコーダがそのエンコーディングを区切って、それを（未知の）拡張子追加の値として識別することが可能になります。

常 すべての ASN.1 エンコード ルールは、送信された値がエンコーダとデコーダによって共通に保持される拡張追加のセットに含まれる場合、正にデコードされるように、 7.3のような拡張可能な制約を持つ型のエンコードとデコードを指定する必要があります。それ以外の場合は、デコーダがエンコードを区切って、それを（未知の）拡張加算の値として識別することが可能になります。

すべての場合において、拡張子追加の存在は、拡張子マーカーを持つ型が他の型の中にネストされている場合に、後の内容を認識する能力に影響を与えないものとします。

注 1 – ASN.1 の基本エンコーディング ルールおよび ASN.1 のパック エンコーディング ルールのすべてのバリアントは、これらの要件をすべて満たしています。 ECN を使用して定義されたエンコーディング ルールは、必ずしもこれらの要件をすべて満たすわけではありませんが、満たす可能性はあります。

注 2 – PER と BER は、拡張機能追加のエンコーディングにおけるバージョン番号を識別しません。 ECN を使用して指定されたエンコーディングは、そのような識別を提供する場合と提供しない場合があります。

8 タグ

8.1タグは、クラスとクラス内の番号を指定することによって (この勧告 | 国際標準のテキスト内で、またはタイプ接頭辞を使用して) 指定されます。クラスは次のいずれかです。

– 普遍的;

- 応用;

- プライベート;

– コンテキスト固有。

8.2 この数値は、10 進数で指定された負ではない整数です。

8.3 ASN.1 のユーザーによって割り当てられるタグの制限は、31.2 で指定されます。

注 – 節 31.2 には、この表記法のユーザーが ASN.1 仕様でユニバーサル クラス タグを明示的に指定できないという制限が含まれています。他の 3 つのクラスのタグの使用には正式な違いはありません。アプリケーション クラス タグが使用される場合、一般に、ユーザーの選択とスタイルの問題として、代わりにプライベートまたはコンテキスト固有のクラス タグを適用できます。 3 つのクラスが存在するのは主に歴史的な理由によるものですが、クラスが通常使用される方法については G.2.12 で説明されています。

8.4 表 1 は、この勧告で指定されているユニバーサル クラスのタグの割り当てをまとめたものです。
|国際標準。

8.5一部のエンコード ルールでは、タグの正規の順序が必要です。均一性を提供するために、タグの正規の順序が 8.6 で定義されています。

8.6 タグの正規の順序は、各タイプの最も外側のタグに基づいており、次のように定義されます。

a) ユニバーサル クラス タグを持つ要素または代替が最初に表示され、次にアプリケーション クラス タグを持つ要素、次にコンテキスト固有のタグを持つ要素、次にプライベート クラス タグを持つ要素が続きます。

b) タグの各クラス内では、要素または代替要素はタグの昇順で表示されます。
数字。

表 1 – ユニバーサル クラス タグの割り当て

ユニバーサル0	エンコードルールで使用するために予約されています
ユニバーサル1	ブール型
ユニバーサル2	整数型
ユニバーサル3	ビット文字列型
ユニバーサル4	オクテットストリング型
ユニバーサル5	ヌル型
ユニバーサル6	オブジェクト識別子のタイプ
ユニバーサル7	オブジェクト記述子のタイプ
ユニバーサル8	外部型とインスタンスオブ型
ユニバーサル9	実数型
ユニバーサル10	列挙型
ユニバーサル11	埋め込み型 pdv タイプ
ユニバーサル12	UTF8文字列型
ユニバーサル13	相対オブジェクト識別子のタイプ
ユニバーサル14	時間タイプ
ユニバーサル15	この推奨事項の将来の版のために予約されています。国際標準
ユニバーサル16	シーケンスとシーケンスオブタイプ
ユニバーサル17	セット型とセット型
ユニバーサル 18-22、25-30	文字列型
ユニバーサル 23-24	UTC時間と一般化時間
ユニバーサル 31-34	それぞれ日付、時刻、日時、期間
ユニバーサル35	OID 国際化リソース識別子のタイプ
ユニバーサル36	相対 OID 国際化リソース識別子タイプ
ユニバーサル37-...	この推奨事項への追加として予約 国際標準

9 エンコード命令

9.1エンコード命令は、型プレフィックス (31.3 を参照) またはエンコード制御セクション (54 節を参照) を使用して型に割り当てられます。

9.2型プレフィックスにはエンコーディング参照が含まれる場合があります。そうでない場合、エンコード参照はモジュールのデフォルトのエンコード参照によって決定されます (13.5 を参照)。

た エンコード制御セクションには、常にエンコード参照が含まれます。複数のエンコード制御がある場合があります
だし、各エンコード制御セクションには個別のエンコード参照が必要です。

参 エンコード命令は、勧告 | で指定された一連の語彙項目で構成されます。 9.4エンコード参照によって決定される国際標準 (付録 E を照)。

9.5同じまたは異なるエンコード参照を持つ複数のエンコード命令を 1 つの型に割り当てることができます (型プレフィックスとエンコード制御セクションのいずれかまたは両方を使用)。特定のエンコード参照で割り当てられたエンコード命令は、別のエンコード参照で割り当てられたエンコード命令や、タグ付けを実行するための型プレフィックスの使用から独立しています。

9.6複数のエンコード命令に同じエンコード参照を割り当てた場合の影響 (型プレフィックスとエンコード制御セクションのいずれかまたは両方を使用) は、勧告 | で指定されています。国際標準はエンコード参照 (付録 E を参照) によって決定され、この勧告では指定されていません。国際標準。

9.7エンコード命令が「TypeAssignment」の「Type」に割り当てられている場合、その命令はその型に関連付けられ、「TypeAssignment」の「typereference」が使用される場所に適用されます。これには、export ステートメントと import ステートメントによる他のモジュールでの使用が含まれます。

10 ASN.1 表記の使用

10.1 タイプ定義の ASN.1 表記は「タイプ」でなければなりません (17.1 を参照)。

10.2 型の値の ASN.1 表記は「値」でなければなりません (17.7 を参照)。

注 – 一般に、型の知識がなければ値の表記を解釈することはできません。

10.3型を型参照名に割り当てるための ASN.1 表記は、「TypeAssignment」 (16.1 を参照)、「ValueSetTypeAssignment」 (16.6 を参照)、「ParameterizedTypeAssignment」 (Rec. ITU-T X.683 | ISO/ を参照) のいずれかでなければなりません。 IEC 8824-4、8.2)、または「ParameterizedValueSetTypeAssignment」(Rec. ITU-T X.683 | ISO/IEC 8824-4、8.2 を参照)。

10.4 値参照名に値を割り当てるための ASN.1 表記は、「ValueAssignment」のいずれかでなければなりません。 (16.2 を参照) または「ParameterizedValueAssignment」(Rec. ITU-T X.683 | ISO/IEC 8824-4、8.2 を参照)。

10.5 「割り当て」という表記のプロダクション代替案は、表記の範囲内でのみ使用されます。 「モジュール定義」 (13.1 の注 2 で指定されている場合を除く)。

11 ASN.1 文字セット

11.1語彙項目は、11.2、11.3、および 11.4 で指定されている場合を除き、表 2 にリストされている一連の文字で構成されます。表 2 では、文字は ISO/IEC 10646 で与えられた名前によって識別されます。

表 2 – ASN.1 文字

AからZまで	(ローマ字大文字 A からローマ字大文字 Z)
aからzまで	(ラテン小文字 A ～ ラテン小文字 Z)
0～9	(数字の 0 ～ 数字 9)
!	（エクスクラメーション•マーク）
」	（クォーテーションマーク）
&	(アンパサンド)
'	（アポストロフィ）
(	(左括弧)
)	(右括弧)
*	（アスタリスク）
、	（コンマ）
-	(ハイフン-マイナス)
。	(終点)
/	(ソリダス)
:	（結腸）
;	（セミコロン）
<	(小なり記号)
=	(等号)
>	(大なり記号)
@	(コマーシャルAT)
[	(左角括弧)
】	(右角括弧)
^	(曲折アクセント)
_	(ローライン)
{	(左中括弧)
|	（垂直線）
}	(右中括弧)
‾	(改行しないハイフン)

注 – 同等の派生標準が国家標準化団体によって開発されている場合、次の語彙項目に追加の文字が表示される場合があります。

- typereference (12.2 を参照);
- 識別子 (12.3 を参照);
- valuereference (12.4 を参照);
- モジュール参照 (12.5 を参照)。

大文字と小文字の区別が意味を持たない言語に対応するために追加の文字が導入される場合、上記の語彙項目の特定の最初の文字の大文字と小文字を区別することによって達成される構文上の区別は、別の方法で。これは、有効な ASN.1 仕様をさまざまな言語で記述できるようにするためです。

囲ま この表記法を使用して文字列タイプの値を指定する場合、定義された11.2文字セットのすべての文字は、ASN.1 表記法で引用符 (34) 文字 (") でれて表示できます (12.14 を参照)。

11.3 追加の (任意の) グラフィック記号が「コメント」語彙項目に表示される場合があります (12.6 を参照)。

11.4 Unicode ラベルの値を指定するために表記法が使用される場合、Unicode ラベルで許可されるすべての文字はASN.1 表記で使用できます。

11.5印刷スタイル、サイズ、色、強度、またはその他の表示特性は重要ではありません。

11.6 大文字と小文字は別のものとして扱われます。

11.7 ASN.1 定義では、字句項目の間に空白文字 (12.1.6 を参照) を含めることもできます。

11.8 非改行ハイフンとハイフンマイナスは、すべての名前で同一のものとして扱われます。
(予約語を含む)。

注 – My-Type などの名前は、ハイフンマイナスが含まれていても、非改行ハイフンが含まれていても同じ名前です。

12 ASN.1 字句項目

12.1 一般規則

12.1.1次の副次節は、字句項目内の文字を指定します。それぞれの場合において、字句項目の名前が、字句項目を形成する文字シーケンスの定義とともに与えられます。

12.1.2この条項 12 の副節で指定される語彙項目 (複数行の "comment"、"bstring"、"xmlbstring"、"hstring"、"xmlhstring"、"cstring"、"xmlcstrin
および "simplestring" を除く)) には空白を含めてはなりません (12.6、12.10、12.11、12.12、12.13、12.14、12.15、および 12.16 を参照)。単一行の「コメント」には「改行」文字を含めることはできません。 「非 integerUnicodeLabel」字句項目には、NON-BREAKING-SPACE 文字が含まれる場合があります。

12.1.3線の長さは制限されません。

12.1.4字句項目は、非スペース標識「&」 (5.4 を参照)が使用されている場合を除き、1 つ以上の空白 (12.1.6 を参照)またはコメント (12.6 を参照)で区切ることができます。 "XMLTypedValue" プロダクション (16.2 を参照) 内では、字句項目の間に空白が表示される場合がありますが、 "comment" 字句項目は存在しません。

注 – これは、「xmlcstring」字句項目内に隣接するハイフンまたはアスタリスクと塗りつぶしが存在することによって生じるあいまいさを避けるためです。このような文字は、「XMLTypedValue」プロダクション内に出現する場合、「コメント」語彙項目の開始を示すことはありません。

12.1.5次の語彙項目の最初の文字が許可された文字である場合、語彙項目は 1 つ以上の空白またはコメントによって後続の語彙項目から分離されなければなりません。前の語彙項目の文字の終わり。

12.1.6この推奨事項 |国際標準では、「改行」と「空白」という用語が使用されます。機械可読仕様で空白や改行 (行末)を表現する際には、以下の文字を1つ以上任意に組み合わせて使用することができます (各文字には、Unicode規格で規定されている文字名と文字コードが与えられています)):

空白の場合:

水平集計 (9)

改行 (10)

垂直集計 (11)

フォームフィード (12)

キャリッジリターン (13)

スペース (32)

休憩禁止スペース ({0,0,0,160})

改行の場合:

改行 (10)

垂直集計 (11)

フォームフィード (12)

キャリッジリターン (13)

注 – 有効な改行である文字または文字シーケンスは、有効な空白でもあります。

12.2 型参照

字句項目の名前 – typereference

12.2.1 「型参照」は、任意の数（1 つ以上）の文字、数字、およびハイフンで構成されます。頭文字は大文字とします。ハイフンを最後の文字にすることはできません。ハイフンの直後に別のハイフンを続けてはなりません。

注 – ハイフンに関するルールは、(おそらくその後に続く) コメントのあいまいさを避けるように設計されています。

12.2.2 「typereference」は、12.38 にリストされている予約文字シーケンスの 1 つであってはなりません。

12.3 識別子

字句項目の名前 - 識別子

「識別子」は、任意の数（1 つ以上）の文字、数字、ハイフンで構成されます。頭文字は小文字とする。ハイフンを最後の文字にすることはできません。ハイフンの直後に別のハイフンを続けてはなりません。

注 – ハイフンに関するルールは、(おそらくその後に続く) コメントのあいまいさを避けるように設計されています。

12.4 値の参照

字句項目の名前 – valuereference

「値参照」は、12.3 で「識別子」に指定された文字のシーケンスで構成されます。この表記法の使用例を分析する際、「値参照」は、それが出現するコンテキストによって「識別子」と区別されます。

12.5 モジュールリファレンス

字句項目の名前 – modulereference

「モジュール参照」は、12.2 の「タイプ参照」に指定された文字のシーケンスで構成されます。この表記法の使用例を分析する際、「モジュール参照」は、それが出現するコンテキストによって「タイプ参照」と区別されます。

12.6 コメント

語彙項目の名前 - コメント

12.6.1 ASN.1 表記の定義では「コメント」が参照されていません。ただし、他の語彙項目の間にいつでも出現する可能性があり、構文的な意味はありません。

注 – それにもかかわらず、推奨事項の文脈では、 ASN.1 を使用する国際標準。ASN.1 コメントには、アプリケーションのセマンティクスに関連する規範的なテキストや構文の制約が含まれる場合があります。

12.6.2語彙項目「コメント」には 2 つの形式があります。

a) 12.6.3 で定義されている「--」で始まる 1 行のコメント。

b) 12.6.4 で定義されている「/*」で始まる複数行のコメント。

12.6.3 「コメント」が隣接するハイフンのペアで始まる場合は常に、次の隣接するハイフンのペアまたは行の終わりのいずれか最初に発生した時点で終了します。コメントには、コメントを開始するペアと終了するペア (存在する場合) 以外の、隣接するハイフンのペアを含めることはできません。 「--」で始まるコメントに隣接する文字「/*」または「*/」が含まれている場合、これらには特別な意味はなく、コメントの一部とみなされます。コメントには、11.1 で指定された文字セットにないグラフィック記号が含まれる場合があります (11.3 を参照)。

12.6.4 「コメント」が「/*」で始まる場合は、この「*/」が同じ行にあるかどうかに関係なく、対応する「*/」で終わるものとします。 「*/」の前に別の「/ *」が見つかった場合、各「/*」に対して一致する「*/」が見つかった時点でコメントは終了します。 「/*」で始まるコメントに2 つの隣接するハイフン「--」が含まれている場合、これらのハイフンには特別な意味はなく、コメントの一部とみなされます。コメントには、11.1 で指定された文字セットにないグラフィック記号が含まれる場合があります (11.3 を参照)。

注 – これにより、ユーザーは、既にコメントが含まれている ASN.1 モジュールの部分 (コメントが"--で始まるかどうか) にコメントすることができます。」 または「/*」)、コメントする部分の先頭に「/*」を挿入し、末尾に「*/」を挿入するだけです。
コメントアウト部分に「/*」または「*/」を含む文字列値。

12.7 空の語彙項目

語彙項目の名前 – 空

「空」の項目には文字が含まれていません。これは、生成シーケンスの代替セットが指定されている場合に、すべての代替が存在しない可能性があることを示すために、第 5 条の表記で使用されます。

12.8 数字

語彙項目の名前 – 番号

「番号」は 1 つ以上の数字で構成されます。 「数字」が 1 桁でない限り、最初の桁はゼロであってはなりません。

注 – 「数値」字句項目は、10 進数表記として解釈されることにより、常に整数値にマップされます。

12.9 実数

字句項目の名前 – 実数

「実数」は、一連の 1 つ以上の数字である整数部分と、オプションで小数点 (.) で構成されます。
オプションで、小数点の後に 1 桁以上の小数部を続けることができます。整数部、小数点、または小数部 (最後に存在するもの) の後には、オプションで e または E、およびオプションで 1 桁以上の符号付き指数を続けることができます。 「実数」の先頭の桁は、それが唯一の桁であるか、直後に小数点が続き、少なくとも 1 桁がゼロでない小数部が続く場合を除き、ゼロであってはなりません。

「数値」も「実数」の有効なインスタンスです。この表記法の使用例を分析する際、「実数」は、それが出現する文脈によって「数値」と区別されます。

12.10 バイナリ文字列

字句項目の名前 – bstring

「bstring」は、任意の数 (おそらくゼロ) の文字で構成されます。

0 1

空白と混在する可能性があり、その前にアポストロフィー (39) 文字(')が続き、その後に次の文字のペアが続きます。

'B

例 – '01101100'B

バイナリ文字列の語彙項目内の空白の出現には意味がありません。

12.11 XML バイナリ文字列項目

項目名 – xmlbstring

「xmlbstring」は、ゼロ、1、または空白の任意の数 (おそらくゼロ) で構成されます。バイナリ文字列項目内に出現する空白文字には意味がありません。

例 – 01101100

この文字シーケンスは、「xmlhstring」および「xmlcstring」の有効なインスタンスでもあります。この表記法の使用例を分析する際、「xmlbstring」は、それが出現するコンテキストによって「xmlhstring」または「xmlcstring」と区別されます。

12.12 16 進文字列

字句項目の名前 – hstring

12.12.1 「hstring」は、任意の数（おそらくゼロ）の文字で構成されます。

ABCDEF 0 1 2 3 4 5 6 7 8 9

空白と混在する可能性があり、その前にアポストロフィー (39) 文字(')が続き、その後に次の文字のペアが続きます。

'H

例 – 'AB0196'H

16 進文字列字句項目内の空白の出現には意味がありません。

12.12.2各文字は、16 進表現を使用してセミオクテットの値を示すために使用されます。

12.13 XML 16 進文字列項目

項目名 – xmlhstring

12.13.1 「xmlhstring」は、任意の数（おそらくゼロ）の文字で構成されます。

0 1 2 3 4 5 6 7 8 9 ABCDEF abcdef

または空白。 16 進文字列項目内に現れる空白文字には意味がありません。

例 – Ab0196

12.13.2各文字は、16 進表現を使用してセミオクテットの値を示すために使用されます。

12.13.3 「xmlhstring」のすべてのインスタンスは「xmlcstring」の有効なインスタンスでもあり、一部のインスタンスは「xmlbstring」の有効なインスタンスでもあります。この表記法の使用例を分析する際、「xmlhstring」は、それが出現するコンテキストによって「xmlbstring」または「xmlcstring」と区別されます。

12.14 文字列

字句項目の名前 – cstring

12.14.1 「cstring」は、文字列タイプによって参照される文字セットからの任意の数 (おそらくゼロ) のグラフィック記号とスペース文字で構成され、前後に引用符 (34) 文字 (") が続きます。文字セットに QUOTATION MARK (34) 文字が含まれている場合、この文字 (「cstring」によって表される文字列に存在する場合) は、「cstring」内で同じ上の QUOTATION MARK (34) 文字のペアによって表されます。 「cstring」は複数行のテキストにまたがる場合があり、その場合、表現される文字列には「cstring」の行末の前後の位置に空白文字が含まれてはなりません。 「cstring」内の行末の直前または直後に現れるスペース文字には意味がありません。

注 1 – 「cstring」は、表現される文字列内のすべての文字にグラフィック記号が割り当てられているか、スペース文字である文字列を (印刷ページ上で) 明確に表現するためにのみ使用できます。制御文字を含む文字列を印刷表現で示す必要がある場合は、代替の ASN.1 構文を使用できます (第 39 条を参照)。

注 2 – 「cstring」で表される文字列は、図記号に関連付けられた文字と空白文字で構成されます。 「cstring」内の行末の直前または直後のスペース文字は、表現される文字列の一部ではありません (無視されます)。 「cstring」にスペース文字が含まれている場合、または文字レパートリー内のグラフィック記号が印刷表現で明確ではない場合、「cstring」で示される文字列はその印刷表現では曖昧になる可能性があります。

例 1 – 」屎 屍 市 弒」

例 2 – 「cstring」：

「ABCDE FGH」

IJK「XYZ」

IA5String型の文字列値を表すために使用できます。表現される値は次の文字で構成されます。

ABCDE FGHIJK"XYZ

ここで、プロポーショナル スペース フォント (上記で使用されているものなど) が印刷仕様で使用されている場合、または文字レパートリーに幅の異なる複数のスペース文字が含まれている場合、 EとFの間に意図されているスペースの正確な数は、印刷表現では曖昧になる可能性があります。

12.14.2文字が結合文字である場合 (付録 H を参照)、それは「cstring」の印刷表現で個別の文字として示されるものとします。結合する文字をオーバープリントしてはなりません。 (これにより、文字列値内の文字を結合する順序が印刷版で明確に定義されます。)

例 – ISO/IEC 10646 では、小文字の「e」とアクセント結合文字は 2 文字であるため、対応する「cstring」は 1 文字の é ではなく 2 文字として出力される必要があります。

12.15 XML文字列項目

項目名 – xmlcstring

12.15.1 「xmlcstring」は、次の ISO/IEC 10646 文字の任意の数 (おそらくゼロ) で構成されます。

a) 水平表 (9);

b) 改行 (10);

c) キャリッジリターン (13);

d) ISO/IEC 10646 文字コードが 32 (16 進数 20) から 55295 (16 進数 D7FF) の範囲にある文字、包括的;

e) ISO/IEC 10646 文字コードが 57344 (E000 hex) から 65533 (FFFD) の範囲にある任意の文字 16 進数) (両端を含む)

f) ISO/IEC 10646 文字コードが 65536 (16 進数 10000) から 1114111 の範囲にある任意の文字 (10FFFF 16 進数) (両端を含む)。

注 – 使用する場合、「xmlcstring」には、規定の文字列タイプで許可されている文字のみを含めるという要件により、追加の制限が課されます。

12.15.2文字「&」(アンパーサンド)、「<」(小なり記号)または「>」(大なり記号)は、12.15.4 または 12.15 で指定された文字シーケンスの 1 つの一部としてのみ出現するものとします。 5.

12.15.3 「xmlcstring」は、制限された文字列 (41.9 を参照) の値を表すために使用され、ISO/IEC 10646 文字のすべての組み合わせを直接または以下に指定するエスケープ シーケンスを使用して表すために使用できます。

注 1 – 「xmlcstring」は、 GeneralStringに現れる可能性のある一部の制御文字など、ISO/IEC 10646 に存在しない文字を表すために使用することはできません。また、ISO/IEC 10646 で定義されている可能性のある文字を表すこともできません。 10FFFF 16 進数を超える文字コード。

注 2 – 文字 LINE FEED (10) と CARRIAGE RETURN (13)、およびCARRIAGE RETURN + LINE FEED のペアは、準拠する XML プロセッサで処理される場合には区別されません。

12.15.4 「xmlcstring」で表される抽象文字列値に文字「&」(アンパーサンド)、「<」(小なり記号)または「>」(大なり記号)が存在する場合 (41.9を参照) 、「xmlcstring」では次のいずれかで表されます。

a) 12.15.8 で指定されたエスケープ シーケンス。または

b) エスケープシーケンス「&」、「<」または「>」それぞれ。これらのエスケープ シーケンスには次の内容を含めてはなりません ホワイトスペース (12.1.6を参照) 。

12.15.5 「xmlcstring」(41.9 参照)で表される抽象文字列値中に、表 3 の 1 列目にある ISO/IEC 10646 文字コードの文字が存在する場合は、その列の文字列で表現するものとする。表 3 の 2。これらの文字シーケンスには空白を含めてはなりません (12.1.6 を参照)。

注 – これには、10 進文字コード 9、10、および 13 の文字は含まれておらず、これらの文字シーケンス内の文字はすべて小文字です。

表 3 – 「xmlcstring」内の制御文字のエスケープ シーケンス

ISO/IEC 10646 文字コード	「xmlcstring」表現	ISO/IEC 10646 文字コード	「xmlcstring」表現
0 (16進数)	<null/>	17 (11 進数)	<dc1/>
1 (16進数)	<そう/>	18 (12 進数)	<dc2/>
2 (2進数) 3	<stx/>	19 (13 進数)	<dc3/>
(3進数) 4 (4	<etx/>	20 (14 進数)	<dc4/>
進数) 5 (5進	<eot/>	21 (15 進数)	<ナック/>
数) 6 (6進	<enq/>	22 (16 進数)	<syn/>
数) 7 (7進	<ack/>	23 (17 進数)	<etb/>
数) 8 (8進	<ベル/>	24 (18 進数)	<できる/>
数)	<bs/>	25 (19 進数)	
11 (B ヘックス)	<vt/>	26 (1A ヘックス)	<sub/>
12 (C 16 進数)	<ff/>	27 (1B 16進数)	<esc/>
14 (E 16 進数)	<だから/>	28 (1C 16進数)	<is4/>
15 (F hex) 16	<si/>	29 (1D 16進数)	<is3/>
(10 hex)	<dle/>	30 (1E ヘックス)	<is2/>
		31(1Fヘクス)	<is1/>

12.15.6 「xmlcstring」が XER エンコーディング (Rec. ITU-T X.693 | ISO/IEC 8825-4 を参照) の一部を形成する「XMLTypedValue」(16.2 を参照) 内で使用される場合、隣接する HYPHEN-MINUS が含まれる場合があります。 (45) 文字。 ASN.1 モジュールの XML 値表記のインスタンス内で使用する場合、隣接する 2 つのハイフンマイナス文字を含めることはできません。この文字シーケンスが ASN.1 モジュールの "xmlcstring" によって表される抽象文字列値に存在する場合、隣接するハイフンマイナス文字の少なくとも 1 つは、12.15.8 で指定されたエスケープ シーケンスによって表されます。

12.15.7 「xmlcstring」が XER エンコーディングの一部を形成する「XMLTypedValue」内で使用される場合 (Rec. ITU-T X.693 | ISO/IEC 8825-4 を参照)、隣接する ASTERISK (42) および SOLIDUS (47) 任意の順序の文字。 ASN.1 モジュールの XML 値表記のインスタンス内で使用する場合、隣接する ASTERISK 文字と SOLIDUS 文字を (順序を問わず) 含めてはなりません。この文字シーケンスが「xmlcstring」で表される抽象文字列値に存在する場合、隣接する ASTERISK 文字と SOLIDUS 文字の少なくとも 1 つは、12.15.8 で指定されたエスケープ シーケンスで表されます。

12.15.8 「xmlcstring」内に直接出現できる文字は、「&#n;」形式のエスケープ シーケンスによって「xmlcstring」内で表すこともできます。 (n は 10 進表記の ISO/IEC 10646 文字コードです) または "&#xn;" 形式の(n は 16 進表記の ISO/IEC 10646 文字コードです)。これらのエスケープ シーケンスには空白を含めてはなりません (12.1.6 を参照)。

注 1 – 「n」の 10 進値および 16 進値では先行ゼロが許可され、16 進値では小文字と大文字の「A」～「F」の両方を使用できます。

注 2 – 基本多言語面 (BMP) にない ISO/IEC 10646 文字にエスケープ シーケンス「&#n」および「&#xn」が使用されている場合、「n」の値は 65535 より大きくなります (FFFF 16 進数）。

例 – 「xmlcstring」：

ABCDé FGHîJK&XYZ

UTF8String 型の文字列値を表すために使用できます。表現される値は次の文字で構成されます。

ABCDé FGHîJK&XYZ

ここで、プロポーショナル スペース フォント (上記のような) が仕様で使用されている場合、é と F の間の正確なスペース文字は印刷媒体では曖昧になる可能性があります。

12.16 単純な文字列字句項目

アイテムの名前 – simplestring

「単純文字列」は、文字コードが 32 ～ 126 の範囲にある 1 つ以上の ISO/IEC 10646 文字で構成され、前後に引用符 (34) 文字 (") が続きます。引用符 (34) は含まれません。 ） キャラクター （"）。 「単純文字列」はテキストの複数行にまたがる場合があり、その場合、行末を表す文字は空白文字として扱われます。この表記法の使用例を分析する際、「simplestring」は、それが出現するコンテキストによって「cstring」と区別されます。

注 – 「simplestring」字句項目は、時間型のサブタイプ表記でのみ使用されます。

12.17 時刻値文字列

項目名 – 文字列

「tstring」は 1 つ以上の文字で構成されます。

0 1 2 3 4 5 6 7 8 9 + - : 。 、 / CDHMRPSTWYZ

前後に引用符 (34) 文字 (") が続きます。

注 – 「tstring」字句項目は、時刻タイプの値表記でのみ使用されます。

12.18 XML時刻値文字列項目

項目名 – xmltstring

「xmltstring」は、1 つ以上の文字で構成されます。

0 1 2 3 4 5 6 7 8 9 + - : 。 、 / CDHMRPSTWYZ

注 – 「xmltstring」字句項目は、時刻タイプの XML 値表記でのみ使用されます。

12.19 プロパティ名と設定名の語彙項目

アイテムの名前 – psname

「psname」は、任意の数 (1 つ以上) の文字、数字、およびハイフンで構成されます。頭文字は大文字とします。ハイフンを最後の文字にすることはできません。ハイフンの直後に別のハイフンを続けてはなりません。

注 – 「psname」字句項目は、時間型のサブタイプ表記で使用される「simplestring」の内容でのみ使用されます。

12.20 代入語彙項目

字句項目の名前 – 「::=」

この語彙項目は、次の文字列で構成されます。

::=

注 – このシーケンスには空白は含まれません (12.1.2 を参照)。

12.21 範囲区切り記号

語彙項目の名前 – 「..」

この語彙項目は、次の文字列で構成されます。

..

注 – このシーケンスには空白は含まれません (12.1.2 を参照)。

12.22 省略記号

語彙項目の名前 – 「...」

この語彙項目は、次の文字列で構成されます。

...

注 – このシーケンスには空白は含まれません (12.1.2 を参照)。

12.23 左バージョン括弧

字句項目の名前 – "[["

この語彙項目は、次の文字列で構成されます。

[[

注 – このシーケンスには空白は含まれません (12.1.2 を参照)。

12.24 右バージョン括弧

語彙項目の名前 – "]]"

この語彙項目は、次の文字列で構成されます。

]]

注 – このシーケンスには空白は含まれません (12.1.2 を参照)。

12.25 エンコーディングのリファレンス

項目名 – エンコードリファレンス

「encodingreference」は、12.2 の「typereference」に指定されている文字のシーケンスで構成されます。ただし、小文字は含まれません。

注 – 現在定義されているエンコード参照は、推奨事項とともに付録 E にリストされています。対応するエンコード命令の構文とセマンティクスを指定する国際標準。 「エンコーディングリファレンス」は、この勧告のこのバージョンまたは将来のバージョンの付録 E にリストされている配列のみから構成されます。国際標準。

12.26 整数値の Unicode ラベル

字句項目の名前 – integerUnicodeLabel

この語彙項目は、国際オブジェクト識別子ツリーのアークを識別する、0 (数字 0) から 9 (数字 9) までの範囲の ISO/IEC 10646 文字の任意の長さのシーケンスで構成されます。文字が 1 つだけで、国際オブジェクト識別子ツリーの関連する弧の主整数値が 0 でない限り、0 (数字ゼロ) 文字で始まってはなりません。

12.27 非整数 Unicode ラベル

字句項目の名前 – 非整数UnicodeLabel

この語彙項目は、Rec. で指定された制約を満たす ISO/IEC 10646 文字の任意の長さのシーケンスで構成されます。 ITU-T X.660 | ISO 9834-1、7.5 に準拠し、国際オブジェクト識別子ツリーのアークを識別します。字句解析の目的で、「integerUnicodeLabel」として識別できる文字だけで構成されてはなりません。

12.28 XML終了タグの開始項目

アイテムの名前 – 「</」

この項目は、次の文字列で構成されます。

</

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.29 XML 単一タグの終了項目

項目名 – 「/>」

この項目は、次の文字列で構成されます。

/>

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.30 XML ブール値 true 項目

アイテム名 – 「true」

12.30.1この項目は、次の文字列で構成されます。

真実

12.30.2この表記法の使用例を分析する際、「true」は、それが出現するコンテキストによって、「値参照」、「識別子」、または XML ブール値「extended-true」のインスタンスと区別されます。

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.31 XML ブール拡張 true 項目

項目名 – extend-true

12.31.1この項目は、次のいずれかの文字シーケンスで構成されます。

真実

または単一の文字:

1 (数字の 1)

12.31.2この表記法の使用例を分析する際、「拡張 true」は、それが出現するコンテキストによって、「値参照」、「識別子」、または XML ブール値「true」のインスタンスと区別されます。

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.32 XML ブール値 false 項目

項目名 – 「false」

12.32.1この項目は、次の文字列で構成されます。

間違い

12.32.2この表記法の使用例を分析する際、「false」は、それが出現するコンテキストによって、「値参照」、「識別子」、または XML ブール値「extended-false」のインスタンスと区別されます。

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.33 XML ブール型拡張 false 項目

項目名 – 拡張 false

12.33.1この項目は、次のいずれかの文字シーケンスで構成されます。

間違い

または単一の文字:

0 (数字のゼロ)

12.33.2この表記法の使用例を分析する際、「false」は、それが出現するコンテキストによって、「値参照」、「識別子」、または XML ブール値「false」のインスタンスと区別されます。

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.34 XML 実数非数値項目

アイテム名 – 「NaN」

12.34.1この項目は、次の文字列で構成されます。

NaN

12.34.2この表記法の使用例を分析する際、「NaN」は、それが出現する文脈によって大文字で始まる他の語彙項目と区別されます。

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.35 XML 実数無限項目

アイテム名 – 「INF」

12.35.1この項目は、次の文字列で構成されます。

INF

12.35.2この表記法の使用例を分析する際、「INF」は、それが出現する文脈によって、大文字で始まる他の語彙項目と区別されます。

注 – このシーケンスには空白文字は含まれません (12.1.2 を参照)。

12.36 ASN.1 タイプの XML タグ名

アイテムの名前 – xmlasn1typename

12.36.1この推奨事項 |国際標準では、ASN.1 組み込み型が XML タグ名として使用される場合、項目「xmlasn1typename」が使用されます。

12.36.2表 4 は、17.2 にリストされている各 ASN.1 組み込みタイプの「xmlasn1typename」を形成する文字シーケンスをリストしています。 ASN.1 組み込みタイプは、表 4 の列 1 でその製品名によって識別されます。 「xmlasn1typename」に使用される文字シーケンスは、表 4 の列 2 で識別されます。これらの文字シーケンスの前後には空白はありません。

12.36.3 「UsefulType」(45.1 を参照) の「xmlasn1typename」は、その定義で使用される「typereference」でなければなりません。

12.36.4 「ObjectClassFieldType」および「InstanceOfType」の項目「xmlasn1typename」の文字列をRec.に指定します。 ITU-T X.681 | ISO/IEC 8824-2、14.1 および付録 C。

12.36.5 ASN.1 組み込み型が「PrefixedType」の場合、「xmlasn1typename」を決定する型は「PrefixedType」の「Type」でなければなりません（31.1.5 を参照）。これ自体が「PrefixedType」である場合、この節 12.36.5 が再帰的に適用されます。

注 – 26.10 の副節は、「SelectionType」および「ConstrainedType」に使用される「Type」を指定します。

表 4 – xmlasn1typename の文字

ASN.1タイプのプロダクション名	xmlasn1typename の文字
ビット文字列タイプ	BIT_STRING
ブール型	ブール値
選択タイプ	選択
日付の種類	日付
日付時刻タイプ	日付時刻
期間タイプ	間隔
埋め込みPDVタイプ	順序
列挙型	列挙された
外部タイプ	順序
タイプのインスタンス	順序
整数型	整数
IRIタイプ	OID_IRI
NullType	ヌル
オブジェクトクラスフィールドタイプ	「記録」を参照してください。 ITU-T X.681	ISO/IEC 8824-2、14.10および14.11
オブジェクト識別子の種類	OBJECT_IDENTIFIER
オクテット文字列型	OCTET_STRING
接頭辞付きタイプ	12.36.5を参照
RealType	本物
相対IRIタイプ	RELATIVE_OID_IRI
相対OIDタイプ	RELATIVE_OID
RestrictedCharacterStringType	型名 (例: IA5String)
シーケンスタイプ	順序
タイプのシーケンス	シーケンス_OF
セットタイプ	セット
タイプのセット	SET_OF
時間の種類	時間
時刻タイプ	時刻
無制限の文字列タイプ	順序

12.37 単一文字の語彙項目

語彙項目の名前 –

「{」
"}"
「<」
>"

「、」
「。」

「/」
"("
")"
「[」
「]」
「-」（ハイフンマイナス）
「:」
「=」

"""" （クォーテーションマーク）
""" (アポストロフィー) ";"

「@」
"|"
「！」
「`」

上記の名前のいずれかの語彙項目は、引用符のない 1 文字で構成されます。

12.38 予約語

予約語の名前 –

不在
抽象構文
全て
応用
自動
始める
少し
BMP文字列
ブール値
による
キャラクター
選択
クラス
成分
コンポーネント
拘束された
含む
日付
日付時刻
デフォルト
定義
間隔
埋め込み
エンコード済み
エンコーディング制御
終わり
列挙された
を除外する
明示的
輸出
拡張性
外部の
間違い
から
一般化された時間
一般文字列
グラフィック文字列
IA5文字列
識別子
暗黙
暗黙的
輸入
含まれるもの
実例
説明書
整数
交差点
ISO646文字列
マックス
最小
マイナス無限大
数字ではありません
ヌル
数値文字列
物体
オブジェクト記述子
オクテット
の
OID-IRI
オプション
パターン
PDV
プラスインフィニティ
現在
印刷可能な文字列
プライベート
本物
相対OID
相対OID-IRI
順序
セット
設定
サイズ
弦
構文
T61ストリング
タグ
テレテックスストリング
時間
時刻
真実
タイプ識別子
連合
個性的
ユニバーサル
ユニバーサル文字列
UTC時間
UTF8文字列
ビデオテックス文字列
可視文字列
と

上記の名前を持つ語彙項目は、名前内の一連の文字で構成され、予約文字です。
シーケンス。

注 1 – これらのシーケンスでは空白は発生しません。

注 2 – キーワードCLASS、 CONSTRAINED、 CONTAINING、 ENCODED、 INSTANCE、 SYNTAX 、およびUNIQUEは、この推奨事項では使用されません。
国際標準;それらはRecで使用されます。 ITU-T X.681 | ISO/IEC 8824-2、Rec. ITU-T X.682 | ISO/IEC 8824-3 および Rec. ITU-T X.683 | ISO/IEC 8824-4。

13 モジュール定義

13.1 「ModuleDefinition」は次のプロダクションによって指定されます。

```
モジュール定義 ::=
        モジュール識別子
        定義
        エンコーディング参照デフォルト
        タグデフォルト
        拡張子デフォルト「::=」

        始める
        モジュール本体
        エンコーディング制御セクション
        終わり

モジュール識別子 ::= モジュ
        ール参照
        決定的な識別

確定的な識別 ::=
        決定的なOID
|決定版OIDとIRI |空の

決定的なOID ::=
        "{" DefinitiveObjIdComponentList "}"

決定的なOIDとIRI ::=
                決定的なOID
        IRI値

DefinitiveObjIdComponentList ::=
        DefinitiveObjIdComponent
| DefinitiveObjIdComponent DefinitiveObjIdComponentList

DefinitiveObjIdComponent ::=
        お名前フォーム
|決定的な番号フォーム
|決定的な名前と番号フォーム

DefinitiveNumberForm ::= 数値

DefinitiveNameAndNumberForm ::= 識別子"(" DefinitiveNumberForm ")"

エンコーディング参照デフォルト ::=
                エンコーディングリファレンス_空の

タグデフォルト::=
        明示的なタグ
|暗黙のタグ
|自動タグ|空の

拡張子のデフォルト::=
        拡張性の暗示
|空の

モジュール本体 ::=
        エクスポート インポート AssignmentList
```

```
|空の

エクスポート ::=
        エクスポートシンボルエクスポート ";"
|すべての「;」をエクスポートし
ます|空の

エクスポートされたシンボル ::=
        シンボルリスト
|空の

インポート ::=
        IMPORTSシンボルインポートされた ";" |
空の

インポートされたシンボル ::=
        モジュールリストからのシンボル |
空の

モジュールリストからのシンボル ::=
        モジュールからのシンボル
|モジュールからのシンボルリストモジュールからのシンボル

モジュールからのシンボル ::=
        GlobalModuleReference 選択オプションからのSymbolList

選択オプション ::= 空

| 「後継者」とともに
| 「子孫」とともに

GlobalModuleReference ::= モジュ
        ール参照 AssignedIdentifier

割り当てられた識別子 ::=
        オブジェクト識別子の値
|定義された値 |空の


シンボルリスト ::=
        シンボル
| SymbolList "," シンボル

記号::=
        参照
|パラメータ化された参照

参照::=
        タイプ参照値参照
        オブジェクトクラス
        参照オブジェクト参照オブ
        ジェクトセット参照


割り当てリスト ::=
        割り当て
|       割り当てリストの割り当て

割り当て ::=
        タイプの割り当て
|値の割り当て
| XML値の割り当て
| ValueSetTypeAssignment
|オブジェクトクラスの割り当て
|オブジェクトの割り当て
|オブジェクトセットの割り当て
|パラメータ化された割り当て
```

注 1 – 「エクスポート」および「インポート」リストでの「ParameterizedReference」の使用は、Rec. で指定されています。 ITU-T X.683 | ISO/IEC 8824-4。

注 2 – 例として (およびこの勧告での定義 | ユニバーサル クラス タグを持つ型の国際標準について)、「ModuleBody」は「ModuleDefinition」の外部で使用できます。

注 3 – 「TypeAssignment」、「ValueAssignment」、「XMLValueAssignment」、および「ValueSetTypeAssignment」の生成は条項 16 で指定されます。

注 4 – モジュール定義の「TagDefault」の値は、モジュール内で明示的に定義されたタイプにのみ影響します。インポートされた型の解釈には影響しません。

注 5 – セミコロンという文字は、割り当てリスト仕様やその従属製品には表示されず、ASN.1 ツール開発者による使用のために予約されています。

13.2 「TagDefault」が「空」の場合、 EXPLICIT TAGSとして扱われます。

注 – サブ条項 31.2 では、明示的タグ、暗黙的タグ、および自動タグの意味を説明します。

13.3 「TagDefault」の代わりにAUTOMATIC TAGSが選択されている場合、モジュールに対して自動タグ付けが選択されているとみなされ、それ以外の場合は選択されていないと言われます。自動タグ付けは、モジュールの定義内で発生する「ComponentTypeLists」および「AlternativeTypeLists」生成物に (追加の条件とともに) 適用される構文変換です。この変換は、シーケンス型、セット型、選択型の表記法に関して、それぞれ 25.8 ～ 25.10、27.3、29.2 ～ 29.5 で正式に規定されています。

13.4 EXTENSIBILITY IMPLIEDオプションは、それが許可されているモジュール内の各タイプの定義に拡張マーカー(「...」)をテキストで挿入することと同等です。暗黙の拡張マーカーの位置は、明示的に指定された拡張マーカーが許可される型内の最後の位置です。 EXTENSIBILITY IMPLIEDが存在しないということは、拡張マーカーが明示的に存在するモジュール内の型に対してのみ拡張性が提供されることを意味します。

注 –暗黙の拡張性は型のみに影響します。オブジェクト セットとサブタイプ制約には影響しません。

13.5 「EncodingReferenceDefault」は、「encodingreference」がデフォルトのエンコード参照であることを指定します。
モジュール。 「EncodingReferenceDefault」が「空」の場合、モジュールのデフォルトのエンコード参照はTAG です。

注 – 付録 E には、推奨事項とともに、許可されるエンコード参照のリストが含まれています。対応するエンコード命令の形式と意味を規定する国際規格。

13.6 「ModuleIdentifier」プロダクション内に現れる「modulereference」をモジュール名と呼びます。

注 – 同じ「モジュール参照」が割り当てられた「ModuleBody」を複数回使用することによって、単一の ASN.1 モジュールを定義する可能性は、(おそらく) 以前の仕様で許可されていました。この勧告では許可されていません |国際標準。

13.7 モジュール名は、モジュールの対象範囲内で 1 回だけ使用されます (13.10 で指定されている場合を除く)。
モジュールの定義。

13.8 「DefinitiveIdentification」が空でない場合、指定されたオブジェクト識別子とオプションの「IRIValue」値は、定義されているモジュールを識別する OID ツリーの同じノードを明確かつ一意に識別します。オブジェクト識別子の値を定義する際に、定義された値を使用することはできません。 「IRIValue」の生成は 34.3 で規定されています。

注 1 – 他のユーザーがそのモジュールを明確に参照できるように、少なくともオブジェクト識別子の値 (できればオブジェクト識別子の値と OID 国際化リソース識別子の値) をモジュールに割り当てることを強くお勧めします。

注 2 – モジュールのどのような変更に新しい「DefinitiveIdentification」が必要になるかという問題は、この推奨事項では扱われていません。国際標準。

13.9 「AssignedIdentifier」が空でない場合、「ObjectIdentifierValue」と「DefinedValue」の代替値は、参照名のインポート元のモジュールを明確かつ一意に識別します。 「AssignedIdentifier」の代わりに「DefinedValue」が使用される場合、それはオブジェクト識別子タイプの値になります。
「AssignedIdentifier」内にテキストで現れる各「valuereference」は、次のルールのいずれかを満たさなければなりません。

a) 定義されているモジュールの「AssignmentList」で定義されており、代入ステートメントの右側にテキストで表示されるすべての「valuereference」も、このルール (ルール "a") または次のルール (ルール ") を満たします。 b")。

b) 「AssignedIdentifier」にテキスト上「valuereference」が含まれていない「SymbolsFromModule」内の「シンボル」として表示されます。

注 1 – 他の人が明確にモジュールを参照できるように、オブジェクト識別子を割り当てることをお勧めします。

注 2 – この構文では、OID 国際化リソース参照 (割り当てられている場合) を参照先モジュールに含めることはできませんが、これを ASN.1 コメントに含めることをお勧めします。

13.10 「SymbolsFromModule」の「GlobalModuleReference」は、空ではない「DefinitiveIdentification」が含まれる場合を除き、別のモジュールの「ModuleDefinition」に表示されます。「modulereference」は 2 つの場合で異なる場合があります。

注 – 他のモジュールで使用されているものとは異なる「モジュール参照」は、同じ名前を持つ 2 つのモジュール (13.7 を無視して名前が付けられているモジュール) からシンボルをインポートする場合にのみ使用する必要があります。代替の個別の名前を使用すると、これらの名前をモジュールの本体で使用できるようになります (13.16 を参照)。

13.11モジュールの参照に「modulereference」と空ではない「AssignedIdentifier」の両方が使用される場合、後者が決定的であるとみなされます。

13.12参照されるモジュールに空ではない「DefinitiveIdentification」がある場合、そのモジュールを参照する「GlobalModuleReference」には空の「AssignedIdentifier」があってはなりません。

13.13 「Exports」の代わりに「SymbolsExported」が選択されている場合:

a) 「SymbolsExported」の各「シンボル」は、次の条件のうち 1 つだけを満たさなければなりません。

i) 構築中のモジュール内でのみ定義されます。または

ii) 「Imports」の代わりの「Symbolsimported」に 1 回だけ出現します。

b) モジュール外部からの参照が適切なすべての「シンボル」は「SymbolsExported」に含まれるものとし、これらの「シンボル」のみがモジュール外部から参照できます (13.14 で指定された緩和の対象)。そして

c) そのような「シンボル」が存在しない場合は、「SymbolsExported」(「Exports」ではない) の空の代替が使用されます。選ばれる。

13.14 「空」の代替案または「エクスポート」の代替案「 EXPORTS ALL 」のいずれかが選択されている場合、モジュール内で定義されている、またはモジュールによってインポートされているすべての「シンボル」は、13.13 a) で指定された制限に従って、他のモジュールから参照できます。

注 – 「エクスポート」の代替「空」は、下位互換性のために含まれています。

13.15 「NamedNumberList」、「Enumeration」、または「NamedBitList」に表示される識別子は、次の場合に暗黙的にエクスポートされます。それらを定義する型参照はエクスポートされるか、エクスポートされた型内のコンポーネント (またはサブコンポーネント) として表示されます。

13.16 「インポート」の代わりに「シンボルインポート」が選択されている場合:

a) 「SymbolsFromModule」の各「Symbol」は、モジュール本体で定義されるか、「SymbolsFromModule」の「GlobalModuleReference」で示されるモジュールの「Imports」句に存在する必要があります。参照先モジュールの「Imports」句にある「シンボル」のインポートは、その句内に「シンボル」が 1 つだけ存在し、その「シンボル」が参照先モジュールで定義されていない場合にのみ許可されます。

注 1 – これは、2 つの異なるモジュールで定義された同じシンボル名を別のモジュールにインポートすることを禁止するものではありません。ただし、同じ「シンボル」名がモジュール A の「Imports」句に複数回出現する場合、その「シンボル」名を A からエクスポートして別のモジュール B にインポートすることはできません。

b) 「SymbolsFromModule」の「GlobalModuleReference」によって示されるモジュールの定義で「Exports」の代替「SymbolsExported」が選択されている場合、「Symbol」はその「SymbolsExported」に表示されます。

c) 「SymbolsFromModule」の「SymbolList」内に表示される「シンボル」のみが、その「SymbolsFromModule」の「GlobalModuleReference」によって示される「モジュール参照」を持つ「外部<X>参照」内のシンボルとして表示されます。 " (<X> は「値」、「タイプ」、「オブジェクト」、「オブジェクトクラス」、または「オブジェクトセット」です)。

d) そのような「シンボル」が存在しない場合は、「Symbolsimported」の「空」の代替が選択されます。

注 2 – c) と d) の効果は、ステートメントIMPORTS;です。これは、モジュールに「外部<X>参照」を含めることができないことを意味します。

e) 「SymbolsFromModuleList」内のすべての「SymbolsFromModule」には、次のような「GlobalModuleReference」:

i) それらの「モジュール参照」は互いに異なり、また参照元モジュールに関連付けられた「モジュール参照」とも異なります。そして

ii) 「AssignedIdentifier」は、空でない場合、互いに、また参照モジュールに関連付けられたオブジェクト識別子の値（存在する場合）とはすべて異なるオブジェクト識別子の値を示す。

f) 「SymbolsFromModule」に空ではない「SelectionOption」がある場合、「GlobalModuleReference」は空であってはならず、参照されるモジュールは次のように決定されます。

私) 「SelectionOption」がWITH SUCCESSORS の場合、 「GlobalModuleReference」で示されるモジュールは、最後のノードが 0 回以上インクリメントされるオブジェクト識別子を持つ DefinitiveIdentification を持つモジュールです。複数のモジュールがこの基準を満たす場合、指定されたモジュールは、オブジェクト識別子に最大の増分数を持つ最後のノードを持つモジュールになります。

ii) 「SelectionOption」がWITH DESCENDANTS である場合、 「GlobalModuleReference」によって示されるモジュールは、「GlobalModuleReference」またはその下位の 1 つによって識別されるノードを識別する DefinitiveIdentification を持つモジュールです。複数のモジュールがこの基準を満たす場合、示されたモジュールが最大のオブジェクト識別子を持つモジュールとなります。この比較では、1 つのアークが異なる (最大のアークを選択) か、1 つのオブジェクト識別子の終わりに達する (長いオブジェクト ID を選択) まで、アークが連続して比較されます。

13.17 「Imports」の「空」の代替が選択されている場合でも、モジュールは「外部<X>参照」を使用して他のモジュールで定義された「シンボル」を参照できます。

注 – 「Imports」の代替「空」は、下位互換性のために含まれています。

13.18 「NamedNumberList」、「Enumeration」、または「NamedBitList」に表示される識別子は、次の場合に暗黙的にインポートされます。それらを定義する型参照はインポートされるか、インポートされた型内のコンポーネント (またはサブコンポーネント) として表示されます。

13.19 「SymbolsFromModule」内の「シンボル」は、「参照」として「ModuleBody」に表示される場合があります。 「シンボル」に関連付けられた意味は、対応する「GlobalModuleReference」によって示されるモジュール内でそのシンボルが持つ意味です。

13.20 「Symbol」が「AssignmentList」（非推奨）にも表示される場合、または「SymbolsFromModule」の 1 つ以上の他のインスタンスに表示される場合は、「External<X>Reference」でのみ使用されます。そのように表示されていない場合は、それを「参考」として直接使用するものとします。

13.21 「割り当て」のさまざまな代替案は、この勧告の次の条項で定義されています。国際規格、特に記載のない限り：

代替割り当て	副節の定義
「タイプ割り当て」	16.1
「値の割り当て」	16.2
「XML値の割り当て」	16.2
「ValueSetTypeAssignment」	16.6
「オブジェクトクラスの割り当て」	記録ITU-T X.681 | ISO/IEC 8824-2、9.1
「オブジェクトの割り当て」	記録ITU-T X.681 | ISO/IEC 8824-2、11.1
「オブジェクトセット割り当て」	記録ITU-T X.681 | ISO/IEC 8824-2、12.1
「パラメータ化された割り当て」	記録ITU-T X.683 | ISO/IEC 8824-4、8.1

すべての「割り当て」の最初のシンボルは、「参照」の代替の 1 つであり、定義されている参照名を示します。 「AssignmentList」内の 2 つの割り当てにおいて、参照名が同じであってはなりません。

13.22 「EncodingControlSections」は第 54 条に規定されています。

14 型と値の定義の参照

14.1 定義された型と値の生成：

```
定義されたタイプ::=
        外部タイプ参照
|タイプリファレンス
|パラメータ化されたタイプ
|パラメータ化された値セットタイプ

定義された値 ::=
        外部値参照
|値参照
|パラメータ化された値
```

型と値の定義を参照するために使用されるシーケンスを指定します。 Recには、「ParameterizedType」および「ParameterizedValueSetType」で識別される型と、「ParameterizedValue」で識別される値が指定される。 ITU-T X.683 | ISO/IEC 8824-4。

14.2 「NonParameterizedTypeName」のプロダクション：

```
NonParameterizedTypeName ::=
        外部タイプ参照
|       タイプリファレンス
```

| xmlasn1タイプ名

ASN.1 タイプを表すために XML タグ名が必要な場合に使用されます。結果の XML タグ名が文字「XML」で始まる場合、LOW LINE (95) が先頭に追加されて「NonParameterizedTypeName」が形成されます。

14.3 3 番目の代替案は、「XMLValueAssignment」（16.2 を参照）または「XMLOpenTypeFieldVal」（Rec. ITU-T X.681 | ISO/IEC 8824-2、14.6 を参照）の「XMLTypedValue」の「NonParameterizedTypeName」として使用してはならない) XML 値表記が ASN.1 モジュールで使用されている場合、「xmlasn1typename」が「CHOICE」、「ENUMERATED」、「SEQUENCE」、「SEQUENCE_OF」、「SET」または「SET_OF」の場合。

注 – これらの「xmlasn1typename」は ASN.1 タイプを定義しないため、この制限は ASN.1 モジュールで使用される XML 値表記に課せられます。「xmlasn1typename」から形成された XML タグ名は決定に使用されないため、エンコード ルール (XER など、Rec. ITU-T X.693 | ISO/IEC 8825-4 を参照) でのこの表記法の使用には制限はありません。エンコードされている型。

14.4 13.19 で指定されている場合を除き、「typereference」、「valuereference」、「ParameterizedType」、「ParameterizedValueSetType」または「ParameterizedValue」の代替は、参照がタイプまたは値が割り当てられている「ModuleBody」内にない限り使用してはならない（ 16.1 および 16.2 を参照)、「typereference」または「valuereference」を参照してください。

14.5 「外部タイプ参照」および「外部値参照」は、対応する「タイプ参照」または「値参照」がない限り使用してはならない。

a) それぞれタイプまたは値が割り当てられています (16.1 および 16.2 を参照)。または

b) 「輸入」条項に存在する、

「ModuleBody」内で、対応する「modulereference」を定義するために使用されます。別のモジュールの「Imports」句での名前の参照は、その句に「Symbol」が複数回出現しない場合にのみ許可されます。

注 – これは、2 つの異なるモジュールで定義された同じ「シンボル」を別のモジュールにインポートすることを禁止するものではありません。ただし、モジュール A のIMPORTS句に同じ「シンボル」が複数回出現する場合、外部参照でモジュールAを使用してその「シンボル」を参照することはできません。

14.6外部参照は、別のモジュールで定義され、以下のプロダクションで指定される参照名を参照するためにのみ、モジュール内で使用されます。

外部タイプ参照 ::=
モジュール参照
「。」
タイプリファレンス

外部値参照 ::=
モジュール参照
「。」
値参照

注 – 追加の外部参照プロダクション (「外部クラスリファレンス」、「外部オブジェクト参照」、および「外部オブジェクトセットリファレンス」) は Rec. で指定されます。 ITU-T X.681 | ISO/IEC 8824-2。

14.7参照モジュールが「Imports」の代替「SymbolsIMported」を使用して定義されている場合、外部参照の「modulereference」は、「SymbolsIMPORTS」内の「SymbolsFromModule」の 1 つの「GlobalModuleReference」に表示されます。参照元モジュールが「Imports」の「空」代替を使用して定義されている場合、外部参照の「modulereference」は、「Reference」が定義されているモジュール (参照元モジュールとは異なる) の「ModuleDefinition」に表示されます。

14.8 「DefinedType」が「Type」によって管理される表記法の一部として使用される場合（たとえば、「SubtypeConstraint」内）、「DefinedType」は、C.6.2 項で指定されている管理対象の「Type」と互換性があるものとします。

14.9 ASN.1 仕様内での「DefinedValue」の出現はすべて「Type」によって管理され、その「DefinedValue」は、C.6.2 項で指定されている準拠する「Type」と互換性のあるタイプの値を参照するものとします。 。

15 ASN.1 コンポーネントへの参照をサポートするための表記

15.1多くの目的のために、ASN.1 のタイプ、値などのコンポーネントへの正式な参照の要件があります。
そのような例の 1 つは、ASN.1 モジュール内の特定のタイプを識別するためのテキストを記述する必要がある場合です。この節は、そのような参照を提供するために使用できる表記法を定義します。

15.2 この表記法により、セットまたはシーケンス型のコンポーネント (型内に必須またはオプションで存在する) を識別できるようになります。

15.3 ASN.1 型定義の任意の部分は、「AbsoluteReference」構文構造を使用して参照できます。

絶対参照 ::=
「@」モジュール識別子
「。」
アイテムスペック

アイテムスペック ::=
タイプリファレンス
|アイテムID 「.」コンポーネントID

アイテムID ::= アイテムスペック

コンポーネント ID ::=
識別子
|番号
| 「*」

注 – AbsoluteReference プロダクションは、この推奨事項の他の場所では使用されていません。国際標準。これは、15.1 に記載されている目的のために提供されます。

15.4 「ModuleIdentifier」は ASN.1 モジュールを識別します (13.1 を参照)。

15.5 「DefinitiveIdentification」の最初または 2 番目の代替が「ModuleIdentifier」の一部として使用される場合、「DefinitiveIdentification」は名前の参照元のモジュールを明確かつ一意に識別します。

15.6 「typereference」は、「ModuleIdentifier」によって識別されるモジュールで定義されている任意の ASN.1 タイプを参照します。

15.7 各「ItemSpec」の「ComponentId」は、「ItemId」によって識別されたタイプのコンポーネントを識別します。識別するコンポーネントが set、sequence、set-of、sequence-of、または Choice タイプではない場合、これは最後の「ComponentId」となります。

15.8 「ComponentId」の「識別子」形式は、親の「ItemId」がセットまたはシーケンス型であり、その「ComponentTypeLists」内の「NamedType」の「識別子」の 1 つである必要がある場合に使用できます。セットまたはシーケンス。また、「ItemId」が選択肢タイプを識別し、その選択肢タイプの「AlternativeTypeLists」内の「NamedType」の「識別子」の 1 つである必要がある場合にも使用できます。それ以外の状況では使用できません。

of ま 「ComponentId」の数値形式は、「ItemId」が sequence-of タイプまたは set-of タイプの場合にのみ使用できます。数値の15.9 の値は、sequence-たは set-of 内の型のインスタンスを識別し、値「1」は型の最初のインスタンスを識別します。値 0 は、外側の型の値に存在する、sequence-of または set-of 内の型のインスタンスの数のカウントを含む概念的な整数型コンポーネント (転送では明示的に存在しない) を識別します。

15.10 「ComponentId」の「*」形式は、「ItemId」がシーケンスまたはセットの場合にのみ使用できます。 「ComponentId」の「*」形式の使用に関連付けられたセマンティクスは、 sequence-of および set-of のすべてのコンポーネントに適用されます。

注 – 次の例では:

M の定義 ::= 開始
T ::= シーケンス {
ブール値、
b 整数のセット
}
終わり

「T」のコンポーネントは、次のような ASN.1 モジュールの外部 (またはコメント内) のテキストによって参照される可能性があります。

-- (@MTb0 が奇数の場合) の場合:
--
(@MTb*は奇数とする)

これは、 bの成分の数が奇数の場合、 bのすべての成分が奇数でなければならないことを示すために使用されます。

16 型と値の割り当て

16.1 「typereference」には、「TypeAssignment」プロダクションによって指定された表記法によって型が割り当てられます。

タイプ割り当て ::=
タイプリファレンス
「::=」
タイプ

「typereference」は ASN.1 予約語であってはなりません (12.38 を参照)。

16.2 「valuereference」には、「ValueAssignment」または「XMLValueAssignment」プロダクションのいずれかで指定された表記法によって値が割り当てられます。

```
値の割り当て::=
    値参照
    タイプ
    「::=」
    価値

XMLValueAssignment ::=
    値参照
    「::=」
    XMLTypedValue

XMLTypedValue ::=
    "<" & NonParameterizedTypeName ">"
    XML値
    "</" & NonParameterizedTypeName ">"
| "<" & NonParameterizedTypeName "/>"
```

「ValueAssignment」の「valuereference」に割り当てられる値は「Value」であり、「Type」によって管理され、「Type」で定義された型の値の表記法となります（16.3で指定）。 「XMLValueAssignment」の「valuereference」に割り当てられる値は「XMLValue」（17.7 を参照）であり、「NonParameterizedTypeName」（16.4 で指定）で定義された型の値の表記となります。これが「xmlasn1typename」項目の場合、表 4 の対応する行の ASN.1 組み込みタイプを識別します (14.3 も参照)。明示的に禁止されている場合を除き、「XMLTypedValue」の「XMLValue」の前後に空白を使用できます (41.9 および Rec. ITU-T X.693 | ISO/IEC 8825-4、31.3.4.1 を参照)。

16.3 「値」は、17.7 で規定されている型の値の表記です。

16.4 「XMLValue」が型の値の表記法である場合、「XMLValue」はその型の「XMLBuiltinValue」表記法です。
タイプ (17.10 を参照)。

16.5 「XMLTypedValue」の 2 番目の選択肢 (XML 空要素タグの使用) は、「XMLValue」プロダクションのインスタンスが空の場合にのみ使用できます。

注 – 「XMLValue」プロダクションが空白のみを含む「xmlcstring」である場合、これは空ではないため、2 番目の代替は使用できません。

16.6 「typereference」には、「ValueSetTypeAssignment」プロダクションで指定された表記法によって設定された値を割り当てることができます。

```
値セットタイプの割り当て ::=
    タイプリファレンス
    タイプ
    「::=」
    値セット
```

この表記法は、「Type」で示される型のサブタイプとして定義され、「ValueSet」で指定または許可される値を正確に含む型を「typereference」に割り当てます。 「typereference」は ASN.1 予約語 (12.38 を参照) であってはならず、型として参照される場合があります。 「ValueSet」は16.7で定義されています。

16.7ある型によって管理される値セットは、「ValueSet」という表記によって指定されます。

```
値セット ::= "{"要素セット仕様"}"
```

値セットは、「ElementSetSpecs」で指定されたすべての値で構成されます。そのうちの少なくとも 1 つは、「ElementSetSpecs」で指定されます (条項 50 を参照)。

16.8 「ValueSetTypeAssignment」プロダクションは次のように展開されます。

```
タイプリファレンス
タイプ
「::=」
"{" ElementSetSpecs "}"
```

エンコード ルールの適用を含むあらゆる目的において、これはプロダクションの使用とまったく同等であるように定義されています。

タイプリファレンス
「::=」
タイプ
"(" ElementSetSpecs ")"

同じ「Type」および「ElementSetSpecs」仕様を持つ。

17 型と値の定義

17.1タイプは「タイプ」という表記によって指定されます。

タイプ ::= ビルトインタイプ |参照型 |制約されたタイプ

17.2 ASN.1 の組み込みタイプは、次のように定義される「BuiltinType」という表記によって指定されます。

ビルトインタイプ::=
ビット文字列タイプ
|ブール型
|文字列タイプ
|選択タイプ
|日付の種類
|日付時刻タイプ
|期間タイプ
|埋め込みPDVタイプ
|列挙型
|外部タイプ
|タイプのインスタンス
| 整数型
| IRIタイプ
| NullType
| オブジェクトクラスフィールドタイプ
| オブジェクト識別子の種類
| オクテット文字列型
| RealType
| 相対IRIタイプ
|相対OIDタイプ
|シーケンスタイプ
|タイプのシーケンス
|セットタイプ
|タイプのセット
|接頭辞付きタイプ
|時間の種類
|時刻タイプ

さまざまな「BuiltinType」表記は次の条項で定義されています (この勧告 | 国際版
特に明記されていない限り標準):

ビット文字列タイプ	22
ブール型	18
文字列タイプ	40
選択タイプ	29
日付の種類	38.4.1
日付時刻タイプ	38.4.3
期間タイプ	38.4.4
埋め込みPDVタイプ	36
列挙型	20
外部タイプ	37
タイプのインスタンス	記録ITU-T X.681 | ISO/IEC 8824-2、付録 C
整数型	19
IRIタイプ	34
NullType	24
オブジェクトクラスフィールドタイプ	記録ITU-T X.681 | ISO/IEC 8824-2、14.1
オブジェクト識別子の種類	32
オクテット文字列型	23

RealType	21
相対IRIタイプ	35
相対OIDタイプ	33
シーケンスタイプ	25
タイプのシーケンス	26
セットタイプ	27
タイプのセット	28
接頭辞付きタイプ	31
時間の種類	38.1.1
時刻タイプ	38.4.2

17.3 ASN.1 の参照型は、「ReferencedType」という表記で指定されます。

参照型 ::=
定義されたタイプ
|便利なタイプ
| 選択タイプ
| TypeFromObject
|オブジェクトからの値セット

「ReferencedType」表記は、他の型 (最終的には組み込み型) を参照するための代替手段を提供します。さまざまな「ReferencedType」表記と、それらが参照する型を決定する方法は、この推奨事項の次の場所で指定されています。特に明記されていない限り、国際規格:

定義されたタイプ	14.1
便利なタイプ	45.1
選択タイプ	30
TypeFromObject	記録ITU-T X.681 | ISO/IEC 8824-2、第 15 項
オブジェクトからの値セット	記録ITU-T X.681 | ISO/IEC 8824-2、第 15 項

17.4 「ConstrainedType」は第 49 条で定義されています。

17.5 この推奨事項 |国際標準では、
セット タイプ、シーケンス タイプ、選択タイプのコンポーネント。 「NamedType」の表記は次のとおりです。

NamedType ::= 識別子のタイプ

17.6 「識別子」は、値の表記、内部サブタイプ制約およびコンポーネント関係制約で、セット タイプ、シーケンス タイプ、または選択タイプのコンポーネントを明確に参照するために使用されます (Rec. ITU-T X.682 | ISO/IEC 8824 を参照) -3)。これはタイプの一部ではないため、タイプには影響しません。

17.7ある型の値は、「Value」という表記または「XMLValue」という表記によって指定されます。

値::=
ビルトインバリュー
|参照値
| オブジェクトクラスフィールド値

XML値 ::=
XML組み込み値
| XMLオブジェクトクラスフィールド値
注 1 – 「ObjectClassFieldValue」および「XMLObjectClassFieldValue」は Rec. ITU-T X.681 | ISO/IEC 8824-2、14.6。

注 2 – 「XMLValue」は「XMLTypedValue」でのみ使用されます。

17.8 「XMLValue」生成の一部が、XML 開始タグの直後に XML 終了タグが続く場合、12.1.4 で許可されているように挿入された空白で区切られる可能性があります (例: <field1></field1>)。)、これら 2 つの XML タグ、および間にある空白は、単一の XML 空要素タグ (<field1/>) で置き換えることができます。

注 – 12.1.4 で許可されている空白文字以外の空白文字が、開始タグの最後の「>」文字と終了タグの最初の「<」文字の間に存在する場合、上記の条件が適用されます。満足していません。

17.9 ASN.1 の組み込み型の値は、次のように定義される「XMLBuiltinValue」(17.10 を参照) または「BuiltinValue」という表記によって指定できます。

ビルトイン値 ::=
ビット文字列値
|ブール値
|文字列値
|選択値

|埋め込みPDV値
|列挙値
|外部値
|値のインスタンス
| 整数値
| IRI値
| Null値
| オブジェクト識別子の値
| オクテット文字列値
|リアルバリュー
| 相対IRI値
|相対OID値
|シーケンス値
|値の順序
|値の設定
|値のセット
| 接頭辞付きの値
|時間値

さまざまな「BuiltinValue」表記のそれぞれは、17.2 にリストされているように、対応する「BuiltinType」表記と同じサブ節で定義されます。

17.10 「XMLBuiltinValue」は次のように定義されます。

XMLBuiltinValue ::=
XMLBitStringValue
| XMLブール値
| XML文字列値
| XMLChoiceValue
| XMLEmbeddedPDVValue
| XML列挙値
| XML外部値
| XMLインスタンスの値
| XML整数値
| XMLIRI値
| XMLNull値
| XMLObjectIdentifierValue
| XMLOctetStringValue
| XMLRealValue
| XMLRelativeIRI値
| XML相対OID値
| XMLシーケンス値
| XML値のシーケンス
| XMLSetValue
| XMLSetOfValue
| XMLPrefixedValue
| XMLTimeValue

さまざまな「XMLBuiltinValue」表記はそれぞれ、上記の 17.2 にリストされているように、対応する「BuiltinType」表記と同じ句で定義されます。

17.11 ASN.1 の参照値は、「ReferencedValue」という表記で指定されます。

参照値 ::=
定義された値
|オブジェクトからの値

「ReferencedValue」表記は、他の値 (最終的には組み込み値) を参照するための代替手段を提供します。さまざまな「ReferencedValue」表記と、それらが参照する値を決定する方法は、次の場所で指定されています (特に明記されていない限り、この勧告 | 国際規格内)。

定義された値	14.1
オブジェクトからの値	記録ITU-T X.681	ISO/IEC 8824-2、第 15 項

17.12型が "BuiltinType"、"ReferencedType"、または "ConstrainedType" であるかどうかに関係なく、その値はその型の "BuiltinValue" または "ReferencedValue" のいずれかによって指定できます。

17.13 「NamedType」表記を使用して参照される型の値は、「NamedValue」表記によって定義されるか、「XMLValue」の一部として使用される場合は「XMLNamedValue」表記によって定義されます。これらの作品は次のとおりです。

NamedValue ::= 識別子の値

XMLNamedValue ::= "<" & 識別子 ">" XMLValue "</" & 識別子 ">"

ここで、「識別子」は「NamedType」表記で使用されるものと同じです。

注 – 「識別子」は表記法の一部であり、値自体の一部を形成するものではありません。を明確に指すために使用されます。セット型、シーケンス型、または選択型のコンポーネント。

17.14型の定義における拡張マーカーの暗黙的な存在 (13.4 を参照) または明示的な存在 (6 節を参照) は、値の表記に影響を与えません。つまり、拡張マーカーを持つ型の値の表記は、拡張マーカーが存在しない場合とまったく同じになります。

注 – サブ条項 50.11 では、サブタイプ制約で使用される値表記が、親タイプの拡張ルートにない値を参照することを禁止しています。

18 ブール型の表記法

18.1 ブール型 (3.8.8 を参照) は、「BooleanType」という表記によって参照されます。

ブール型 ::=ブール型

18.2 この表記法で定義された型のタグはユニバーサル クラス、番号 1 です。

18.3ブール型の値 (3.8.85 および 3.8.44 を参照) は、「BooleanValue」という表記によって定義されるか、「XMLValue」として使用される場合は「XMLBooleanValue」という表記によって定義されます。これらの作品は次のとおりです。

ブール値 ::= TRUE |間違い

XMLブール値::=
空の要素ブール値
|テキストブール値

空の要素ブール値 ::=
"<" & "true" "/>"
| 「<」&「false」「/>」

テキストブール値 ::=
拡張された真
| 拡張-偽

ん。 「XMLValueAssignment」に「EmptyElementBoolean」が出現する場合、その「XMLValueAssignment」には18.4 「TextBoolean」が出現してはなりませ

19 整数型の表記法

19.1 整数型 (3.8.48 を参照) は、「IntegerType」という表記によって参照されます。

整数型::=
整数
| INTEGER "{" NamedNumberList "}"

名前付き番号リスト ::=
名前付き番号
| NamedNumberList "," NamedNumber

名前付き番号 ::=
識別子"(" SignedNumber ")"
| 識別子"("定義値")"

署名付き番号 ::=
番号
| "-" 番号

19.2 「数値」がゼロの場合、「SignedNumber」の 2 番目の代替は使用されません。

19.3 「NamedNumberList」は型の定義において重要ではありません。 19.9 で指定された値の表記でのみ使用されます。

19.4 「DefinedValue」の「valuereference」は整数型でなければなりません。

注 – 「識別子」を使用して「NamedNumber」に関連付けられた値を指定することはできないため、「DefinedValue」が「IntegerValue」と誤って解釈されることはありません。したがって、次の場合には

```
整数 ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= INTEGER { a(3), b(a) }
c T2 ::= b
d T2 ::= a
```

c は、a の 2 番目または 3 番目の出現への参照にはならないため、値 1 を示し、 dは値 3 を示します。

「NamedNumberList」に現れる各「SignedNumber」または「DefinedValue」の値は19.5ずつ異なり、整数型の識別値を表します。

19.6 「NamedNumberList」に含まれる各「識別子」は異なるものとする。

19.7 「NamedNumberList」内の「NamedNumber」の順序は重要ではありません。

19.8 この表記法で定義された型のタグはユニバーサル クラス、番号 2 です。

19.9整数型の値は、「IntegerValue」という表記によって定義されるか、「XMLValue」として使用される場合は「XMLIntegerValue」という表記によって定義されます。これらの作品は次のとおりです。

```
整数値::=
        署名付き番号
|       識別子

XMLIntegerValue ::=
        XMLSignedNumber
|空の要素整数
|       テキスト整数

XMLSignedNumber ::=
        番号
|       "-" & 番号

空の要素整数 ::=
        「<」&識別子「/>」

テキスト整数 ::=
        識別子
```

19.10 「EmptyElementInteger」が「XMLValueAssignment」に出現する場合、
その「XMLValueAssignment」内の「TextInteger」 。

19.11 「IntegerValue」および「XMLIntegerValue」の最後の 2 つの選択肢の「識別子」は、値が関連付けられている「IntegerType」の「識別子」の 1 つであり、対応する番号を表すものとします。

注 – 「識別子」が定義されている整数値を参照する場合は、「IntegerValue」の「識別子」形式と「XMLIntegerValue」の「識別子」形式のいずれかを使用することを優先する必要があります。

19.12 「NamedNumberList」を使用した整数型の値表記のインスタンス内で、「NamedNumberList」の「識別子」と参照名の両方である名前が出現すると、「識別子」として解釈されます。

19.13 「数値」がゼロの場合、「XMLSignedNumber」の 2 番目の代替は使用されません。

20 列挙型の表記法

20.1 列挙型 (3.8.30 を参照) は、「EnumeratedType」という表記によって参照されます。

```
列挙型::=
        ENUMERATED "{"列挙型"}"

列挙型::=
        ルート列挙
| RootEnumeration "," "..." ExceptionSpec
| RootEnumeration "," "..." ExceptionSpec "," AdditionalEnumeration
```

RootEnumeration ::= 列挙型

AdditionalEnumeration ::= 列挙型

列挙 ::= 列挙項目 | EnumerationItem "," 列挙型

EnumerationItem ::= 識別子 |名前付き番号

注 1 – 「EnumeratedType」の各値には、個別の整数に関連付けられた識別子があります。ただし、値自体は整数のセマンティクスを持つことは期待されていません。 「EnumerationItem」の代わりに「NamedNumber」を指定すると、互換性のある拡張機能を容易にするために値の表現を制御できます。

注 2 – 「RootEnumeration」の「NamedNumber」内の数値は必ずしも順序付けされているわけではなく、連続している必要はありません。また、「AdditionalEnumeration」の「NamedNumber」内の数値は順序付けされていますが、必ずしも連続しているわけではありません。

20.2各「NamedNumber」について、「識別子」および「SignedNumber」は、「列挙」内の他のすべての「識別子」および「SignedNumber」とは区別されなければなりません。サブ条項 19.2 および 19.4 も各「NamedNumber」に適用されます。

20.3 「識別子」である各「EnumerationItem」(「EnumeratedType」内)には、個別の非負の整数が連続的に割り当てられます。 「RootEnumeration」には、「NamedNumber」である「EnumerationItem」で使用されるものを除いた、0から始まる連続した整数が代入されます。

注 – 整数値は、エンコード ルールの定義を支援するために「EnumerationItem」に関連付けられています。それ以外の場合、ASN.1 仕様では使用されません。

20.4 新しい各「EnumerationItem」の値は、その型内で以前に定義されたすべての「AdditionalEnumeration」よりも大きくなければなりません。

20.5 「AdditionalEnumeration」内の「EnumerationItem」の定義に「NamedNumber」が使用される場合、それに関連付けられた値は、以前に定義されたかどうかに関係なく、以前に定義されたすべての「EnumerationItem」(このタイプ内)の値とは異なるものとします。 「EnumerationItem」は列挙ルートに発生するかどうかに関係なく発生します。例えば：

```
A ::= ENUMERATED {a, b, ..., c(0)} -- 'a' と 'c' が両方とも 0 に等しいため無効です
B ::= ENUMERATED {a, b, ..., c, d(2)} -- 'c' と 'd' は両方とも 2 に等しいため無効です
C ::= ENUMERATED {a, b(3), ..., c(1)} -- 有効、 'c' = 1
D ::= ENUMERATED {a, b, ..., c(2)} -- 有効、 'c' = 2
```

20.6 「AdditionalEnumeration」代替の最初の「EnumerationItem」に関連付けられた「識別子」(「NamedNumber」ではない)の値は、「EnumerationItem」が「RootEnumeration」で定義されていない最小値でなければならず、すべての"AdditionalEnumeration" (存在する場合) 内の先行する "EnumerationItem" の方が小さいです。たとえば、次はすべて有効です。

```
A ::= 列挙型 {a, b, ..., c} -- c = 2
B ::= 列挙型 {a, b, c(0), ..., d} -- d = 3
C ::= 列挙型 {a, b, ..., c(3), d} -- d = 4
D ::= ENUMERATED {a, z(25), ..., d} -- d = 1
```

20.7 列挙型には、ユニバーサル クラス番号 10 のタグがあります。

20.8 列挙型の値は、「EnumeratedValue」という表記によって定義されるか、または列挙型として使用される場合に定義されます。「XMLValue」、表記法「XMLEnumeratedValue」。これらの作品は次のとおりです。

EnumeratedValue ::= 識別子

XMLEnumeratedValue ::=
空の要素列挙型
|テキスト列挙型

EmptyElementEnumulated ::= "<" & 識別子 "/>"

TextEnumulated ::= 識別子

その 「EmptyElementEnumerated」が「XMLValueAssignment」に出現する場合、出現はありません。「XMLValueAssignment」の「TextEnumerated」の20.9 。

20.10 「EnumeratedValue」および「XMLEnumeratedValue」の 2 つの選択肢の「識別子」は、値が関連付けられている「EnumeratedType」シーケンスの「識別子」と等しくなければなりません。

20.11列挙型の値表記のインスタンス内で、「列挙型」の「識別子」と参照名の両方である名前が出現すると、「識別子」として解釈されます。

21 実数型の表記法

21.1 実数型 (3.8.60 を参照) は、「RealType」という表記で参照されます。

```
RealType ::= REAL
```

21.2 実数タイプには、ユニバーサル クラスである番号 9 のタグが付いています。

21.3実数型の抽象値は、特殊な値PLUS-INFINITY、 MINUS-INFINITY、およびNOT-A-NUMBERと、プラス ゼロまたはマイナス ゼロで構成される数値実数、または以下の式で指定できる数値です。 3 つの整数、M、B、E:

$$M \times B^{E}$$

ここで、M (ゼロ以外) は仮数、B (2 または 10) は基数、E は指数と呼ばれます。 B = 2 (「ベース」 2 抽象値) および B = 10 (「ベース」 10 抽象値) の値は、個別の抽象値として定義されます。それ以外の場合、M B の値
E 同じ数値に評価されるものは、単一の抽象値です。

注 – マイナス ゼロとプラス ゼロは、数学的ゼロの 2 つの異なる抽象値であり、 「基数2」と「基数10」の抽象値は、他のすべての数値実数に対する個別の抽象値です。

21.4実数型には、実数型の数値の値とサブタイプの表記法をサポートするために使用される関連型があります (実数型の特殊な値とプラス ゼロとマイナス ゼロの表記法に加えて)。

注 – エンコーディング ルールは、エンコーディングの指定に使用される別のタイプを定義したり、関連するタイプを参照せずにエンコーディングを指定したりする場合があります。特に、BER および PER のエンコーディングは、「base」が 10 の場合は 2 進化 10 進数 (BCD) エンコーディングを提供し、「base」が 2 の場合はハードウェア浮動小数点表現との間の効率的な変換を可能にするエンコーディングを提供します。

21.5数値の値定義 (およびサブタイプの目的) に関連付けられた型は次のとおりです (規範的なコメント付き)。

```
順序 {
仮数 INTEGER、
基本 INTEGER (2|10)、
指数 INTEGER
    -- 関連する数学的実数は「仮数」です
    -- 「底」の「指数」乗算
}
```

注 1 – "base" 2 と "base" 10 で表される値は、同じ実数値に評価される場合でも、別個の抽象値とみなされ、異なるアプリケーション セマンティクスを持つ可能性があります。

注 2 – REAL (WITH COMPONENTS { ... , base (10)})という表記は、値のセットを "base" 10 の数値 (および "base" 2 の数値) に制限するために使用できます。この表記には、関連付けられた型で表現できない値 (特別な実数値とプラスとマイナスのゼロ) は含まれません。必要に応じて、セット算術を使用してこれらを追加できます。

注 3 – この型は、一般的な浮動小数点ハードウェアに格納できる任意の数値の正確な有限表現、および有限の文字 10 進数表現を持つ任意の数値を運ぶことができます。

21.6 実数型の値は、「RealValue」という表記によって定義されるか、「XMLValue」で使用される場合は次のように定義されます。「XMLRealValue」という表記:

```
リアルバリュー::=
    数値実数値
|   スペシャルリアルバリュー

数値実数値 ::=
    実数
| "-" 実数
|シーケンス値

特別な実値 ::=
    プラスインフィニティ
|マイナス無限大
|数字ではありません
```

注 – 「NumericRealValue」の 3 番目の選択肢は、プラス ゼロまたはマイナス ゼロの値には使用できません。これらの抽象値は、それぞれ最初または 2 番目の選択肢を使用して指定され、「実数」には 1 つの「0」文字が使用されます。

```
XMLRealValue ::=
    XML数値実数値 | XMLSpecialRealValue
```

```
XMLNumericRealValue ::=
        実数
|       "-" & 実数

XMLSpecialRealValue ::=
        空の要素実数
|       テキストリアル

空の要素実数 ::=
        "<" & PLUS-INFINITY "/>"
| "<" &マイナス無限大"/>"
| "<" &非数字"/>"

テキストリアル ::=
        「INF」
|       「-」と「INF」
| 「な〜ん」
```

21.7 「XMLValueAssignment」に「EmptyElementReal」が出現する場合、その「XMLValueAssignment」には「TextReal」は存在しません。

21.8 「実数」表記が使用される場合、対応する「基数」 10 の抽象値、またはプラス 0 を識別します。
「実数」値の前に「-」が付いている場合、負の数またはマイナス 0 である対応する「基数」 10 の抽象値を識別します。 "RealType" が"base" 2に制限されている場合、"realnumber" または "-" "realnumber" は、"realnumber" で指定された 10 進数値またはローカルに定義された数値のいずれかに対応する"base" 2 の抽象値を識別します。正確な表現が不可能な場合の精度。

22 ビット文字列型の表記法

22.1 ビット文字列タイプ (3.8.7 を参照) は、「BitStringType」という表記によって参照されます。

```
ビット文字列タイプ ::=
        ビット文字列

|ビット文字列"{" NamedBitList "}"

名前付きビットリスト ::=
        名前付きビット

| NamedBitList "," NamedBit

名前ビット ::=
        識別子「(」数値「)」
|       識別子"("定義値")"
```

22.2 ビット列の最初のビットは先頭ビットと呼ばれます。ビット列の最後のビットはトレーリング ビットと呼ばれます。

注 – この用語は、値の表記の指定とエンコード規則の定義に使用されます。

22.3 「DefinedValue」は、整数型の非負の値への参照でなければなりません。

値」ゼ 「NamedBitList」に現れる各「数値」または「DefinedValue」の値は異なり、ビット文字列値内の識別ビットの22.4数値です。ビット列の先頭ビットは「数ロによって識別され、後続のビットは連続した値を持ちます。

22.5 「NamedBitList」に含まれる各「識別子」は異なるものとする。

注 1 – 「NamedBitList」内の「NamedBit」生成シーケンスの順序は重要ではありません。

注 2 – 「NamedBitList」内に表示される「識別子」を使用して「NamedBit」に関連付けられた値を指定することはできないため、「DefinedValue」が「IntegerValue」と誤って解釈されることはありません。したがって、次の場合は次のようになります。

```
整数 ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= ビット文字列 { a(3), b(a) }
```

aの最後の出現は値 1 を示します。これは、a の 2 番目または 3 番目の出現への参照にはならないためです。

22.6 「NamedBitList」の存在は、このタイプの抽象値のセットには影響を与えません。指定されたビット以外の 1 ビットを含む値は許可されます。

22.7ビット文字列型の定義に「NamedBitList」が使用される場合、ASN.1 エンコード ルールは、エンコードまたはデコードされる値に末尾の 0 ビットを自由に追加 (または削除) できます。アプリケーション設計者がすべきことは、

したがって、後続の 0 ビットの数だけが異なる値には、異なるセマンティクスが関連付けられていないことを確認してください。

22.8 このタイプにはユニバーサル クラス番号 3 のタグが付いています。

22.9ビット文字列型の値は、「BitStringValue」という表記によって定義されるか、「XMLValue」として使用される場合は「XMLBitStringValue」という表記によって定義されます。これらの作品は次のとおりです。

```
ビット文字列値 ::=
        bstring
| hstring
| "{"識別子リスト"}"
| 「{」 「}」
|値を含む

識別子リスト ::=
        識別子
|       IdentifierList "," 識別子

XMLBitStringValue ::=
        XMLTypedValue
        | xmlbstring
        | XMLIdentifierList
|空の

XMLIdentifierList ::=
        空の要素リスト
|       テキストリスト

空の要素リスト ::=
        「<」&識別子「/>」
| EmptyElementList "<" & 識別子 "/>"

テキストリスト ::=
        識別子
|       TextList 識別子
```

22.10 「XMLValueAssignment」に「EmptyElementList」が出現する場合、その「XMLValueAssignment」には「TextList」が出現してはなりません。

22.11 「XMLTypedValue」代替は、ビット文字列に ASN.1 タイプを含み、ENCODED BY を含まない内容制約がある場合を除き、使用してはならない。この代替手段が使用される場合、「XMLTypedValue」はコンテンツ制約内の ASN.1 タイプの値になります。

22.12ビット文字列に「NamedBitList」がない限り、代替「XMLIdentifierList」は使用されません。

22.13 「BitStringValue」または「XMLBitStringValue」の代替の各「識別子」は、その値が関連付けられている「BitStringType」生成シーケンスの「識別子」と同じであるものとします。

22.14 「空」の代替は、ビットのないビット文字列を示します。

22.15ビット文字列に名前付きビットがある場合、「BitStringValue」または「XMLBitStringValue」の表記は、「識別子」に対応する番号で指定されたビット位置に 1 があり、他のすべてのビットが含まれるビット文字列値を示します。
ゼロ。

注 – 「NamedBitList」を持つ「BitStringType」の場合、 「BitStringValue」の「{」 「}」生成シーケンスと「XMLBitStringValue」の「空」は、1 ビットを含まないビット文字列を示すために使用されます。

22.16 「bstring」または「xmlbstring」表記を使用する場合、ビット文字列値の先頭ビットは左側にあり、ビット文字列値の末尾ビットは右側にあります。

22.17 「hstring」表記を使用する場合、各 16 進数の最上位ビットはビット文字列の左端のビットに対応します。

注 – この表記法は、エンコーディング ルールが転送のためにビット文字列をオクテットに配置する方法を決して制限するものではありません。

22.18 「hstring」表記は、ビット文字列値が 4 ビットの倍数で構成されていない限り使用してはならない。

例

'A98A'H

そして

'1010100110001010'B

は、同じビット文字列値の代替表記です。型が「NamedBitList」を使用して定義されている場合、(単一の) 末尾のゼロは値の一部を形成しないため、長さは 15 ビットになります。型が「NamedBitList」なしで定義されている場合、末尾のゼロは値の一部を形成するため、長さは 16 ビットになります。

22.19 CONTAINING代替は、CONTAININGを含むビット文字列型に内容制約がある場合にのみ使用できます。 「値」は、「ContentsConstraint」の「Type」の値の値表記となります (Rec. ITU-T X.682 | ISO/IEC 8824-3、第 11 節を参照)。

注 – この値の表記は、サブタイプ制約に現れることはありません。 ITU-T X.682 | ISO/IEC 8824-3 の 11.3 項では、「ContentsConstraint」の後のさらなる制約を禁止しており、上記のテキストでは、ガバナーが「ContentsConstraint」を持たない限り、その使用を禁止しています。

22.20 ENCODED BYを含まないビット文字列型に内容制約がある場合は、CONTAINING 代替が使用されます。

23 オクテット文字列型の表記法

23.1 オクテット文字列型 (3.8.55 を参照) は、「OctetStringType」という表記で参照されます。

オクテット文字列タイプ ::=オクテット文字列

23.2 このタイプには、ユニバーサル クラス番号 4 のタグが付いています。

23.3 オクテット文字列型の値は、「OctetStringValue」という表記によって定義されるか、または「XMLValue」、表記法「XMLOctetStringValue」。これらの作品は次のとおりです。

```
オクテット文字列値 ::=
        bstring
| hstring
|値を含む

XMLOctetStringValue ::=
        XMLTypedValue
|       xmlhstring
```

23.4 「XMLTypedValue」代替は、オクテット文字列に ASN.1 タイプを含み、ENCODED BY を含まない内容制約がある場合を除き、使用してはならない。この代替手段が使用される場合、「XMLTypedValue」はコンテンツ制約内の ASN.1 タイプの値になります。

23.5 オクテット文字列のエンコード規則を指定する場合、オクテットは最初のオクテットと末尾のオクテットという用語によって参照されます。オクテットであり、オクテット内のビットは最上位ビットと最下位ビットという用語で参照されます。

23.6 「bstring」表記を使用する場合、「bstring」表記の左端のビットは、オクテット文字列値の最初のオクテットの最上位ビットとなります。「bstring」が 8 ビットの倍数でない場合は、次の 8 の倍数にするために末尾にゼロが追加されているかのように解釈されます。

23.7 「hstring」または「xmlhstring」表記を使用する場合、左端の 16 進数は最初のオクテットの最上位セミオクテットでなければなりません。

23.8 「hstring」が奇数の 16 進数の場合、単一の文字列が含まれているかのように解釈されます。追加の末尾ゼロの 16 進数。 「xmlhstring」は奇数の 16 進数であってはなりません。

23.9 CONTAINING代替は、CONTAININGを含むオクテット文字列型に内容制約がある場合にのみ使用できます。 「値」は、「ContentsConstraint」の「Type」の値の値表記となります (Rec. ITU-T X.682 | ISO/IEC 8824-3、第 11 節を参照)。

注 – この値の表記は、サブタイプ制約に現れることはありません。 ITU-T X.682 | ISO/IEC 8824-3 の 11.3 項では、「ContentsConstraint」の後のさらなる制約を禁止しており、上記のテキストでは、ガバナーが「ContentsConstraint」を持たない限り、その使用を禁止しています。

23.10 ENCODED BYを含まないオクテット文字列型に内容制約がある場合は、CONTAINING 代替が使用されます。

24 null型の表記法

24.1 null 型 (3.8.51 を参照) は、「NullType」という表記によって参照されます。

NullType ::= NULL

24.2 このタイプにはユニバーサルクラスのタグ番号5が付いています。

24.3 null 型の値は、「NullValue」という表記によって参照されるか、「XMLValue」として使用される場合は、次のように参照されます。「XMLNullValue」という表記。これらの作品は次のとおりです。

Null値 ::= NULL

XMLNullValue ::= 空

25 シーケンス型の表記法

25.1 シーケンス タイプを定義するための表記法 (3.8.67 を参照) は、「SequenceType」となります。

シーケンスタイプ::=
順序"{" "}"
| SEQUENCE "{" ExtensionAndException OptionalExtensionMarker "}"
| SEQUENCE "{" ComponentTypeLists "}"

ExtensionAndException ::= "..." | 「...」例外仕様

OptionalExtensionMarker ::= "," "..." |空の

コンポーネントタイプリスト ::=
RootComponentTypeList
| RootComponentTypeList "," ExtensionAndException ExtensionAdditions
オプションの拡張マーカー
| RootComponentTypeList "," ExtensionAndException ExtensionAdditions
ExtensionEndMarker "," RootComponentTypeList
| ExtensionAndException ExtensionAdditions ExtensionEndMarker ","
RootComponentTypeList
| ExtensionAndException ExtensionAdditions OptionalExtensionMarker

ルートコンポーネントタイプリスト ::= コンポーネントタイプリスト

ExtensionEndMarker ::= "," "..."

拡張機能の追加 ::=
"," 拡張機能追加リスト |空の

拡張機能追加リスト ::=
拡張子追加
| ExtensionAdditionList "," ExtensionAddition

拡張機能の追加 ::=
コンポーネントの種類
|拡張機能追加グループ

ExtensionAdditionGroup ::= "[[" VersionNumber ComponentTypeList "]]"

バージョン番号 ::= 空 |番号"："

コンポーネントタイプリスト ::=
コンポーネントの種類
| ComponentTypeList "," コンポーネントタイプ

コンポーネントタイプ::=
名前付きタイプ
| NamedTypeオプション
| NamedType のデフォルト値
|タイプのコンポーネント

25.2 以下の節の目的上、「PrefixedType」は、次のいずれかの場合にテキストでタグ付けされたタイプとして定義されます。 a) 「PrefixedType」が「TaggedType」である。または b) 「PrefixedType」の「Type」がテキストでタグ付けされたタイプである。

25.3 「ComponentTypeLists」生成が、自動タグ付けが選択されているモジュールの定義内で発生し (13.3 を参照)、「ComponentType」の最初の 3 つの選択肢のいずれにも「NamedType」が出現せず、テキストでタグ付けされたタイプではない場合(25.2 を参照)、その場合、自動タグ付け変換は「ComponentTypeLists」全体に対して選択されます。それ以外の場合は、選択されません。

注 1 – シーケンス タイプのコンポーネントのリストの定義内で「TaggedType」表記を使用すると、自動タグ付けメカニズムによる自動割り当てとは対照的に、タグの制御が指定子に与えられます。したがって、以下では、
場合：

T ::= SEQUENCE { a 整数、 b [1] ブール値、 c オクテット文字列 }

シーケンスタイプTのこの定義が自動タグ付けが選択されているモジュール内で発生する場合でも、コンポーネントa、 b、 c のリストには自動タグ付けは適用されません。

注 2 – 自動タグ付けが選択されているモジュール内に出現する「ComponentTypeLists」プロダクションのみが、自動タグ付けによる変換の候補となります。

25.4自動タグ付け変換を適用するかどうかの決定は、「ComponentTypeLists」が出現するたびに、 25.5 で指定されたCOMPONENTS OF変換の前に個別に行われます。ただし、25.8 ～ 25.10 で指定されているように、自動タグ付け変換 (適用されている場合) は COMPONENTS OF 変換の後に適用されます。

注 – この結果、自動タグの適用は「ComponentTypeLists」にテキストで存在するタグによって抑制されますが、 COMPONENTS OF に続く「Type」に存在するタグによっては抑制されません。

25.5 「COMPONENTS OF Type」表記における「Type」はシーケンス型とする。 「COMPONENTS OF Type」表記は、コンポーネントのリストのこの時点で、拡張マーカーおよび拡張マーカーを除く、参照される型のすべてのコンポーネント タイプ (少なくとも 1 つ存在する) が含まれることを定義するために使用されます。 「タイプ」に存在する可能性のある拡張子の追加。 (「COMPONENTS OF Type 」の「Type」の「RootComponentTypeList」のみが含まれます。拡張マーカーと拡張機能の追加がある場合でも、 「COMPONENTS OF Type」表記では無視されます。) 参照される型に適用されるサブタイプ制約。はこの変換では無視されます。

注 – この変換は、次の節の要件が満たされる前に論理的に完了します。

25.6以下の副次節はそれぞれ、ルートまたは拡張機能の追加、あるいはその両方における一連の「ComponentType」の出現を識別します。 25.6.1 の規則は、そのようなシリーズすべてに適用されます。

25.6.1すべてOPTIONALまたはDEFAULTとマークされている「ComponentType」が 1 つ以上連続して出現する場合、それらの「ComponentType」のタグと、一連の直後に続くコンポーネント タイプのタグは別のものとします (31.2 を参照)。自動タグ付けが選択されている場合、タグが個別であるという要件は自動タグ付けが実行された後にのみ適用され、常に満たされます。

25.6.2サブ条項 25.6.1 は、ルート内の一連の「ComponentType」に適用されます。

25.6.3サブ条項 25.6.1 は、ルートまたは拡張機能の追加にある完全な一連の「ComponentType」に、タイプ定義内で出現するテキストの順序で適用されます (すべてのバージョン括弧と省略記号表記は無視されます)。 (52.7も参照してください。)

25.7 「ComponentTypeLists」の 3 番目または 4 番目の代替が使用される場合、拡張機能追加内のすべての「ComponentType」は、最初の「ComponentType」を含む、テキスト上に続く「ComponentType」のタグとは異なるタグを持たなければなりません。末尾の「RootComponentTypeList」にOPTIONALまたはDEFAULTのマークが付いている場合でも、そのマークが付いていません。 (52.7も参照してください。)

25.8 「ComponentTypeLists」の出現の自動タグ付け変換は、論理的に次の後に実行されます。
25.5 で指定された変換。ただし、25.3 が「ComponentTypeLists」の出現に適用すると決定した場合に限ります。自動タグ付け変換は、「NamedType」プロダクションに元々あった「Type」を 25.10 で指定された置換「TaggedType」オカレンスに置き換えることにより、「ComponentTypeLists」の各「ComponentType」に影響を与えます。

25.9自動タグ付けが有効で、拡張機能ルートの「ComponentType」にタグがない場合、「ExtensionAdditionList」内の「ComponentType」はテキストでタグ付けされたタイプであってはなりません。

10月25日 自動タグ付けが有効な場合、置換「TaggedType」は次のように指定されます。

a) 代替の「TaggedType」表記では、代替の「Tag Type」が使用されます。

b) 置換「TaggedType」の「Class」が空である(すなわち、タグ付けはコンテキスト固有である)。

c) 置換「TaggedType」の「ClassNumber」は、「RootComponentTypeList」の最初の「ComponentType」のタグ値は 0、2 番目の「ComponentType」のタグ値は 1、というようにタグ番号が増加していきます。

d) 「ExtensionAdditionList」の最初の「ComponentType」の置換「TaggedType」の「ClassNumber」は、「RootComponentTypeList」が欠落している場合はゼロ、そうでない場合は「RootComponentTypeList」の最大の「ClassNumber」より 1 大きくなります。 「ExtensionAdditionList」内の次の「ComponentType」の「ClassNumber」が最初のものより 1 つ大きくなるなど、タグ番号が増加していきます。

e) 置換後の「TaggedType」の「Type」は、置換される元の「Type」です。

注 1 – 置換「TaggedType」の暗黙的タグ付けまたは明示的タグ付けの仕様を管理する規則は、31.2.7 によって提供されます。自動タグ付けは、「タイプ」が選択タイプ、オープンタイプ表記、または「DummyReference」（Rec. ITU-T X.683 | ISO/IEC 8824-4、8.3 を参照）である場合を除き、常に暗黙的なタグ付けです。それは明示的なタグ付けです。

注 2 – 25.8 が満たされると、コンポーネントのタグは完全に決定され、自動タグ付け変換が適用される別の「ComponentTypeLists」内のコンポーネントの定義でシーケンス タイプが参照される場合でも変更されません。したがって、次の場合:

```
T ::= SEQUENCE { a Ta, b Tb, c Tc }
E ::= SEQUENCE { f1 E1, f2 T,                    f3 E3 }
```

Eのコンポーネントに適用される自動タグ付けは、 Tのタグ付け環境がどのようなものであっても、Tのコンポーネントa、 b 、およびcに付加されたタグには決して影響しません。Tが自動タグ付け環境で定義され、 Eが自動タグ付け環境にない場合、自動タグ付けは、 Tのコンポーネントa、 b 、およびcに引き続き適用されます。

注 3 – シーケンス タイプが「タイプのコンポーネント」の「タイプ」として表示される場合、その中の「ComponentType」の各出現箇所は、参照シーケンス タイプへの自動タグ付けの適用が可能になる前に、25.5 の適用によって複製されます。
したがって、次の場合:

```
T ::= SEQUENCE { a Ta, b SEQUENCE { b1 T1, b2 T2, b3 T3}, c Tc }
W ::= SEQUENCE { x Wx, T の成分, y Wy }
```

W が自動タグ付け環境で定義されている場合、 T内のa、 b 、およびc のタグは、 W内のa 、b、およびcのタグと同じである必要はありませんが、 b1、 b2 、およびb3のタグはTとWの両方で同じです。言い換えると、自動タグ付け変換は、指定された「ComponentTypeLists」に 1 回だけ適用されます。

注 4 – サブタイプは自動タグ付けには影響しません。

注 5 – 自動タグ付けが設定されている場合、拡張挿入ポイント (3.8.35 を参照) 以外の場所に新しいコンポーネントを挿入すると、タグ変更の副作用により他のコンポーネントが変更され、相互作用の問題が発生する可能性があります。仕様の古いバージョン。

11月25日 OPTIONALまたはDEFAULTが存在する場合、対応する値は新しいタイプの値から省略される場合があります。

25.12 DEFAULTが発生した場合、その型の値の省略は、「Value」で定義された値の挿入とまったく同じになります。これは、「NamedType」の「Type」で定義された型の値の値表記となります。 「制作の流れ。

25.13 「ExtensionAdditionGroup」（すべてのコンポーネントを合わせたもの）に対応する値はオプションです。ただし、そのような値が存在する場合は、括弧で囲まれた「ComponentTypeList」内の、 OPTIONALまたはDEFAULTとマークされていないコンポーネントに対応する値が存在するものとします。

25.14 「ComponentTypeList」のすべての「NamedType」生成シーケンス内の「識別子」（COMPONENTS OF の展開によって得られるものと合わせて）は、すべて別個のものでなければなりません。

25.15拡張追加タイプと拡張ルートの間に論理的に位置する、 OPTIONALまたはDEFAULTのマークが付いていないすべての拡張追加タイプに指定された値がない限り、特定の拡張追加タイプの値は指定されません。

注 1 – 型が拡張機能の追加によって拡張機能のルート (バージョン 1) からバージョン 2 を経てバージョン 3 に成長した場合、バージョン 3 からの追加のエンコーディングに存在するには、バージョン 3 のすべての追加のエンコーディングが存在する必要があります。 OPTIONALまたはDEFAULTのマークが付いていないバージョン 2 。

注 2 – 拡張機能の追加であるが、「ExtensionAdditionGroup」に含まれていない「ComponentType」は、 OPTIONALまたはDEFAULT とマークされていない場合は、常にエンコードする必要があります。ただし、以前のバージョンの を使用している送信者から抽象値が中継される場合を除きます。 「ComponentType」が定義されていない抽象構文。

注 3 – 次の理由から、「ExtensionAdditionGroup」プロダクションの使用が推奨されます。

a) エンコード規則 (PER など) に応じて、よりコンパクトなエンコードが得られる可能性があります。

b) 「ExtensionAdditionList」で定義され、 OPTIONALまたはDEFAULTとマークされていないタイプの値は、その値が定義されている拡張追加グループがエンコードされている場合、その値が常にエンコードに存在する必要があることを明確に示しているため、構文はより正確です (注1)と比較してください。

c) 構文により、「ExtensionAdditionList」内のどの型がグループとしてアプリケーションによってサポートされる必要があるかが明確になります。

25.16 「VersionNumber」は、モジュール内のすべての「ExtensionAdditions」および「ExtensionAdditionAlternatives」が「ExtensionAdditionGroup」または「VersionNumber」を持つ「ExtensionAdditionAlternativesGroup」である場合にのみ使用されます。「ExtensionAdditionGroup」の各「VersionNumber」の「数値」は 2 以上であり、挿入ポイント内の先行する「ExtensionAdditionGroup」の「数値」より大きくなければなりません。

注 1 – ここで使用される規則では、拡張機能追加グループのない仕様はバージョン 1 であるため、最初に追加される拡張機能追加グループの番号は 2 以上になります。 ここで、「ExtensionAdditions」には 1 つの「ExtensionAddition」が必要です。 "、"ExtensionAdditionGroup" は 1 つの "ExtensionAddition" で使用できます。

注 2 – 「VersionNumber」の使用に関する制限は単一モジュール内にのみ適用され、インポートされた型には制限が課されません。

25.17すべてのシーケンス型には、ユニバーサル クラス、番号 16 のタグがあります。

注 – Sequence-of タイプには、sequence タイプと同じタグが付いています (26.2 を参照)。

25.18シーケンス型の値を定義するための表記は「SequenceValue」、または「XMLValue」として使用される場合は「XMLSequenceValue」となります。これらの作品は次のとおりです。

```
シーケンス値 ::=
        "{" コンポーネント値リスト "}"
| 「{」 「}」

コンポーネント値リスト ::=
        名前付き値
| ComponentValueList "," NamedValue

XMLシーケンス値 ::=
        XMLコンポーネント値リスト
|空の

XMLコンポーネント値リスト ::=
        XMLNamedValue
| XMLComponentValueList XMLNamedValue
```

25.19 「{」 「}」または「空」表記は、次の場合にのみ使用されます。

a) 「SequenceType」内のすべての「ComponentType」シーケンスはDEFAULTまたはOPTIONAL とマークされ、すべての値が省略されます。または

b) 型表記はSEQUENCE{} でした。

25.20 OPTIONALまたはDEFAULTのマークが付いていない「SequenceType」内の「NamedType」ごとに 1 つの「NamedValue」または「XMLNamedValue」が存在し、値は対応する「NamedType」シーケンスと同じ順序でなければなりません。

26 sequence-of 型の表記法

26.1 別の型から sequence-of 型 (3.8.68 を参照) を定義するための表記は、「SequenceOfType」とします。

```
SequenceOfType ::=タイプのシーケンス| NamedTypeのシーケンス
```

注 – 「SequenceOfType」の XML 値表記で使用される XML タグ名に大文字の頭文字が必要な場合は、最初の代替文字を使用する必要があります。(XML タグ名は「タイプ」の名前から形成されます。)

26.2 すべての sequence-of 型には、ユニバーサル クラス、番号 16 のタグがあります。

注 – シーケンス型には、sequence-of 型と同じタグが付いています (25.17 を参照)。

26.3 sequence-of タイプの値を定義するための表記は、「SequenceOfValue」、または「XMLValue」として使用される場合は「XMLSequenceOfValue」となります。これらの作品は次のとおりです。

```
値の順序 ::=
        "{"値リスト"}"
| "{" NamedValueList "}"
| 「{」 「}」

値リスト ::=
        価値
|       ValueList "," 値

名前付き値リスト ::=
        名前付き値
| NamedValueList "," NamedValue

XMLシーケンス値::=
        XMLValueList
| XMLDelimitedItemList
|空の

XMLValueList ::=
        XMLValueOrEmpty
| XMLValueOrEmpty XMLValueList
```

```
XMLValueOrEmpty ::=
        XML値
| "<" & NonParameterizedTypeName "/>"

XMLDelimitedItemList ::=
        XMLDelimitedItem
| XMLDelimitedItem XMLDelimitedItemList

XMLDelimitedItem ::=
        "<" & NonParameterizedTypeName ">" XMLValue
                "</" & NonParameterizedTypeName ">" | "<" & 識別
子 ">" XMLValue "</" & 識別子 ">"
```

「{」 「}」または「空」の表記は、「SequenceOfValue」または「XMLSequenceOfValue」が空のリストの場合に使用されます。

注 – 意味上の重要性は、これらの値の順序に基づいて配置される場合があります。

26.4コンポーネントの「XMLValue」が「空」の場合、コンポーネントのその値を表すために「XMLValueOrEmpty」の 2 番目の選択肢が選択されます。

注 – これはSEQUENCE OF NULL の場合にのみ発生します。

26.5 「XMLValueList」または「XMLDelimitedItemList」プロダクションは、表 5 の列 2 に従って使用されます。コンポーネントの「タイプ」は列 1 にリストされています。

表 5 – ASN.1 タイプの「XMLSequenceOfValue」および「XMLSetOfValue」の表記

ASN.1タイプ	XML値の表記
ビット文字列タイプ	XMLDelimitedItemList
ブール型	26.6を参照
文字列タイプ	XMLDelimitedItemList
選択タイプ	XMLValueList
埋め込みPDVタイプ	XMLDelimitedItemList
列挙型	26.7を参照
外部タイプ	XMLDelimitedItemList
タイプのインスタンス	「記録」を参照してください。 ITU-T X.681 | ISO/IEC 8824-2、C.9
整数型	XMLDelimitedItemList
IRIタイプ	XMLDelimitedItemList
NullType	XMLValueList
オブジェクトクラスフィールドタイプ	「記録」を参照してください。 ITU-T X.681 | ISO/IEC 8824-2、14.10および14.11
オブジェクト識別子の種類	XMLDelimitedItemList
オクテット文字列型	XMLDelimitedItemList
RealType	XMLDelimitedItemList
相対IRIタイプ	XMLDelimitedItemList
相対OIDタイプ	XMLDelimitedItemList
シーケンスタイプ	XMLDelimitedItemList
タイプのシーケンス	XMLDelimitedItemList
セットタイプ	XMLDelimitedItemList
タイプのセット	XMLDelimitedItemList
接頭辞付きタイプ	26.10.1を参照
UsefulType (GeneralizedTime)	XMLDelimitedItemList
UsefulType (UTCTime)	XMLDelimitedItemList
UsefulType (オブジェクト記述子)	XMLDelimitedItemList
TypeFromObject	「記録」を参照してください。 ITU-T X.681 | ISO/IEC 8824-2、15.6
オブジェクトからの値セット	「記録」を参照してください。 ITU-T X.681 | ISO/IEC 8824-2、15.6

26.6 ブール型の値に「EmptyElementBoolean」が使用される場合は、「XMLValueList」が使用されます。それ以外の場合は、「XMLDelimitedItemList」が使用されます。

26.7 列挙型の値に「EmptyElementEnumerated」が使用される場合、「XMLValueList」は次のようになります。
中古品。それ以外の場合は、「XMLDelimitedItemList」が使用されます。

26.8 コンポーネントの「タイプ」が「DefinedType」の場合、コンポーネントを決定するタイプ「XMLSequenceOfValue」表記は、「DefinedType」によって参照される型でなければなりません（14.1 を参照）。

26.9 「XMLDelimitedItem」の 2 番目の代替は、「SequenceOfType」に「識別子」が含まれており、「XMLDelimitedItem」内の「識別子」がその「識別子」である場合にのみ使用されます。

26.10 "XMLDelimitedItem" の最初の代替が使用される場合、sequence-of type のコンポーネント ("TypePrefix" の出現を無視した後) が "typereference" または "ExternalTypeReference" である場合、 "NonParameterizedTypeName" は次のようになります。それぞれ「typereference」または「ExternalTypeReference」の「typereference」。それ以外の場合は、コンポーネントの組み込みタイプに対応する表 4 で指定された「xmlasn1typename」になります。

26.10.1コンポーネントの「Type」が「PrefixedType」の場合、「XMLSequenceOfValue」の代替および「xmlasn1typename」（必要な場合）を決定するタイプは、「PrefixedType」の「Type」となります（31.1を参照） .5）。これ自体が「PrefixedType」、「ConstrainedType」、または「SelectionType」である場合、26.10 のこれらの副節は再帰的に適用されます。

26.10.2コンポーネントの「Type」が「ConstrainedType」の場合、「XMLSequenceOfValue」の代替および「xmlasn1typename」（必要な場合）を決定するタイプは、「ConstrainedType」の「Type」となります（49.1を参照）。 ）。これ自体が「PrefixedType」、「ConstrainedType」、または「SelectionType」である場合、26.10 のこれらの副節は再帰的に適用されます。

26.10.3コンポーネントの「Type」が「SelectionType」の場合、「XMLSequenceOfValue」代替および「xmlasn1typename」（必要な場合）表記を決定するタイプは、「SelectionType」によって参照されるタイプでなければなりません（節を参照） 30）。これ自体が「PrefixedType」、「ConstrainedType」、または「SelectionType」である場合、26.10 のこれらの副節は再帰的に適用されます。

26.11 「SequenceOfType」の最初の代替が使用される場合、「SequenceOfValue」の最初の代替が使用されます。 「SequenceOfValue」の「ValueList」内の各「Value」、および「XMLSequenceOfValue」の代替内の各「XMLValue」は、「SequenceOfType」で指定されたタイプでなければなりません。

26.12 「SequenceOfType」の 2 番目の代替が使用される場合、「SequenceOfValue」の 2 番目の代替が使用され、「NamedValueList」の各「NamedValue」には、「NamedType」で指定されたタイプの「Value」が含まれます。 「SequenceOfType」。 「NamedValue」の「識別子」は、「SequenceOfType」の「NamedType」の「識別子」となります。

27 セットタイプの表記法

27.1 他の型からセット型を定義するための表記法 (3.8.72 を参照) は、「SetType」となります。

```
セットタイプ::=
        セット"{" "}"
| SET "{" ExtensionAndException OptionalExtensionMarker "}"
| SET "{" ComponentTypeLists "}"
```

「ComponentTypeLists」、「ExtensionAndException」、「OptionalExtensionMarker」は 25.1 で規定されています。

27.2 「COMPONENTS OF Type」表記における「Type」は集合型とする。 「COMPONENTS OF Type」表記は、コンポーネントのリストのこの時点で、「Type」に存在する可能性のある拡張マーカーと拡張機能の追加を除く、参照される型のすべてのコンポーネント タイプの包含を定義するために使用されます。 」。 (「COMPONENTS OF Type 」の「Type」の「RootComponentTypeList」のみが含まれます。拡張マーカーと拡張機能の追加がある場合でも、「COMPONENTS OF Type」表記では無視されます。) 参照される型に適用されるサブタイプ制約。はこの変換では無視されます。

注 – この変換は、次の節の要件が満たされる前に論理的に完了します。

27.3セットタイプ内の「ComponentType」タイプはすべて異なるタグを持つ必要があります (31.2 を参照)。 「ExtensionAdditions」に追加されるそれぞれの新しい「ComponentType」のタグは、「ExtensionAdditions」内の他のコンポーネントのタグよりも標準的に大きくなければなりません (8.6 を参照)。

注 – この表記が表示されるモジュールの「TagDefault」がAUTOMATIC TAGS である場合、これは 25.8 の適用の結果、実際の「ComponentType」に関係なく実現されます。 (52.7も参照してください。)

27.4 節 25.3 および節 25.8 ～ 25.14 もセット型に適用されます。

27.5 すべてのセット タイプには、ユニバーサル クラス、番号 17 のタグが付いています。

注 – タイプのセットには、タイプのセットと同じタグが付いています (28.2 を参照)。

27.6 セットタイプ内の値の順序に関連付けられたセマンティクスは存在しません。

27.7セットタイプの値を定義するための表記は「SetValue」、または「XMLValue」として使用される場合は「XMLSetValue」となります。これらの作品は次のとおりです。

```
値を設定::=
        "{" コンポーネント値リスト "}"
| 「{」 「}」

XMLSetValue ::=
        XMLコンポーネント値リスト
|空の
```

「ComponentValueList」と「XMLComponentValueList」は25.18で規定されています。

27.8 「SetValue」と「XMLSetValue」は、次の場合にのみそれぞれ「{」 「}」と「空」になります。

a) 「SetType」内のすべての「ComponentType」シーケンスはDEFAULTまたはOPTIONAL としてマークされ、すべての値は省略;または

b) 型表記はSET{} でした。

27.9 OPTIONALまたはDEFAULTのマークが付いていない「SetType」内の「NamedType」ごとに 1 つの「NamedValue」または「XMLNamedValue」が存在するものとします。

注 – これらの「NamedValue」または「XMLNamedValue」は任意の順序で表示できます。

28 型のセットの表記法

28.1 別の型から型のセット (3.8.73 を参照) を定義するための表記は、「SetOfType」とします。

```
タイプのセット::=
        タイプのセット
| NamedTypeのセット
```

注 – 「SetOfType」の XML 値表記で使用される XML タグ名に大文字の頭文字が必要な場合は、最初の代替文字を使用する必要があります。 (XML タグ名は「タイプ」の名前から形成されます。)

28.2 すべてのタイプのセットには、ユニバーサル クラス、番号 17 のタグがあります。

注 – セットタイプには、セットタイプと同じタグが付いています (27.5 を参照)。

28.3 set-of タイプの値を定義するための表記は、「SetOfValue」、または「XMLValue」として使用される場合は「XMLSetOfValue」となります。これらの作品は次のとおりです。

```
値の設定::=
        "{" 値リスト "}"
| "{" NamedValueList "}"
| 「{」 「}」

XMLSetOfValue ::=
        XMLValueList
| XMLDelimitedItemList
|空の
```

「ValueList」、「NamedValueList」、および「XMLSetOfValue」の代替は 26.3 で指定されており、代替の選択は「XMLSequenceOfValue」が使用された場合と同じです。 「{」 「}」または「空」の表記は、「SetOfValue」または「XMLSetOfValue」が空のリストの場合に使用されます。

注 1 – 意味上の重要性は、これらの値の順序に優先されるべきではありません。

注 2 – これらの値の順序を保持するためにエンコーディング ルールは必要ありません。

注 3 – set-of タイプは数学的な値のセットではないため、例として、 SET OF INTEGERの場合、値{ 1 }と{ 1 1 }は別個です。

28.4 「SetOfType」の最初の代替が使用される場合、「SetOfValue」の最初の代替が使用されます。 「SetOfValue」の「ValueList」内の各「Value」、および「XMLSetOfValue」の代替内の各「XMLValue」は、「SetOfType」で指定されたタイプでなければなりません。

28.5 「SetOfType」の 2 番目の代替が使用される場合、「SetOfValue」の 2 番目の代替が使用され、「NamedValueList」内の各「NamedValue」シーケンスには、「NamedType」で指定されたタイプの「Value」が含まれなければなりません。 「SetOfType」の。 「NamedValue」の「識別子」は、「SetOfType」の「NamedType」の「識別子」となります。

29 選択肢タイプの表記

29.1 他のタイプから選択タイプ (3.8.14 を参照) を定義するための表記は、「ChoiceType」となります。

```
ChoiceType ::= CHOICE "{" AlternativeTypeLists "}"

代替タイプリスト ::=
        RootAlternativeTypeList
|       RootAlternativeTypeList ","
                ExtensionAndException ExtensionAdditionAlternatives
                オプションの拡張マーカー

RootAlternativeTypeList ::= AlternativeTypeList

ExtensionAdditionAlternatives ::=
        "," ExtensionAdditionAlternativesList
|空の

ExtensionAdditionAlternativesList ::=
        拡張機能追加代替
|       ExtensionAdditionAlternativesList "," ExtensionAdditionAlternative

ExtensionAdditionAlternative ::=
        ExtensionAdditionAlternativesGroup
|名前付きタイプ

ExtensionAdditionAlternativesGroup ::=
        "[[" VersionNumber AlternativeTypeList "]]"

AlternativeTypeList ::=
        名前付きタイプ
|       AlternativeTypeList "," NamedType
```

注 – 「T ::= CHOICE { a A }」とA は同じ型ではなく、エンコード規則によって異なる方法でエンコードされる可能性があります。

29.2自動タグ付けが選択されているモジュールの定義内で「AlternativeTypeLists」生成が発生し (13.3 を参照)、「AlternativeTypeList」内の「NamedType」の出現がいずれもテキストでタグ付けされた型ではない場合 (25.2 を参照)、自動タグ付け変換は「AlternativeTypeLists」全体に対して選択されます。それ以外の場合は選択されません。

29.3 「AlternativeTypeLists」内の「AlternativeTypeList」プロダクションで定義されたタイプには、個別のタグが必要です（31.2 および 52.7 を参照）。自動タグ付けが選択されている場合、タグが個別であるという要件は自動タグ付けが実行された後にのみ適用され、常に満たされます。

29.4 自動タグ付けが有効で、拡張機能ルートの「NamedType」にタグがない場合は、「ExtensionAdditionAlternativesList」内の「NamedType」は、テキストでタグ付けされたタイプでなければなりません。

29.5自動タグ付け変換は、「NamedType」プロダクションに元々あった「Type」を置換用の「TaggedType」に置き換えることにより、「AlternativeTypeLists」の各「NamedType」に影響を与えます。置換「TaggedType」は次のように指定します。

a) 代替の「TaggedType」表記では、代替の「Tag Type」が使用されます。

b) 置換「TaggedType」の「Class」が空である(すなわち、タグ付けはコンテキスト固有である)。

c) 置換「TaggedType」の「ClassNumber」は、「RootAlternativeTypeList」の最初の「NamedType」のタグ値は 0、2 番目の「NamedType」のタグ値は 1、というようにタグ番号が増えていきます。

d) 「ExtensionAdditionAlternativesList」の最初の「NamedType」の置換「TaggedType」の「ClassNumber」は、「RootAlternativeTypeList」の最大の「ClassNumber」より 1 大きく、「ExtensionAdditionAlternativesList」の次の「NamedType」は最初のものより 1 つ大きい「ClassNumber」というように、タグ番号が増加していきます。

e) 置換後の「TaggedType」の「Type」は、置換される元の「Type」です。

注 1 – 置換「TaggedType」の暗黙的タグ付けまたは明示的タグ付けの仕様を管理する規則は、31.2.7 によって提供されます。自動タグ付けは、「タイプ」がタグなしの選択タイプ、タグなしのオープンタイプ表記、またはタグなしの「DummyReference」（Rec. ITU-T X.683 | ISO/IEC 8824-4、8.3 を参照）でない限り、常に暗黙的なタグ付けです。この場合、それは明示的なタグ付けです。

注 2 – 自動タグ付けが適用されると、コンポーネントのタグは完全に決定され、自動タグ付け変換が適用される別の「AlternativeTypeLists」内の代替の定義で選択タイプが参照されている場合でも、変更されません。したがって、次の場合:

```
T ::= CHOICE { a Ta, b Tb, c Tc }
E ::= 選択肢 {f1 E1, f2 T, f3 E3}
```

Eのコンポーネントに適用される自動タグ付けは、 Tのタグ付け環境がどのようなものであっても、Tのコンポーネントa、 b 、およびcに付加されたタグには決して影響しません。Tが自動タグ付け環境で定義され、 Eが自動タグ付け環境にない場合、自動タグ付けは、 Tのコンポーネントa、 b 、およびcに引き続き適用されます。

注 3 – サブタイプは自動タグ付けには影響しません。

注 4 – 自動タグ付けが設定されている場合、拡張挿入ポイント (3.8.35 を参照) 以外の場所に新しい代替を挿入すると、タグ変更の副作用により他の代替が変更される可能性があり、その結果、タグとの相互作用の問題が発生する可能性があります。仕様の古いバージョン。

29.6 「VersionNumber」は 25.1 で定義されており、25.16 で指定されているモジュール全体での「VersionNumber」の一貫した使用に関する制限は、このプロダクション内での「番号」の使用に適用されます。

29.7 「ExtensionAdditionAlternativesList」に追加されるそれぞれの新しい「NamedType」のタグは、標準的に「ExtensionAdditionAlternativesList」内の他の代替タグよりも大きくなければならず（8.6を参照）、「ExtensionAdditionAlternativesList」内の最後の「NamedType」でなければなりません。

29.8選択タイプには、すべてが同じタグを持たない値が含まれています。 (タグは、選択肢タイプに値を提供した代替案に依存します。)

29.9この型に拡張子マーカーがなく、この推奨事項が適用される場所で使用される場合国際標準では、個別のタグを持つ型の使用が要求されており (29.3 を参照)、そのような要件では、選択型の値の考えられるすべてのタグが考慮されます。 「TagDefault」が自動タグではないことを前提とした次の例は、この要件を示しています。

例

```
                A ::= 選択肢 {
1b_             B、
c               ヌル}

                B ::= 選択肢 {
d               [0] NULL、
e               [1] NULL}

2               A ::= 選択肢 {
b               B、
c               C }

                B ::= 選択肢 {
d               [0] NULL、
e               [1] NULL}

                C ::= 選択肢 {
f               [2] NULL、
g               [3] NULL}

3               （正しくない）
                A ::= 選択肢 {
b               B、
c               C }

                B ::= 選択肢 {
d               [0] NULL、
e               [1] NULL}

                C ::= 選択肢 {
f               [0] NULL、
g               [1] NULL}
```

例 1 と 2 は、この表記法の正しい使用法です。例 3 は、タイプdとfのタグ、およびeとgのタグが同一であるため、自動タグ付けなしでは正しくありません。

29.10 「AlternativeTypeLists」内のすべての「NamedType」の「識別子」は、そのリスト内の他の「NamedType」の「識別子」とは異なるものとする。

29.11選択肢タイプの値を定義するための表記は、「ChoiceValue」 、または「XMLValue」として使用される場合は「XMLChoiceValue」となります。これらの作品は次のとおりです。

```
ChoiceValue ::= 識別子":"値

XMLChoiceValue ::= "<" & 識別子 ">" XMLValue "</" & 識別子 ">"
```

29.12 「Value」または「XMLValue」は、「識別子」によって指定される「AlternativeTypeLists」内の型の値の表記法とします。

30 選択タイプの表記

30.1 選択タイプを定義するための表記法 (3.8.66 を参照) は、「SelectionType」とします。

SelectionType ::= 識別子 "<" タイプ

ここで、「Type」は選択タイプを表し、「識別子」はその選択タイプの定義の「AlternativeTypeLists」に表示される「NamedType」の識別子です。

30.2 「タイプ」が制約されたタイプを示す場合、選択は親タイプで実行され、親タイプのサブタイプ制約は無視されます。

30.3 「SelectionType」が「NamedType」として使用される場合、「SelectionType」の「識別子」だけでなく「NamedType」の「識別子」も存在します。

30.4 「SelectionType」が「Type」として使用される場合、「識別子」は保持され、示されるタイプは選択された代替のタイプです。

30.5選択タイプの値の表記は、「SelectionType」によって参照されるタイプの値の表記となります。

31 接頭辞付き型の表記法

31.1 一般

31.1.1プレフィックス付き型 (3.8.78 を参照) は、古い型と同型であるが、異なるタグまたは追加のタグを持ち、異なるまたは追加の関連エンコード命令を持つ可能性がある新しい型です。

31.1.2接頭辞付きタイプは、「TaggedType」または「EncodingPrefixedType」のいずれかです。

31.1.3タグ付き型であるプレフィックス付き型は、主にこの推奨事項 | で使用される場合に使用されます。国際標準では、個別のタグを持つ型の使用が要求されています (25.6 ～ 25.7、27.3、および 29.3 を参照)。モジュール内でAUTOMATIC TAGSの「TagDefault」を使用すると、そのモジュール内に「TaggedType」を明示的に出現させずにこれを実現できます。

注 – プロトコルによって、複数のデータ型の値がいつでも送信される可能性があると判断される場合、受信者が値を正しくデコードできるようにするために個別のタグが必要になる場合があります。

31.1.4 「EncodingPrefixedType」を使用したエンコード命令の割り当ては、関連するエンコード参照によって識別されるエンコードにのみ関連し、型の抽象値には影響しません。

31.1.5接頭辞付きタイプの表記は「PrefixedType」とします。

PrefixedType ::=
タグ付きタイプ
| EncodingPrefixedType

注 – タグ付けをエンコード命令の割り当てとして説明すると、「PrefixedType」の構文の仕様がより単純かつ明確になります。ただし、歴史的には、タグ付けは ASN.1 仕様の初期バージョンで導入されており、型定義の合法性に影響を与える可能性があります。エンコード接頭辞付き型が導入されたときに、タグ付けの概念 (および関連する構文の説明) に対する最小限の変更が行われました。タグ付けは、エンコード命令の割り当てとも構文的に異なります。タグ付けがEXPLICITまたはIMPLICITであるという指定は、タグの終了"]"の後に発生し、通常のエンコードの場合のように"["と"]"のペアには含まれません。説明書。

31.1.6 「PrefixedType」の値の表記は「PrefixedValue」とするか、「XMLValue」として使用する場合は「XMLPrefixedValue」とします。これらの作品は次のとおりです。

PrefixedValue ::= 値

XMLPrefixedValue ::= XMLValue

ここで、「Value」または「XMLValue」は、「PrefixedType」の「TaggedType」または「EncodingPrefixedType」の「Type」の値の表記です。

注 1 – この表記法には、「タグ」も「EncodingPrefix」の一部も表示されません。

注 2 – エンコード命令は、エンコード制御セクションのタイプに割り当てることもできます (54 節を参照)。そんな
代入は、型の値の表記には影響しません。

31.2 タグ付きタイプ

31.2.1タグ付きタイプの表記は「TaggedType」とします。

タグ付きタイプ ::=
タグの種類
|タグの暗黙的なタイプ
|タグの明示的なタイプ

タグ ::= "[" EncodingReference クラス ClassNumber "]"

エンコーディング参照 ::=
エンコード参照「:」
|空の

クラス番号 ::=
番号
|定義された値

クラス ::=
ユニバーサル
|応用
|プライベート
|空の

31.2.2 「タグ」で使用する場合、「エンコーディング参照」はTAG となります。モジュールのデフォルトのエンコーディング参照がTAGでない限り、「Tag」の「EncodingReference」は「空」であってはなりません(13.5 を参照)。

31.2.3 「DefinedValue」の「valuereference」は整数型であり、負でない値が割り当てられます。

31.2.4新しい型は古い型と同型ですが、クラス "Class" と番号 "ClassNumber" のタグがあります。ただし、 "Class" が "empty" の場合、タグはコンテキスト固有のクラスと番号になります。は「クラス番号」です。

31.2.5 「クラス」は、この勧告で定義されているタイプを除き、ユニバーサルであってはなりません |国際標準。

注 1 – ユニバーサル クラス タグの使用は、ITU-T および ISO によって随時合意されています。

注 2 – サブ条項 G.2.12 には、タグ クラスのスタイル上の使用に関するガイダンスとヒントが含まれています。

31.2.6タグの適用はすべて、暗黙的なタグ付けまたは明示的なタグ付けのいずれかです。暗黙的なタグ付けは、オプションを提供するエンコード ルールについて、転送中に「TaggedType」内の「Type」の元のタグを明示的に識別する必要がないことを示します。

注 – これがユニバーサル クラスであった古いタグを保持すると、新しい型の ASN.1 定義を知らなくても古い型を明確に識別できるため便利です。ただし、最小転送オクテットは通常、 IMPLICIT を使用することで達成されます。 IMPLICITを使用したエンコーディングの例は、 Rec. に示されています。 ITU-T X.690 | ISO/IEC 8825-1。

31.2.7次のいずれかが当てはまる場合、タグ付け構造では明示的なタグ付けを指定します。

a) 「Tag EXPLICIT Type」の代替手段が使用されます。

b) 「タグタイプ」の代替が使用され、モジュールの「TagDefault」の値が明示的であるか
TAGSまたは空です。

c) 「タグ タイプ」代替が使用され、モジュールの「TagDefault」の値がIMPLICIT TAGSまたはAUTOMATIC TAGSですが、「Type」で定義されたタイプがタグなしの選択タイプ、タグなしのオープン タイプ、またはタグなしの「 DummyReference」（Rec. ITU-T X.683 | ISO/IEC 8824-4、8.3 を参照）。

それ以外の場合、タグ付け構造は暗黙的なタグ付けを指定します。

31.2.8 「クラス」が「空」の場合、25.6 ～ 25.7、27.3、および 29.3 の個別のタグの要件によって暗示される制限を除き、「タグ」の使用に制限はありません。

31.2.9 「Type」で定義された型が、タグなしの選択型、タグなしのオープン型、またはタグなしの「DummyReference」である場合、 IMPLICIT代替は使用されません (Rec. ITU-T X.683 | ISO/IEC 8824-を参照) 4、8.3）。

31.3 エンコーディングプレフィックス付きタイプ

31.3.1エンコーディング接頭辞付きタイプの表記は「EncodingPrefixedType」とします。

EncodingPrefixedType ::=
エンコーディングプレフィックスタイプ

エンコーディングプレフィックス ::=
"[" エンコーディング参照エンコーディング命令 "]"

「EncodingReference」は31.2.1で定義されています。

31.3.2 「Encodingstruction」プロダクションは勧告 | で指定されています。 「EncodingReference」(付録 E を参照) によって識別される国際標準であり、ASN.1 語彙項目 (コメント、cstring、および空白を含む) の任意のシーケンスで構成できます。

注 1 – 「[」および「]」の語彙項目は「Encodingstruct」には決して現れません。

注 2 – この推奨事項の将来のバージョン |国際標準は、附属書 E にさらにエンコーディング参照を追加する可能性があります。ASN.1 ツールは、「エンコーディング参照」が附属書 E で指定されているもののいずれでもない場合に警告 (のみ) を提供し、「]を使用して「EncodingPrefix」全体を無視することをお勧めします。 "をターミネータとして使用します (上記の注 1 を参照)。

31.3.3 「EncodingReference」が空の場合、エンコード接頭辞のエンコード参照がモジュールのデフォルトのエンコード参照になります。

注 – モジュールのデフォルトのエンコード参照がTAG (31.2.2 を参照) で、「EncodingReference」が「空」の場合、「PrefixedType」は「EncodingPrefixedType」ではなく「TaggedType」になります。

31.3.4一般に、組み合わせて使用できるエンコード命令 (同じエンコード参照を持つ)、および特定の命令または命令の組み合わせを適用できるタイプには制限があります。
これらの制限は、推奨事項で指定されています。エンコーディング参照に関連する国際標準 (付録 E を参照) であり、この勧告では指定されていません。国際標準。

32 オブジェクト識別子のタイプの表記

32.1 オブジェクト識別子のタイプ (3.8.54 を参照) は、「ObjectIdentifierType」という表記によって参照されます。

オブジェクト識別子タイプ ::=
オブジェクト識別子

32.2 このタイプにはユニバーサルクラスのタグ番号6が付いています。

32.3 オブジェクト識別子の値の表記は「ObjectIdentifierValue」とするか、「XMLValue」として使用する場合は、「XMLObjectIdentifierValue」。これらの作品は次のとおりです。

オブジェクト識別子値 ::=
"{" ObjIdComponentsList "}"
| "{" DefinedValue ObjIdComponentsList "}"

ObjIdComponentsList ::=
ObjIdコンポーネント
| ObjIdComponentsObjIdComponentsList

ObjIdコンポーネント ::=
お名前フォーム
|ナンバーフォーム
|名前と番号フォーム
|定義された値

名前フォーム ::= 識別子

NumberForm ::= 数値 |定義された値

名前と番号フォーム ::=
識別子"(" NumberForm ")"

XMLObjectIdentifierValue ::=
XMLObjIdComponentList

XMLObjIdComponentList ::=
XMLObjIdコンポーネント
| XMLObjIdComponent & "." & XMLObjIdComponentList

XMLObjIdコンポーネント ::=
お名前フォーム
| XML番号フォーム
| XML名前と番号フォーム

XMLNumberForm ::= 数値

XML 名と番号フォーム ::=
識別子 & "(" & XMLNumberForm & ")"

32.4 「NumberForm」の「DefinedValue」の「valuereference」は整数型であり、負でない値が割り当てられます。

32.5 「ObjectIdentifierValue」の「DefinedValue」の「valuereference」はオブジェクト識別子型でなければなりません。

32.6 「ObjIdComponents」の「DefinedValue」は、相対オブジェクト識別子のタイプであり、オブジェクト識別子ツリーのある開始ノードからオブジェクト識別子ツリーの後のノードまでのアークの順序付きセットを識別する必要があります。
開始ノードは前の「ObjIdComponents」によって識別され、後の「ObjIdComponents」(存在する場合)によって後のノードからのアークが識別されます。開始ノードは、ルートでも、ルートの直下のノードでもありません。

注 – オブジェクトを明確に識別するには、相対オブジェクト識別子の値を特定のオブジェクト識別子の値に関連付ける必要があります。オブジェクト識別子の値には、少なくとも 2 つのコンポーネントが必要です (32.11 を参照)。このため、開始ノードに制限があります。

32.7 「NameForm」は、数値と識別子が Rec. で指定されているオブジェクト識別子コンポーネントにのみ使用されます。 ITU-T X.660 | ISO/IEC 9834-1、付録 A (この勧告の付録 F も参照 | 国際規格)、Rec. 9834-1 に指定された識別子の 1 つでなければなりません。 ITU-T X.660 | ISO/IEC 9834-1、付録 A ～ C。

注 – 「NameForm」が許可されている場合、状況によっては代わりに「NameAndNumberForm」の使用が Rec によって推奨されています。 ITU-T X.660 | ISO/IEC 9834-1、A.1.2。

32.8どこで Rec. ITU-T X.660 | ISO/IEC 9834-1 は同義語識別子を指定しており、同義語は Rec. に従って同義語が登録されたときに確立された条件の下で使用できます。 ITU-T X.660 | ISO/IEC 9834-1。
ここで、同じ名前は両方とも Rec で指定された識別子です。 ITU-T X.660 | ISO/IEC 9834-1 および「NameForm」を含むモジュール内の ASN.1 値参照では、オブジェクト識別子の値内の名前は Rec として扱われます。 ITU-T X.660 | ISO/IEC 9834-1 識別子。

32.9 「NumberForm」および「XMLNumberForm」の「数値」は、
オブジェクト識別子コンポーネント。

32.10 3 つのトップレベル アークの下の「NameAndNumberForm」および「XMLNameAndNumberForm」で使用できる「識別子」には柔軟性があります。これらの識別子はエンコードには含まれず、時間の経過とともに変更される可能性があります。これは、組織の名前が変更される可能性があることを認識しているためです。アークの識別子は、通常、アーク上のノードを担当する登録局と、後続のアークに対する責任が割り当てられている登録局との間で合意される必要があります。

注 – 3 つのトップレベルアークの下にあるアークを担当する登録局は、Rec. で特定されています。 ITU-T X.660 | ISO/IEC 9834-1。

32.11オブジェクト識別子の値に関連付けられたセマンティクスは、Rec で指定されます。 ITU-T X.660 | ISO/IEC 9834-1。

注 – 記録ITU-T X.660 | ISO/IEC 9834-1 では、オブジェクト識別子の値に少なくとも 2 つの円弧が含まれる必要があります。

32.12オブジェクト識別子コンポーネントの重要な部分は、オブジェクト識別子コンポーネントの数値を提供する「NameForm」または「NumberForm」または「XMLNumberForm」です。
Rec.で指定された円弧を除きます。 ITU-T X.660 | ISO/IEC 9834-1、附属書 A から C (この勧告の附属書 F も参照 | 国際規格) では、オブジェクト識別子コンポーネントの数値は常にオブジェクト識別子値表記のインスタンスに存在します。

32.13 「ObjectIdentifierValue」にオブジェクト識別子の値の「DefinedValue」が含まれる場合、それが参照するオブジェクト識別子コンポーネントのリストが、値内に明示的に存在するコンポーネントの前に付加されます。

注 – 記録ITU-T X.660 | ISO/IEC 9834-1 では、オブジェクトを識別するためにオブジェクト識別子の値が割り当てられるときは常に、オブジェクト記述子の値も割り当てられることを推奨しています。

例

Rec. で指定されているように割り当てられた識別子を使用します。 ITU-T X.660 | ISO/IEC 9834-1、値:

{ ISO 標準 8571 アプリケーションコンテキスト (1) }
そして

{ 1 0 8571 1 }

それぞれは、 ISO 8571 で定義されているオブジェクト、アプリケーション コンテキストを識別します。

iso.standard.8571.アプリケーションコンテキスト(1)

そして

1.0.8571.1

「XMLObjectIdentifierValue」内。

次の追加定義を使用します。

ftam オブジェクト識別子 ::= { ISO 標準 8571 }

次の値は上記の値と同等です。

{ftamアプリケーションコンテキスト(1)}

33 相対オブジェクト識別子のタイプの表記

33.1 相対オブジェクト識別子のタイプ (3.8.64 を参照) は、「RelativeOIDType」という表記によって参照されます。

RelativeOIDType ::= RELATIVE-OID

33.2 このタイプにはユニバーサルクラスのタグ番号13が付いています。

相対オブジェクト識別子の値の表記は「RelativeOIDValue」、または「XMLValue」として使用される場合は、 33.3 「XMLRelativeOIDValue」となります。これらの作品は次のとおりです。

相対OID値::=
"{" RelativeOIDComponentsList "}"

RelativeOIDComponentsList ::=
相対OIDコンポーネント
| RelativeOIDComponents RelativeOIDComponentsList

相対OIDコンポーネント ::=
ナンバーフォーム
|名前と番号フォーム
|定義された値

XMLRelativeOID値 ::=
XMLRelativeOIDComponentList

XMLRelativeOIDComponentList ::=
XMLRelativeOIDコンポーネント
| XMLRelativeOIDComponent & "." & XMLRelativeOIDComponentList

XMLRelativeOIDコンポーネント ::=
XML番号フォーム
| XML名前と番号フォーム

33.4プロダクションの「NumberForm」、「XMLNameAndNumberForm」、 "名前と番号フォーム", "XML数値フォーム", およびそれらのセマンティクスは、サブ条項 32.3 から 32.12 で定義されています。

の開 「RelativeOIDComponents」の「DefinedValue」は、相対オブジェクト識別子タイプであるものとし、オブジェクト識別子ツリー始ノードからオブジェクト識別子ツリーの後のノードまでのアークの順序付きセットを識別するものとする。開始ノードは、前の「RelativeOIDComponents」(存在する場合)によって識別され、後の「RelativeOIDComponents」(存在する場合)は、後のノードからのアークを識別します。

33.6最初の「RelativeOIDComponents」または「XMLRelativeOIDComponent」は、オブジェクト識別子ツリーの開始ノードからオブジェクト識別子ツリーの後のノードまでの 1 つ以上のアークを識別します。出発点として考えられるのは、
型定義に関連付けられたコメントによって定義されます。タイプ定義に関連付けられたコメント内に開始ノードの定義がない場合は、通信のインスタンスでオブジェクト識別子の値として送信する必要があります (G.2.21 を参照)。開始ノードは、ルートでも、ルートの直下のノードでもありません。

注 – オブジェクトを明確に識別するには、相対オブジェクト識別子の値を特定のオブジェクト識別子の値に関連付ける必要があります。オブジェクト識別子の値には、少なくとも 2 つのコンポーネントが必要です (32.11 を参照)。このため、開始ノードに制限があります。

例

次の定義を使用します。

この大学オブジェクト識別子 ::=
{joint-iso-itu-t 例(999) 大学(56) この大学(32)}

firstgroup RELATIVE-OID ::= {science-fac(4) maths-dept(3)}

または XML 値表記では次のようになります。

この大学 ::= <OBJECT_IDENTIFIER>2.999.56.32</OBJECT_IDENTIFIER>

最初のグループ ::= <RELATIVE_OID>4.3</RELATIVE_OID>

相対オブジェクト識別子:

relOID RELATIVE-OID ::= {最初のグループの部屋(4) ソケット(6)}

または XML 値表記では次のようになります。

relOID ::= <RELATIVE_OID>4.3.4.6</RELATIVE_OID>

現在のルート (アプリケーションによって既知であるか、アプリケーションによって送信される) がthisUniversityである場合、 OBJECT IDENTIFIER値{2 999 56 32 4 3 4 6}の代わりに使用できます。

34 OID 国際化リソース識別子タイプの表記

34.1 OID 国際化リソース識別子タイプ (3.8.47 を参照) は、「IRIType」という表記によって参照されます。

IRIタイプ ::= OID-IRI

34.2 このタイプにはユニバーサルクラスのタグ番号35が付いています。

34.3 OID 国際化リソース識別子の値の表記は、「IRIValue」であるか、または「XML値」、「XMLIRI値」。これらの作品は次のとおりです。

```
IRI値::=
        「」
        最初のアーク識別子
        後続のアーク識別子
        「」

FirstArcIdentifier ::=
        "/" ArcIdentifier

後続のArcIdentifier ::=
        "/" ArcIdentifier 後続の ArcIdentifier
|空の

ArcIdentifier ::=
        整数Unicodeラベル
|       非整数Unicodeラベル

XMLIRI値 ::=
        最初のアーク識別子
        後続のアーク識別子
```

34.4 「FirstArcIdentifier」は、OID ツリーのルートからのアーク (おそらく長いアーク) を識別します。

34.5 各「SubsequentArcIdentifier」は、前の「ArcIdentifier」からアークを識別します。

例

Rec. で指定されているように割り当てられた識別子を使用します。 ITU-T X.660 | ISO/IEC 9834-1 および ISO/IEC 19785 により識別されるオブジェクト:

{iso 登録機関 cbeff (19785) 組織 (0) jtc1-sc37(257) パトロン形式 (1) tlv エンコード (5)}

または XML 値表記では次のようになります。

<OBJECT_IDENTIFIER>1.1.19785.0.257.1.5</OBJECT_IDENTIFIER>

これは、TLV でエンコードされた CBEFF Patron フォーマットを識別しますが、ASN.1 OID-IRI 識別子も持つ可能性があります。

「/ISO/Registration_Authority/19785.CBEFF/Organizations/JTC1-SC37/Patron-formats/TLV-encoded」

または、XML 値表記では次のようになります。

<OID-IRI>/ISO/Registration_Authority/19785.CBEFF/Organizations/JTC1-SC37/Patron-formats/TLV エンコード</OID-IRI>

35 相対 OID 国際化リソース識別子タイプの表記

35.1相対 OID 国際化リソース識別子タイプ (3.8.62 を参照) は、「RelativeIRIType」という表記によって参照されます。

RelativeIRIType ::= RELATIVE-OID-IRI

35.2 このタイプにはユニバーサル クラスのタグ番号 36 が付いています。

相対 OID 国際化リソース識別子の値の表記は「RelativeIRIValue」でなければなりません、または「XMLValue」、「XMLRelativeIRIValue」として使用する場合は35.3 。これらの作品は次のとおりです。

相対IRI値 ::=
「」
FirstRelativeArcIdentifier
後続のアーク識別子
「」

FirstRelativeArcIdentifier ::=
アーク識別子

XMLRelativeIRI値 ::=
FirstRelativeArcIdentifier
後続のアーク識別子

35.4 「FirstRelativeArcIdentifier」は、オブジェクト識別子ツリー内の開始ノードからオブジェクト識別子ツリー内のその後のノードまでのアークを識別するものとする。開始点は、型定義に関連付けられたコメントによって定義できます。タイプ定義に関連付けられたコメント内に開始ノードの定義がない場合は、通信のインスタンスで OID 国際化リソース識別子の値として送信する必要があります。

注 – リソースを明確に識別するには、相対 OID 国際化リソース識別子の値を特定の OID 国際化参照識別子の値に関連付ける必要があります。

例

以下の特定のノードを使用します。

cbeffPatronFormats OID-IRI ::=

「/ISO/Registration_Authority/19785.CBEFF/Patron-formats」

相対 OID 国際化リソース識別子:

tlv エンコードされた RELATIVE-OID-IRI ::= "TLV エンコードされた"

TLV でエンコードされたパトロン フォーマットを識別します。

36 embedded-pdv タイプの表記法

36.1 embedded-pdv タイプ (3.8.24 を参照) は、「EmbeddedPDVType」という表記によって参照されます。

EmbeddedPDVType ::=埋め込み PDV

注 – 「埋め込み PDV」という用語は、メッセージに埋め込まれている、おそらく異なる抽象構文 (本質的には、別個の識別されたプロトコルで定義されたメッセージの値とエンコード) からの抽象値を意味します。歴史的には、OSI プレゼンテーション層での使用から「埋め込みプレゼンテーション データ値」を意味していましたが、この拡張は現在では使用されていないため、「埋め込み値」として解釈する必要があります。

36.2 このタイプにはユニバーサルクラスのタグ番号11が付いています。

36.3 タイプは以下を表す値で構成されます。

a) ASN.1 タイプの値であってもよいが、必ずしもそうである必要はない、単一のデータ値のエンコーディング。そして

b) 以下のものを（個別にまたは一緒に）特定します。

1) 抽象構文。そして

2) 転送構文。

注 1 – データ値は、ASN.1 タイプの値である場合もあれば、静止画像や動画のエンコーディングである場合もあります。この識別は、1 つまたは 2 つのオブジェクト識別子で構成されるか、(OSI 環境では) 抽象構文と転送構文を指定する OSI プレゼンテーション コンテキスト識別子を参照します。

注 2 – 抽象構文および/またはエンコーディングの識別は、アプリケーション設計者によって固定値として決定されることもあります。その場合、通信のインスタンスではエンコードされません。

36.4embedded -pdv タイプには、関連付けられたタイプがあります。この関連タイプは、embedded-pdv タイプの値とサブタイプの表記をサポートするために使用されます。

36.5 自動タグ付け環境を想定した、値の定義とサブタイプに関連付けられた型は次のとおりです (規範的なコメント付き)。

```
順序 {
識別構文の概要                                選択 {
                                                順序 {
                                                    オブジェクト識別子、
                移行                                オブジェクト識別子 }
                -- 抽象化および転送構文のオブジェクト識別子--、

                構文                                オブジェクト識別子
                -- 要約を識別するための単一のオブジェクト識別子
                -- および転送構文--,

                プレゼンテーションコンテキストID          整数
                -- (OSI 環境にのみ適用)
                -- ネゴシエートされた OSI プレゼンテーション コンテキストによって、
                -- 抽象構文と転送構文 --、

                コンテキストネゴシエーション              順序 {
                プレゼンテーションコンテキストID          整数、
                転送構文                            オブジェクト識別子 }
                -- (OSI 環境にのみ適用)
                -- コンテキスト ネゴシエーションが進行中、プレゼンテーション コンテキスト ID
                -- 抽象構文のみを識別します
                -- したがって、転送構文を指定する必要があります --,

                transfer-syntax                     オブジェクト識別子
                -- 値のタイプ (たとえば、値が
                -- ASN.1 タイプの値)
                -- アプリケーション設計者によって修正されます (したがって、両方の設計者に知られています)
                -- 送信者と受信者) 。これ
                -- ケースは主にサポートを目的として提供されています
                -- 選択フィールド暗号化 (または他のエンコーディング)
                -- 変換) ASN.1 タイプ --、

                固定--                              ヌル
                データ値は固定 ASN.1 タイプの値です (したがって、
                -- 両方の送信者に知られている
                -- および受信者) -- },

            データ値記述子                          オブジェクト記述子は省略可能
                -- これにより、人間が判読できるクラスの識別が提供されます。
                - 価値    --、
データ値                                        オクテット文字列 }

( コンポーネントあり {
        ...、
        データ値記述子 ABSENT } )
```

注 –embedded-pdv タイプでは、 data-value-descriptor値を含めることはできません。ただし、ここで提供される関連型の定義は、embedded-pdv 型、外部型、および無制限の文字列型の間に存在する共通点の基礎となります。

子で 代替プレゼンテーションコンテキスト ID は、整数36.6 の値が OSI 定義コンテキスト セット内の OSI プレゼンテーション コンテキスト識別ある場合に、OSI 環境でのみ適用されます。この代替手段は、OSI コンテキストのネゴシエーション中には使用されません。

36.7コンテキストネゴシエーションの代替手段は OSI 環境でのみ適用され、OSI コンテキスト ネゴシエーション中にのみ使用されます。整数値は、OSI 定義コンテキスト セットに追加するために提案された OSI プレゼンテーション コンテキスト識別子でなければなりません。オブジェクト識別子の転送構文は、値をエンコードするために使用される OSI プレゼンテーション コンテキストに対して提案された転送構文を識別します。

36.8embedded -pdv 型の値の表記法は、36.5 で定義された関連する型の値表記法でなければなりません。ここで、 OCTET STRING型のデータ値コンポーネントの値は、識別で指定された転送構文を使用したエンコーディングを表します。

```
EmbeddedPDVValue ::= SequenceValue

XMLEmbeddedPDVValue ::= XMLSequenceValue
```

例 – 構文の使用など、単一のオプションを強制する場合は、次のように記述します。

```
埋め込み PDV (コンポーネント { 付き)
        …、
        識別 (コンポーネント {
                構文 PRESENT } ) } )
```

37 外部型の表記

37.1 外部型 (3.8.43 を参照) は、「ExternalType」という表記によって参照されます。

```
外部タイプ ::=外部
```

37.2 このタイプにはユニバーサルクラスのタグ番号8が付いています。

37.3 タイプは以下を表す値で構成されます。

a) ASN.1 タイプの値であってもよいが、必ずしもそうである必要はない、単一のデータ値のエンコーディング。そして

b) 以下の識別:

1) 抽象構文。そして

2) 転送構文。そして

c) (オプションで) データ値のカテゴリーについて人間が判読できる説明を提供するオブジェクト記述子。オプションのオブジェクト記述子は、「ExternalType」表記の使用に関連するコメントによって明示的に許可されない限り、存在してはならない。

注 – 36.3 の注 1 は外部タイプにも適用されます。

外部 外部型には関連付けられた型があります。この型は、抽象の定義に精度を与えるために使用されます。
型の37.4 の値であり、外部型の値とサブタイプの表記をサポートするためにも使用されます。

注 – エンコーディング ルールは、エンコーディングを導出するために使用される別のタイプを定義したり、関連するタイプを参照せずにエンコーディングを指定したりする場合があります。たとえば、BER のエンコーディングでは、歴史的な理由から異なるシーケンス タイプが使用されます。

37.5自動タグ付け環境を想定した、値の定義とサブタイプに関連付けられた型は次のとおりです (規範的なコメント付き)。

```
順序 {
識別                                    選択 {
        構文抽象転                              順序 {
                送                                      オブジェクト識別子、
                                                        オブジェクト識別子 }
        -- 抽象化および転送構文のオブジェクト識別子 --、

        構文                                    オブジェクト識別子
        -- 要約を識別するための単一のオブジェクト識別子
        -- および転送構文 --,

        プレゼンテーションコンテキストID        整数
        -- (OSI 環境にのみ適用)
        -- ネゴシエートされた OSI プレゼンテーション コンテキストによって、
        -- 抽象構文と転送構文 --、

        コンテキストネゴシエーシ                順序 {
        ョンプレゼンテーションコンテキ          整数、
        ストID転送構文                          オブジェクト識別子 }
        -- (OSI 環境にのみ適用)
        -- コンテキスト ネゴシエーションが進行中、プレゼンテーション コンテキスト ID
        -- 抽象構文のみを識別します
        -- したがって、転送構文を指定する必要があります --,

        転送構文                                オブジェクト識別子
        -- 値のタイプ (たとえば、値が
```

```
                -- ASN.1 タイプの値)
                -- アプリケーション設計者によって修正されます (したがって、両方の設計者に知られています)
                -- 送信者と受信者）。これ
                -- ケースは主にサポートを目的として提供されています
                -- 選択フィールド暗号化 (または他のエンコーディング)
                -- 変換) ASN.1 タイプ --、
                修理済み                                ヌル
                -- データ値は固定 ASN.1 タイプの値です (したがって、
                -- 両方の送信者に知られている
                -- および受信者) -- },
データ値記述子                          オブジェクト記述子は省略可能
                -- これにより、人間が判読できるクラスの識別が提供されます。
                - 値             --、
データ値                                オクテット文字列 }
( コンポーネントあり {
        ...、
        識別 (コンポーネント {
                ...、
                構文転送                                不在、
                構文が修正されま                        不在、
                した                    不在 } ） } ）
```

注 – 歴史的な理由により、外部型では構文、転送構文、または識別の固定代替手段は使用できません。これらのオプションを必要とするアプリケーション設計者は、embedded-pdv タイプを使用する必要があります。ここで提供される関連型の定義は、外部型、無制限文字列型、およびembedded-pdv型の間に存在する共通点の基礎となります。

37.6 36.6 および 36.7 のテキストは外部タイプにも適用されます。

37.7外部型の値の表記は、37.5 で定義された関連型の値表記でなければなりません。ここで、 OCTET STRING型のデータ値コンポーネントの値は、識別で指定された転送構文を使用したエンコーディングを表します。

```
外部値 ::= シーケンス値

XML外部値 ::= XMLシーケンス値
```

注 – 歴史的な理由により、エンコード ルールは、エンコードが 8 ビットの正確な倍数ではないEXTERNALに埋め込まれた値を転送できます。このような値は、上記の関連付けられた型を使用した値表記では表現できません。

38 時間タイプ

38.1 一般

38.1.1時間タイプ (3.8.83 を参照) は、「TimeType」という表記によって参照されます。

```
時間タイプ ::= TIME
```

38.1.2この表記法で定義された型のタグはユニバーサル クラス、番号 14 です。

38.1.3時間型の値は、「TimeValue」という表記によって定義されるか、「XMLValue」として使用される場合は「XMLTimeValue」という表記によって定義されます。これらの表記の構文は、ISO 8601、3.4 で定義された表記を使用して、「単純文字列」の内容として 38.3 で定義されています。

38.2 時間プロパティと時間抽象値の設定

38.2.1表 6 の列 1 には、時間抽象値の時間プロパティの説明と名前が指定されています。列 2 では、列 1 の時間プロパティに対して可能な時間プロパティ設定の名前を指定します。列 3 は、時間プロパティが適用され、対応する時間プロパティ設定を持つ抽象値を (通常は ISO 8601 を参照して) 指定します。

注 1 – ASN.1 は、ISO 8601 表現でサポートされていない抽象値を指定しません。

注 2 – 時間プロパティとその設定の名前は、プロパティ設定サブタイプ表記のプロパティ アサーションに表示されます (51 節を参照)。

注 3 – ASN.1 は、TIME 抽象値が同じプロパティを持ち、それらのプロパティの同じ設定がある場合、それらの値の間の順序関係を認識します。時間差を含む抽象値については、同じ時間差を持つ抽象値間でのみ順序関係が認められる。

表 6 – 時間抽象値のプロパティと設定

時間プロパティ	プロパティ設定の名前	このプロパティ設定を持つ抽象値
抽象値の基本的な性質 名前:基本 コメント:このプロパティの設定により、抽象値の基本的な性質が識別されます。常に抽象値にはこの特性があります。	日付	ISO 8601、4.1を参照してください。 日付のみであるすべての抽象値。
	時間	ISO 8601、4.2を参照してください。 時刻のみを示すすべての抽象値。
	日付時刻	ISO 8601、4.3を参照してください。 日付と時刻であるすべての抽象値。
	間隔	ISO 8601、4.4を参照してください。 すべての時間間隔の抽象値。
	記録間隔	ISO 8601、4.5を参照してください。 すべての繰り返し間隔の抽象値。
日付の時間スケールと精度 名前:日付 コメント:これは、日付の識別を含む抽象値にのみ適用されます。それは時間を特定します その日付のスケールと正確さ。 注 – 複数の日付を識別する抽象値 (間隔など) には、両方の日付に適用される単一の日付設定があります。	C (センチュリー)	ISO 8601、4.1.2.3 c) を参照してください。 世紀のみを表す日付を含むすべての抽象値。
	Y (年のみ)	ISO 8601、4.1.2.3 b) を参照してください。 年のみを表す日付を含むすべての抽象値。
	YM (年-月)	ISO 8601、4.1.2.3 a) を参照してください。 年月のタイムスケールを使用する日付を含むすべての抽象値。
	YMD (年-月-日)	ISO 8601、4.1.2.2を参照してください。 年-月-日のタイム スケールを使用する日付を含むすべての抽象値。
	YD (年-日)	ISO 8601、4.1.3.2を参照してください。 年-日のタイム スケールを使用する日付を含むすべての抽象値。
	YW (年-週)	ISO 8601、4.1.4.3 を参照してください。 年週のタイム スケールを使用する日付を含むすべての抽象値。
	YWD (年-週-日)	ISO 8601、4.1.4.2を参照してください。 年-週-日のタイム スケールを使用する日付を含むすべての抽象値。

表 6 – 時間抽象値のプロパティと設定

時間プロパティ	プロパティ設定の名前	このプロパティ設定を持つ抽象値
関連する年のタイプ 名前:年 コメント:これは、1 年以上の年または世紀の識別を含む抽象的な値にのみ適用されます。その設定は、年 (または世紀) の識別が「通常の」年であるか、予兆グレゴリオ暦 (J.2.2 を参照) の年であるか、負の年であるか、またはそれを表すために 4 桁以上を必要とする年であるかを識別します。 。 注 – 複数の年に関係する抽象値 (たとえば、間隔) には、両方の年に適用される単一の年の設定があります。	基本	1582 ～ 9999 の範囲の年 (または 15 ～ 99 の範囲の世紀) を含むすべての抽象値。
	予防薬	0 ～ 1581 の範囲の年 (または 00 ～ 14 の範囲の世紀) を含むすべての抽象値。 注 – 予兆グレゴリオ暦では、年の値 0 は紀元前 1 年にほぼ対応する意味を持ちます (J.2.2 を参照)。
	ネガティブ	–9999 ～ –0001 の範囲の年 (または –99 ～ –01 の範囲の世紀) を含むすべての抽象値。
	L5、 L6、 L7•••無限大（大）	正または負に関係なく、10 進表現に 5、6、7 などの桁が必要な年 (または 10 進表現に 3、4、5 桁などが必要な世紀) をそれぞれ含むすべての抽象値。
一時的な精度 名前:時間 コメント:これは、時刻の識別を含む抽象値にのみ適用されます。それはその時刻の正確さを識別します。 注 – 複数の時刻を識別する抽象値 (たとえば、間隔) には、両方の時刻に適用される単一の時刻設定があります。	H （時間）	ISO 8601、4.2.2.3 b) を参照してください。 時間単位の精度で時刻を含むすべての抽象値。
	HM (時-分)	ISO 8601、4.2.2.3 a) を参照してください。 分単位の精度で時刻を含むすべての抽象値。
	HMS (時-分-秒)	ISO 8601、4.2.2.2を参照してください。 秒単位の精度で時刻を含むすべての抽象値。
	HF1、 HF2、 HF3など、無限大まで (時間の小数部)	ISO 8601、4.2.2.4 c) を参照してください。 小数点以下 1、2、3 などの時間単位の精度で時刻を含むすべての抽象値。
	HMF1、 HMF2、 HMF3など、無限大 (時分数)	ISO 8601、4.2.2.4 b) を参照してください。 小数点以下 1、2、3 などの分単位の精度で時刻を含むすべての抽象値。
	HMSF1、 HMSF2、 HMSF3など無限大まで 時分秒の小数部	ISO 8601、4.2.2.4 a) を参照してください。 小数点以下 1、2、3 などの秒単位の精度で時刻を含むすべての抽象値。

表 6 – 時間抽象値のプロパティと設定

時間プロパティ	プロパティ設定の名前	このプロパティ設定を持つ抽象値
時間のローカルまたは UTC タイムスケール 名前:ローカルまたは UTC コメント:これは、時刻の識別を含む抽象値にのみ適用されます。これは、その時刻のタイム スケール (現地時間、UTC、または現地時間に UTC との差を加えたもの) を識別します。時差は地方自治体によって決定されます。 ASN.1 は、-15 時間から +15 時間の範囲の時差をサポートします。現地時刻が UTC 以上である場合、差は正になります (ISO 8601、4.2.5.1 を参照)。 J.2.11も参照してください。 注 – 複数の時間を識別する抽象値 (間隔など) には、両方の時間に適用されるLocal-or-UTCの単一の設定があります。	L (現地時間のみ)	38.2.2 および ISO 8601、4.2.2 および 4.2.3 を参照してください。 ローカル時刻のみを指定する時刻を含むすべての抽象値。
	Z (UTC のみ)	ISO 8601、4.2.4を参照してください。 ローカル時刻ではなく UTC を指定する時刻を含むすべての抽象値。
	LD (現地時刻および UTC との差)	ISO 8601、4.2.5を参照してください。 現地時刻を指定する時刻と、現地時刻を取得するために UTC に追加される時刻 (負の場合もある) を含むすべての抽象値。
間隔指定の形式 名称 :インターバル型 コメント:これは、間隔または繰り返し間隔である抽象値にのみ適用されます。これは、間隔指定の形式 (開始点と終了点、期間、開始点と期間、または終了点のある期間) を識別します。	SE (始点と終点)	ISO 8601、4.4.1 a) を参照してください。 開始点と終了点を使用して間隔を指定するすべての抽象値。
	D (持続時間のみ)	ISO 8601、4.4.1 b) および 4.4.3 を参照してください。 期間のみを使用して間隔を指定するすべての抽象値。
	SD (開始点と継続時間)	ISO 8601、4.4.1 c)を参照してください。 開始点と期間を使用して間隔を指定するすべての抽象値。
	DE (期間と終了点)	ISO 8601、4.4.1 d) を参照してください。 期間と終了点を使用して間隔を指定するすべての抽象値。
開始点および/または終了点の指定の性質 名前： SEポイント コメント:これは、開始点または終了点、またはその両方を使用する間隔または繰り返し間隔にのみ適用されます。 このプロパティの設定は、この抽象値の一部を形成する開始点および/または終了点の性質を識別します。 注 – 開始点と終了点の両方を持つすべての間隔抽象値には、このプロパティと、日付または時刻に関連する関連プロパティに対する単一の設定があります。 開始点と終了点の形式が異なる間隔抽象値はありません。したがって、間隔の開始点と間隔の終了点の両方を持つすべての抽象値は、開始点と終了点の同じ時間コンポーネントのセットを持ちます (ただし、終了点の値の表記については表 7 を参照してください)。これがISO 8601との違いです。	日付	ISO 8601、4.1を参照してください。 日付のみを使用して開始点および/または終了点を指定するすべての抽象値。
	時間	ISO 8601、4.2を参照してください。 時刻のみを使用して開始点および/または終了点を指定するすべての抽象値。
	日付時刻	ISO 8601、4.3を参照してください。 日付と時刻を使用して開始点および/または終了点を指定するすべての抽象値。
繰り返しの仕様 名前:再発 コメント:これは、定期的な間隔である抽象値にのみ適用されます。これは、繰り返しの回数に関する合意された制限 (または無制限) を識別します。	無制限(繰り返し回数に制限はありません。繰り返し回数は空の文字列で表されます)	ISO 8601、4.5を参照してください。 間隔の無制限の繰り返しを表すすべての抽象値。
	R1、 R2、 R3など、無限大まで (繰り返し桁数)	ISO 8601、4.5を参照してください。 間隔の繰り返しを表すすべての抽象値。繰り返しの回数を表すためにそれぞれ 1、2、3 などの桁が必要です。

表 6 – 時間抽象値のプロパティと設定

時間プロパティ	プロパティ設定の名前	このプロパティ設定を持つ抽象値
真夜中の一日の始まりまたは終わり 名前：ミッドナイト コメント:これは、午前 0 時を表す時刻を含む抽象値にのみ適用されます。この午前 0 時の値が 1 日の始まり (通常 00:00:00 として表される) であるか、または 1 日の終わり (通常 24:00:00 として表される) であるかを識別します。	スタート（一日の始まり）	ISO 8601、4.2.3 a) を参照してください。 1 日の始まりの午前 0 時を表す時刻を含む抽象値。
	終了（一日の終わり）	ISO 8601、4.2.3 b) を参照してください。 1 日の終わりの午前 0 時を表す時刻を含む抽象値。
注 – ASN.1 では、開始点と終了点の両方の構文を制御するSE ポイント設定が 1 つだけであるため、異なる時間プロパティを持つ間隔の開始点と終了点の使用はサポートされていません。開始点と終了点は同じ時間形式を使用する必要があります。これがISO 8601との違いです。		

38.2.2 ISO 8601 は、午前 0 時を表す 2 つの基本的な表現を提供しています。1 日の終わりの午前 0 時を表す「2400」と、一日の始まりの午前 0 時を表す「0000」です (秒または秒の小数部分はゼロ桁のみを含みます)。 。これらは、単一の抽象値の異なる表現とは見なされず、別個の抽象値として見なされます。

注 1 – これは、独立した時間として明確に区別され、1 日の始まりと終わりを表すためです。日と組み合わせて使用する場合、時間軸上でまったく同じ位置にあるにもかかわらず、x 日の「2400」は x+1 日の「0000」よりも小さいと見なされます。

注 2 – それぞれ、時間プロパティ設定「Midnight=End」と「Midnight=Start」があります。

注 3 – 他の時間と同様に、秒と秒の小数部分の精度に応じて、特定の日の開始時と終了時の午前 0 時を表す、無限に多くの個別の抽象値が存在します。さらに、秒ではなく時間または分の端数の使用に基づいた真夜中の抽象値の無限のセットもあります。 (抽象値が午前 0 時の値の場合、これらの小数部はすべて 0 からさまざまな精度になります。)

38.2.3 ISO 8601 は、時間間隔および繰り返し時間間隔の構成要素として、期間の 2 つの基本的な表現 (週、または年、月、日、時、分、秒の組み合わせ) を提供しています。 ISO 8601 で期間を表す異なる文字列は、ASN.1 では異なる抽象値を表すものとみなされます。 ただし、唯一の違いが、表現される期間 (期間の精度を含む) を変更しないゼロ時間コンポーネントの省略または包含である場合を除きます。 。ゼロ時間コンポーネントの包含または省略は、正規のエンコード規則および Rec. のすべてのエンコード規則で完全に指定されています。 ITU-T X.691 | ISO/IEC 8825-2。期間に関連付けられた時間プロパティ ("Basic=Interval Interval-type=D"以外) はありませんが、期間の時間コンポーネントに制限を適用して、要素が存在しないことを要求したり、その値を制限したりすることができます (38.4 を参照)。 4)。

注 1 – ISO 8601 には、コンポーネント (特に小数部分) のサイズに関する事前の合意に関する要件があります。
これは通常、さまざまな精度のプロパティ設定によって処理されます。ただし、 DURATIONの場合、簡単にするために、コンポーネントの精度を決定するためのプロパティ設定は導入されませんでした。代わりに、38.4.4 で指定されているように、同等のシーケンス型に対する内部サブタイプ制約を適用して、DURATION のコンポーネントに関する事前の合意を記録できます。

注 2 – ISO 8601 では、週コンポーネントの使用を、他の日付コンポーネント (年、月、日) の使用や、時間、分、または秒の時間コンポーネントの使用と組み合わせてはいけないと規定しています。この制限は、ISO 8601 との一貫性を保つために ASN.1 にも適用されます。

38.2.4異なるDURATION抽象値の間には、単一の時間要素 (たとえば、週、月、または日のみ) を使用して表現されない限り、定義された順序関係はありません。これは、1 か月または 1 か月の期間について合意された国際定義がないためです。秒に換算すると 1 年。

38.3 指定されたプロパティ設定を持つ時間抽象値の基本的な値の表記法と XML 値の表記法

38.3.1同じ時間プロパティ設定を持つすべての時間抽象値は同じ値表記を持ち、年、月、週、日、時、分、秒など (関連する時間スケール上の) の値によってのみ異なります。その抽象値を、同じプロパティ設定を持つ他の抽象値と区別するために使用されます。

38.3.2時間タイプの値の表記は、「TimeValue」および「XMLTimeValue」とします。

時間値 ::= 文字列

XMLTimeValue ::= xmltstring

「tstring」と「xmltstring」の内容は、表 7 の列 3 で定義されている時間コンポーネント構文を使用して 38.3.4 で定義されます。表 7 は、さまざまなコンポーネントに対して使用可能な多数の表記法を定義しています (たとえば、年のコンポーネント)。使用される正確な表記法は、で指定された抽象値のプロパティ設定によって異なります。

列 2。列 2 にリストされていないプロパティは、コンポーネントに使用される表記に影響を与えません。これらの時間コンポーネントの表記は通常、(準拠している) ISO 8601 表現を参照して定義されますが、値の表記における曖昧さを避けるために、年ではなく世紀を示す時間コンポーネントにC文字が追加されます。 、表 7 の列 3 に指定されているとおりです。

38.3.3表 7 は、時間コンポーネント (列 1 にリストされている) の値表記と XML 値表記を (列 3 に) 指定します。列 1 は時間コンポーネントを示します。列 2 は、抽象値に関連付けられたプロパティの設定に関して、特定の行が適用される条件を指定します。列 3 は、その時間コンポーネントに使用される表記法を指定します。列 3 で使用される表記法は、ISO 8601、3.4 で定義されているものに、世紀指定子としてCが追加されています。

注 1 – 列 3 で使用される ISO 8601 表記は、次のように要約できます。 Y は年の数字、M は月の数字または月指定子、D は日の数字、w は週の数字、h は時間の数字、mは分の桁、s は秒の桁、n は 0 ～ 9 のいずれか、± はプラスまたはマイナス、下線は 0 回以上の繰り返しを表します (たとえば、「±YYYYY」)。 ISO 8601 表記は、この推奨事項で使用される他の表記よりも優先して使用されます。 ISO 8601との連携を明確にするための国際規格。 —

注 2 – 条項 J.2 は、この表記法を理解するのに役立つ ISO 8601 の主要な概念に関するチュートリアルを提供します。結果の値の表記例については、G.3 項も参照してください。

表 7 – 特定のプロパティと設定を持つ時間抽象値の値表記

時間コンポーネント	財産	値の表記構文
年のコンポーネント	「年＝基本」 および「日付=C」 または 「年=プロレプティック」 および「日付=C」	ISO 8601、4.1.2.3 c) ラテン大文字 C: [YYC] の文字が続く
年のコンポーネント	「年=マイナス」 および「日付=C」 または 「年=Ln」および「日付=C」	ISO 8601、4.1.2.4 d) ラテン大文字 C の文字が続く: [±YYYC] — Y の繰り返し数は、 「年 = 負」の場合は 0_ 「年 = Ln」の場合は n-4に等しくなります。
年のコンポーネント	「年＝基本」 そして日付はCではありません または 「年=プロレプティック」 そして日付はCではありません	ISO 8601、4.1.2.2: [YYYY]
年のコンポーネント	「年=負」と日付 Cではありません または 「年=Ln」かつ日付がCではありません	ISO 8601、4.1.2.4 c): [±YYYYY] — Y の繰り返し数は、 「年 = 負」の場合は 0_ 「年 = Ln」の場合は n-4に等しくなります。
月コンポーネント	どれでも	ISO 8601、4.1.2.3 a): [-MM]
週のコンポーネント	どれでも	ISO 8601、4.1.4.3: [-Www]
日の構成要素	「年=YMD」	ISO 8601、4.1.2.2 拡張フォーマット: [-DD]
日の構成要素	「年=YD」	ISO 8601、4.1.3.2 拡張フォーマット: [-DDD]
日の構成要素	「年=YWD」	ISO 8601、4.1.4.2 拡張フォーマット: [-D]
時間コンポーネント	「基本＝時間」 または 「基本=間隔」および 「SEポイント=時間」 または 「Basic=Rec-Interval」および 「SEポイント=時間」	ISO 8601、4.2.2.3 b): [hh] 時間コンポーネント値の表記24 は、抽象値「一日の終わりの午前 0 時」に常に使用され、 時間コンポーネント値の表記00は「一日の始まりの午前 0 時」に使用されます。

表 7 – 特定のプロパティと設定を持つ時間抽象値の値表記

時間コンポーネント	財産	値の表記構文
時間コンポーネント	「基本=日付/時刻」 または 「基本=間隔」および 「SEポイント=日付/時刻」 または 「Basic=Rec-Interval」および 「SEポイント=日付/時刻」	ISO 8601、4.3.2 拡張フォーマット: [Thh] 値表記T24 は、抽象値「一日の終わりの午前 0 時」の時間コンポーネントに常に使用され、値表記T00は「一日の始まりの午前 0 時」に使用されます。
分のコンポーネント	どれでも	ISO 8601、4.3.2 拡張形式: [:mm]
秒コンポーネント	どれでも	ISO 8601、4.3.2 拡張形式: [:ss]
時、分、または秒の小数 部分	どれでも	ISO 8601、4.2.2.4: [,hh] または [.hh]、[,mm] または [.mm]、または [,ss] または [.ss] 注 – どの ASN.1 モジュールでも、小数点記号にはコンマまたはピリオドを一貫して使用することをお勧めします。
期間内の年、月、週、また は日の小数部分（J.2.6、注を参照）	「基本=間隔」および 「インターバルタイプ=D」 または 「基本=間隔」および 「インターバルタイプ=SD」 または 「基本=間隔」および 「インターバルタイプ=DE」	ISO 8601、4.4.3.2: [,nn] または [.nn] 注 – どの ASN.1 モジュールでも、小数点記号にはコンマまたはピリオドを一貫して使用することをお勧めします。
UTC 指定子コンポーネ ント	「ローカルまたは UTC=Z」	ISO 8601、4.2.4: [Z]
時差成分	「ローカルまたは UTC=LD」	ISO 8601、4.2.5.2 拡張形式: [±hh] または [±hh:mm] 時差コンポーネントは、時間の正確な倍数ではない場合、分単位の正確な時差となります。 注 – これは、現地時刻と UTC の差が整数の時間数でない限り、分コンポーネントが存在する必要があることを意味します。
期間コンポーネント	「インターバルタイプ=D」 または 「インターバルタイプ=SD」 または 「インターバルタイプ=DE」	ISO 8601、4.4.3.2: 38.3.6を参照
時間間隔	「インターバルタイプ=SE」 または 「インターバルタイプ=SD」 または 「インターバルタイプ=DE」	ISO 8601、4.4 拡張フォーマット: 開始点コンポーネント("Interval-type=SE"または"Interval-type=SD")または継続時間コンポーネント("Interval-type=DE")の後に[/] が続き、その後に継続時間コンポーネント("Interval-type=SD") が続きます")またはエンドポイントコンポーネント("Interval-type=SE"または"Interval-type=DE")。
始点コンポーネント	SEポイント設定に依存	これはSE-pointの設定によって決定され、このコンポーネントを表すBasicプロパティの設定として解釈されます。次に、日付、年、時刻、ローカルまたは UTC のプロパティ設定を使用して、開始点コンポーネントの形式を決定します。

表 7 – 特定のプロパティと設定を持つ時間抽象値の値表記

時間コンポーネント	財産	値の表記構文
エンドポイントコンポーネント	SEポイント設定に依存	これはSE-pointの設定によって決定され、このコンポーネントを表すBasicプロパティの設定として解釈されます。次に、日付、年、時刻、ローカルまたは UTC のプロパティ設定を使用して、エンドポイント コンポーネントの形式を決定します。終了ポイントの UTC と現地時間の差が開始ポイントの差と同じである場合、時差コンポーネントを省略することが (オプションで) 許可されます。 注 – これは ISO 8601 ほど一般的ではありませんが、簡単にするためにこれらのケースに限定されています。
繰り返しの時間間隔	「Recurrence=Unlimited」 ISO 8601、4.5 拡張形式: [R/] の後に時間間隔コンポーネントが続きます。	
繰り返しの時間間隔	「繰り返し=R1」、「繰り返し=R2」、「繰り返し=R3」など。	ISO 8601、4.5 拡張形式: [Rnn/] の後に時間間隔コンポーネントが続きます。

38.3.4 「tstring」の値は、時間コンポーネントの文字エンコーディング（表 6 に従ってプロパティの設定によって決定される）を連結したものとし、その前後に引用符 (34) 文字 (" "xmltstring" の値は、時間コンポーネントの文字エンコーディング (表 6 に従ってプロパティの設定によって決定される) を引用符で囲まずに連結したものとします。

注 1 – 値の表記法と XML 値の表記法は、次の場合を除いて正規のものです。

a) 期間のさまざまな表現。そして

b) 小数点区切り文字としてのコンマまたはピリオドの使用の変化。そして

c) 時間と分、または時間の整数である時差要素のみの時間のさまざまな使用。
時間;そして

d) 終了点の時間差が開始点の時間差と同じである場合の、（開始点と終了点の両方を含む）区間の終了点における時間差成分の包含または省略。

注 2 – 値の表記例は G.3 に示されています。

38.3.5時間コンポーネントの表記は、ISO 8601 で指定された順序で連結されます。

注 – これは、最も重要な時間コンポーネントが最初で、ゾーン指定子 (時間差コンポーネントまたはZ) が最後であることを意味します。

38.3.6期間コンポーネントの基本的な値の表記法と XML 値の表記法は、次の副節で指定されます。

38.3.6.1値の表記は、[P] (ISO 8601、4.4.3 を参照) の後に次のいずれかを続けるものとします。

a) 年、月、日の指定 (38.3.6.2 を参照) にオプションで時間、分、秒の指定が続く
(38.3.6.3を参照);または

b) 週の指定（38.3.6.4 を参照）。または

c) 時間、分、秒の指定 (38.3.6.3 を参照)。

38.3.6.2 年、月、日の指定は、次の 1 つ以上 (順番に) でなければなりません。

a) 年の指定 (38.3.6.5 を参照)。

b) 月の指定（38.3.6.6 を参照）。

c) 曜日の指定（38.3.6.7 を参照）。

38.3.6.3時分秒の指定は、[T] の後に次の 1 つ以上 (順番に) が続くものとします。

a) 時間指定（38.3.6.8 を参照）。または

b) 議事録の指定 (38.3.6.9 を参照)。または

c) 秒指定（38.3.6.10 を参照）。

38.3.6.4週の指定は、1 つ以上の数字と、その後に任意で小数部 (38.3.6.12 を参照) が続き、その後に [W] が続くもので構成されます。

38.3.6.5 年の指定は、1 つ以上の数字と、オプションで小数部分 (38.3.6.12 を参照)、その後に [Y] が続くもので構成されます。

38.3.6.6月の指定は、1 つ以上の数字と、その後に任意で小数部 (38.3.6.12 を参照) が続き、その後に [M] が続くもので構成されます。

38.3.6.7曜日の指定は、1 つ以上の数字と、オプションで小数部分 (38.3.6.12 を参照)、その後に [D] が続くもので構成されます。

38.3.6.8時間指定は、1 つ以上の数字と、必要に応じて小数部分 (38.3.6.12 を参照)、その後に続く [H] で構成されます。

38.3.6.9分の指定は、1 つ以上の数字と、その後に任意で小数部分 (38.3.6.12 を参照) が続き、その後に [M] が続くもので構成されます。

38.3.6.10秒の指定は、1 つ以上の数字と、オプションで小数部 (38.3.6.12 を参照) が続き、その後に [S] が続くもので構成されます。

38.3.6.11指定の整数部には、任意で小数部が続く 1 桁のゼロでない限り、先頭にゼロを含めてはなりません。後続の小数部がある場合は、整数部に少なくとも 1 桁必要です。

38.3.6.12小数部は、小数点区切り文字 (ピリオドまたはコンマのいずれか) と、その後に続く 1 つ以上の 10 進数で構成されます。

38.3.6.13指定に小数部分が含まれる場合、以下の指定はありません。

38.3.6.14異なる精度で期間を表す値の表記は、異なる抽象値を表します。

例 1: 次の値の表記はすべて、異なる抽象値を表します。

a) P29M (またはP0Y29M) -- 0 年 29 か月、精度は 1 か月です。

b) P29M0D (またはP0Y29M0D) -- 0 年 29 か月 0 日 (精度は 1 日)。

c) P29MT0S (またはP0Y29M0DT0H0M0S) -- 0 年、29 か月、0 日、0 時間、0 分、0 秒、1 秒の精度。

d) P29MT0.00H (またはP0Y29M0DT0,00H) -- 0 年、29 か月、0 日、0 時間、100 分の 1 時間の精度。

e) P29MT0.000S (またはP0Y29M0DT0H0M0.000S) -- 0 年、29 か月、0 日、0 時間、0 分、0 秒、1 ミリ秒の精度。

例 2: 次の値の表記はすべて、同じ抽象値 (0 年、29 か月、0 日、0 時間、0 分) を 100 分の 1 の精度で表します。

a) P0Y29M0DT0H0.00M

b) P0Y29M0DT0.00M

c) P0Y29MT0H0.00M

d) P0Y29MT0.00M

e) P29M0DT0H0.00M

f) P29M0DT0.00M

g) P29MT0H0.00M

h) P29MT0.00M

38.4 便利な時間の種類

次の便利な時間タイプが定義されており、アプリケーション設計者のほとんどの通常の要件をカバーすることが期待されます。

注 – これらの定義では、第 51 項で指定されているプロパティ設定サブタイプの表記を使用します。たとえば、年と日カレンダーの使用など、代替の時間スケールが必要な場合は、定義された時間タイプ (付録 B を参照) を使用するか、プロパティ設定サブタイプを使用できます。表記法を使用して、TIME型の追加のサブタイプを定義できます(使用できるプロパティと設定の例については、G.3 を参照してください)。

38.4.1日付型は次の表記によって参照されます。

DateType ::=日付

そして次のように定義されます。

日付 ::= [ユニバーサル 31] 暗黙の時間
(設定 "基本=日付 日付=YMD 年=基本")

38.4.2時刻型は次の表記によって参照されます。

TimeOfDayType ::=時刻

そして次のように定義されます。

時刻 ::= [UNIVERSAL 32] 暗黙的な時刻
(設定 "Basic=Time Time=HMS Local-or-UTC=L")

注 – このタイプでは、一日の始まりの午前 0 時(00:00:00)だけでなく、一日の終わりの午前 0 時(24:00:00) も許可されます。

38.4.3日時型は次の表記によって参照されます。

DateTimeType ::=日付-時刻

そして次のように定義されます。

日時 ::= [UNIVERSAL 33] 暗黙的な時刻
(設定 "基本=日付-時刻 日付=YMD 年=基本時刻=HMS
ローカルまたは UTC=L")

注 – このタイプでは、一日の始まりの午前 0 時(00:00:00)だけでなく、一日の終わりの午前 0 時(24:00:00) も許可されます。

38.4.4期間タイプは次の表記によって参照されます。

期間タイプ ::=期間

そして次のように定義されます。

持続時間 ::= [ユニバーサル 34] 暗黙的な時間
(設定「基本=インターバル インターバルタイプ=D」)

すべての抽象値が「Basic=Interval Interval-type=D」というプロパティ設定を持つTIME型のサブセット(UNIVERSAL 34またはUNIVERSAL 14) は、duration サブタイプと呼ばれます。このタイプは、次の節に従って制約できます。

38.4.4.1内部サブタイプ制約は、同等のシーケンス タイプを使用して任意の期間サブタイプに適用できます (38.4.4.2 を参照)。

注 – 同等のシーケンス型に適用される内部サブタイプ制約は、期間型の特定の時間コンポーネントを禁止または要求したり、期間型の一部またはすべての時間コンポーネントの値に範囲制約を設定したりするために使用できます (51.11 も参照) .2）。

38.4.4.2 DURATION -EQUIVALENT の同等のシーケンス タイプは次のとおりです。

```
持続時間相当 ::= シーケンス {
年月週          整数 (0..MAX) オプション、
                整数 (0..MAX) オプション、
                整数 (0..MAX) オプション、
日、時          整数 (0..MAX) オプション、整数 (0..MAX) オプシ
間、分          ョン、
                整数 (0..MAX) オプション、
SEQUENCE        INTEGER (0..MAX) OPTIONAL、秒の小数部
{
    桁数 INTEGER(1..MAX),
    小数値 INTEGER(0..MAX) } OPTIONAL }
```

ここで、同等のシーケンス型の年コンポーネントは、duration 型の抽象値の年時間コンポーネントに対応します。

38.4.4.3等価シーケンス タイプのコンポーネントに課される制約は、継続時間タイプの対応する時間コンポーネントに対する制約です。

注 1 – 期間タイプの規則では、時間コンポーネントの少なくとも 1 つが存在することが必要ですが (38.2.3 を参照)、週が存在する場合には他の時間コンポーネントが存在しないことが必要です。これらのルールに違反する内部サブタイプ制約の使用は、不正な仕様となります。

注 2 – 小数部は常に、抽象値に存在する最下位の時間コンポーネントに適用されます。

38.4.5すべての有用な時間型の基本値表記および XML 値表記は、TIME型の値表記(38.3.2 を参照) とし、有用な時間型に存在する抽象値の表記に限定されます。

39 文字列の種類

これらのタイプは、指定された文字レパートリーからの文字列で構成されます。 1 つ以上のテーブルのセルを使用して文字レパートリーとそのエンコーディングを定義するのが通常であり、各セルはレパートリー内の文字に対応します。

通常、各セルには図形記号と文字名も割り当てられますが、一部のレパートリーでは、セルが空のままであるか、名前はあるが形状がない場合があります（名前はあるが形状がないセルの例には、ISO/IEC の EOF などの制御文字が含まれます） 646 および ISO/IEC 10646 の THIN-SPACE や EN-SPACE などのスペース文字）。

一般に、セルに関連付けられた情報は、その情報が null (そのセルにグラフィック シンボルや名前が割り当てられていない) であっても、レパートリー内の個別の抽象文字を示します。

文字列型の ASN.1 の基本値表記には 3 つのバリアント (組み合わせ可能) があり、以下に正式に指定します。

a) 割り当てられたグラフィック記号を使用した文字列内の文字の表現。スペース文字も含まれる可能性があります。これは「cstring」表記です。

注 1 – レパートリー内の複数の文字に同じグラフィック記号が使用されている場合、そのような表現は印刷表現において曖昧になる可能性があります。

注 2 – 異なる幅のスペース文字がレパートリーに存在する場合、または仕様書がプロポーショナルスペースフォントで印刷されている場合、このような表現は印刷表現において曖昧になる可能性があります。

b) 文字が割り当てられている一連の ASN.1 値参照を指定することによる、文字列値内の文字のリスト。このような値参照のセットは、ISO/IEC 10646 文字レパートリーおよびIA5String文字レパートリーに対して、42 節のモジュールASN1-CHARACTER-MODULE で定義されます。この形式は、ユーザーが上記の a) または下記の c) で説明されている値表記を使用してそのような値参照に割り当てない限り、他の文字レパートリーには使用できません。

c) 文字レパートリーテーブル内のセルの位置によって各抽象文字を識別することによる、文字列値内の文字のリスト。このフォームは、IA5String、 UniversalString、 UTF8String 、およびBMPStringでのみ使用できます。

文字列型の ASN.1 XML 値表記は、「xmlcstring」表記を使用します。これには、特定の特殊文字にエスケープ シーケンスを使用したり、10 進数または 16 進数を使用して文字を指定したりする機能が含まれています (12.15 を参照)。

40 文字列型の表記

40.1 文字列型を参照するための表記法 (3.8.12 を参照) は次のとおりです。

```
文字列タイプ ::=
        RestrictedCharacterStringType
|       無制限の文字列タイプ
```

「RestrictedCharacterStringType」は制限された文字列型の表記法であり、第 41 条で定義されています。
「UnrestrictedCharacterStringType」は、無制限の文字列型の表記法であり、44.1で定義されています。

40.2タ 各制限文字列型のタグは41.1に規定されています。無制限の文字列のタグ
イプは 44.2 で規定されています。

40.3 文字列値の表記は次のとおりです。

```
文字列値 ::=
        制限された文字列値
|       無制限の文字列値

XML文字列値 ::=
        XMLRestrictedCharacterStringValue
| XMLUnrestrictedCharacterStringValue
```

「RestrictedCharacterStringValue」と「XMLRestrictedCharacterStringValue」はそれぞれ 41.8 と 41.9 で定義されています。
「UnrestrictedCharacterStringValue」および「XMLUnrestrictedCharacterStringValue」は、無制限の文字列値の表記法であり、44.7 で定義されています。

41 制限される文字列型の定義

この句は、値が、指定された文字のコレクションからの 0 個、または 1 個以上の文字のシーケンスに制限される型を定義します。制限された文字列型を参照するための表記は、「RestrictedCharacterStringType」とします。

```
RestrictedCharacterStringType ::=
        BMP文字列
├一般文字列
```

|グラフィック文字列
| IA5文字列
| ISO646文字列
|数値文字列
|印刷可能な文字列
|テレテックスストリング
| T61ストリング
|ユニバーサル文字列
| UTF8文字列
|ビデオテックス文字列
|可視文字列

各「RestrictedCharacterStringType」代替は、次を指定することによって定義されます。

a) タイプに割り当てられたタグ。そして

b)型を参照する名前 (例: NumericString) 。そして

c) 文字グラフィックをリストした表を参照するか、ISO 国際コード化文字セット登録番号を参照して、型の定義に使用される文字のコレクション内の文字 (ISO 国際コード化文字セット登録を参照)エスケープ シーケンスとともに使用することもできます)、または ISO/IEC 10646 を参照することもできます。

表 8 – 制限される文字列タイプのリスト

型を参照するための名前	ユニバーサル クラス番号	定義登録番号a)、テーブル番号、 またはRec. ITU-T X.680 | ISO/IEC 8824-1 条項	ノート
UTF8文字列	12	第 41.16 条	
数値文字列	18	表9	(注1)
印刷可能な文字列	19	表10	(注1)
テレテックスストリング(T61ストリング)	20	6、87、102、103、106、107、126、144、150、153、156、164、165、168 + スペース + 削除	(注2)
ビデオテックス文字列	21	1、13、72、73、87、89、102、108、126、128、129、144、150、153、164、165、168 + スペース + 削除	（注3）
IA5文字列	22	1、6 + スペース + 削除	
グラフィック文字列	25	全Gセット+SPACE	
可視文字列(ISO646文字列)	26	6+スペース	
一般文字列	27	すべての G およびすべての C セット + スペース + DELETE	
ユニバーサル文字列	28	41.6を参照	
BMP文字列	30	41.15 を参照	

a) 定義する登録番号は、エスケープ シーケンスで使用される ISO International Register of Coded Character Sets にリストされています。

注 1 – 活字スタイル、サイズ、色、強度、またはその他の表示特性は重要ではありません。

注 2 – レジスタ エントリ 6 と 156 は、102 と 103 の代わりに使用できます。

注 3 – これらの登録番号に対応するエントリは、CCITT Rec の機能を提供します。 T.100 と Rec. ITU-T T.101。

41.1表 8 は、各制限文字列タイプが参照される名前、そのタイプに割り当てられたユニバーサル クラス タグの番号、定義する登録番号またはテーブル、または定義するテキスト句、および必要に応じて注記の識別をリストします。テーブルのエントリに関連する。表記法で同義の名前が定義されている場合、これは括弧内にリストされます。

41.2表 9 に、 NumericString型およびNumericString文字の抽象構文に使用できる文字を示します。

表 9 – 数値文字列

名前	グラフィック
数字	0、1 、 ... 9
空間	（空間）

41.3 次のオブジェクト識別子、OID 国際化リソース識別子、およびオブジェクト記述子の値は次のとおりです。
NumericString文字の抽象構文を識別および説明するために割り当てられています。

{ ジョイント-iso-itu-t asn1(1) 仕様(0) 文字列(1) 数値文字列(0) }

"/Joint-ISO-ITU-T/ASN.1/仕様/文字列/数値文字列"

そして

「NumericString 文字の抽象構文」

注 1 – このオブジェクト識別子の値は、 CHARACTER STRING値で使用でき、また、値とは別に文字列タイプの ID を保持する必要がある場合にも使用できます。

注 2 – NumericString文字抽象構文の値は、次のようにエンコードできます。

a) ISO/IEC 10646 で規定されている、抽象文字のエンコードに関する規則の 1 つ。この場合、文字転送構文は、ISO/IEC 10646、Annex N の規則に関連付けられたオブジェクト識別子によって識別されます。

b) 組み込み型NumericString の ASN.1 エンコード規則。この場合、文字転送構文はオブジェクト識別子の値{joint-iso-itu-t asn1(1) Basic-encoding(1)} によって識別されます。

41.4 表 10 に、 PrintableString型とPrintableStringに使用できる文字を示します。
文字の抽象構文。

表 10 – PrintableString

名前	グラフィック
ラテン大文字	A、 B、 ... Z
ラテン語の小文字	a、 b、 ... z
数字	0、1 、 ... 9
空間	（空間）
アポストロフィ	'
左括弧	(
右括弧	)
プラス記号	+
コンマ	、
ハイフンマイナス	-
フルストップ	。
ソリダス	/
結腸	:
等号	=
疑問符	?

41.5 次のオブジェクト識別子、OID 国際化リソース識別子、およびオブジェクト記述子の値は次のとおりです。
は、 PrintableString文字抽象構文を識別して記述するために割り当てられています。

{ジョイント-iso-itu-t asn1(1)仕様(0)characterStrings(1)printableString(1)}

"/Joint-ISO-ITU-T/ASN.1/仕様/Character_Strings/Printable_String"

そして

「PrintableString 文字抽象構文」

注 1 – このオブジェクト識別子の値は、 CHARACTER STRING値で使用でき、また、値とは別に文字列タイプの ID を保持する必要がある場合にも使用できます。

注 2 – PrintableString文字抽象構文の値は、次のようにエンコードできます。

a) ISO/IEC 10646 で規定されている、抽象文字のエンコードに関する規則の 1 つ。この場合、文字転送構文は、ISO/IEC 10646、Annex N の規則に関連付けられたオブジェクト識別子によって識別されます。

b) 組み込み型PrintableString の ASN.1 エンコード規則。この場合、文字転送構文は次のようになります。
オブジェクト識別子{ Joint-iso-itu-t asn1(1) Basic-encoding(1) } によって識別されます。

41.6 UniversalString型に使用できる文字は、 UniversalString型で許可されている文字のいずれかです。
ISO/IEC 10646。

41.7 このタイプを使用すると、ISO/IEC 10646 で指定された適合要件が適用されます。

注 – 第 42 条は、ISO/IEC 10646、付録 A で定義されている「サブセット用のグラフィックス文字のコレクション」について、このタイプの多数のサブタイプを含む ASN.1 モジュールを定義します。

制限文字列型の「RestrictedCharacterStringValue」の表記は、「cstring」(41.8 12.14 参照）、「CharacterStringList」、「Quadruple」、または「Tuple」とする。 「Quadruple」は長さ 1 の文字列のみを定義でき、 UniversalString、 UTF8String 、またはBMPString型の値表記でのみ使用できます。 「タプル」は長さ 1 の文字列のみを定義でき、 IA5Stringの値表記でのみ使用できます。

種類。

```
制限された文字列値 ::=
        cstring
|       文字列リスト
|クワドルプル
|タプル

CharacterStringList ::= "{" CharSyms "}"

チャーシムズ ::=
        文字定義
| CharSyms "," CharsDefn

文字定義::=
        cstring
|クワドルプル
|タプル
|定義された値

4 倍::= "{"グループ "、" 平面 "、" 行 "、" セル"}"

グループ ::= 番号

平面 ::= 数値

行 ::= 番号

細胞            ::= 数値

タプル ::= "{" TableColumn "," TableRow "}"

テーブル列 ::= 数値

テーブル行 ::= 数値
```

注 1 – 「cstring」表記は、値に含まれる文字のグラフィック シンボルを表示できる媒体上でのみ明確に使用できます。逆に、メディアにそのような機能がない場合、そのようなグラフィック シンボルを使用する文字列値を明確に指定する唯一の手段は、型がUniversalString、 UTF8String、 BMPString 、またはIA5String の場合に限り、「CharacterStringList」表記法を使用することです。 "CharsDefn" の代わりに "DefinedValue" が使用されます (42.1.2 を参照)。

注 2 – 条項 42 は、 BMPString型の単一文字 (サイズ 1 の文字列) を示す多数の「valuereference」を定義しています。
(したがって、 UniversalStringとUTF8String)およびIA5String。

例 – 文字「 」が利用可能な媒体上で表現できないUniversalStringに値「abc def」を指定したいとします。この値は次のように表すこともできます。

```
ASN1-CHARACTER-MODULE から BasicLatin、greekCapitalLetterSigma をインポートします
{ Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) iso10646(0) };

MyAlphabet ::= UniversalString (FROM (BasicLatin | greekCapitalLetterSigma))

mystring MyAlphabet ::= { "abc" 、 greekCapitalLetterSigma 、 "def" }
```

注 3 – UniversalString、 UTF8String 、またはBMPString型の値を指定する場合、同様の形状を持つ異なるグラフィック文字から生じる曖昧さが解決されない限り、「cstring」表記を使用しないでください。

例 – 次の「cstring」表記は使用しないでください。グラフィック記号「H」、「O」、「P」、および「E」は、BASIC LATIN、CYRILLIC、BASIC GREEK のアルファベットに出現し、曖昧であるためです。

```
ASN1-CHARACTER-MODULE から BasicLatin、キリル文字、BasicGreek をインポートします
{ Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) iso10646(0) };

MyAlphabet ::= UniversalString (FROM (BasicLatin | Cyrillic | BasicGreek))
```

mystring MyAlphabet ::= "HOPE"

mystringの代替の明確な定義は次のようになります。

mystring MyAlphabet(BasicLatin) ::= "HOPE"

正式には、 mystring はMyAlphabetのサブセットの値への値参照ですが、付録 C の値マッピング規則に従って、MyAlphabet 内のこの値への値参照が必要な場合はどこでも使用できます。

41.9 「XMLRestrictedCharacterStringValue」の表記は次のとおりです。

XMLRestrictedCharacterStringValue ::= xmlcstring

「XMLRestrictedCharacterStringValue」の「XMLTypedValue」(16.2 を参照) の「XMLValue」の周囲に空白を使用してはなりません。 ただし、この表記法がエンコードで使用され、エンコード規則で明示的に空白が許可されている場合を除きます (Rec. ITU-T X.693 | ISO を参照)。 /IEC 8825-4、39.3.2)。

41.10 「xmlcstring」で直接表現できない文字があります。これらは、12.15 で指定されたエスケープ シーケンスを使用して表現されます。

注 – 制限された文字列値に 12.15.1 で指定されている ISO/IEC 10646 文字ではない文字が含まれている場合、これらの文字は「xmlcstring」で表すことができず、そのような値は XML エンコーディング ルールを使用して転送できません (Rec. ITU-T を参照) X.693 | ISO/IEC 8825-4)。

41.11 「CharsDefn」の「DefinedValue」は、その型の値への参照でなければなりません。

41.12 「平面」、「行」、および「セル」プロダクションの「数」は 256 未満、「グループ」プロダクションの場合は 128 未満でなければなりません。

41.13 「グループ」は UCS のコーディング空間内のグループを指定し、「プレーン」はグループ内のプレーンを指定し、「行」はプレーン内の行を指定し、「セル」は行内のセルを指定します。この表記法によって識別される抽象文字は、「Group」、「Plane」、「Row」、および「Cell」の値によって指定されるセルの抽象文字です。どのような場合でも、許可される文字のセットはサブタイプによって制限される可能性があります。

注 – アプリケーション設計者は、制約を適用せずにGeneralString、 GraphicString、 UniversalStringなどのオープンエンド文字列型を使用する場合、適合性への影響を慎重に考慮する必要があります。 TeletexStringなど、境界があるが大きな文字列タイプについても、適合性に関する注意深いテキストが必要です。

41.14 「TableColumn」プロダクションの「数値」は 0 から 7 の範囲でなければならず、「TableRow」プロダクションの「数値」は 0 から 15 の範囲でなければなりません。 ISO/IEC 2022 の図 1 に準拠した文字コード表の「TableColumn」は列を指定し、「TableRow」は行を指定します。この表記は、コード表の列 0 と列に Register Entry 1 が含まれる場合に IA5Stringにのみ使用されます。 1 と列 2 ～ 7 のレジスタ エントリ 6 (エスケープ シーケンスで使用されるコード化文字セットの ISO 国際登録を参照)。

41.15 BMPStringは、独自の一意のタグを持ち、基本多言語面の文字のみを含む (最初の 64K-2 セルに対応し、基本多言語面外の文字をアドレス指定するためにエンコーディングが使用されているセルは除く) 41.15 BMPString は UniversalString のサブタイプです。 ISO/IEC 10646。次のように定義された関連タイプがあります。

ユニバーサル文字列 (Bmp)

ここで、 Bmp は、ISO/IEC 10646、付録 A で定義された「BMP」コレクション名に対応するUniversalStringのサブタイプとして、 ASN.1 モジュールASN1-CHARACTER-MODULE (条項 42 を参照) で定義されています。

注 1 – BMPString は組み込み型であるため、 ASN1-CHARACTER-MODULE では定義されていません。

注 2 – BMPString を組み込み型として定義する目的は、32 ビット エンコーディングではなく 16 ビット エンコーディングを使用するための制約を考慮しないエンコーディング ルール (BER など) を有効にすることです。

注 3 – 値の表記では、すべてのBMPString値は有効なUniversalString値およびUTF8String値です。

41.16 UTF8String は、抽象レベルではUniversalStringと同義であり、 UniversalString が使用される場所であればどこでも使用できますが (個別のタグを必要とするルールに従う)、異なるタグを持ち、個別の型です。

注 – BER および PER で使用されるUTF8Stringのエンコーディングは、 UniversalStringのエンコーディングとは異なり、ほとんどのテキストでは冗長性が低くなります。

42 文字、コレクション、プロパティ カテゴリ セットに名前を付ける

この句は、ISO/IEC 10646 の各文字の値参照名の定義を含む ASN.1 組み込みモジュールを指定します。各名前は、サイズ 1 のUniversalString値を参照します。このモジュールには、型参照の定義も含まれます。 ISO/IEC 10646 の文字の各コレクションの名前。各名前はUniversalString型のサブセットを参照します。最後に、Unicode 標準の 4.5 にリストされている文字プロパティの一般カテゴリの文字セットの「typereference」名の定義が含まれており、各名前はUniversalString型のサブセットを参照します。

注 – これらの値は、 UniversalString型およびそこから派生した型の値表記で使用できます。 42.1 で指定されたモジュールで定義されているすべての値参照と型参照はエクスポートされ、それらを使用するモジュールによってインポートされる必要があります。

42.1 ASN.1モジュール「ASN1-CHARACTER-MODULE」の仕様

ここではモジュールの全文は掲載されていません。代わりに、それを定義する手段が指定されます。

注 – この推奨事項 |国際規格は ISO/IEC 10646:2003 に基づいています。この規格の新しいバージョンを使用して適用することはできません。 「ASN1-CHARACTER-MODULE」を定義する手段の仕様は、ISO/IEC 10646:2003 でのみ適用できます。

42.1.1モジュールは次のように始まります。

```
ASN1-CHARACTER-MODULE { Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) iso10646(0) }
"/Joint-ISO-ITU-T/ASN.1/仕様/モジュール/ISO_10646"
定義 ::= 開始

        -- この内で定義されているすべての値参照と型参照
        -- モジュールは暗黙的にエクスポートされ、任意のモジュールでインポートできます。
        -- ISO/IEC 646 制御文字:

        null IA5String ::= {0, 0}
そう    IA5文字列 ::= {0, 1}
stx     IA5文字列 ::= {0, 2}
etx     IA5文字列 ::= {0, 3}
えっと  IA5文字列 ::= {0, 4}
enq     IA5文字列 ::= {0, 5}
ack     IA5文字列 ::= {0, 6}
bel     IA5文字列 ::= {0, 7}
bs      IA5文字列 ::= {0, 8}
ht      IA5文字列 ::= {0, 9}
lf      IA5文字列 ::= {0,10}
vt      IA5文字列 ::= {0,11}
ff      IA5文字列 ::= {0,12}
cr      IA5文字列 ::= {0,13}
それで  IA5文字列 ::= {0,14}
シ      IA5文字列 ::= {0,15}
ド      IA5文字列 ::= {1, 0}
ル      IA5文字列 ::= {1, 1}
DC1 DC2 IA5文字列 ::= {1, 2}
dc3     IA5文字列 ::= {1, 3}
dc4     IA5String ::= {1, 4}
ナク    IA5String ::= {1, 5}
シン    IA5文字列 ::= {1, 6}
ETB     IA5String ::= {1, 7}
できる  IA5String ::= {1, 8}
エム    IA5文字列 ::= {1, 9}
サブ    IA5文字列 ::= {1,10}
ESC     IA5文字列 ::= {1,11}
is4     IA5文字列 ::= {1,12}
is3     IA5文字列 ::= {1,13}
is2     IA5文字列 ::= {1,14}
is1     IA5文字列 ::= {1,15}
デル    IA5文字列 ::= {7,15}
```

42.1.2 ISO/IEC 10646 の第 24 節および第 25 節に示されているグラフィック文字 (グリフ) の文字名の各リストの各エントリについて、モジュールには次の形式のステートメントが含まれます。

```
<名前付き文字> BMPString ::= <テーブルセル>
-- 文字 <iso10646name> を表します。ISO/IEC 10646 を参照してください。
```

どこ：

a) <iso10646name> はISO/IEC 10646 にリストされているキャラクター名から派生したキャラクター名です。

b) <namedcharacter> は、 <iso10646name>に指定された手順を適用することによって取得された文字列です。 42.2;

c) <tablecell> は、リスト エントリに対応する ISO/IEC 10646 のテーブル セル内のグリフです。

例

```
latinCapitalLetterA BMPString ::= {0, 0, 0, 65}
```

-- ラテン大文字 A を表します。ISO/IEC 10646 を参照してください。

greekCapitalLetterSigma BMPString ::= {0, 0, 3, 163}
-- ギリシャ語大文字シグマを表します。ISO/IEC 10646 を参照してください。

42.1.3 ISO/IEC 10646、付録 A で指定されているグラフィック文字のコレクションの名前ごとに、次の形式のモジュールにステートメントが含まれます。

<名前付きコレクション文字列> ::= BMPString
(FROM (<alternativelist>))
-- 文字のコレクション <collectionstring> を表します。
-- ISO/IEC 10646 を参照。

どこ：

a) <collectionstring> は、 ISO/IEC 10646 で割り当てられた文字のコレクションの名前です。

b) <namedcollectionstring> は、 42.3 の手順を<collectionstring>に適用することによって形成されます。

c) <alternativelist> は、 42.2 で生成された<namedcharacter> を使用して形成されます。
ISO/IEC 10646 で規定されている文字。

結果として得られる型参照<namedcollectionstring> は、限定されたサブセットを形成します。 (付録 H のチュートリアルを参照してください。)

注 – 限定されたサブセットとは、指定されたサブセット内の文字のリストです。これを、ISO/IEC 10646、付録 A にリストされている文字のコレクションに BASIC LATIN コレクションを加えた選択されたサブセットと比較してください。

例（一部）

スペース BMPString ::= {0、0、0、32}
感嘆符 BMPString ::= {0, 0, 0, 33}
quoteMark BMPString ::= {0, 0, 0, 34}
... - 等々
チルダ BMPString ::= {0、0、0、126}

BasicLatin ::= BMPString
(から(スペース
|エクスクラメーション•マーク
|クォーテーションマーク
| ... - 等々
|チルダ)
)
-- BASIC LATIN 文字の集合を表します。ISO/IEC 10646 を参照してください。
-- この例の省略記号は簡潔にするために使用されており、「など」を意味します。
-- 実際の ASN.1 モジュールではこれを使用できません。

42.1.4 ISO/IEC 10646 は 3 つの実装レベルを定義しています。デフォルトでは、 Level1とLevel2を除くASN1-CHARACTER-MODULEで定義されたすべての型は実装レベル 3 に準拠します。これは、そのような型には結合文字の使用に制限がないためです。 Level1 は実装レベル 1 が必要であることを示し、 Level2
は実装レベル 2 が必要であることを示し、 Level3 は実装レベル 3 が必要であることを示します。したがって、 ASN1-CHARACTER-MODULEでは以下が定義されます。

Level1 ::= BMPString (FROM (BMPString(SIZE(1)) EXCEPT CombiningCharacters))

Level2 ::= BMPString (FROM (BMPString(SIZE(1)) EXCEPT CombiningCharactersType-2))

レベル 3 ::= BMPString

注 1 – CombiningCharactersおよびCombiningCharactersType-2 は、 ISO/IEC 10646、付録 A で定義されている、それぞれ「COMBINING CHARACTERS」および「COMBINING CHARACTERS B-2」に対応する <namedcollectionstring> です。

注 2 – Level1とLevel2 は、 「IntersectionMark」（第 50 項を参照）の後に使用されるか、「ConstraintSpec」内の唯一の制約として使用されます。 (例については、G.2.7.1 を参照してください。)

注 3 – このトピックの詳細については、H.2.5 を参照してください。

42.1.5 Unicode 標準の表 4-5 にリストされている各略語と各説明について、次の形式のモジュールに 2 つのステートメントが含まれています。

<カテゴリの略語> ::= UniversalString (FROM (<alternativelist>))
-- プロパティを持つ文字のセットを表します
--カテゴリ<カテゴリの略語>。

<カテゴリの説明> UniversalString ::= <カテゴリの略語>

どこ：

a) <categoryabbreviation> は、次の文書にリストされている文字プロパティの一般的なカテゴリの略語です。
Unicode 標準、表 4-5 (たとえば、 Lu 、 Nd 、またはPi)。

b) <categorydescription> は、同じ一般カテゴリの文字の説明であり、すべての単語の頭文字が大文字になり、カンマとすべてのスペースが削除され、括弧内のすべての説明が削除されます (たとえば、LetterUppercase 、 NumericDigit 、 PunctuationInitialQuote)。

c) 各 <categoryabbreviation> の <alternativelist> は、Unicode 標準の Unicode Character Database (バージョン 3.2.0) にリストされている、対応する <categoryabbreviation を持つ文字ごとに 42.2 によって生成された <namedcharacter> 名のリストです。 >。
注 – 文字の Unicode 名は、その文字の <iso10646name> と同じです。

42.1.6 Unicode 標準の表 4-5 にリストされている各略語の頭文字については、次の形式のモジュールに 2 つのステートメントが含まれています。

<カテゴリの略語文字> ::= UniversalString (FROM (<alternativelist>))
-- 任意のカテゴリ プロパティを持つ文字のセットを表します
--頭文字 <categoryabbreviationletter> を付けます。

<メインカテゴリの説明> UniversalString ::= <カテゴリの略語文字>

どこ：

a) <categoryabbreviationletter> は、文字の一般カテゴリの略語の最初の文字です。
Unicode 標準の表 4-5 にリストされているプロパティ (L 、 N 、Pなど)。

b) <categorydescription> は、同じ一般的な文字カテゴリ (文字、数字、句読点など)の説明の最初の単語です。

c) 各 <categoryabbreviationletter> の <alternativelist> は、<namedcharacter> 名のリストです。
Unicode 標準の Unicode 文字データベース (バージョン 3.2.0) にリストされている、対応する <categoryabbreviationletter> を持つ文字ごとに、42.2 によって生成されます。
注 – 文字の Unicode 名は、その文字の <iso10646name> と同じです。

42.1.7モジュールは次のステートメントによって終了します。

終わり

42.1.8 42.1.3 の例と同等のユーザー定義は次のとおりです。

BasicLatin ::= BMPString (FROM (スペース..チルダ))
-- 文字の集合を表す BASIC LATIN、
-- ISO/IEC 10646 を参照。

42.2 <namedcharacter> は、 <iso10646name> (42.1.2 を参照) を取得し、次のアルゴリズムを適用することによって取得される文字列です。

a) <iso10646name>の各大文字は、対応する小文字に変換されます。ただし、大文字の前にスペースがある場合は大文字は変更されません。

b) 各数字と各ハイフンマイナスは変更されません。

c) 各 SPACE が削除されます。

注 – ISO/IEC 10646 の付録 K の文字命名ガイドラインと組み合わせた上記のアルゴリズムは、ISO/IEC 10646 にリストされているすべての文字名に対して常に明確な値の表記になります。

例 – ISO/IEC 10646、行 0、セル 60 の文字は、「LESS-THAN SIGN」という名前で、グラフィック表現「<」を持ち、次の「DefinedValue」を使用して参照できます。

小なり記号

42.3 <namedcollectionstring> は、 <collectionstring>を取得し、次のアルゴリズムを適用することによって取得される文字列です。

a) ISO/IEC 10646 コレクション名の各大文字は、対応する小文字に変換されます。ただし、大文字の前にスペースがあるか、名前の最初の文字である場合は除きます。大文字は変更されません。

b) 各数字と各ハイフンマイナスは変更されません。

c) 各 SPACE が削除されます。

例

1) ISO/IEC 10646 の付録 A で識別されるコレクションは次のとおりです。

ベーシックラテン語

ASN.1 タイプ参照があります。

ベーシックラテン語

2) BASIC LATIN コレクションの文字と BASIC の組み合わせで構成される文字列型。
ARABIC コレクションは次のように定義できます。

私の文字列 ::= BMPString (FROM (BasicLatin | BasicArabic))

注 – 上記の構造が必要なのは、次の構造のほうが明らかに単純であるためです。

My-Character-String ::= BMPString (BasicLatin | BasicArabic)

完全に BASIC LATIN または BASIC ARABIC である文字列のみが許可されますが、両方の混合は許可されません。

43 正規の文字順序

43.1 「ValueRange」サブタイプの目的と、エンコード規則による使用の可能性のために、 UniversalString、 UTF8String、 BMPString、 NumericString、 PrintableString、 VisibleString、およびIA5String に対して正規の文字順序が指定されています。

び制 43.2 この節の目的に限り、文字はコード表のセルと 1 対 1 に対応します。43.2そのセルに文字名または形状が割り当てられているかどうか、およ御文字か印刷文字かどうか、結合文字または非結合文字。

43.3抽象文字の正規順序は、ISO/IEC 10646 の 32 ビット表現におけるその値の正規順序によって定義され、正規順序では小さい数値が最初に現れ、大きい数値が最後に現れます。

43.4 「PermittedAlphabet」表記（または個々の文字)内の「ValueRanges」のエンドポイントは、モジュールASN1-CHARACTER-MODULEで定義された ASN.1 値参照を使用して指定できます。または（グラフィック記号は仕様のコンテキストで明確です)およびそれを表現するために使用される媒体)は、「cstring」 (ASN1-CHARACTER-MODULE は42.1 で定義されています)でグラフィック シンボルを指定するか、41.8 の「Quadruple」または「Tuple」表記を使用します。

43.5 NumericStringの場合、左から右に増加する正規の順序は次のように定義されます (41.2 の表 9 を参照)。

(スペース) 0 1 2 3 4 5 6 7 8 9

文字セット全体には正確に 11 文字が含まれています。 「ValueRange」（または個々の文字)の終点は、「cstring」内のグラフィック シンボルを使用して指定できます。

注 – この順序は、ISO/IEC 10646 の BASIC LATIN コレクション内の対応する文字の順序と同じです。

43.6 PrintableStringの場合、左から右、上から下に増加する正規の順序は次のように定義されます (41.4 の表 10 を参照)。

(スペース) (アポストロフィ) (左括弧) (右括弧) (プラス記号)
(カンマ) (ハイフン-マイナス) (フルストップ) (ソリッド) 0123456789 (コロン) (等号)
(疑問符) ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

文字セット全体には正確に 74 文字が含まれています。 「ValueRange」（または個々の文字)の終点は、「cstring」内のグラフィック シンボルを使用して指定できます。

注 – この順序は、ISO/IEC 10646 の BASIC LATIN コレクション内の対応する文字の順序と同じです。

43.7 VisibleStringの場合、セルの正規の順序は ISO/IEC 646 エンコーディング (ISO 646 ENCODING と呼ばれる) から次のように定義されます。

(ISO 646 エンコーディング) - 32

注 – つまり、正規の順序は ISO/IEC 646 コード テーブルのセル 2/0 ～ 7/14 の文字と同じです。

文字セット全体には正確に 95 文字が含まれています。 「ValueRange」（または個々の文字)の終点は、「cstring」内のグラフィック シンボルを使用して指定できます。

43.8 IA5Stringの場合、セルの正規の順序は ISO/IEC 646 エンコーディングから次のように定義されます。

(ISO 646 エンコーディング)

文字セット全体には正確に 128 文字が含まれています。 「ValueRange」（または個々の文字)のエンドポイントは、「cstring」内のグラフィック シンボル、または 42.1.1 で定義されている ISO 646 制御文字値参照を使用して指定できます。

44 無制限の文字列型の定義

この句は、値が任意の文字抽象構文の値である型を定義します。 OSI 環境では、この抽象構文は OSI 定義のコンテキスト セットの一部である場合があります。それ以外の場合は、無制限の文字列型を使用するたびに直接参照されます。

注 1 – 文字抽象構文 (および 1 つ以上の対応する文字転送構文) は、ASN.1オブジェクト識別子を割り当てることができる任意の組織によって定義できます。

注 2 – 関心のあるコミュニティによって作成されたプロファイルは、通常、CHARACTER STRINGの特定のインスタンスまたはインスタンスのグループに対してサポートされる文字抽象構文と文字転送構文を決定します。 OSI アプリケーションでは、サポートされている構文への参照を OSI プロトコル実装適合性ステートメントに含めるのが通常です。

44.1無制限の文字列タイプ (3.8.89 を参照) は、「UnrestrictedCharacterStringType」という表記によって参照されます。

UnrestrictedCharacterStringType ::=文字列

44.2 このタイプにはユニバーサルクラスのタグ番号29が付いています。

44.3 タイプは以下を表す値で構成されます。

a) ASN.1 文字列タイプの値である可能性はありますが、必ずしもそうである必要はない文字列値。そして

b) 以下のものを（個別にまたは一緒に)特定します。

1) 文字抽象構文。そして

2) 文字転送構文。

す。 無制限の文字列型には型が関連付けられています。この関連タイプは、その値44.4とサブタイプ表記をサポートするために使用されま

44.5自動タグ付け環境を想定した、値の定義とサブタイプに関連付けられた型は次のとおりです (規範的なコメント付き)。

```
順序 {
識別構文 抽象転                                選択 {
        送                                          順序 {
                                                        オブジェクト識別子、
                                                        オブジェクト識別子 }
                -- 抽象化および転送構文のオブジェクト識別子 --、

        構文                                            オブジェクト識別子
                -- を識別するための単一のオブジェクト識別子。
                -- 抽象構文と転送構文 --、
プレゼンテーションコンテキストID                整数
                -- (OSI 環境にのみ適用)
                -- ネゴシエートされた OSI プレゼンテーション コンテキストによって、
                -- 抽象構文と転送構文 --、

        コンテキストネゴシエーシ                順序 {
        ョンプレゼンテーションコンテキ          整数、
        ストID転送構文                          オブジェクト識別子 }
                -- (OSI 環境にのみ適用)
                -- コンテキスト ネゴシエーションが進行中、プレゼンテーション コンテキスト ID
                -- は、 --abstract-
                syntax のみを識別するため、転送構文を指定する必要があります --,

                transfer-syntax                         オブジェクト識別子
                -- 値のタイプ (たとえば、値が
                -- ASN.1 タイプの値) はアプリケーションによって固定されます
                -- デザイナー (したがって、送信者と受信者の両方に知られています)。これ
                -- ケースは主にサポートを目的として提供されています
                -- 選択フィールド暗号化 (または他のエンコーディング)
                -- 変換) ASN.1 タイプ --、

                修理済み                                ヌル
```

-- データ値は固定 ASN.1 タイプの値です (したがって、
-- 送信者と受信者の両方に知られています) -- },

データ値記述子 オブジェクト記述子は省略可能
-- これにより、人間が判読できるクラスの識別が提供されます。
- 値 --、
文字列値 オクテット文字列 }
(コンポーネントあり {
...、
データ値記述子 ABSENT })

注 – 無制限の文字列タイプでは、データ値記述子の値を ID と一緒に含めることはできません。ただし、ここで提供される関連型の定義は、embedded-pdv 型、外部型、および無制限の文字列型の間に存在する共通点の基礎となります。

44.6 36.6、36.7の本文は無制限文字列型にも適用されます。

識別 値の表記は、44.5 で定義された関連型の値の表記でなければなりません。ここで、 OCTET STRING型の44.7文字列値コンポーネントの値は、で指定された転送構文を使用したエンコーディングを表します。

UnrestrictedCharacterStringValue ::= シーケンス値

XMLUnrestrictedCharacterStringValue ::= XMLSequenceValue

44.8無制限の文字列型の例を G.2.8 に示します。

45 第 46 条から第 48 条で定義されるタイプの表記

45.1 第 46 条から第 48 条で定義された型を参照するための表記は次のとおりです。

UsefulType ::= タイプリファレンス

ここで、「typereference」は、ASN.1 表記法を使用して条項 46 ～ 48 で定義されているものの 1 つです。

45.2 各「UsefulType」のタグは第46項から第48項で指定します。

46 一般化された時間

注 1 – この推奨事項の以前のバージョン |国際標準では (ISO 時間標準の進化により) 異なるテキストが使用されていますが、技術的な内容はこの勧告の最初のバージョンから変わっていません。国際標準。

注 2 – 時間タイプ (第 38 条を参照) はより柔軟性があり、優先されるべきです。

46.1 この型は次の名前で参照されます。

一般化された時間

46.2 このタイプは、カレンダーの日付と次のもので構成されます。

a) 一日の始まりの午前 0 時を含むが、一日の終わりの午前 0 時を除く現地時刻。
精度:

1) 時、分、秒 (または秒と秒の小数点以下の任意の桁数)。

または

2) 時と分（または分と小数点以下の端数）。または

3) 時間（または時間と小数点以下の端数）。または

b) 上記 a) にリストされているいずれかの精度の UTC 時刻 (1 日の始まりの午前 0 時を含むが、一日の終わりの午前 0 時を除く)。または

c) 上記 a) で指定した現地時刻と現地時刻との差
UTC。

注 – 現地時刻が UTC より進んでいる場合、時差コンポーネントは正の値になります。

46.3 タイプは、ASN.1 を使用して次のように定義されます。

GeneralizedTime ::= [UNIVERSAL 24] 暗黙的な VisibleString

VisibleStringの値は、次のいずれかの文字列に制限されます。

a) 暦日とそれに続く現地時刻の指定。以下で構成されます。 1) ISO 8601、4.1.2.2 – 基本形式で指定されてい

る、暦日を表す文字列)。フォローしました

による：

注 1 – これは、区切り文字を使用せずに、年を 4 桁で表現し、月を 2 桁で表現し、日を 2 桁で表現することを指定します。

2) 1 時間、1 分、1 秒、または 1 秒未満の精度 (任意の精度) で時刻を表す文字列。小数点記号としてカンマまたはピリオドを使用します (ISO で指定されているとおり)。 8601、4.2.2.2 および 4.2.2.3 – 基本フォーマット);オプションで、その後に次のものが続きます。

3) 秒が省略されている場合は分の小数部、分が省略されている場合は時間の小数部

秒は省略されます (ISO 8601、4.2.2.4 で規定されているとおり)。または

注 2 – ISO 8601 では、小数点記号としてカンマまたはピリオドの使用を指定しています。他に区切り文字はありません。特定の ASN.1 仕様では、カンマまたはピリオドを使用することをお勧めします。
一貫して小数点記号として使用されます。

b) 上記 a) の文字の後に大文字の Z が続く暦日と UTC 時間の指定。または

c) ISO 8601 で指定されているカレンダー日付、現地時間、および現地時間と UTC の正確な差の指定。差が整数の時間数である場合は、分コンポーネントはオプションで省略されます。

注 3 – ASN.1 の標準エンコード規則に関する初期の作業では、実際には精度の概念がないと想定されていたため、3.000 の秒コンポーネントで表現される抽象値は、秒コンポーネントで表現される抽象値と同じ抽象値とみなされていました。秒のコンポーネントは 3 であり、正規エンコードの小数部分での末尾のゼロの使用を禁止し、秒または分と秒の省略を禁止します。また、現地時間や現地時間ではなく、UTC 時間の使用のみをサポートしました。

時差成分あり。これは、下位互換性のために、ASN.1 標準の以降のエディションでも変更されていません。 TIME型 (2004 年に ASN.1 に導入) は、抽象値が関連する精度を持つ可能性があること、(たとえば) 3.000 と 3 としての秒の表現が異なる抽象値を生成すること、および現地時刻と UTC 仕様を表すことを認識します。さまざまな抽象的な値。 TIMEの正規のエンコード規則は、その抽象値の全範囲をエンコードするため、 TIMEの使用

新しい仕様では、GeneralizedTime の使用よりも優先される可能性があります。

ケース c) の場合、ケース a) と同様に形成された文字列の部分は現地時刻(t1) を表し、 (符号付き) 時差(t2)によって UTC を決定できます。 t2が正の場合、現地時間は UTC より進んでいます。したがって、UTC は次のように決定できます。

UTC はt1 − t2

例

ケースa)

「19851106210627.3」

現地時間、1985年11月6日午後9時過ぎ6分27.3秒。

ケースb)

「19851106210627.3Z」
上記の協定世界時。

ケースc)

「19851106210627.3-0500」

例 a) の現地時間。協定世界時は 1985 年 11 月 7 日の午前 2 時から 6 分 27.3 秒です。

ケース d)

「198511062106.456」

現地時間、1985 年 11 月 6 日午後 9 時過ぎ 6.456 分。

ケースe)

「1985110621.14159」

現地時間、1985 年 11 月 6 日午後 9 時から 0.14159 時間。

46.4 タグは 46.3 で定義されているものとします。

46.5 値の表記は46.3で定義されたVisibleStringの値の表記とする。

47 世界時

47.1 この型は次の名前で参照されます。

UTC時間

47.2 タイプは以下を表す値で構成されます。

a) 暦日。そして

b) 1 分または 1 秒の精度の時間。そして

c) (オプションで) 協定世界時との現地時間の差。

47.3 タイプは、ASN.1 を使用して次のように定義されます。

UTCTime ::= [UNIVERSAL 23] 暗黙的な VisibleString

VisibleStringの値は、次のものを並べた文字列に制限されます。

a) 6 桁の YYMMDD (YY は西暦の下 2 桁、MM は月)
(1 月を 01 として数えます)、DD は月の日 (01 から 31) です。そして

b) 次のいずれか:

1) 4 桁の hhmm。hh は時間 (00 ～ 23)、mm は分 (00 ～ 59) を表します。または

2) 6 桁の hhmmss。hh と mm は上記 1) と同じで、ss は秒 (00 ～ 59)。そして

c) 次のいずれか:

1) 文字Z。または

2) +または-のいずれかの文字と、その後に hhmm が続きます。hh は時間、mm は分です。

上記の b) の代替案では、時間の指定の精度を変えることができます。

代替 c) 1) では、時刻は協定世界時です。代替 c) 2) では、上記の a) と b) で指定された時刻(t1)は現地時刻です。上記の c) 2) で指定された時差(t2)により、協定世界時を次のように決定できます。

協定世界時はt1 − t2

例 1 – 現地時刻が 1982 年 1 月 2 日の午前 7 時、協定世界時が 1982 年 1 月 2 日の正午 12 時である場合、 UTCTimeの値は次のいずれかになります。

– "8201021200Z";または
–「8201020700-0500」。

例 2 – 現地時刻が 2001 年 1 月 2 日の午前 7 時、協定世界時が 2001 年 1 月 2 日の正午 12 時である場合、 UTCTimeの値は次のいずれかになります。

– "0101021200Z";または
–「0101020700-0500」。

47.4 タグは 47.3 で定義されているものとします。

47.5 値の表記は47.3で定義されたVisibleStringの値の表記とする。

48 オブジェクト記述子のタイプ

48.1 この型は次の名前で参照されます。

オブジェクト記述子

48.2 このタイプは、オブジェクトを説明する人間が読めるテキストで構成されます。テキストは明確ではありません
オブジェクトの識別ですが、異なるオブジェクトの同一テキストは一般的ではないことを意図しています。

注 –オブジェクトにタイプOBJECT IDENTIFIERの値を割り当てる権限は、そのオブジェクトにタイプObjectDescriptorの値も割り当てる必要があることが推奨されます。

48.3 タイプは、ASN.1 を使用して次のように定義されます。

ObjectDescriptor ::= [UNIVERSAL 7] IMPLICIT GraphicString

GraphicStringには、オブジェクトを説明するテキストが含まれます。

48.4 タグは 48.3 で定義されているものとします。

48.5 値の表記は48.3で定義されたGraphicStringの値の表記とする。

49 制約されたタイプ

49.1 「ConstrainedType」表記を使用すると、(親) 型に制約を適用して、その値のセットを親のサブタイプに制限したり、(セットまたはシーケンス型内で) コンポーネントの関係が次の値に適用されるように指定したりできます。親の型と、同じセットまたはシーケンス値内の他のコンポーネントの値へ。また、例外識別子を制約に関連付けることもできます。

```
制約タイプ ::=
        型制約
|制約付きタイプ
```

最初の選択肢では、親タイプは「Type」であり、制約は 49.6 で定義されている「Constraint」によって指定されます。 2 番目の代替案は 49.5 で定義されています。

49.2 「制約」表記が set-of または sequence-of タイプ表記に続く場合、set-of または sequence-of ではなく、(最も内側の) set-of または sequence-of 表記の「Type」に適用されます。タイプ。

注 – たとえば、次の制約(SIZE(1..64))は、 SEQUENCE OFではなくVisibleStringに適用されます。

```
NamesOfMemberNations ::= VisibleString の配列 (SIZE(1..64))
```

49.3 「制約」の表記が選択タイプの表記の後に続く場合、それは選択された選択肢のタイプではなく、選択タイプに適用されます。このような制約は無視されます (30.2 を参照)。

注 – 次の例では、制約(WITH COMPONENTS {..., a ABSENT}) は、選択されたSEQUENCEタイプではなく、 CHOICEタイプTに適用され、 Vの値には影響しません。

```
T ::= CHOICE
{ SEQUENCE
        { INTEGER オプション、
        b ブール値
}、
b NULL
}

V ::= a < T (コンポーネントあり {..., a ABSENT})
```

49.4 「Constraint」表記が「PrefixedType」表記に続く場合、「PrefixedType」または「Type」のどちらが親タイプとみなされているかに関係なく、表記全体の解釈は同じです。

49.5 49.2 で指定された解釈の結果として、set-of 型または sequence-of 型に制約を適用できるようにする特別な表記法が提供されます。これは「TypeWithConstraint」です。

```
制約付きタイプ::=
        SET制約OFタイプ
| SETサイズ拘束OFタイプ
| SEQUENCE制約OFタイプ
| SEQUENCEサイズ制約OFタイプ
| NamedTypeのSET制約
| NamedTypeのSizeConstraintを設定する
| NamedTypeのSEQUENCE制約
| NamedTypeのシーケンスサイズ制約
```

1 番目と 2 番目の選択肢では、親タイプは「SET OF Type」ですが、3 番目と 4 番目の選択肢では「SEQUENCE OF」です。Type"。5 番目と 6 番目の選択肢では、親の型は"SET OF NamedType" で、7 番目と 8 番目の選択肢では"SEQUENCE OF NamedType" です。1 番目、3 番目、5 番目、7 番目の選択肢では、制約は "Constraint" (49.6 を参照)、2 番目、4 番目、6 番目、および 8 番目では「SizeConstraint」です (51.5 を参照)。

注 – 「Constraint」の代替案には対応する「SizeConstraint」の代替案が含まれていますが、「SizeConstraint」の代替案は歴史的な理由から提供されています。

49.6制約は「制約」という表記で指定します。

```
制約 ::= "("制約仕様例外仕様")"
```

制約仕様 ::=
サブタイプ制約
| 一般的な制約

「ExceptionSpec」は第 53 条で定義されています。「拡張マーカー」（第 52 条を参照）と組み合わせて使用されない限り、「ConstraintSpec」に「DummyReference」の出現が含まれる場合にのみ存在します（Rec. ITU-を参照）。 T X.683 | ISO/IEC 8824-4、8.3)、または「UserDefinedConstraint」です (Rec. ITU-T X.682 | ISO/IEC 8824-3、9 節を参照)。 「GeneralConstraint」は Rec. で定義されています。 ITU-T X.682 | ISO/IEC 8824-3、8.1。

49.7 「SubtypeConstraint」という表記は、汎用の「ElementSetSpecs」という表記です (50 節を参照)。

サブタイプ制約 ::= 要素セット仕様

このコンテキストでは、要素は親タイプの値です (要素セットのガバナーが親タイプです)。セットには少なくとも 1 つの要素が存在する必要があります。

50 要素セット仕様

50.1 一部の表記法では、特定のタイプまたは情報オブジェクト クラス (ガバナ) の要素のセットを指定できます。このような場合、「ElementSetSpec」という表記が使用されます。

要素セット仕様 ::=
RootElementSetSpec
| RootElementSetSpec "," "..."
| RootElementSetSpec "," "..." "," AdditionalElementSetSpec

ルート要素セット仕様 ::= 要素セット仕様

追加要素セット仕様 ::= 要素セット仕様

ElementSetSpec ::= 共用体
|すべての除外事項

ユニオン ::= 交差
| UElems ユニオンマーク交差点

UElems ::= ユニオン

交差点 ::= IntersectionElements
| IElems IntersectionMark IntersectionElements

IElems ::= 交差

IntersectionElements ::= 要素 |要素の除外

要素 ::= 要素

除外 ::= EXCEPT要素

ユニオンマーク ::= "|" |連合

交差点マーク ::= "^" |交差点

注 1 – キャレット文字「^」と単語INTERSECTIONは同義です。文字「|」とUNIONという単語は同義です。文体上、文字または単語のいずれかをユーザー仕様全体で使用することをお勧めします。 EXCEPT はどちらのスタイルでも使用できます。

注 2 – 優先順位は最高から最低まで次のとおりです: 「^」、「|」は除きます。 ALL EXCEPTが指定されているため、 「ALL EXCEPT xxx」を括弧で囲まない限り、他の制約と混在できないことに注意してください。

注 3 – 「要素」が出現する場所には、括弧のない制約 [例: INTEGER (1..4)]または括弧で囲まれたサブタイプ制約 [例: INTEGER ((1..4 | 9))] が出現する可能性があります。

注 4 – 2 つのEXCEPT演算子は「|」、 「^」、 「(」、または「)」で区切る必要があるため、 (A EXCEPT B EXCEPT C)は許可されないことに注意してください。これを((A EXCEPT B) EXCEPT C)または(A EXCEPT (B EXCEPT C)) に変更する必要があります。

注 5 – ((A EXCEPT B) EXCEPT C) は(A EXCEPT (B | C))と同じであることに注意してください。

注 6 – 「ElementSetSpecs」によって参照される要素は、「RootElementSetSpec」および「AdditionalElementSetSpec」（存在する場合）によって参照される要素の結合です。

注 7 – 要素が情報オブジェクトである場合 (つまり、ガバナーが情報オブジェクト クラスである場合)、Rec. で定義されている「ObjectSetElements」という表記が使用されます。 ITU-T X.681 | ISO/IEC 8824-2、12.10 が使用されます。

50.2 セットを構成する要素は次のとおりです。

a) "ElementSetSpec" の最初の選択肢が選択されている場合は、"Unions" で指定されているもの [b) を参照]。それ以外の場合は、ガバナーのすべての要素 (ただし、"Elements" 表記で指定されているものを除く)
「除外」;

b) 「Unions」の最初の選択肢が選択された場合、「Intersections」で指定されたもの [c) を参照]、
それ以外の場合は、「UElems」または「Intersections」のいずれかで少なくとも 1 回指定されたもの。

c) 「Intersections」の最初の選択肢が選択されている場合は、「IntersectionElements」で指定されているもの [d) を参照]。それ以外の場合は、「IElems」で指定されており、「IntersectionElements」でも指定されているもの。

d) 「IntersectionElements」の最初の選択肢が選択されている場合は、「Elements」で指定されているもの、それ以外の場合
「除外項目」で指定されたものを除く、「要素」で指定されたもの。

50.3要素が情報オブジェクトである場合 (つまり、ガバナーが情報オブジェクト クラスである場合)、「ElementSetSpec」の 2 番目の代替は使用されません。

50.4 次の条件が満たされる場合、値のセットは拡張可能であると定義されます。

a) 「ElementsSetSpecs」の場合: 外側のレベルに拡張マーカーがあります。

注 – これは、親のすべての値が新しい制約付きタイプのルートに含まれている場合にも当てはまります。

b) 「Unions」の場合: 「UElem」の少なくとも 1 つは拡張可能です。

c) 「交差」の場合: 「IElem」の少なくとも 1 つが拡張可能です。

d) 「除外」の場合: EXCEPTに先行する一連の要素は拡張可能です。

それ以外の場合、値のセットは拡張できません (I.4 も参照)。

50.5値のセットが拡張可能な場合、ルート値は、50.2 で指定されているように、セット算術に関与する値セットのルート値のみを使用してセット算術を実行することによって決定できます。拡張加算は、セット算術に関与する値のセットごとに、拡張加算によって増補されたルート値を使用してセット算術を実行し、ルート値であると判定された値を除外することによって決定できます。

50.6 「要素」表記は次のように定義されます。

```
要素::=
        サブタイプ要素
|オブジェクトセット要素
| "(" ElementSetSpec ")"
```

この表記法で指定される要素は次のとおりです。

a) 「SubtypeElements」代替手段が使用される場合は、以下の第 51 項に記載されているとおりです。この表記は、ガバナーがタイプである場合にのみ使用され、関係する実際のタイプによって表記の可能性がさらに制限されます。この文脈では、ガバナーは親タイプと呼ばれます。

b) Rec. に記載されているとおり。 ITU-T X.681 | ISO/IEC 8824-2、12.10 (「ObjectSetElements」表記が使用されている場合)。
この表記は、ガバナーが情報オブジェクト クラスである場合にのみ使用されます。

c) 3 番目の選択肢が使用される場合は、「ElementSetSpec」によって指定されるもの。

50.7サブタイプ制約内で集合演算を実行する場合、または支配型が拡張可能でないときに値セットを実行する場合、集合演算では支配型の抽象値のみが使用されます。この場合、集合演算で使用される値表記のすべてのインスタンス (値参照を含む) は、支配型の抽象値を参照する必要があります。範囲制約のエンドポイントは、支配型の値を参照する必要があり、範囲仕様全体として、支配型の抽象値である範囲内のすべて (およびのみ) の値を参照します。

50.8サブタイプ制約内で集合演算を実行する場合、または支配型が拡張可能な場合に値セットを実行する場合、集合演算では支配型の拡張ルートにある抽象値のみが使用されます。この場合、集合演算で使用される値表記のすべてのインスタンス (値参照を含む) は、支配型の拡張ルートの抽象値を参照する必要があります。範囲制約のエンドポイントは、支配型の拡張ルートに存在する値を参照する必要があり、範囲指定は全体として、その拡張ルート内にある範囲内のすべて (およびのみ) の値を参照します。統治タイプ。

50.9情報オブジェクトのセットを含む集合演算を実行する場合、すべての情報オブジェクトが集合演算で使用されます。集合算術に寄与する情報オブジェクト集合のいずれかが拡張可能である場合、または 「ElementSetSpecs」の最外部レベルに拡張マーカーがある場合、集合算術の結果は拡張可能である。

50.10サブタイプ制約が拡張不可能な親タイプに適用される場合、その中で使用される値の表記は、親タイプの抽象値ではない値を参照してはならない。

50.11サブタイプ制約が、拡張可能な制約の適用を通じて拡張可能な親タイプに連続的に適用される場合、その中で使用される値の表記は、親タイプの拡張ルートにない値を参照してはならない。 2 番目の (シリアルに適用された) 制約の結果は、拡張マーカーや拡張子の追加の可能性がない親型に制約が適用された場合と同じになるように定義されます。

例

```
Foo ::= INTEGER ( 1..6, ..., 73..80)
バー ::= Foo (73) --不正です
foo Foo ::= 73 --これは Foo の値表記であり、制約の一部ではないため正当です。
```

73 が Foo の拡張ルートにないため、 Barは不正です。 73 がFooの拡張ルートにあった場合、この例は正当なものとなり、 Bar には 73 という単一の値が含まれることになります。

注 – この副節は「SubtypeConstraint」にのみ適用されます。 「GeneralConstraint」 (Rec. ITU-T X.682 | ISO/IEC 8824-3、8.1 を参照) が親タイプに適用される場合、その親タイプの拡張性は影響を受けません。

51 サブタイプ要素

51.1 一般

「SubtypeElements」のさまざまな表記形式が多数提供されています。それらは以下で識別され、その構文とセマンティクスは次の副節で定義されます。表 11 と表 12 は、どの表記法をどの親タイプに適用できるかをまとめています。いずれかのテーブルに「SubtypeElements」が存在しない場合は、対応するサブタイプ要素をそのテーブルにリストされている親タイプのいずれにも適用できないことを意味します。

```
サブタイプ要素 ::=
        単一値
|含まれるサブタイプ
|値の範囲
|許可されたアルファベット
|       サイズ制約
|タイプ制約
| InnerTypeConstraints
|       パターン制約
|       プロパティ設定
|期間範囲
|時間点範囲
|繰り返し範囲
```

表 11 – Time 型以外の型への「SubtypeElements」の適用性

タイプ (またはタグ付けまたはサブタイプによってそのようなタイプから派生)	シングル価値	含まれているサブタイプ	価値範囲	サイズ制約	許可されるアルファベット	タイプ制約	インナーサブタイプ	パターン制約
ビット列	はい	はい	いいえ	はい	いいえ	いいえ	いいえ	いいえ
ブール値	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
選択	はい	はい	いいえ	いいえ	いいえ	いいえ	はい	いいえ
埋め込み型 pdv	はい	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ
列挙型	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
外部の	はい	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ
インスタンスの	はい	はい	いいえ	いいえ	いいえ	いいえ	はい	いいえ
整数	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ
ヌル	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
オブジェクトクラスのフィールドタイプ	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
オブジェクト記述子	はい	はい	いいえ	はい	はい	いいえ	いいえ	いいえ
オブジェクト識別子	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
オクテット文字列	はい	はい	いいえ	はい	いいえ	いいえ	いいえ	いいえ
OID 国際化リソース識別子	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
オープンタイプ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ	いいえ
本物	はい	はい	はい	いいえ	いいえ	いいえ	はい	いいえ
相対オブジェクト識別子	はいB)	はいB)	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
相対 OID 国際化リソース識別子	はいB)	はいB)	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
制限される文字列の種類	はい	はい	はい)	はい	はい	いいえ	いいえ	はい
順序	はい	はい	いいえ	いいえ	いいえ	いいえ	はい	いいえ
シーケンス	はい	はい	いいえ	はい	いいえ	いいえ	はい	いいえ
セット	はい	はい	いいえ	いいえ	いいえ	いいえ	はい	いいえ
セット	はい	はい	いいえ	はい	いいえ	いいえ	はい	いいえ
一般化された時間と UTCTime タイプ	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
無制限の文字列型	はい	いいえ	いいえ	はい	いいえ	いいえ	はい	いいえ

a) BMPString、 IA5String、 NumericString、 PrintableString、 VisibleString、 UTF8String 、およびUniversalStringの「PermittedAlphabet」内でのみ許可されます。

b)すべての相対オブジェクト識別子および相対 OID 国際化リソース識別子のタイプまたは値の開始ノード
制約または値セットは、ガバナーの開始ノードと同じでなければなりません。

表 12 – Time タイプへの「SubtypeElements」の適用性

タイプ (またはタグ付けまたはサブタイプによってそのようなタイプから派生)	シングル価値	含まれているサブタイプ	プロパティ設定	間隔範囲	時点範囲	再発範囲	内部サブタイピング
時間タイプ	はい	はい	はい	はい	はい	はい	(注記)

注 – 親タイプのすべての抽象値がプロパティ設定「Basic=Interval Interval-type=D」を持つ場合にのみ許可されます(38.4.4 を参照)。

51.2 単一の値

51.2.1 「SingleValue」表記は次のとおりです。

単一値 ::= 値

ここで、「Value」は親タイプの値表記です。

51.2.2 「SingleValue」は、「Value」で指定された親型の単一の値を指定します。

51.3 含まれるサブタイプ

51.3.1 「ContainedSubtype」の表記は次のとおりです。

ContainedSubtype ::= タイプを含む

含まれます ::=含まれます|空の

「ContainedSubtype」の「Type」が null 型の表記である場合、「includes」プロダクションの「空」代替は使用されません。

51.3.2 「ContainedSubtype」は、「Type」のルートにもある親タイプのルートのすべての値を指定します。 「型」は、親型と同じ組み込み型から派生する必要があります。

51.3.3含まれるサブタイプ制約で使用される拡張可能な「タイプ」によって参照される値のセットは、「タイプ」から拡張マーカーを継承しません。その型の拡張ルートにない「Type」の値はすべて無視され、制約された型の値には影響しません。

注 – 拡張可能な「型」を使用しても、それ自体では制約付き型が拡張可能になるわけではありません。

51.4 値の範囲

51.4.1 「ValueRange」の表記は次のとおりです。

ValueRange ::= LowerEndpoint ".." UpperEndpoint

51.4.2 「ValueRange」は、範囲の端点の値を指定することによって指定される値の範囲内の値を指定します。この表記法は、整数型、特定の制限された文字列型 (IA5String、 NumericString、 PrintableString、 VisibleString、 BMPString、 UniversalString 、およびUTF8Stringのみ)の "PermittedAlphabet"、および実数型にのみ適用できます。 「ValueRange」で指定されたすべての値は、親タイプのルートに存在する必要があります。

注 – サブタイプの目的で、 NOT-A-NUMBER はすべての実数値を超え、 PLUS-INFINITY はNOT-A-NUMBERを除くすべての実数値を超え、マイナス ゼロはすべての負の実数値を超え、プラス ゼロより小さく、 MINUS-INFINITY はすべての負の実数値を超えます。すべての実際の値よりも小さいです。それ以外の場合は、通常の数学的順序付けが適用されます。

51.4.3範囲の各端点は、閉じている (その端点が指定されている場合) か、開いている (この場合、端点が指定されていない) かのいずれかです。開いた場合、エンドポイントの仕様には小なり記号 (「<」) が含まれます。

下端点 ::= 下端値 | LowerEndValue "<"

アッパーエンドポイント ::= アッパーエンド値 | "<" 上限値

51.4.4エンドポイントを指定しないこともできます。その場合、範囲は親タイプが許可する限りその方向に拡張されます。

LowerEndValue ::= 値 |最小

上限値 ::= 値 |マックス

注 – 「ValueRange」が「PermittedAlphabet」制約として使用される場合、「LowerEndValue」と「UpperEndValue」のサイズは 1 でなければなりません。

51.5 サイズの制約

51.5.1 「SizeConstraint」の表記は次のとおりです。

サイズ制約 ::=サイズ制約

51.5.2 「SizeConstraint」は、ビット文字列型、オクテット文字列型、文字列型、型のセットまたは型のシーケンスにのみ適用できます。

51.5.3 「制約」は、指定された値の長さとして許可される整数値を指定し、次の親タイプに適用できる任意の制約の形式をとります。

整数 (0 .. 最大)

「Constraint」には、「ConstraintSpec」の代わりに「SubtypeConstraint」を使用します。

51.5.4測定単位は、次のように親タイプによって異なります。

タイプ	測定単位
ビット文字列	少し
オクテット文字列	オクテット
文字列	キャラクター
セットの	成分値
シーケンスの	成分値

注 – 文字列値のサイズを決定するためにこの副節で指定される文字数のカウントは、オクテットのカウントとは明確に区別されなければなりません。文字数は、その型で使用される文字のコレクションの定義に従って、特に、そのような定義に現れる可能性がある規格、テーブル、またはレジスタ内の登録番号への参照に関連して解釈されます。

51.6 型制約

51.6.1 「TypeConstraint」表記は次のとおりです。

TypeConstraint ::= タイプ

51.6.2この表記法はオープン型表記にのみ適用され、オープン型を「Type」の値に制限します。

51.7 許可されるアルファベット

51.7.1 「PermittedAlphabet」表記は次のとおりです。

許可されたアルファベット ::= FROM制約

51.7.2 「PermittedAlphabet」は、親文字列のサブアルファベットを使用して構築できるすべての値を指定します。
この表記は、制限された文字列型にのみ適用できます。

51.7.3 「制約」は、「ConstraintSpec」の代わりに「SubtypeConstraint」を使用するものとします。 「SubtypeConstraint」内の各「SubtypeElements」は、「SingleValue」、「ContainedSubtype」、「ValueRange」、および「SizeConstraint」の 4 つの選択肢のいずれかになります。サブアルファベットには、「制約」によって許可される親文字列型の 1 つ以上の値に現れる文字が正確に含まれます。

51.7.4 「制約」が拡張可能な場合、許可されたアルファベット制約によって選択された値のセットは拡張可能です。
ルート内の値のセットは、「制約」のルートによって許可される値であり、拡張追加は、ルートに既に存在する値を除いた、「制約」の拡張追加とともにルートによって許可される値です。

51.8 内部サブタイピング

51.8.1 「InnerTypeConstraints」の表記は次のとおりです。

InnerTypeConstraints ::=
WITH COMPONENT SingleTypeConstraint
|コンポーネント付き

51.8.2 「InnerTypeConstraints」は、親タイプのコンポーネントの存在および/または値に関する制約のコレクションを満たす値のみを指定します。親タイプの値は、明示または暗黙のすべての制約を満たさない限り、指定されません (51.8.7 を参照)。この表記は、set-of、sequence-of、set、sequence、choice タイプに適用できます。

注 – セットまたはシーケンス型に適用された「InnerTypeConstraints」は、 COMPONENTS OF変換では無視されます (25.5 および 27.2 を参照)。

51.8.3 「InnerTypeConstraints」に「GeneralConstraint」（49.6 項を参照）が含まれる場合、それはプロダクションの最初の代替手段「ElementSetSpecs」（50 項を参照）の一部として（直接的または間接的に）のみ使用されます。 「ObjectSetSpec」 (Rec. ITU-T X.681 | ISO/IEC 8824-2、12.3 を参照)、およびプロダクションの最初の代替案である「ElementSetSpec」、「Unions」、「Intersections」、および「IntersectionElements」 (節を参照) 50）。

51.8.4単一の他の (内部) 型 (set-of および sequence-of) に関して定義される型の場合、サブタイプ値仕様の形式をとる制約が提供されます。これの表記法は「SingleTypeConstraint」です。

SingleTypeConstraint ::= 制約

「制約」は、単一の他の (内部) 型のサブタイプを定義します。親型の値は、各内部値が内部型に「制約」を適用することによって取得されたサブタイプに属する場合にのみ指定されます。

51.8.5複数の他の (内部) 型 (choice、set、sequence) に関して定義されている型の場合、これらの内部型に対する多くの制約を提供できます。これの表記法は「MultipleTypeConstraints」です。

MultipleTypeConstraints ::=
フルスペック
| 部分仕様

フルスペック ::= "{" TypeConstraints "}"

PartialSpec ::= "{" "..." "," TypeConstraints "}"

タイプ制約::=
名前付き制約
| NamedConstraint "," TypeConstraints

名前付き制約::=
識別子ComponentConstraint

51.8.6 「TypeConstraints」には、親タイプのコンポーネントタイプに対する制約のリストが含まれています。シーケンス タイプの場合、制約は順番に現れる必要があります。制約が適用される内部型は、その識別子によって識別されます。特定のコンポーネントに対して、最大 1 つの「NamedConstraint」が存在する必要があります。

51.8.7 「MultipleTypeConstraints」は、「FullSpecific」または「PartialSpecific」のいずれかで構成されます。 「FullSpecific」が使用される場合、存在しないように制約できる (51.8.10 を参照) が明示的にリストされていないすべての内部型にABSENTという暗黙の存在制約があります。 「PartialSpecific」が使用される場合、暗黙の制約はなく、任意の内部型をリストから省略できます。

51.8.8特定の内部型は、その存在 (親型の値内)、その値、またはその両方に関して制約される場合があります。表記は「ComponentConstraint」です。

ComponentConstraint ::= ValueConstraint PresenceConstraint

51.8.9内部型の値に対する制約は、「ValueConstraint」という表記で表されます。

値制約 ::= 制約 |空の

制約は、内部値が内部型に適用される「制約」によって指定されたサブタイプに属している場合にのみ、親型の値によって満たされます。

51.8.10内部型の存在に関する制約は、「PresenceConstraint」という表記で表現されます。

プレゼンス制約 ::=現在|不在|オプション|空の

これらの代替手段の意味、およびそれらが許可される状況は、51.8.10.1 から 51.8.10.3 で定義されています。

51.8.10.1親タイプがシーケンスまたはセットである場合、 OPTIONALとマークされたコンポーネントタイプは、 PRESENT (この場合、対応するコンポーネント値が存在する場合にのみ制約が満たされます) またはABSENT (この場合)に制約される場合があります。この場合、制約は対応するコンポーネント値が存在しない場合にのみ満たされます）、またはOPTIONALになります（この場合、対応するコンポーネント値の存在には制約が課されません）。

51.8.10.2親タイプが選択である場合、コンポーネントタイプはABSENT (この場合、対応するコンポーネントタイプが値で使用されていない場合にのみ制約が満たされます)、またはPRESENT (この場合)に制約できます。制約は、対応するコンポーネント タイプが値で使用されている場合にのみ満たされます)。「MultipleTypeConstraints」にはPRESENTキーワードが1 つだけ存在します。

注 – 明確な例については、G.5.6 を参照してください。

51.8.10.3空の「PresenceConstraint」の意味は、「FullSpecific」が使用されているか「PartialSpecific」が使用されているかによって異なります。

a) 「完全仕様」では、これはセットまたはシーケンスコンポーネントに対するPRESENTの制約と同等です。
OPTIONALとマークされており、それ以外の場合はそれ以上の制約は課されません。

b) 「部分仕様」では、制約は課されません。

51.9 パターン制約

51.9.1 「PatternConstraint」の表記は次のとおりです。

PatternConstraint ::= PATTERN値

51.9.2 「Value」は、付録 A で定義されている ASN.1 正規表現を含むUniversalString型の「cstring」（またはそのような文字列への参照）でなければなりません。 「PatternConstraint」は、親型の値を選択します。 ASN.1 正規表現を満たすもの。値全体が ASN.1 正規表現全体を満たす必要があります。つまり、「PatternConstraint」は、先頭の文字が (全体) ASN.1 正規表現に一致するが、末尾の文字がさらに含まれる値を選択しません。

注 – 「値」は正式にはUniversalString型の値として定義されていますが、 UniversalString型とUTF8String型の値のセットは同じです (41.16 を参照)。したがって、完全に同等の定義は、「Value」が UTF8String 型の値であるということになる可能性があります。

51.10 プロパティ設定

51.10.1 「PropertySettings」の表記は次のとおりです。

PropertySettings ::= SETTINGS単純文字列

51.10.2 「simplestring」の内容は「PropertySettingsList」となります。

プロパティ設定リスト ::=
プロパティと設定ペア
| PropertySettingsList PropertyAndSettingPair

PropertyAndSettingPair ::= プロパティ名 "= 設定名

プロパティ名 ::= psname

設定名 ::= psname

51.10.3 「PropertyName」は、表 6 の列 1 にリストされている時間プロパティ名の 1 つであり、「PropertySettingsList」に最大 1 回出現するものとします。

51.10.4 「PropertyAndSettingPair」の「SettingName」は、その「PropertyAndSettingPair」の「PropertyName」を含む行（列 1 に）の表 6 の列 2 にリストされているプロパティ設定名の 1 つであるものとします。

51.10.5抽象値は、すべての「PropertyAndSettingPair」について次の条件を満たす場合にのみ、サブタイプに含まれます。どちらか：

a) 抽象値には「PropertyName」のプロパティ設定がありません (次の列 2 と列 3 を参照)
表 6 は、特定の「PropertyName」のプロパティ設定を持つ抽象値を示します。または

b) 抽象値には、「SettingName」と同じプロパティ設定があります。

注 – 人間が読みやすいように、 Basic time プロパティの設定を常に最初の "PropertyAndSettingPair" として含めることをお勧めしますが、必須ではありません。

例: TIME(SETTINGS "Midnight=Start") は、プロパティ設定"Midnight=End"を除くすべての抽象値 (日付のみを表す値を含む) が存在するTIME型のサブセットを生成します。

51.10.6 TIME型のすべての抽象値には、基本時間プロパティの設定があります(これは他の時間プロパティには当てはまりません)。 "PropertyAndSettingPair" が結果の抽象値のセットに影響を与えないという誤解を招く表記を防ぐために、Basic timeプロパティの特定の設定で使用できる "PropertyName" にいくつかの制限が設けられています。制限事項を表 13 に示します。

注 – 表 13 は、「PropertyAndSetting」ペアの使用を防止するためのルールの完全なセットではなく、その一部は冗長です (一般に違法ではありません)。

表 13 –基本プロパティ設定でのプロパティ名の使用の制限

基本的なプロパティの設定	この基本プロパティ設定で禁止されているプロパティ名
日付	時刻、ローカルまたは UTC、深夜、間隔タイプ、 SEポイント、再発
時間	日付、年、インターバルタイプ、 SEポイント、繰り返し
日付時刻	インターバル型、 SEポイント、再発
間隔	再発
記録間隔	制限なし

51.11 持続時間の範囲

51.11.1 「DurationRange」サブタイプの表記は次のとおりです。

期間範囲 ::= 値範囲

51.11.2 「ValueRange」の両方の「Value」は、サブタイプに存在する時間抽象値を（値表記または値参照によって）識別します。

TIME(SETTINGS "基本=インターバル インターバルタイプ=D")

51.11.3 「ValueRange」の両方の「Value」は、次のいずれかを使用して期間を指定します。

a) 同じ単一時間成分を同じ精度で使用する（小数部分がない、または小数部分の桁数が同じ）。または

b) 最下位の時間コンポーネントを除いて同一の値を持つ複数の時間コンポーネント (値は異なる場合がありますが、精度は同じでなければなりません)。

51.11.4選択された期間抽象値は次のとおりです。

a) 「ValueRange」内の 2 つの「Value」の同一時間コンポーネントに同じ値があります。そして

b) 内の 2 つの「値」の最下位時間コンポーネントが指定された範囲内にある。
"値の範囲";そして

c) 内の 2 つの「値」の最下位時間コンポーネントと同じ精度を持っています。
「値の範囲」。

注 – これは、指定された精度で単一の時間コンポーネントのみを使用する期間を指定する手段として、内部サブタイピング (38.4.4 を参照) の使用に代わる方法を提供します。

例: TIME("PT2M0.000S".."PT2M59.000S") は、 2 分と 0 ～ 59 秒の期間を 1 ミリ秒の精度で表す抽象値のみで構成されるTIMEサブタイプを定義します。

51.12 時点範囲

51.12.1 「TimePointRange」の表記は次のとおりです。

TimePointRange ::= ValueRange

51.12.2 「ValueRange」の両方の「Value」は、サブタイプに存在する時間抽象値 (値表記または値参照) を識別します。

時間 ((設定「基本=日付」)
|(設定「基本=時間」)
|(設定 "基本=日付/時刻"))

51.12.3 「ValueRange」内の 2 つの「Value」は、深夜時間プロパティを除くすべての時間プロパティで同一の設定を持つものとします。

51.12.4 「ValueRange」内の 2 つの「Value」のプロパティ設定が「Local-or-UTC=LD」である場合、2 つの「Value」の時差は同じでなければなりません。

注 – これにより、たとえば次のようなサブタイプが可能になります。

時間 ("00:00".."09:00")

または：

TIME (「21:00」..「24:00」)。

51.12.5このサブタイプ表記は、すべての時間プロパティ (深夜時間プロパティを除く) の設定が、「ValueRange」の「値」の設定と同一である (また、同じ時差がある場合は同じ時差を持つ) 抽象値を親タイプから選択します。 「Value」にLocal-or-UTC=LDのプロパティ設定があり、指定された「ValueRange」内の値を持ちます (51.4 を参照)。

注 – 関連するすべての抽象値の要件は、すべての時間プロパティ (深夜を除く) に対して同一の設定を持つことです。
time プロパティ)、それらに時間差がある場合、それは同じ時間差であるため、すべてが同じ時間スケールを使用していることが保証され、したがってそれらの間に順序関係が存在することが保証されます。

51.13 再発範囲

51.13.1 「RecurrenceRange」の表記は次のとおりです。

RecurrenceRange ::= ValueRange

51.13.2 「ValueRange」の「Value」は両方とも整数値でなければなりません。

51.13.3順序付けのみを目的として、 「Recurrence=Unlimited」のプロパティ設定を持つ時間値は、無限回数の繰り返し (MAX の整数値) を指定するものとして扱われます。

51.13.4このサブタイプ表記は、サブタイプにも存在する抽象値を親タイプから選択します。

時間(設定「基本=記録間隔」)

指定された「ValueRange」内にある繰り返し桁数を持つもの (51.4 を参照)。

52 拡張マーカー

注 – 一般的な制約表記と同様、拡張マーカーは、ASN.1 の一部のエンコーディング ルール (基本エンコーディング ルールなど) には影響しませんが、パック エンコーディング ルールなどの他のエンコーディング ルールには影響します。 ECN を使用して定義されたエンコーディングに対するその影響は、ECN 仕様によって決まります。

52.1拡張マーカーの省略記号は、拡張の追加が予想されることを示します。このような追加がデコードプロセス中にエラーとして扱われないこと以外に、そのような追加がどのように処理されるべきかについては何も述べられていません。

52.2 拡張マーカーと例外識別子の併用 (第 53 条を参照) は、拡張の追加が期待されていることを示すとともに、制約違反があった場合にアプリケーションがとるべきアクションを識別する手段も提供します。ストア アンド フォワードやその他の形式の中継が使用されている状況では、(たとえば) 認識されない拡張子の追加が再エンコードの可能性のためにアプリケーションに返されることを示すために、この表記を使用することをお勧めします。そして中継する。

52.3拡張可能なサブタイプ制約、値セット、または情報オブジェクトセットを含むセット算術の結果は、条項 50 で指定されます。

張機 拡張可能な制約で定義された型が「ContainedSubtype」で参照される場合、新しく定義された型52.4は、拡張マーカーまたはその拡能の追加を継承しません (51.3.3 を参照)。新しく定義された型は、その「ElementSetSpecs」の最外部レベルに拡張マーカーを含めることによって拡張可能にすることができます (50.4 も参照)。例えば：

```
A ::= INTEGER (0..10, ..., 12) -- A は拡張可能です。
B ::= 整数 (A) -- B は拡張不可能で、0 ～ 10 に制限されます。
C ::= 整数 (A, …) -- C は拡張可能ですが、0 ～ 10 に制限されます。
```

52.5 拡張可能な制約で定義された型が「ElementSetSpecs」でさらに制約される場合、結果の型は拡張マーカーも、前の制約に存在する可能性のある拡張の追加も継承しません (50.11 を参照)。例えば：

```
A ::= INTEGER (0..10, ...) -- A は拡張可能です。
B ::= A (2..5) -- B は拡張不可能です。
C ::= A -- C は拡張可能です。
```

52.6存在しないように制約されているセット、シーケンス、または選択タイプのコンポーネントは、セット、シーケンス、または選択タイプが拡張可能なタイプであるかどうかに関係なく、存在してはならない。

注 – 内部型制約は拡張性に影響しません。

例えば：

```
A ::= シーケンス {
        整数
        b ブール型オプション、
        …
        }

B ::= A (コンポーネント {b なし} あり)
                -- B は拡張可能ですが、「b」は拡張できません
                -- いずれかの値に存在します。
```

52.7 この推奨事項 |国際規格では個別のタグが必要です (25.6 ～ 25.7、27.3、および 29.3 を参照)。
タグの一意性のチェックを実行する前に、概念的には次の変換が適用されます。

52.7.1新しい要素または代替要素 (概念的に追加された要素と呼ばれます。52.7.2 を参照) は、次の場合に拡張挿入ポイントに概念的に追加されます。

a) 拡張マーカーはありませんが、拡張性がモジュール ヘッダーに暗示されており、拡張マーカーが追加され、その拡張マーカーの後の最初の追加として新しい要素が追加されます。または

b) CHOICE 、 SEQUENCE 、またはSET内に拡張マーカーが 1 つあり、新しい要素がCHOICE 、 SEQUENCE 、またはSETの終わりの閉じ中括弧の直前に追加されます。または

c) CHOICE 、 SEQUENCE 、 SETに 2 つの拡張マーカーがあり、 2 番目の拡張マーカーの直前に新しい要素が追加されます。

52.7.2この概念的に追加された要素は、個別のタグを必要とするルールの適用を通じて合法性をチェックすることのみを目的としています（25.6 ～ 25.7、27.3 および 29.3 を参照）。これは、自動タグ付け (該当する場合) の適用と COMPONENTS OF の拡張の後に概念的に追加されます。

52.7.3概念的に追加された要素は、すべての通常の ASN.1 タイプのタグとは異なるタグを持つように定義されますが、このタグは、そのようなすべての概念的に追加された要素のタグと一致し、オープン タイプの不確定タグと一致します。 Rec.に指定されているとおり。 ITU-T X.681 | ISO/IEC 8824-2、14.2、注 2。

注 – 概念的に追加された要素およびオープン型に関連するタグの一意性に関する規則は、別個のタグを必要とする規則 (25.6 ～ 25.7、27.3 および 29.3 を参照) とともに、次のことを保証するために必要かつ十分です。

a) BER エンコードが行われている場合、未知の拡張子の追加は、明確に単一の挿入ポイントに起因すると考えられます。
デコードされた。そして

b) 不明な拡張機能の追加をOPTIONAL要素と混同することはできません。

PER では上記のルールで十分ですが、これらのプロパティを保証するためには必須ではありません。それにもかかわらず、それらはルールとして課せられます
ASN.1 のエンコード規則からの表記の独立性を確保するため。

52.7.4これらの概念的に追加された要素によって、個別のタイプを必要とするルールに違反する場合、仕様は拡張性表記を違法に使用したことになります。

注 – 上記のルールの目的は、挿入ポイント (特にSEQUENCE 、 SET 、またはCHOICEの最後にない挿入ポイント) の使用から生じる正確な制限を設けることです。この制限は、BER、DER、および CER において、バージョン 1 システムが受信した未知の要素を特定の挿入ポイントに明確に帰属させることができるように設計されています。このような追加された要素の例外処理が挿入ポイントごとに異なる場合、これは重要になります。

52.8 例

52.8.1 例 1

```
A ::= セット {
...     あ、
b 選択肢 {
        c       C、
        dD、
        ...
    }
}
```

追加されたマテリアルはbの一部でなければならないため、曖昧さはなく、合法です。

52.8.2 例 2

```
A ::= セット {
...     あ、
b 選択肢 {
                C、
        CDD、
        ...
    }、
    ...、
    d D
}
```

は違法です。追加されたマテリアルはb の一部であるか、または A の外側レベルにある可能性があり、バージョン 1 システムではどちらであるかを判断できません。

52.8.3 例 3

```
A ::= セット {
...     あ、
b 選択肢 {
        c       C、
        ...
    }、
    d 選択肢 {
        e       E、
        ...
    }
```

}

追加された素材がbまたはd の一部である可能性があるため、これも違法です。

52.8.4拡張可能な選択肢内に拡張可能な選択肢を使用したり、 OPTIONALまたはDEFAULT とマークされたシーケンスの要素内に拡張可能な選択肢を使用して、より複雑な例を構築することもできますが、バージョン 1 に存在しない要素を確実に使用できるようにするには、上記のルールが必要かつ十分です。バージョン 1 システムでは、正確に 1 つの挿入ポイントに起因すると明確に認識されています。

53 例外識別子

53.1複雑な ASN.1 仕様では、デコーダがその中で完全には規定されていない内容を処理する必要があることが明確に認識されている箇所が数多くあります。このようなケースは、特に、抽象構文のパラメーターを使用して定義された制約の使用から発生します (Rec. ITU-T X.683 | ISO/IEC 8824-4、第 10 項を参照)。

53.2 このような場合、アプリケーション設計者は、実装に依存する制約に違反したときに実行するアクションを特定する必要があります。例外識別子は、実行されるアクションを示すために ASN.1 仕様の一部を参照する明確な手段として提供されます。識別子は「!」で構成されます。文字の後に、オプションの ASN.1 タイプとそのタイプの値が続きます。型が指定されていない場合は、 INTEGER が値の型として想定されます。

53.3 「ExceptionSpec」が存在する場合、 「!」に関連付けられた制約違反を処理する方法を示すテキストが標準の本文にあることを示します。キャラクター。これが存在しない場合、実装者は、実行するアクションを説明するテキストを特定するか、制約違反が発生したときに実装に依存したアクションを実行する必要があります。

53.4 「ExceptionSpec」表記は次のように定義されます。

例外仕様 ::= "!"例外識別 |空の

例外識別 ::=
署名付き番号
|定義された値
| 「:」値を入力します

最初の 2 つの選択肢は、整数型の例外識別子を示します。 3 番目の選択肢は、任意のタイプ (「Type」) の例外識別子 (「Value」) を示します。

53.5 型が複数の制約によって制約されており、そのうちの 1 つ以上に例外識別子がある場合、
最も外側の制約内の例外識別子は、その型の例外識別子とみなされます。

53.6集合演算で使用される型に例外マーカーが存在する場合、例外識別子は無視され、集合演算の結果として制約される型には継承されません。

54 エンコード制御セクション

54.1 「EncodingControlSections」は次のプロダクションによって指定されます。

エンコーディング制御セクション ::=
EncodingControlSection EncodingControlSections |空

エンコーディング制御セクション ::=
エンコーディング制御

エンコーディングリファレンス
エンコーディング命令割り当てリスト

54.2 ASN.1 モジュール内の各「EncodingControlSection」は、異なる「encodingreference」を持ち、そのエンコード参照のエンコード命令をモジュール内の 1 つ以上のタイプに割り当てます。

54.3 「encodingreference」はTAGであってはなりません。

「EncodingstructAssignmentList」の生成と関連するセマンティクスは、 54.4 Recommendation |で指定されています。
「encodingreference」（付録 E を参照）によって識別される国際標準であり、ASN.1 語彙項目（コメント、cstring、および空白を含む）の任意のシーケンスで構成できます。ただし、語彙項目ENDおよびENCODING-CONTROLは除きます。 「EncodingInstructionAssignmentList」。

注 1 – この推奨事項の将来のバージョン |国際標準は、付録 E にさらにエンコード参照を追加する可能性があります。ASN.1 ツールは、「EncodingControlSection」の「encodingreference」が付録 E で指定されているものの 1 つではない場合に警告 (のみ) を提供し、「EncodingControlSection」を無視することをお勧めします。次にENDまたはENCODING-CONTROL が発生するまで、どちらか先に起こるまで。

注2 – 「EncodingControlSection」内の「encodingreference」は省略できません。モジュールのデフォルトのエンコーディング参照は、「EncodingControlSection」には影響しません。

54.5 型プレフィックスおよび「EncodingControlSection」を使用した、エンコード命令 (同じエンコード参照を持つ) の型への割り当てには、相互作用と制限があります。 (スタイルの問題として) "EncodingControlSection" のみを使用することは常に可能ですが、一般に、"EncodingControlSection" でのみ割り当てることができるエンコード命令 (特にモジュール内のすべての型に適用されるもの) がいくつかあります。特定の命令または命令の組み合わせを適用できる種類にも制限があります。これらの相互作用と制限は、推奨事項で指定されています。エンコーディング参照に関連する国際標準 (付録 E を参照) であり、この勧告では指定されていません。国際標準。

付録A

ASN.1正規表現

(この付録は、この勧告 | 国際規格の不可欠な部分を形成します。)

A.1 意味

A.1.1 ASN.1 正規表現は、このパターンに準拠する形式を持つ文字列のセットを記述するパターンです。
正規表現自体は文字列です。これは、さまざまな演算子を使用して小さな式を組み合わせることで、算術式と同様に構築されます。 (通常) 1 つまたは 2 つの文字で構成される最小の式は、一連の文字を表すプレースホルダーです。

ここで紹介する正規表現は、Perl などのスクリプト言語の正規表現や XML スキーマの正規表現に非常に似ており、他の使用例もいくつかあります。

A.1.2すべての文字と数字を含むほとんどの文字は、それ自体と一致する正規表現です。

例

正規表現「fred」は文字列「fred」のみに一致します。

A.1.3 2 つの正規表現を連結できます。結果の正規表現は、連結された部分表現にそれぞれ一致する 2 つの部分文字列を連結して形成された任意の文字列と一致します。

A.2 メタキャラクター

A.2.1メタキャラクター シーケンス (またはメタキャラクター) は、正規表現のコンテキストで特別な意味を持つ 1 つ以上の連続した文字のセットです。次のリストには、すべてのメタキャラクター シーケンスが含まれています。それらの意味については、次の節で説明します。

[ ]		範囲が「-」で示されているセット内の任意の文字と一致します。 左括弧の後の「^」は、それに続くセットを補完します。
{g、p、r、c}		ISO/IEC 10646 の文字を識別する Quadruple (41.8 を参照)
\N{名前}		指定された文字 (または指定された文字セットの任意の文字) と一致します。 A.2.4
。		任意の文字と一致します(12.1.6で定義された改行文字のいずれかである場合を除く)
\d		任意の数字と一致します (「[0-9]」に相当)
\w		任意の英数字と一致します (「[a-zA-Z0-9]」と同等)
\t		HORIZONTAL TABULATION (9) 文字と一致します (12.1.6 を参照)
\n		12.1.6 で定義された改行文字のいずれかと一致します
\r		CARIAGE RETURN (13) 文字と一致します (12.1.6 を参照)
\s		任意の空白文字と一致します (12.1.6 を参照)
\b		単語の境界を一致させる
\		(プレフィックス) 次のメタキャラクタを引用符で囲み、文字通りに解釈させます。
\\		REVERSE SOLIDUS (92) 文字「\」と一致します。
「」		引用符 (34) 文字(")と一致します。
|	(中置)	2 つの式の代替
( )		囲まれた式のグループ化
*		(後置) 前の式と 0 回、1 回または複数回一致します。
+		(後置) 前の式と 1 回または複数回一致します。
?		(後置) 前の式と 1 回一致するか、まったく一致しません
#n		(後置) 前の式と正確に n 回一致します (n は 1 桁の数字)
#(n)		(後置) 前の式と正確に n 回一致します
#(n,)		(後置) 前の式と少なくとも n 回一致します
#(n,m)		(後置) 前の式と少なくとも n 回、最大 m 回一致します。
#(,m)		(後置) 前の式との一致は m 回以内です

注 1 – 文字 CIRCUMFLEX ACCENT (94) "^"および HYPHEN-MINUS (45) "-"は、A.2.2 で定義された文字列の特定の位置にある追加のメタ文字です。

注 2 – この付録の文字名の後の丸括弧内の値は、ISO/IEC 10646 における文字の 10 進数値です。

注 3 – この表記法では、文字列の先頭と末尾にそれぞれ一致するメタ文字「^」と「$」は提供されません。したがって、文字列は、正規表現の先頭、末尾、または両側に「.*」が含まれている場合を除き、正規表現全体と一致します。

注 4 – 次のメタ文字シーケンスには、空白が改行の直前または直後に表示されない限り、空白を含めることはできません (12.1.6 を参照)。

{g,p,r,c}
\N{名前}
#n
#(n)
#(n,)
#(n,m)
#(,m)

正規表現に改行が含まれている場合、改行の直前または直後に現れるスペース文字は意味を持たず、何も一致しません (12.14.1 を参照)。

A.2.2 「[」と「]」で囲まれた文字のリストは、そのリスト内の任意の 1 文字と一致します。リストの最初の文字がキャレット「^」の場合、リストにない任意の文字と一致します。文字の範囲は、(43.3 で定義された順序関係に従って) ハイフンで区切って最初と最後の文字を指定することによって指定できます。 「]」と「\」を除くすべてのメタキャラクター シーケンスは、リスト内での特別な意味を失います。リテラルの CIRCUMFLEX ACCENT (94) "^" を含めるには、最初の位置以外の任意の場所に配置するか、その前にバックスラッシュを付けます。リテラルのハイフンマイナス (45) "-" を含めるには、リストの最初または最後に置くか、その前にバックスラッシュを置きます。リテラルの CLOSING SQUARE BRACKET (93) "]" を含めるには、それを最初に配置します。リストの最初の文字がキャレット「^」の場合、そのキャレットの直後にある文字「-」と「]」もそれら自体と一致します。 A.2.3、A.2.4、A.2.6、および A.2.7 で定義されたメタキャラクター シーケンスは、その意味が保たれる角括弧内で使用できます。

例

正規表現「[0123456789]」、または同等の「[0-9]」は、任意の 1 桁の数字と一致します。

正規表現「[^0]」は、 0を除く任意の 1 文字と一致します。

正規表現「[\d^.-]」は、任意の 1 つの数字、キャレット、ハイフン、またはピリオドと一致します。

A.2.3同じグリフを持つ 2 つの ISO/IEC 10646 文字間のあいまいさを避けるために、2 つの表記法が提供されています。 「{group,plane,row,cell}」という形式の表記は、 41.8 で定義されている「Quadruple」プロダクションに従って (単一の) 文字を参照します。

A.2.4 「valuereference 」が現在のモジュールで定義またはインポートされているサイズ 1 の制限された文字列値 (41 節を参照) への参照である場合、「\N{valuereference}」形式の表記は参照文字と一致します。 「\N{typereference}」という形式の表記は、 「typereference」が現在のモジュールで定義されている「RestrictedCharacterStringType」のサブタイプへの参照である場合、または「RestrictedCharacterStringType」の 1 つである場合、参照される文字セットの任意の文字と一致します。 「第 41 条で定義されています。

正規表現「\N{LetterUppercase}」および「\N{Lu}」は、 Unicode 標準で定義されている一般カテゴリ「Letter、uppercase」(「Lu」と省略)の任意の (単一)文字に一致します。

注 – 特に、「valuereference」または「typereference」は、モジュールASN1-CHARACTER-MODULE (42.1 を参照) で定義され、現在のモジュール (41.8 を参照) にインポートされる参照の 1 つにすることができます。

例

正規表現「\N{greekCapitalLetterSigma}」は、ギリシャ語の大文字のシグマに一致します。

正規表現「\N{BasicLatin}」は、 BASIC LATIN 文字セットの任意の (単一の) 文字と一致します。

\N{Cyrillic}\N{BasicGreek}]+"、または\N{Cyrillic} | \N{BasicGreek})+」は、次の 同等の「(\N{BasicLatin} | "[\N{BasicLatin}
正規表現です。指定された 3 つの文字セットの任意の (null 以外の) 数の文字で構成される文字列と一致します。

A.2.5ピリオド「。」 12.1.6 で定義された改行文字のいずれかである場合を除き、任意の 1 文字と一致します。

A.2.6記号「\d」は「[0-9]」の同義語です。つまり、任意の 1 桁に一致します。記号「\t」は、HORIZONTAL TABULATION (9) 文字と一致します。記号「\w」は「[a-zA-Z0-9]」の同義語です。つまり、任意の 1 つの文字 (小文字または大文字) または任意の 1 つの数字に一致します。

例

正規表現「\w+(\s\w+)*\」。少なくとも 1 つの (英数字) 単語で構成される文に一致します。
12.1.6 で定義されているように、単語は 1 つの空白文字で区切られます。終了ピリオドの前に空白文字はありません。

A.2.7記号「\r」は、CARRIAGE RETURN (13) 文字と一致します。記号「\n」は、12.1.6 で定義されている改行文字のいずれかと一致します。記号「\s」は、12.1.6 で定義されている空白文字のいずれかと一致します。記号「\b」は、単語の先頭または末尾にある空の文字列と一致します。

例

正規表現「.*\bfred\b.*」は、 「fred」という単語を含む任意の文字列と一致します(この単語は一連の 4 文字だけではなく、区切り文字で区切られています)。したがって、 「fred」や「I am fred the first」のような文字列には一致しますが、 「My name is freddy」や「I am afred I don't know how to spaid 'afraid'!」のような文字列には一致しません。

A.2.8通常メタキャラクタとして機能する文字は、接頭辞「\」を付けることで文字通り解釈できます。
正規表現に引用符 (34) が含まれる場合、この文字は 2 つの引用符文字で表されます。

例

正規表現「\.」 (単一の) 文字列「.」には一致しますが、単一文字の文字列には一致しません。

正規表現「""」は、単一の引用符を含む文字列と一致します。

正規表現「\)」は文字列「)」と一致します。

正規表現「\a」は文字「a」と一致します。

注 – 4 番目の例は、メタキャラクタではない文字の前にバックスラッシュを使用できることを示していますが、この使用は非推奨です (この勧告 | 国際標準の将来のバージョンでは他のメタキャラクタが許可される可能性があるため)。

A.2.9 2 つ以上の正規表現を中置演算子「|」で結合できます。結果の正規表現は、いずれかの部分表現に一致する任意の文字列と一致します。

A.2.10繰り返し演算子で終わらない正規表現の後には、繰り返し演算子が続く場合があります。演算子が「?」の場合、前の項目はオプションであり、最大 1 回一致します。演算子が「*」の場合、前の項目は 0 回以上一致します。演算子が「+」の場合、前の項目が 1 回以上一致します。演算子の形式が「#(n)」の場合、前の項目は正確に n 回一致します。この特定のケースでは、n が 1 桁の場合は括弧を省略できます。 「#(n,)」の形式の場合、項目は n 回以上一致します。 「#(,m)」の形式の場合、項目はオプションであり、最大 m 回一致します。最後に、形式が「#(n,m)」の場合、項目は少なくとも n 回、最大 m 回一致します。

注 – メタキャラクター「*」、 「+」、 「?」を使用することは違法です。または「#」を正規表現の最初の文字として、または繰り返し演算子の直後に使用します。メタ文字「#」または「|」を使用することも違法です。正規表現の最後の文字として使用します。

例

「555-1212」のような電話番号は、正規表現「\d#3-\d#4」、または同等の「\d#(3)-\d#(4)」によって照合されます。

「$12345.90」のようなドル単位の価格は、正規表現「$\d#(1,)(\.\d#(1,2))?」と一致します。
「#」記号の後に範囲が続く場合は、その後に括弧が必要であることに注意してください。

「123-45-5678」のような社会保障番号は、正規表現「\d#3-?\d#2-?\d#4」と一致します。

A.2.11繰り返し（A.2.10 を参照）は連結（A.1.3 を参照）より優先され、連結は交互（A.2.9 を参照）よりも優先されます。これらの優先規則をオーバーライドするために、部分式全体を括弧で囲むことができます。

A.2.12正規表現に括弧内の部分式が含まれる場合、各 (引用符で囲まれていない) 開き括弧には、正規表現の左から右に個別の (厳密に正の) 整数が連続的に割り当てられます。各部分式は、関連付けられた整数を使用する「\1」、 「\2」のような表記を使用してコメント内で参照できます。空の部分式「()」は許可されません。

例

```
"((\d#2)(\d#2)(\d#4))" -- \1 は日付です。\2 は月、\3 は日です。
                       -- そして年に 4 円。
```

注 – さまざまな目的で、正規表現の部分表現への正式な参照が必要です。そのような例の 1 つは、ASN.1 モジュール内の正規表現を文書化するテキストを記述する必要があることです。これは、そのような参照を提供するために使用できる表記です。この表記法は、この推奨事項の他の場所では使用されていません。国際標準。

付録B

定義された時間タイプ

(この付録は、この勧告 | 国際規格の不可欠な部分を形成します。)

B.1 一般的な

B.1.1この付録には、定義された時間タイプを指定する ASN.1 モジュールが含まれています。これらのタイプは、ASN.1 仕様にインポートしてその仕様で使用したり、追加の時間タイプを定義するためのモデルとして使用したりできます。インポートしないと使用できません。

このモ 場合によっては、定義された時刻タイプは、日付または時刻のサブセットのいずれかでサブタイプ化されている場合にのみ役に立ちます。
ジュールで指定されているB.1.2 (または両方)。この場合は、型の定義に明確に記載されています。

例: 使用

アプリケーション日時 ::= 日時(年-月-日-サブセット)(秒-サブセット)

年、月、日、時、分、秒の日時を定義します。これを使用するには、タイプと 2 つのサブタイプをインポートする必要があります。

B.2 ASN.1 で定義された時間タイプ モジュール

DefinedTimeTypes {joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) 定義されたタイプ-モジュール(3)}

定義 自動タグ ::= 開始

すべてをエクスポートします。

```
-- 日付タイプ

CENTURY ::= TIME((SETTINGS "基本=日付 日付=C 年=基本")|
                 (設定 "基本=日付 日付=C 年=Proletic"))

ANY-CENTURY ::= TIME((SETTINGS "基本=日付 日付=C 年=負")|
                     (設定 "基本=日付 日付=C 年=L5"))
  -- これにより、正の場合は 3 桁の世紀のみが許可されます。
  -- 桁数が多い型は、
  -- 追加の時間タイプとして定義されます。
  -- 年の指定に 5 桁が必要な場合は、L5 が世紀に使用されるこ
  とに注意してください。表 6 を参照してください。

YEAR ::= TIME((SETTINGS "基本=日付 日付=Y 年=基本")|
              (設定 "基本=日付 日付=Y 年=予防的"))

任意の年 ::= TIME((SETTINGS "基本=日付 日付=Y 年=負")|
                  (設定 "基本=日付 日付=Y 年=L5"))
  -- これにより、正の場合は 5 桁の年のみが許可されます。
  -- 桁数が多い型は、
  -- 追加の時間タイプとして定義されます。

年-月 ::= TIME((SETTINGS "基本=日付 日付=YM 年=基本")|
               (設定 "基本=日付 日付=YM 年=Proletic"))

任意の年-月 ::= TIME((SETTINGS "基本=日付 日付=YM 年=負")|
                     (設定 "基本=日付 日付=YM 年=L5"))
  -- これにより、正の場合は 5 桁の年のみが許可されます。
  -- 桁数が多い型は、
  -- 追加の時間タイプとして定義されます。

年-月-日 ::= TIME((SETTINGS "Basic=日付 Date=YMD Year=Basic")|
                  (設定 "基本=日付 日付=YMD 年=Proletic"))

任意の年-月-日 ::= TIME((SETTINGS "Basic=日付 Date=YMD Year=Negative")|
                        (設定 "基本=日付 日付=YMD 年=L5"))
  -- これにより、正の場合は 5 桁の年のみが許可されます。
  -- 桁数が多い型は、
  -- 追加の時間タイプとして定義されます。
```

```
年-週 ::= TIME((SETTINGS "基本=日付 日付=YW 年=基本")|
                    (設定 "基本=日付 日付=YW 年=Proletic"))

任意の年-週 ::= TIME((SETTINGS "Basic=日付 Date=YW Year=Negative")| (SETTINGS "Basic=Date Date=YW Year=L5"))

                    -- これにより、正の場合は 5 桁の年のみが許可されます。
                    -- より多くの桁数を持つ型は、 -- 追加の時間型として定義できま
                    す。

年-週-日 ::= TIME((SETTINGS "基本=日付 日付=YWD 年=基本")|
                      (設定 "基本=日付 日付=YWD 年=Proletic"))

任意の年-週-日 ::= TIME((SETTINGS "Basic=日付 Date=YWD Year=Negative")| (SETTINGS "Basic=Date Date=YWD Year=L5"))

                    -- これにより、正の場合は 5 桁の年のみが許可されます。
                    -- より多くの桁数を持つ型は、 -- 追加の時間型として定義できま
                    す。

-- 時刻に関連するタイプ

HOURS ::= TIME(SETTINGS "Basic=Time Time=H Local-or-UTC=L")

HOURS-UTC ::= TIME(SETTINGS "Basic=Time Time=H Local-or-UTC=Z")

時間と差分 ::= TIME(SETTINGS "Basic=Time Time=H Local-or-UTC=LD")

MINUTES ::= TIME(SETTINGS "Basic=Time Time=HM Local-or-UTC=L")

MINUTES-UTC ::= TIME(SETTINGS "Basic=Time Time=HM Local-or-UTC=Z")

MINUTES-AND-DIFF ::= TIME(SETTINGS "Basic=Time Time=HM Local-or-UTC=LD")

SECONDS ::= TIME(SETTINGS "Basic=Time Time=HMS Local-or-UTC=L")

SECONDS-UTC ::= TIME(SETTINGS "Basic=Time Time=HMS Local-or-UTC=Z")

SECONDS-AND-DIFF ::= TIME(SETTINGS "Basic=Time Time=HMS Local-or-UTC=LD")

時間と小数部 ::= TIME(SETTINGS "Basic=Time Time=HF3 Local-or-UTC=L") -- 3 桁の小数部


HOURS-UTC-AND-FRACTION ::= TIME(SETTINGS "Basic=Time Time=HF3 Local-or-UTC=Z")
                    -- 3 桁の小数

時間と差分と分数 ::= TIME(SETTINGS "Basic=Time Time=HF3
                        Local-or-UTC=LD") -- 3 桁の小
                        数

分と小数部 ::= TIME(SETTINGS "Basic=Time Time=HMF3 Local-or-UTC=L")
                    -- 3 桁の小数

MINUTES-UTC-AND-FRACTION ::= TIME(SETTINGS "Basic=Time Time=HMF3 Local-or-UTC=Z")
                    -- 3 桁の小数

分と差分と分数 ::= TIME(SETTINGS "Basic=Time Time=HMF3
                          Local-or-UTC=LD") -- 3 桁の小
                          数

SECONDS-AND-FRACTION ::= TIME(SETTINGS "Basic=Time Time=HMSF3 Local-or-UTC=L") -- 3 桁の小数


SECONDS-UTC-AND-FRACTION ::= TIME(SETTINGS "Basic=Time Time=HMSF3 Local-or-UTC=Z") -- 3 桁の小数


秒と差分と分数 ::= TIME(SETTINGS "Basic=Time Time=HMSF3
                            Local-or-UTC=LD") -- 3 桁の小
                            数

-- 間隔タイプ (DURATION は便利なタイプなので含まれていません)。

開始-終了日-間隔 ::= TIME(SETTINGS "Basic=間隔 間隔タイプ=SE
                        SEポイント=日付")
        -- これは、DATE サブセットでサブタイプ化されている場合にのみ役立ちます (以下を参照)。

START-END-TIME-INTERVAL ::= TIME(SETTINGS "Basic=Interval 間隔の種類=SE
                        SEポイント=時間")
        -- これは、TIME-OF-DAY サブセットでサブタイプ化されている場合にのみ役立ちます --
        (以下を参照)。
```

```
開始-終了日-時刻-間隔 ::= TIME(SETTINGS "Basic=間隔 間隔の種類=SE
                                SE-point=日付/時刻")
        -- これは、DATE サブセットおよび -- TIME-OF-DAY サブセットでサブタイプ化され
        ている場合にのみ役立ちます (以下を参照)。

開始日-期間-間隔 ::= TIME(SETTINGS "基本=間隔 間隔タイプ=SD
                                SEポイント=日付")
        -- これは、DATE サブセットでサブタイプ化されている場合にのみ役立ちます (以下を参照)。

開始時間-期間-間隔 ::= TIME(SETTINGS "基本=間隔 間隔タイプ=SD
                                SEポイント=時間")
        -- これは、TIME-OF-DAY サブセットでサブタイプ化されている場合にのみ役立ちます --
        (以下を参照)。

開始日時-期間-間隔 ::= TIME(SETTINGS "Basic=間隔
                                      インターバルタイプ=SD
                                      SE-point=日付/時刻")
        -- これは、DATE サブセットおよび -- TIME-OF-DAY サブセットでサブタイプ化され
        ている場合にのみ役立ちます (以下を参照)。


DURATION-END-DATE-INTERVAL ::= TIME(SETTINGS "Basic=Interval 間隔の種類=DE
                                SEポイント=日付")
        -- これは、DATE サブセットでサブタイプ化されている場合にのみ役立ちます (以下を参照)。

DURATION-END-TIME-INTERVAL ::= TIME(SETTINGS "Basic=Interval 間隔の種類=DE
                                SEポイント=時間")
        -- これは、TIME-OF-DAY サブセットでサブタイプ化されている場合にのみ役立ちます --
        (以下を参照)。

DURATION-END-DATE-TIME-INTERVAL ::= TIME(SETTINGS "Basic=Interval 間隔の種類=DE
                                SE-point=日付/時刻")
        -- これは、DATE サブセットおよび -- TIME-OF-DAY サブセットでサブタイプ化され
        ている場合にのみ役立ちます (以下を参照)。

-- 定期的な間隔タイプ。

REC-START-END-DATE-INTERVAL ::= TIME(SETTINGS "Basic=Rec-Interval 間隔タイプ=SE
                                SEポイント=日付")
        -- これは、DATE サブセットでサブタイプ化されている場合にのみ役立ちます (以下を参照)。

REC-START-END-TIME-INTERVAL ::= TIME(SETTINGS "Basic=Rec-Interval Interval-type=SE SE-point=Time")


        -- これは、TIME-OF-DAY サブセットでサブタイプ化されている場合にのみ役立ちます --
        (以下を参照)。

REC-START-END-DATE-TIME-INTERVAL ::= TIME(SETTINGS "Basic=録画間隔
                                      インターバルタイプ=SE
                                      SE-point=日付/時刻")
        -- これは、DATE サブセットおよび -- TIME-OF-DAY サブセットでサブタイプ化され
        ている場合にのみ役立ちます (以下を参照)。

REC-DURATION-INTERVAL ::= TIME(SETTINGS "Basic=Rec-Interval Interval-type=D")

REC-START-DATE-DURATION-INTERVAL ::= TIME(SETTINGS "Basic=REC-Interval
                                      インターバルタイプ=SD
                                      SEポイント=日付")
        -- これは、DATE サブセットでサブタイプ化されている場合にのみ役立ちます (以下を参照)。

REC-START-TIME-DURATION-INTERVAL ::= TIME(SETTINGS "Basic=REC-INTERVAL
                                      インターバルタイプ=SD SEポ
                                      イント=時間")
        -- これは、TIME-OF-DAY サブセットでサブタイプ化されている場合にのみ役立ちます --
        (以下を参照)。

REC-START-DATE-TIME-DURATION-INTERVAL ::= TIME(SETTINGS "Basic=REC-Interval
                                      インターバルタイプ=SD
                                      SE-point=日付/時刻")
        -- これは、DATE サブセットおよび -- TIME-OF-DAY サブセットでサブタイプ化され
        ている場合にのみ役立ちます (以下を参照)。


記録期間-終了日-間隔 ::= TIME(SETTINGS "Basic=記録間隔
                                      インターバルタイプ=DE
                                      SEポイント=日付")
        -- これは、DATE サブセットでサブタイプ化されている場合にのみ役立ちます (以下を参照)。
```

```
REC-DURATION-END-TIME-INTERVAL ::= TIME(SETTINGS "Basic=Rec-Interval
                                        インターバルタイプ=DE SE-
                                        point=時間")
        -- これは、TIME-OF-DAY サブセットでサブタイプ化されている場合にのみ役立ちます --
        (以下を参照)。

REC-DURATION-END-DATE-TIME-INTERVAL ::= TIME(SETTINGS "Basic=Rec-Interval
                                        インターバルタイプ=DE
                                        SE-point=日付/時刻")
        -- これは、DATE サブセットおよび -- TIME-OF-DAY サブセットでサブタイプ化され
        ている場合にのみ役立ちます (以下を参照)。

-- 日付のサブセット

CENTURY-SUBSET ::= TIME((SETTINGS "日付=C 年=基本")|
                        (設定 "日付=C 年=Proletic"))

任意の世紀のサブセット ::= TIME((SETTINGS "日付=C 年=負")|
                          (設定「日付=C 年=L5」))

YEAR-SUBSET ::= TIME((SETTINGS "日付=Y 年=基本")|
                     (設定 "日付=Y 年=Proletic"))

任意の年のサブセット ::= TIME((SETTINGS "日付=Y 年=負")|
                       (設定 "日付=Y 年=L5"))

年-月-サブセット ::= TIME((SETTINGS "日付=YM 年=基本")|
                      (設定 "日付=YM 年=Proletic"))

任意の年-月-サブセット ::= TIME((SETTINGS "日付=YM 年=負")|
                         (設定 "日付=YM 年=L5"))

年-月-日-サブセット ::= TIME((SETTINGS "日付=YMD 年=基本")|
                         (設定 "日付=YMD 年=Proletic"))

任意の年-月-日-サブセット ::= TIME((SETTINGS "日付=YMD 年=負")|
                           (設定 "日付=YMD 年=L5"))

年-週-サブセット ::= TIME((SETTINGS "日付=YW 年=基本")|
                      (設定 "日付=YW 年=Proletic"))

任意の年週サブセット ::= TIME((SETTINGS "日付=YW 年=負")|
                         (設定 "日付=YW 年=L5"))

年-週-日-サブセット ::= TIME((SETTINGS "日付=YWD 年=基本")|
                         (設定 "日付=YWD 年=Proletic"))

任意の年の曜日のサブセット ::= TIME((SETTINGS "日付=YWD 年=負")|
                           (設定 "日付=YWD 年=L5"))

-- 時間サブセット

HOURS-SUBSET ::= TIME(SETTINGS "Time=H Local-or-UTC=L")

HOURS-UTC-SUBSET ::= TIME(SETTINGS "Time=H Local-or-UTC=Z")

HOURS-AND-DIFF-SUBSET ::= TIME(SETTINGS "Time=H Local-or-UTC=LD")

MINUTES-SUBSET ::= TIME(SETTINGS "Time=HM Local-or-UTC=L")

MINUTES-UTC-SUBSET ::= TIME(SETTINGS "Time=HM Local-or-UTC=Z")

MINUTES-AND-DIFF-SUBSET ::= TIME(SETTINGS "Time=HM Local-or-UTC=LD")

SECONDS-SUBSET ::= TIME(SETTINGS "Time=HMS Local-or-UTC=L")

SECONDS-UTC-SUBSET ::= TIME(SETTINGS "Time=HMS Local-or-UTC=Z")

SECONDS-AND-DIFF-SUBSET ::= TIME(SETTINGS "Time=HMS Local-or-UTC=LD")

時間と小数部のサブセット ::= TIME(SETTINGS "Time=HF3 Local-or-UTC=L")

HOURS-UTC-AND-FRACTION-SUBSET ::= TIME(SETTINGS "Time=HF3
                                                ローカルまたは UTC=Z")

時間と差分と分数のサブセット ::= TIME(SETTINGS "Time=HF3
                                  ローカルまたは UTC=LD")
```

```
分と小数部のサブセット ::= TIME(SETTINGS "Time=HMF3
                                        ローカルまたは UTC=L")

MINUTES-UTC-AND-FRACTION-SUBSET ::= TIME(SETTINGS "Time=HMF3
                                        ローカルまたは UTC=Z")

分と差分と分数のサブセット ::= TIME(SETTINGS "Time=HMF3
                                        ローカルまたは UTC=LD")

秒小数部サブセット ::= TIME(SETTINGS "Time=HMSF3
                                        ローカルまたは UTC=L")

SECONDS-UTC-AND-FRACTION-SUBSET ::= TIME(SETTINGS "Time=HMSF3
                                        ローカルまたは UTC=Z")

秒と差分と小数部のサブセット ::= TIME(SETTINGS "Time=HMSF3
                                        ローカルまたは UTC=LD")

終わり
```

付録C

型と値の互換性に関する規則

(この付録は、この勧告 | 国際規格の不可欠な部分を形成します。)

この付録は、主にツール作成者が言語を同一に解釈できるようにするために役立つことが期待されています。これは、ASN.1 で何が合法で何がそうでないかを明確に指定し、値参照名が識別する正確な値、および型または値セット参照名が識別する正確な値のセットを指定できるようにするために存在します。 。上記以外の目的で ASN.1 表記の有効な変換の定義を提供することを目的としたものではありません。

C.1 値マッピングの概念の必要性 (チュートリアルの導入)

C.1.1次の ASN.1 定義を考慮してください。

```
A ::= 整数

B ::= [1] 整数

C ::= [2] 整数 (0..6,...)

D ::= [2] 整数 (0..6,...,7)

E ::= 整数 (7..20)

F ::= INTEGER {赤(0)、白(1)、青(2)、緑(3)、紫(4)}

a A ::= 3

b B ::= 4

c C ::= 5

d D ::= 6

e E ::= 7

f F ::= 緑
```

C.1.2値参照a、 b、 c、 d、 e、およびf は、それぞれA、 B、 C、 D、 E、およびFによって管理される値表記で使用できることは明らかです。例えば：

```
W ::= SEQUENCE {w1 A デフォルト a}

そして：

×A ::= a

そして：

Y ::= A(1..a)
```

C.1.1 の定義を考慮すると、すべて有効です。しかし、上記のA がB、 C、 D、 E、またはFに置き換えられた場合、その結果のステートメントは違法になりますか?同様に、上記の値参照a がこれらの各ケースでb、またはc、またはd、またはe、またはf に置き換えられた場合、結果のステートメントは合法ですか?

C.1.3より洗練された質問は、それぞれの場合において、型参照をその割り当ての右側にある明示的なテキストで置き換えることを検討することです。たとえば、次のことを考えてみましょう。

```
f INTEGER {赤(0)、白(1)、青(2)、緑(3)、紫(4)} ::= 緑

W ::= シーケンス {
                w1 INTEGER {赤(0)、白(1)、青(2)、緑(3)、紫(4)}
                デフォルト f}

x INTEGER {赤(0)、白(1)、青(2)、緑(3)、紫(4)} ::= f

Y ::= INTEGER {赤(0)、白(1)、青(2)、緑(3)、紫(4)}(1..f)
```

上記は合法的な ASN.1 でしょうか?

C.1.4上記の例のいくつかは、たとえ合法であっても (ほとんどが合法である - 後の本文を参照)、少なくともわかりにくく、最悪の場合は混乱を招くため、ユーザーが同様のテキストを書くことは賢明ではないケースです。ただし、ある型 (必ずしもINTEGER型だけではない) の値への値参照が、そのデフォルト値として頻繁に使用されます。

ガバナでタグ付けまたはサブタイプが適用されたタイプ。値マッピングの概念は、上記のようなどの構成が合法であるかを決定する明確かつ正確な手段を提供するために導入されています。

C.1.5 もう一度考えてみましょう:

C ::= [2] 整数 (0..6,...)

E ::= 整数 (7..20)

F ::= INTEGER {赤(0)、白(1)、青(2)、緑(3)、紫(4)}

それぞれの場合において、新しいタイプが作成されます。 Fについては、その値とユニバーサル型 INTEGER の値との間の 1 対 1 の対応を明確に識別できます。 CとEの場合、それらの値とユニバーサル型INTEGER の値のサブセットとの間の 1 対 1 の対応を明確に識別できます。この関係を、 2 つのタイプの値間の値マッピングと呼びます。さらに、 F、 C、およびEの値はすべて INTEGER の値への (1-1) マッピングを持っているため、これらのマッピングを使用して、F、 C、およびEの値自体の間のマッピングを提供できます。これは、図 C.1 のFとCについて示されています。

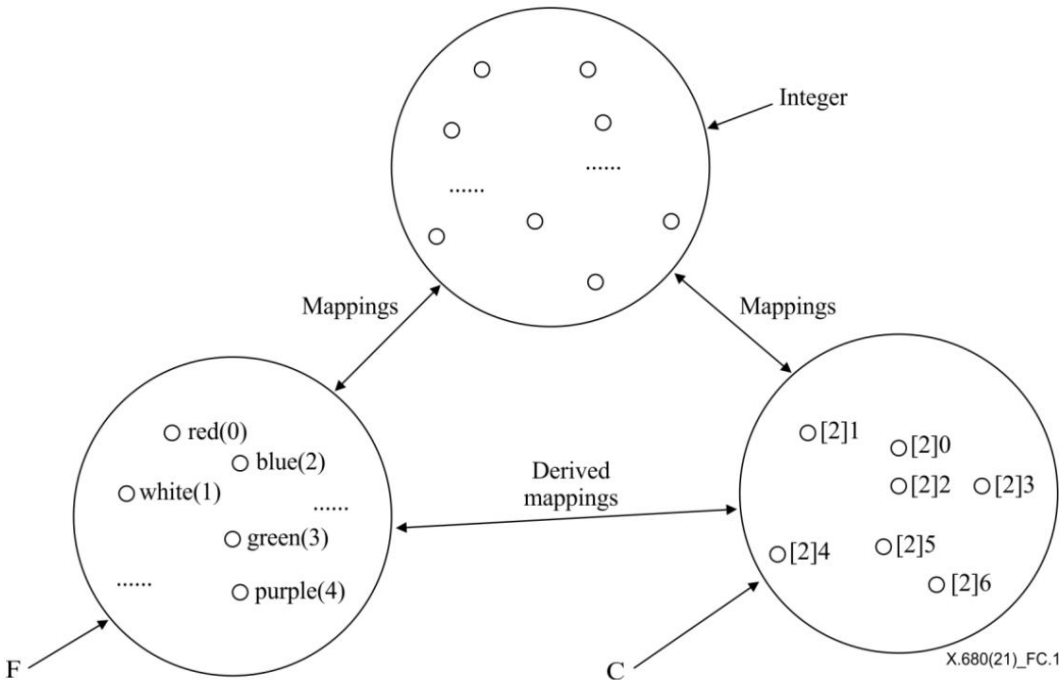

図 C.1 – 派生マッピング

C.1.6次のような値参照があるとします。

c C ::= 5

Fの値を特定するために何らかのコンテキストで必要なCの値にマッピングすると、 Cのその値と F の (単一の) 値の間に値マッピングが存在する場合、 c をa として定義できます (実際に定義します)。これは、図 C.2 に示されています。値参照 c は、 Fの値を識別するために使用され、次のように定義する必要がある直接参照f1の代わりに使用できます。

f1F ::= 5

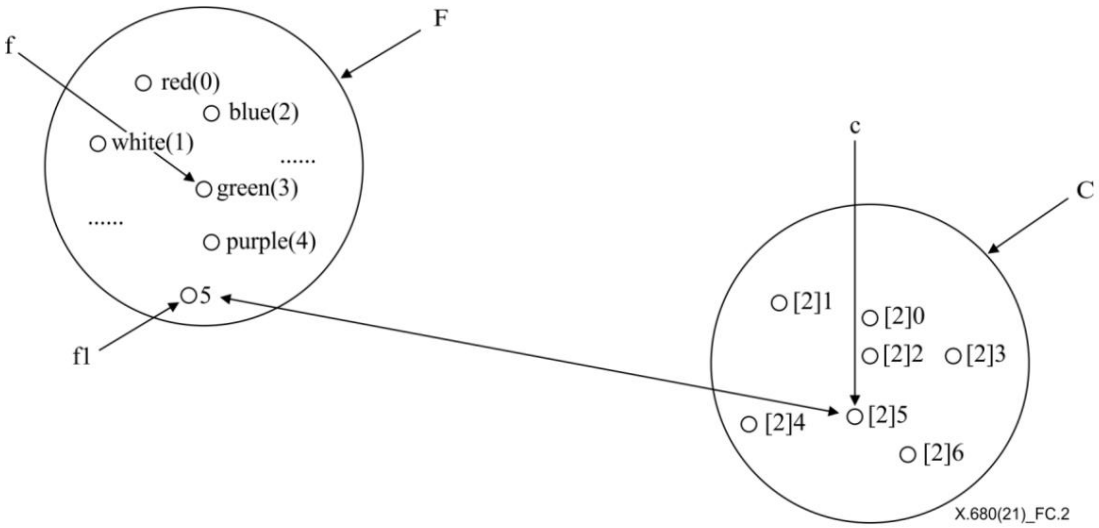

図 C.2 – 値参照のマッピング

C.1.7場合によっては、ある型の値 (たとえば、C.1.1 のAの 7 ～ 20) が別の型の値 (C.1.1 のEの 7 ～ 20) にマッピングされる場合があることに注意してください。たとえば)、そのようなマッピングを持たない他の値 (A より上の 21)。 Aのそのような値への参照は、E の値への有効な参照にはなりません。 (この例では、 E全体がAのサブセットにマッピングされる値を持っています。一般に、値のサブセットが存在する可能性があります。マッピングのある両方の型と、マッピングされていない両方の型の他の値。)

C.1.8 ASN.1 標準の本文では、上記および同様の場合の合法性を指定するために通常の英語テキストが使用されます。サブ条項 C.6 は合法性に関する正確な要件を示しており、複雑な構造について疑問がある場合は常に参照する必要があります。

注 – 値のマッピングが「Type」構造の 2 つの出現の間に存在するように定義されているという事実により、1 つの「Type」構造を使用して確立された値参照を使用して、十分に類似した別の「Type」構造の値を識別することができます。これにより、ダミー パラメータと実際のパラメータの互換性に関するルールに違反することなく、テキスト的に別個の 2 つの「Type」構造を使用してダミー パラメータと実際のパラメータを型指定できます。また、情報オブジェクト クラスのフィールドを 1 つの「Type」構成体を使用して指定し、情報オブジェクト内の対応する値を、十分に類似した個別の「Type」構成体を使用して指定することもできます。 (これらの例は網羅的であることを意図したものではありません。) ただし、この自由度を利用するのは、 SEQUENCE OF INTEGERやCHOICE {int INTEGER, id OBJECT IDENTIFIER} などの単純な場合にのみ使用し、より複雑な場合には使用しないことをお勧めします。「型」の構成要素。

C.2 値のマッピング

C.2.1基礎となるモデルは、値を含む、重複しないコンテナとしての型であり、ASN.1 の "Type" 構造が出現するたびに、別個の新しい型を定義します (図 C.1 および C.2 を参照)。この付録では、そのような型間に値マッピングが存在する場合を指定し、他の型の値への参照が必要な場合に、ある型の値への参照を使用できるようにします。

例: 以下を考慮してください。

X ::= 整数

Y ::= 整数

XとYは 2 つの異なる型への型参照名 (ポインター) ですが、これらの型の間には値のマッピングが存在するため、 Yによって管理される場合(たとえば、DEFAULT に続く)、 Xの値への任意の値参照を使用できます。

C.2.2可能なすべての ASN.1 値のセットにおいて、値マッピングは値のペアを関連付けます。値のマッピングのセット全体は数学的な関係です。この関係には次の特性があります。再帰的 (各 ASN.1 値はそれ自体に関連しています)、対称的です (値マッピングが値x1から値x2 に存在するように定義されている場合、自動的に値マッピングが存在します) x2からx1)、推移的です (値x1からの値マッピングがある場合)

x2への値、およびx2からx3への値マッピングが存在する場合、 x1から x3への値マッピングが自動的に存在します)。

C.2.3さらに、任意の 2 つの型X1およびX2 が値のセットとして見られる場合、 X1の値からX2の値への値マッピングのセットは1 対 1 の関係になります。つまり、 X1のすべての値x1についてです。 、 X2にx2 、 x1から x2への値マッピングがある場合、次のようになります。

a) x1からx2とは異なる X2 の別の値にマッピングされる値はありません。そして

b) X1の値(x1 以外)から x2への値マッピングはありません。

C.2.4値x1と値x2 の間に値マッピングが存在する場合、何らかの管理型で必要な場合は、いずれか一方への値参照を自動的に使用して他方を参照できます。

注 – 値マッピングが一部の「Type」構造の値間に存在するように定義されているという事実は、ASN.1 表記の使用に柔軟性を提供することのみを目的としています。このようなマッピングの存在は、2 つの型が同じアプリケーション セマンティクスを保持するという意味をまったく持ちませんが、値マッピングなしでは不正となる ASN.1 構造は、対応する型が実際に同じアプリケーション セマンティクスを保持する場合にのみ使用することが推奨されます。値のマッピングは、大規模な仕様の中で、同一の ASN.1 構成要素であるがアプリケーション セマンティクスがまったく異なる 2 つの型の間に頻繁に存在し、これらの値のマッピングの存在が仕様全体の正当性を判断する際に使用されることは決してないことに注意してください。

C.3 同一の型定義

C.3.1同一型定義の概念は、同一であるか、通常は交換可能であると予想されるほど十分に類似している「Type」の 2 つのインスタンス間で値マッピングを定義できるようにするために使用されます。 「十分に類似している」の意味を正確にするために、この副節は、「Type」のインスタンスのそれぞれに適用される一連の変換を指定して、「 Type」のインスタンスの標準形式を生成します。 "Type" の 2 つのインスタンスは、それらの通常形式が同じ語彙項目の同一の順序リストである場合に限り、同一の型定義であると定義されます (12 節を参照)。

C.3.2 ASN.1 仕様内の各「Type」の出現は、第 12 項で定義された語彙項目の順序付きリストです。正規形式は、C.3.2.1 から C.3.2.6 で定義された変換を適用することによって取得されます。その順番で。

C.3.2.1すべてのコメント (12.6 を参照) が削除されます。

C.3.2.2次の変換は再帰的ではないため、任意の順序で一度だけ適用する必要があります。

a) 「ValueSetTypeAssignment」によって定義された型の場合、その定義は、同じ「Type」と、16.6 で指定されている「ValueSet」の内容であるサブタイプ制約を使用する「TypeAssignment」によって置き換えられます。

b) 各整数型について: 「NamedNumberList」(19.1 を参照) があれば、「識別子」が次のように並べ替えられます。
アルファベット順 (「a」が最初、「z」が最後)。

c) 各列挙型について: 20.3 で指定されているように、「識別子」(数字のない) である「EnumerationItem」(20.1 を参照) に数字が追加されます。次に、「識別子」がアルファベット順 (最初に「a」 、最後に「z」) になるように「RootEnumeration」が並べ替えられます。

d) ビット文字列タイプごとに、「NamedBitList」 (22.1 を参照)があれば、その「識別子」が次の順序に収まるように並べ替えられます。
アルファベット順 (「a」が最初、「z」が最後)。

e) 各オブジェクト識別子の値について: 各「ObjIdComponents」は、第 32 条のセマンティクスに従って、対応する「NumberForm」に変換されます (32.13 の例を参照)。

f) 各相対オブジェクト識別子の値について (33.3 を参照): 各「RelativeOIDComponents」は次のように変換されます。
第 33 条のセマンティクスに従って、対応する「NumberForm」 。

g) シーケンス型 (第 25 節を参照) およびセット型 (第 27 節を参照) の場合: "ExtensionAndException"、 "ExtensionAdditions" 形式の拡張子はすべて切り取られ、 "ComponentTypeLists" の末尾に貼り付けられます。 「OptionalExtensionMarker」が存在する場合は削除されます。

「TagDefault」がIMPLICIT TAGS の場合、次のいずれかの場合を除き、キーワードIMPLICITが「Tag」のすべてのインスタンスに追加されます (31.2 を参照)。

- それはすでに存在しています。または
- 予約語EXPLICITが存在します。または
- タグ付けされているタイプがCHOICEタイプである、または;
- オープンタイプです。

「TagDefault」がAUTOMATIC TAGS の場合、自動タグ付けを適用するかどうかは 25.3 に従って決定されます (自動タグ付けは後で実行されます)。

注 – 節 25.4 および 27.2 では、「Components of Type」の置換の結果として挿入された「ComponentType」内の「Tag」の存在自体が自動タグ付け変換を妨げるものではないことを指定しています。

「ExtensionDefault」がEXTENSIBILITY IMPLIED の場合、省略記号(「...」)が「ComponentTypeLists」の後に追加されます (存在しない場合)。

h) 選択タイプの場合 (条項 29 を参照): 「NameType」の識別子がアルファベット順 (最初に「a」、最後に「z」) になるように、「RootAlternativeTypeList」が並べ替えられます。 「OptionalExtensionMarker」が存在する場合は削除されます。「TagDefault」がIMPLICIT TAGS の場合、次のいずれかの場合を除き、キーワードIMPLICITが「Tags」のすべてのインスタンスに追加されます (31.2 を参照)。

- それはすでに存在しています。または
- 予約語EXPLICITが存在します。または
- タグ付けされている型はCHOICE型です。または
- オープンタイプです。

「TagDefault」がAUTOMATIC TAGS の場合、自動タグ付けを適用するかどうかは 29.5 に従って決定されます。「ExtensionDefault」がEXTENSIBILITY IMPLIED の場合、省略記号(「...」)が「AlternativeTypeLists」の後に追加されます(存在しない場合)。

C.3.2.3以下の変換は、固定点に到達するまで、指定された順序で再帰的に適用されます。

a) 各オブジェクト識別子の値について (32.3 を参照): 値の定義が「DefinedValue」で始まる場合、「DefinedValue」はその定義で置き換えられます。

b) 各相対オブジェクト識別子の値について (33.3 を参照): 値の定義に「DefinedValue」が含まれている場合、「DefinedValue」はその定義で置き換えられます。

c) シーケンス型およびセット型の場合: 「COMPONENTS OF Type」(第 25 節を参照) のすべてのインスタンスは、第 25 節および第 27 節に従って変換されます。

d) シーケンス、セット、選択タイプの場合: 自動的にタグ付けすることが事前に決定されている場合 (C.3.2.2 g を参照)、および h))、自動タグ付けは第 25 条、第 27 条、および第 29 条に従って適用されます。

e) 選択タイプの場合: 構成は、第 30 項に従って選択された代替案に置き換えられます。

f) すべての型参照は、次の規則に従ってその定義に置き換えられます。

- 置換する型が変換される型への参照である場合、その型参照は、それ自体以外のどの項目にも一致しない特別な項目に置き換えられます。
- 置換する型が sequence-of 型または set-of 型の場合、置換される型に続く制約があれば、その制約はキーワードOF の前に移動されます。
- 置換されたタイプがパラメータ化されたタイプまたはパラメータ化された値セットである場合 (Rec. ITU-T X.683 | ISO/IEC 8824-4、8.2 を参照)、すべての「DummyReference」は対応する「ActualParameter」に置き換えられます。

g) すべての値参照はその定義に置き換えられます。置換された値がパラメータ化された値である場合 (Rec. ITU-T X.683 | ISO/IEC 8824-4、8.2 を参照)、すべての「DummyReference」は対応する「ActualParameter」に置き換えられます。

注 – 値参照を置き換える前に、この付録の手順を適用して、値参照が値マッピングを通じて、または直接、その支配型の値を確実に識別するものとします。

C.3.2.4セット タイプの場合: 「ComponentType」がアルファベット順 (最初に「a」、最後に「z」) になるように、「RootComponentTypeList」が並べ替えられます。

C.3.2.5次の変換は値の定義に適用されます。

a) 整数値が識別子を使用して定義されている場合、その識別子は関連する数値に置き換えられます。 b) ビット文字列値が識別子を使用して定義されている場合、ビット文字列値は、すべての値を含む対応する「bstring」に置き換えられます。末尾のゼロビットが削除されます。

c) 「cstring」内の各改行 (改行を含む) の直前と直後のすべての空白は、削除されました。

d) 「bstring」と「hstring」のすべての空白が削除されます。

e) 基数 2 で定義された各実数値は、仮数部が奇数になるように正規化され、各実数値が定義されます。基数 10 の場合、仮数の最後の桁が 0 にならないように正規化されます。

f) GeneralizedTime、 UTCTime、 TIME、 TIME-OF-DAY、 DATE、DATE-TIME、およびDURATIONの各値は、DER および CER でエンコードするときに使用される規則に準拠する文字列に置き換えられます (Rec. ITU-T X を参照)。 690 | ISO/IEC 8825-1、11.7、11.8、および 11.9)。

g) c) を適用した後、 UTF8String、 NumericString、 PrintableString、 IA5String、 VisibleString (ISO646String)、BMPStringおよびUniversalStringの各値は、「Quadruple」表記法を使用して記述されたUniversalString型の同等の値に置き換えられます(41.8 節を参照)。

C.3.2.6 「実数」の出現はすべて、「基数」10 に関連付けられた「SequenceValue」に変換されます。 「SequenceValue」に関連付けられた「RealValue」の出現は、仮数の最後の桁がゼロにならないように、同じ「基数」の関連付けられた「SequenceValue」に変換されます。

C.3.3 「Type」の 2 つのインスタンスが、通常の形式に変換されたときに、同一の語彙項目のリストである場合 (12 節を参照)、「Type」の 2 つのインスタンスは、次の例外を除き、同一の型定義であると定義されます。 「オブジェクトクラス参照」 (Rec. ITU-T X.681 | ISO/IEC 8824-2、7.1 を参照)、「オブジェクト参照」 (Rec. ITU-T X.681 | ISO/IEC 8824-2、7.2 を参照) の場合または、「objectsetreference」 (Rec. ITU-T X.681 | ISO/IEC 8824-2、7.3 を参照) が「Type」の正規化形式内に出現する場合、2 つのタイプは同一のタイプ定義として定義されていません。値のマッピング (以下の C.4 を参照) はそれらの間に存在しません。

注 – この例外は、情報オブジェクト クラス、情報オブジェクト、および情報オブジェクト セットの表記法に関連する構文の要素に対して標準形式への変換規則を提供する必要性を回避するために挿入されました。同様に、全値表記および集合算術表記の正規化の仕様は現時点では含まれていません。そのような仕様に対する要件があることが判明した場合、これはこの推奨事項の将来のバージョンで提供される可能性があります。国際標準。

同一型定義と値マッピングの概念は、参照名を使用するかテキストをコピーすることによって単純な ASN.1 構造を確実に使用できるようにするために導入されました。情報オブジェクト クラスなどを含む「Type」のより複雑なインスタンスにこの機能を提供する必要はないと考えられました。

C.4 値マッピングの仕様

C.4.1 C.3 の規則に基づく 2 つの「Type」の出現が同一の型定義である場合、値のマッピングが存在します。
ある型のすべての値と他の型の対応する値の間。

C.4.2タグ付け (31.2 を参照) によって任意の型 X2 から作成された型X1の場合、 X1のすべてのメンバーとX2の対応するメンバーの間に値マッピングが存在するように定義されます。

注 – 値マッピングは、上記の C.4.2 ではX1とX2の値の間、および C.4.3 ではX3とX4の値の間に存在するように定義されていますが、そのような型がその他の点では同一であるが別個の型定義 (たとえば、 SEQUENCEまたはCHOICEタイプ定義)、結果のタイプ定義 (SEQUENCEまたはCHOICEタイプ) は同一のタイプ定義ではなく、それらの間に値のマッピングはありません。

C.4.3要素セット構成またはサブタイプによって、任意の支配型X4から値を選択して作成された型X3の場合、値マッピングは、新しい型のメンバーと支配型のメンバーの間に存在するように定義されます。要素セットまたはサブタイプ構成によって選択されました。拡張マーカーの有無は、このルールには影響しません。

C.4.4 C.5 では、一部の文字列型間で追加の値のマッピングが指定されています。

C.4.5値マッピングは、名前付きの値を持つ整数型として定義された任意の型のすべての値と、名前付きの値なし、または異なる名前付きの値付き、または名前付き値の異なる名前付きで定義された整数型、またはその両方の間に存在するように定義されています。 。

注 – 値マッピングの存在は、名前付き値の名前の使用に関するスコープ ルール要件には影響しません。これらは、それらが定義されている型、またはその型への型参照名によって管理されるスコープ内でのみ使用できます。

C.4.6値マッピングは、名前付きビットを持つビット文字列型として定義された任意の型のすべての値と、名前付きビットなし、または異なる名前付きビットを持つ、または名前付きビットの異なる名前で定義されたビット文字列型のすべての値の間に存在するように定義されます。または両方。

注 – 値マッピングの存在は、名前付きビットの名前の使用に関するスコープ ルール要件には影響しません。
これらは、それらが定義されている型、またはその型への型参照名によって管理されるスコープ内でのみ使用できます。

C.5 文字列タイプに対して定義された追加の値マッピング

C.5.1制限された文字列タイプには、グループ A (C.5.2 を参照) とグループ B (C.5.3 を参照) の 2 つのグループがあります。値のマッピングは、グループ A 内のすべての型の間に存在するように定義されており、これらの型の値への値参照は、他の型のいずれかによって管理される場合に使用できます。グループ B の型の場合、これらの異なる型間、またはグループ A の型とグループ B の型の間に値のマッピングは存在しません。

C.5.2グループ A は以下で構成されます。

UTF8文字列
数値文字列
印刷可能な文字列
IA5文字列
可視文字列(ISO646文字列)
ユニバーサル文字列
BMP文字列

C.5.3グループ B は以下で構成されます。

テレテックスストリング(T61ストリング)
ビデオテックス文字列
グラフィック文字列
一般文字列

C.5.4グループ A の値マッピングは、各型の文字列値をUniversalString にマッピングし、値マッピングの推移性プロパティを使用することによって指定されます。グループ A 型のいずれかの値をUniversalString にマップするには、文字列は、以下に指定されているようにマップされた各文字を含む同じ長さのUniversalStringに置き換えられます。

C.5.5正式には、 UTF8Stringの抽象値のセットは、 UniversalStringで発生する抽象値のセットと同じですが、タグが異なります (41.16 を参照)。また、 UTF8Stringの各抽象値は、 UniversalStringの対応する抽象値にマップされるように定義されています。 。

C.5.6 NumericStringおよびPrintableString型を形成するために使用される文字のグリフ (印刷された文字の形状) には、 ISO/IEC 10646 の最初の 128 文字に割り当てられたグリフのサブセットへの認識可能かつ明確なマッピングがあります。これらの型のマッピングは定義されています。このグリフのマッピングを使用します。

C.5.7 IA5StringとVisibleStringは、UniversalStringの BER エンコーディングで同じ (32 ビット) 値を持つ UniversalString 文字に各文字をマッピングすることによって、UniversalStringにマッピングされます。
bit) IA5StringおよびVisibleStringの BER エンコーディングの値。

C.5.8 BMPString は正式にはUniversalStringのサブセットであり、対応する抽象値には値マッピングがあります。

C.6 特定の型と値の互換性要件

この節では、値マッピングの概念を使用して、特定の ASN.1 構造の合法性に関する正確なテキストを提供します。

C.6.1支配型Yを持つ任意の「値」出現x-notation は、 x-notationで指定された値x-valにマッピングされる値を持つ支配型Yの値y-valを識別します。このような値が存在することが要件です。

たとえば、次の行の最後の行にあるxを考えてみましょう。

```
X ::= [0] 整数 (0..30)

x X ::= 29

Y ::= [1] 整数 (25..35)

Z1 ::= Y (x | 30)
```

これらの ASN.1 構造は正当であり、最後の割り当てでは、 x 表記xはXのx-val 29を参照し、値マッピングを通じてYのy-val 29を識別します。x 表記30 は、 Y のy-val 30 、 Z1は値 29 と 30 のセットです。一方、代入は次のようになります。

```
Z2 ::= Y (x | 20)
```

x表記20 が参照できるy 値が存在しないため、これは不正です。

C.6.2支配型 V を持つ任意の「型」出現 (t 表記) は、支配型Vのルートにある値の完全なセットを識別し、「 「 t 表記」と入力します。このセットには少なくとも 1 つの値が含まれている必要があります。

たとえば、次の行の最後の行にあるWについて考えてみましょう。

```
V ::= [0] 整数 (0..30)

W ::= [1] 整数 (25..35)

Y ::= [2] 整数 (31..35)

Z1 ::= V (W | 24)
```

W は値 25 ～ 30 をセット演算に提供し、 Z1の値は 24 ～ 30 になります。一方、割り当ては次のとおりです。

```
Z2 ::= V (Y | 24)
```

Vの値にマップされるYの値がないため、これは不正です。

C.6.3実パラメータとして指定される値の型は、その値からダミー パラメータを管理する型の値の 1 つにマッピングする値を持つ必要があり、識別されるのはその管理型の値です。

C.6.4 「タイプ」が値セットのダミーパラメータであるダミーパラメータの実パラメータとして指定される場合、その「タイプ」のすべての値には、値セットダミーのガバナ内の値への値マッピングが必要です。パラメータ。実際のパラメータは、「タイプ」にマッピングされているガバナ内の値のセット全体を選択します。

側で 値または値セット パラメーターであるダミー パラメーターのタイプ A を指定する場合、Aのすべての値、および右Aを使用するすべてのインスタンスを除いて、これは不正なC.6.5仕様です。代入の場合、 Aの値はダミー パラメータの代わりに合法的に適用できます。

C.7 例

C.7.1 この節では、C.3 と C.4 を説明する例を示します。

C.7.2 例 1

```
X ::= SEQUENCE {名前                    X1 ::= シーケンス
VisibleString、年齢                     {名前のVisibleString、
INTEGER}                                -コメント-
                                        年齢 INTEGER}

X2 ::= [8] SEQUENCE {name               X3 ::= シーケンス
VisibleString, age                      {名前のVisibleString、
INTEGER}                                年齢 年齢タイプ}

                                        AgeType ::= INTEGER
```

X、 X1、 X2、およびX3はすべて同一の型定義です。空白とコメントの違いは表示されません。また、 X3でのAgeType型参照の使用は型定義に影響しません。ただし、シーケンスの要素の識別子のいずれかが変更された場合、型は同一の定義ではなくなり、型間に値のマッピングがなくなることに注意してください。

C.7.3 例 2

```
B ::= SET {名前                         B1 ::= セット
VisibleString、年齢                     {年齢 INTEGER,
INTEGER}                                名前VisibleString}
```

どちらもモジュール ヘッダーにAUTOMATIC TAGSを持つモジュール内にない限り、これらは同一の型定義です。そうでない場合、それらは同一の型定義ではなく、それらの間に値のマッピングは存在しません。同様の例は、 CHOICEおよびENUMERATED (「EnumerationItem」の「識別子」形式を使用)を使用して作成できます。

C.7.4 例 3

```
C ::= SET {名前                         C1 ::= セット
[0]VisibleString、年齢 INTEGER}          {名前のVisibleString、
                                        age INTEGER (1..64)}
```

は同一の型定義ではなく、それらのいずれもBまたはB1 のいずれかと同一の型定義ではありません。また、 CとC1 の値の間、またはそれらのいずれかとBまたはB1のいずれかとの間に値のマッピングはありません。

C.7.5 例 4

```
x INTEGER { y (2) } ::= 3

z 整数 ::= x
```

は正当であり、 C.4.5 で定義された値マッピングを通じて値 3 をzに割り当てます。

C.7.6 例 5

```
b1 ビット文字列 ::= '101'B

b2 ビット文字列 {バージョン 1(0)、バージョン 2(1)、バージョン 3(2)} ::= b1
```

は正当であり、値{version1, version3}を b2に割り当てます。

C.7.7 例 6

C.1.1 の定義では、 SEQUENCE要素は次の形式になります。

X デフォルト y

ここで、 XはA、 B、 C、 D、 E、 Fのいずれか、またはこれらの名前への型割り当ての右側にあるテキストのいずれかであり、 yはa、 b、 c、 d、 e、またはf。ただし、次の例外があります。 E DEFAULT y は、a、 b、 c、 d、 f、およびCのすべてに対して不正です。 DEFAULT e は、これらの場合、デフォルト値参照から使用できる値のマッピングがないため、不正です。デフォルトのタイプに変換されます。

付録D

割り当てられたオブジェクト識別子と OID 国際化リソース識別子の値

(この付録は、この勧告 | 国際規格の不可欠な部分を形成します。)

この付録には、ASN.1 シリーズの勧告 | で割り当てられたオブジェクト識別子、OID 国際化リソース識別子、およびオブジェクト記述子の値が記録されます。国際標準であり、それらの値を参照するために使用する ASN.1 モジュールを提供します。

D.1 この推奨事項で割り当てられた値 |国際標準

この推奨事項では次の値が割り当てられています。国際標準：

41.3項

オブジェクト識別子の値:
{ ジョイント-iso-itu-t asn1(1) 仕様(0) 文字列(1) 数値文字列(0) }

OID 国際化リソース識別子の値: "/Joint-ISO-ITU-T/ASN.1/仕様/Character_Strings/Numeric_String"

オブジェクト記述子の値: 「NumericString ASN.1 タイプ」

41.5条

オブジェクト識別子の値:
{ジョイント-iso-itu-t asn1(1)仕様(0)characterStrings(1)printableString(1)}

OID 国際化リソース識別子の値: "/Joint-ISO-ITU-T/ASN.1/仕様/Character_Strings/Printable_String"

オブジェクト記述子の値: 「PrintableString ASN.1 タイプ」

42.1項

オブジェクト識別子の値:
{ Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) iso10646(0) }

OID 国際化リソース識別子の値: "/Joint-ISO-ITU-T/ASN.1/仕様/モジュール/ISO_10646"

オブジェクト記述子の値: 「ASN.1 文字モジュール」

D.2 項

オブジェクト識別子の値:
{ Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) オブジェクト識別子(1) }

OID 国際化リソース識別子の値: "/Joint-ISO-ITU-T/ASN.1/仕様/モジュール/Object_Identifiers"

オブジェクト記述子の値: 「ASN.1 オブジェクト識別子モジュール」

D.2 ASN.1 およびエンコード ルール標準のオブジェクト識別子

この節は、ASN.1 標準 (Rec. ITU-T X.680 | ISO/IEC 8824-1 から Rec. ITU- T X.693 | ISO/IEC 8825-4)。

注 – これらの値は、OBJECT IDENTIFIER タイプおよびそこから派生したタイプの値表記で使用できます。この句で指定されたモジュールで定義された値参照はすべてエクスポートされ、それらを使用するモジュールによってインポートされる必要があります。

```
ASN1-オブジェクト識別子-モジュール { Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) オブジェクト識別子(1) }
"/Joint-ISO-ITU-T/ASN.1/仕様/モジュール/Object_Identifiers"
定義 ::= 開始

    -- NumericString ASN.1 タイプ ( 41.3 を参照) --
    数値文字列オブジェクト識別子 ::= {ジョイント-iso-itu-t
    asn1(1) 仕様(0) 文字列(1) 数値文字列(0) }
```

```
-- PrintableString ASN.1 タイプ ( 41.5 を参照) --
printableString オブジェクト識別子 ::=
{ジョイント-iso-itu-t asn1(1)仕様(0)characterStrings(1)printableString(1)}

-- ASN.1 文字モジュール ( 42.1 を参照) --
asn1CharacterModule オブジェクト識別子 ::=
{ Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) iso10646(0) }

-- ASN.1 オブジェクト識別子モジュール (本モジュール) --
asn1ObjectIdentifierModule オブジェクト識別子 ::=
{ Joint-iso-itu-t asn1(1) 仕様(0) モジュール(0) オブジェクト識別子(1) }

-- 単一の ASN.1 タイプの BER エンコーディング --
ber オブジェクト識別子 ::= { Joint-iso-itu-
t asn1(1) Basic-encoding(1) }

-- 単一の ASN.1 タイプの CER エンコーディング --
cer オブジェクト識別子 ::=
{ Joint-iso-itu-t asn1(1) ber-derived(2) canonical-encoding(0) }

-- 単一の ASN.1 タイプの DER エンコーディング --
オブジェクト識別子 ::= { Joint-iso-itu-t
asn1(1) ber-derived(2)distinguished-encoding(1) }

-- 単一の ASN.1 タイプの PER エンコーディング (基本アライメント) --
perBasicAligned オブジェクト識別子 ::= { Joint-iso-itu-t
asn1(1) Packed-encoding(3) Basic(0) aligned(0) }

-- 単一の ASN.1 タイプの PER エンコーディング (基本的なアライメントなし) --
perBasicUnaligned OBJECT IDENTIFIER ::= { Joint-iso-itu-t
asn1(1) Packed-encoding(3) Basic(0) unaligned(1) }

-- 単一の ASN.1 タイプの PER エンコーディング (標準アライメント) --
CanonicalAligned ごとのオブジェクト識別子 ::=
{ Joint-iso-itu-t asn1(1) パックエンコーディング(3) canonical(1) aligned(0) }

-- 単一の ASN.1 タイプの PER エンコーディング (標準的な非整列) --
CanonicalUnaligned ごとのオブジェクト識別子 ::=
{ Joint-iso-itu-t asn1(1) パックエンコーディング(3) canonical(1) unaligned(1) }

-- 単一の ASN.1 タイプの XER エンコーディング (基本) --
xerBasic オブジェクト識別子 ::= {joint-iso-itu-t
asn1(1) xml-encoding(5) Basic(0) }

-- 単一の ASN.1 タイプの XER エンコーディング (標準) --
xerCanonical オブジェクト識別子 ::= {joint-iso-itu-t
asn1(1) xml-encoding(5) canonical(1) }

-- 単一 ASN.1 タイプの EXER エンコード (拡張) --
xerExtended オブジェクト識別子 ::= {joint-iso-itu-t
asn1(1) xml-encoding(5) extend(2) }

終了-- ASN1 オブジェクト識別子モジュール --
```

付録E

エンコーディングのリファレンス

(この付録は、この勧告 | 国際規格の不可欠な部分を形成します。)

E.1この付録では、現在定義されているエンコード参照と推奨事項を指定します。そのエンコード参照を使用してエンコード命令の構文形式 (およびセマンティクス) を指定する国際標準 (関連するエンコード命令を持たないTAGエンコード参照を除く)。

注 – ここで指定されていないエンコード参照が ASN.1 仕様に記載されている場合は、関連するエンコード命令を無視し、警告診断 (のみ) を行うことをお勧めします。

E.2表 E.1 の列 1 のエンコーディング参照は現在定義されています。関連するエンコード命令 (該当する場合) の構文とセマンティクスは、勧告 | で定義されています。表 E.1 の列 2 で参照される国際規格。

表 E.1 – 特定のエンコーディング参照に関連付けられたセマンティクスを定義する標準

エンコーディングのリファレンス	標準を参照
鬼ごっこ	この推奨事項 |国際標準
XER	記録ITU-T X.693 (2021) | ISO/IEC 8825-4 :2021
PER	記録ITU-T X.695 (2021) | ISO/IEC 8825-6 :2021

付録F

国際オブジェクト識別子ツリーでのアークの割り当てと使用

(この附属書は、この勧告 | 国際規格の不可欠な部分を形成するものではありません。)

F.1 一般的な

F.1.1国際オブジェクト識別子ツリーは Rec. で指定されています。 ITU-T X.660 | ISO/IEC 9834-1 付属書 A。登録機関の階層を定義しており、それぞれが以下を割り当てます。

a) ノードを識別するための各従属アークに対する主整数識別子 (曖昧さのない一意の)
担当するノードの下にあります。

b) (オプションで) 下位アークの識別を人間が読みやすくすることができる、各下位アークの二次識別子。ただし、必ずしも明確である必要はありません。

c) 主整数の文字エンコーディングである整数値の Unicode ラベル (明確)
円弧の値。

d) (オプションで) 下位の代替 ID を提供する追加の Unicode ラベル (明確な)
円弧。

F.1.2 Unicode ラベルは、(若干の制限はありますが) Unicode 文字の任意のシーケンスです。

F.1.3 記録ITU-T X.660 | ISO/IEC 9834-1、付録 A (および推奨事項 | 国際規格と
が参照する手順) は、国際オブジェクト識別子ツリーを定義します。

注 – OID 割り当てに関する情報の非公式リポジトリは、 http://www.oid-info.com で入手できます。

F.2 オブジェクト識別子(OBJECT IDENTIFIER)タイプによる国際オブジェクト識別子ツリーの使用

F.2.1 このタイプ (およびその値表記とエンコード) は、各アークの主整数値のみを使用して (オプションで値表記、XML 値表記、および XML に二次識別子を含めて) 国際オブジェクト識別子ツリーのノードを識別する手段を提供します。エンコーディング) 。

F.2.2 これは長期にわたって使用されており、国際オブジェクトのノードを識別するコンパクトな手段を提供します。
バイナリエンコードされたコンピュータ通信における識別子ツリー。

F.3 OID 国際化リソース識別子による国際オブジェクト識別子ツリーの使用 (OID-IRI)タイプ

F.3.1 このタイプ (およびその値の表記とエンコーディング) は、国際ネットワークのノードを識別する手段を提供します。
各アークの Unicode ラベルのみを使用するオブジェクト識別子ツリー。

F.3.2同じ構文は、IANA に登録された「oid」IRI および URI スキームの本体も形成します (Rec. ITU-T X.660 | ISO/IEC 9834-1 の付録 F を参照)。

付録G

例とヒント

(この附属書は、この勧告 | 国際規格の不可欠な部分を形成するものではありません。)

この付録には、(仮説的な) データ構造の説明における ASN.1 の使用例が含まれています。また、ASN.1 のさまざまな機能を使用するためのヒントまたはガイドラインも含まれています。特に明記されていない限り、 AUTOMATIC TAGSの環境が想定されます。

G.1 人事記録の例

ASN.1 の使用は、単純な仮想の人事記録を使用して説明されています。

G.1.1 人事記録の非公式な説明

人事記録の構造と特定の個人のその値を以下に示します。

名前： ジョン•P•スミス

タイトル： 監督

従業員番号: 51

入社日： 1971 年 9 月 17 日

配偶者の名前： メアリー•T•スミス

子どもの数： 2

お子様の情報

名前： ラルフ•T•スミス

生年月日 1957 年 11 月 11 日

お子様の情報

名前： スーザン•B•ジョーンズ

生年月日 1959 年 7 月 17 日

G.1.2 レコード構造の ASN.1 記述

すべての人事レコードの構造は、データ型の標準表記を使用して以下に正式に説明されます。

```
人事レコード ::= [アプリケーション 0] セット
{
    名前          名前、
    タイトル      可視文字列、
    番号          従業員番号、
    入社日        日付、
    配偶者の名前、
    子供たち      ChildInformation のシーケンス デフォルト {}
}

子情報 ::= セット
{
    名前生        名前、
    年月日        日付
}

名前 ::= [アプリケーション 1] シーケンス
{
    名            可視文字列、
    イニシャル    可視文字列、
    苗字          可視文字列
}

従業員番号 ::= [アプリケーション 2] 整数

日付 ::= [アプリケーション 3] VisibleString -- YYYY MMDD
```

この例は、ASN.1 構文の解析の一側面を示しています。構文構造DEFAULT はSEQUENCEまたはSETのコンポーネントにのみ適用でき、 SEQUENCE OFの要素には適用できません。したがって、 DEFAULT {

PersonalRecordの}は、ChildInformationではなく、子に適用されます。

G.1.3 レコード値の ASN.1 記述

ジョン•スミスの人事記録の値は、データ値の標準表記を使用して以下に正式に説明されます。

```
                名前                    {名「ジョン」、イニシャル「P」、家族名「スミス」}、
{ 役職                  "監督"、
番号 入社日         51、
配偶者の名前 {名        "19710917",
「メアリー」、イニシャル「T」、家族名「スミス」}、
子供たち
{ {名前 {名「ラルフ」、イニシャル「T」、家族名「スミス」},
   生年月日 "19571111"},
  {名前 {名「スーザン」、イニシャル「B」、家族名「ジョーンズ」},
   誕生日 "19590717" }
}
            }
```

または XML 値表記では次のようになります。

```
人::=
<人事記録>
  <名前>
    <指定された名前>ジョン</指定された名前>
    <initial>P</initial>
    <familyName>スミス</familyName>
  </名前>
  <title>ディレクター</title>
  <数字>51</数字>
  <採用日>19710917</採用日>
  <配偶者の名前>
    <指定された名前>メアリー</指定された名前>
    <initial>T</initial>
    <familyName>スミス</familyName>
  </nameOf配偶者>
  <子供たち>
    <お子様の情報>
      <名前>
        <与えられた名前>ラルフ</与えられた名前>
        <initial>T</initial>
        <familyName>スミス</familyName>
      </名前>
      <誕生日>19571111</誕生日>
    </子供情報>
    <お子様の情報>
      <名前>
        <givenName>スーザン</givenName>
        <initial>B</initial>
        <familyName>ジョーンズ</familyName>
      </名前>
      <誕生日>19590717</誕生日>
    </子供情報>
  </子供>
</人事記録>
```

G.2 表記の使用に関するガイドライン

この勧告で定義されているデータ型と正式な表記法 |国際規格は柔軟なので、
それらを使用して幅広いプロトコルを設計できます。ただし、この柔軟性は、特に表記法に初めて取り組む場合に混乱を招くことがあります。この付録では、表記の使用に関するガイドラインと例を示すことで混乱を最小限に抑えるよう努めています。組み込みデータ型ごとに、1 つ以上の使用ガイドラインが提供されます。文字列型（たとえば、 VisibleString)および第 46 項から第 48 項で定義されている型については、ここでは扱いません。

G.2.1 ブール値

G.2.1.1ブール型を使用して、論理 (つまり 2 つの状態) 変数の値 (たとえば、「はい」または「いいえ」の質問に対する答え) をモデル化します。

例

雇用されている ::= ブール値

G.2.1.2ブール型に参照名を割り当てる場合は、真の状態を表す名前を選択してください。

例

既婚 ::= ブール値

ない

婚姻状況 ::= ブール値

G.2.2 整数

G.2.2.1整数型を使用して、基本変数または整数変数の値 (すべての実用的な目的で、大きさに制限はありません) をモデル化します。

例

CheckingAccountBalance ::= INTEGER -- セント単位。マイナスはオーバードローを意味します。

残高 CheckingAccountBalance ::= 0

または XML 値表記を使用します。

残高 ::= <CheckingAccountBalance>0</CheckingAccountBalance>

G.2.2.2整数型の許容最小値と最大値を名前付き数値として定義します。

例

月の日 ::= INTEGER {最初(1)、最後(31)}

今日の月の日::= 最初

不明な月の日 ::= 0

または XML 値表記を使用します。

今日 ::= <月の日><最初/></月の日>

不明 ::= <月の日>0</月の日>

最初と最後の名前付きの数字は、読者にとっての意味上の重要性を理由に選択されており、 DayOfTheMonth が1 未満、31 より大きい、または 1 と 31 の間の他の値を持つ可能性を排除するものではないことに注意してください。

DayOfTheMonthの値を最初と最後だけに制限するには、次のように記述します。

DayOfTheMonth ::= INTEGER {first(1), last(31)} (最初 | 最後)

DayOfTheMonthの値を1 から 31 までのすべての値に制限するには、次のように記述します。

DayOfTheMonth ::= INTEGER {first(1), last(31)} (最初 .. 最後)

月の日 月の日 ::= 4

または XML 値表記を使用します。

月の日 ::= <月の日>4</月の日>

G.2.3 列挙型

G.2.3.1列挙型を使用して、3 つ以上の状態を持つ変数の値をモデル化します。唯一の制約が区別性である場合は、ゼロから始まる値を割り当てます。

例

```
DayOfTheWeek ::= ENUMERATED {日曜日(0)、月曜日(1)、火曜日(2)、
                    水曜日(3)、木曜日(4)、金曜日(5)、土曜日(6)}
```

firstDay DayOfTheWeek ::= 日曜日

または XML 値表記を使用します。

```
firstDay ::= <DayOfTheWeek><sunday/></DayOfTheWeek>
```

sunday、 mondayなどの列挙は読者にとっての意味上の重要性から選択されていますが、 DayOfTheWeekはこれらの値の 1 つを想定し、他の値は想定しないことに制限されていることに注意してください。さらに、値に割り当てられるのは、 sunday、 mondayなどの名前のみです。同等の整数値は許可されません。

G.2.3.2拡張可能な列挙型を使用して、現在は 2 つの状態しか持たないが、プロトコルの将来のバージョンでは追加の状態を持つ可能性がある変数の値をモデル化します。

例

```
MaritalStatus ::= ENUMERATED {独身、既婚}
        -- MaritalStatus の最初のバージョン
```

見越して：

```
MaritalStatus ::= ENUMERATED {独身、既婚、 …、未亡人}
        -- MaritalStatus の 2 番目のバージョン
```

そしてさらに後:

```
MaritalStatus ::= ENUMERATED {独身、既婚、 …、死別、離婚}
        -- MaritalStatus の 3 番目のバージョン
```

G.2.4 実数

G.2.4.1実数型を使用して近似数値をモデル化します。

例

```
角度ラジアン ::= REAL

pi REAL ::= {仮数 31415926535897932384626433832 79、基数 10、指数－30}
```

または、REAL の代替値表記を使用します。

```
円周率 ::= 3.14159265358979323846264338327
```

または XML 値表記を使用します。

```
パイ::=
<リアル>
    3.14159265358979323846264338327 </REAL>
```

G.2.4.2アプリケーション設計者は、浮動小数点ハードウェアの違いや、アプリケーションで単長浮動小数点を使用するか倍長浮動小数点を使用するかなどの実装決定の違いにもかかわらず、実数値との完全な相互作用を保証したい場合があります。これは次のようにして実現できます。

```
App-X-Real ::= REAL (WITH COMPONENTS {仮数
                (－16777215..16777215)、基数 (2)、指
                数
                (－125..128) } )
/*
  送信者はこれらの範囲外の値を送信してはならず、準拠する
  受信者はこれらの範囲内のすべての値を受信して処理できる
  必要があります。 */


周囲 App-X-Real ::= {仮数 16、基数 2、指数 1}
```

または XML 値表記を使用します。

```
胴回り::=
<アプリ-X-リアル>
  32
</App-X-Real>
```

G.2.5 ビット列

G.2.5.1ビット文字列型を使用して、形式と長さが指定されていない、または他の場所で指定され、ビット単位の長さが必ずしも 8 の倍数であるとは限らないバイナリ データをモデル化します。

例

G3FacsimilePage ::= BIT STRING --
Rec. に準拠するビットのシーケンス。 ITU-T T.4。

画像 G3FacsimilePage ::= '100110100100001110110'B

トレーラービット文字列 ::= '0123456789ABCDEF'H

body1 G3ファクシミリページ ::= '1101'B

body2 G3FacsimilePage ::= '1101000'B

または XML 値表記を使用します。

```
画像 ::= <G3FacSimile>100110100100001110110</G3FacSimile>

トレーラー::=
    <BIT_STRING>
        0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101
        1110 1111
    </BIT_STRING>

body1 ::= <G3FacSimile>1101</G3FacSimile>

body2 ::= <G3FacSimile>1101000</G3FacSimile>
```

body1とbody2 は、末尾の 0 ビットが重要であるため、別個の抽象値であることに注意してください(G3FacsimilePage の定義に「NamedBitList」がないため)。

G.2.5.2サイズ制約のあるビット文字列型を使用して、固定サイズのビット フィールドの値をモデル化します。

例

```
ビットフィールド ::= ビット文字列 (サイズ (12))

マップ1 ビットフィールド ::= '100110100100'B

マップ2ビットフィールド::= '9A4'H

map3 BitField ::= '1001101001'B -- 不正 - サイズ制約に違反しています。
```

または XML 値表記を使用します。

```
マップ1 ::= <ビットフィールド>100110100100</ビットフィールド>
```

map2の末尾 4 ビットは重要ではないため、 map1とmap2 は同じ抽象値であることに注意してください。

G.2.5.3ビット文字列タイプを使用して、ビット マップの値をモデル化します。ビット マップは、対応する順序付けされたオブジェクトのコレクションのそれぞれに対して特定の条件が当てはまるかどうかを示す、順序付けされた論理変数のコレクションです。

```
曜日 ::= ビット文字列 {
    日曜日(0)、月曜日(1)、火曜日(2)、水曜日(3)、木曜日
    (4)、金曜日(5)、土曜日(6) } (SIZE (0..7))


SunnyDaysLastWeek1 DaysOfTheWeek ::= {日曜日、月曜日、水曜日} SunnyDaysLastWeek2
DaysOfTheWeek ::= '1101'B sundayDaysLastWeek3
DaysOfTheWeek ::= '1101000'B

SunnyDaysLastWeek4 DaysOfTheWeek ::= '11010000'B -- 違法です
```

または XML 値表記を使用します。

```
晴れた日先週1 ::=
<曜日> <日曜日/><月曜日/
  ><水曜日/>
        </曜日>

SunnyDaysLastWeek2 ::= <DaysOfTheWeek>1101</DaysOfTheWeek>

SunnyDaysLastWeek3 ::= <DaysOfTheWeek>1101000</DaysOfTheWeek>
```

ビット文字列値の長さが 7 ビット未満の場合、欠落しているビットはその日が曇りの日であることを示しているため、上記の最初の 3 つの値は同じ抽象値を持つことに注意してください。

G.2.5.4ビット文字列タイプを使用して、ビット マップの値をモデル化します。ビットマップは、対応する順序付けされたオブジェクトのコレクションのそれぞれに特定の条件が当てはまるかどうかを示す、固定サイズの順序付けされた論理変数のコレクションです。

```
曜日 ::= ビット文字列 {
        日曜日(0)、月曜日(1)、火曜日(2)、水曜日(3)、木曜日(4)、
        金曜日(5)、土曜日(6) } (サイズ(7))

SunnyDaysLastWeek1 DaysOfTheWeek ::= {日曜日、月曜日、水曜日} SunnyDaysLastWeek2
DaysOfTheWeek ::= '1101'B -- 不正 -- サイズ制約に違反しています。晴れた日先週今週の
                                                3 日 ::= '1101000'B


SunnyDaysLastWeek4 DaysOfTheWeek ::= '11010000'B -- 不正 -- サイズ制約に違反しています。
```

1 番目と 3 番目の値は同じ抽象値を持つことに注意してください。

G.2.5.5名前付きビットを持つビット文字列タイプを使用して、関連する論理変数のコレクションの値をモデル化します。

例

```
個人ステータス ::= ビット文字列
        {既婚(0)、就業中(1)、退役軍人(2)、大学卒業生(3)}

billClinton PersonalStatus ::= {既婚、雇用、大学卒業}

ヒラリー•クリントンの個人ステータス ::= '110100'B
```

または XML 値表記を使用します。

```
ビル•クリントン ::=
<個人ステータス> <結婚/
   > <雇用/> <大学
   卒業/>

</個人ステータス>

ヒラリークリントン ::= <個人ステータス>110100</個人ステータス>
```

billClintonとHillaryClinton は同じ抽象値を持っていることに注意してください。

G.2.6 オクテット文字列

G.2.6.1オクテット文字列型を使用して、形式と長さが未指定、または他の場所で指定され、ビット長が 8 の倍数であるバイナリ データをモデル化します。

例

```
G4FacsimileImage ::= OCTET STRING --
Rec. に準拠するオクテットのシーケンス。 ITU-T T.5 および CCITT Rec. T.6

画像 G4FacsimileImage ::= '3FE2EBAD471005'H
```

または XML 値表記を使用します。

```
画像 ::= <G4FacSimileImage>3FE2EBAD471005</G4FacSimileImage>
```

G.2.6.2適切な文字列型が利用可能な場合は、オクテット文字列型よりも制限された文字列型を使用します。

例

```
姓 ::= PrintableString

大統領の姓 ::= 「クリントン」
```

または XML 値表記を使用します。

```
大統領 ::= <姓>クリントン</姓>
```

G.2.7 UniversalString、BMPString、およびUTF8String

ISO/IEC 10646 Basic Multilingual Plane (BMP) の文字のみで構成される情報文字列をモデル化するにはBMPString型またはUTF8String型を使用し、制限されていない ISO/IEC 10646 文字で構成される文字列をモデル化するには UniversalStringまたはUTF8Stringを使用します。 BMPに。

G.2.7.1 Level1またはLevel2を使用して、実装レベルが結合文字の使用に制限を設けていることを示します。

例

```
ロシア名 ::= キリル文字 (レベル 1)
    -- RussianName は結合文字を使用しません。

SaudiName ::= BasicArabic (SIZE (1..100) ^ Level2)
    -- SaudiName は結合文字のサブセットを使用します。
```

文字 の表現:

```
greekCapitalLetterSigma BMPString ::= {0, 0, 3, 163}
```

または XML 値表記を使用します。

```
greekCapitalLetterSigma ::= <BMPString>&#x03a3;</BMPString>
```

文字列「f → 」の表現:

```
rightwardsArrow UTF8String ::= {0, 0, 33, 146}

無限 UTF8String ::= {0, 0, 34, 30}

プロパティ UTF8String ::= {"f "、rightwardsArrow、 " "、無限大}
```

または XML 値表記を使用します。

```
プロパティ ::= <UTF8String>f &#x2192; &#x221E;</UTF8String>
```

G.2.7.2コレクションは、「UnionMark」を使用することにより、選択したサブセット（つまり、BASIC LATIN コレクション内のすべての文字を含む）に拡張できます（条項 50 を参照）。

例

```
KatakanaAndBasicLatin ::= UniversalString (FROM (カタカナ | BasicLatin))
```

G.2.8 文字列

制限なしの文字列タイプを使用して、制限された文字列タイプのいずれかを使用してモデル化できない情報の文字列をモデル化します。文字のレパートリーとそのオクテットへのコーディングを必ず指定してください。

例

```
PackedBCDString ::= 文字列 (コンポーネントあり {
                                                識別 (WITH COMPONENTS {固定 PRESENT })

        /* 抽象構文および転送構文は、以下に定義されている
           packedBCDString-AbstractSyntaxIdおよび
      packedBCDString-TransferSyntaxIdでなければなりません。
    */
                    })

    /* 文字抽象構文のオブジェクト識別子の値
       (文字セット) アルファベットが 0
       ～ 9 の数字です。 */

PackedBCDString-AbstractSyntaxId オブジェクト識別子 ::= { Joint-iso-itu-t
    example(999) PackedBCD(2) charSet(0) }

    /* オクテットごとに 2 桁をパックする文字転送構文のオブジェクト識別
       子の値。各桁は 0000 ～ 1001、11112 としてエンコードされ、パディン
       グに使用されます。 */

PackedBCDString-TransferSyntaxId オブジェクト識別子 ::= { Joint-iso-itu-t
                    example(999) PackedBCD(2)
                        CharacterTransferSyntax(1) }
/* PackedBCDString のエンコーディングには、定義された文字のエンコーディ
   ングのみが含まれ、必要な長さフィールドが含まれます。BER の場合はタグを含
   むフィールドが含まれます。 「fixed」が指定されているため、オブ
   ジェクト識別子の値は引き継がれません。
    */
```

または XML 値表記を使用します。

```
パックされたBCDString-AbstractSyntaxId ::=
<OBJECT_IDENTIFIER>
        Joint-iso-itu-t.example(999).packedBCD(2).charSet(0)
</OBJECT_IDENTIFIER>


PackedBCDString-TransferSyntaxId ::=
<OBJECT_IDENTIFIER> Joint-iso-itu-
t.example(999).packedBCD(2).characterTransferSyntax(1)
                </OBJECT_IDENTIFIER>
```

または：

```
パックされたBCDString-AbstractSyntaxId ::=
<OBJECT_IDENTIFIER>2.999.2.0</OBJECT_IDENTIFIER>

PackedBCDString-TransferSyntaxId ::=
<OBJECT_IDENTIFIER>2.999.2.1</OBJECT_IDENTIFIER>
```

注 – エンコーディング規則は、抽象値がエンコーディングで保持されることを保証しますが、必ずしもオブジェクト識別子の値を含む形式でCHARACTER STRING型の値をエンコードするとは限りません。

G.2.9 ヌル

シーケンスのコンポーネントが事実上存在しないことを示すには、null タイプを使用します。

例

```
患者識別子 ::= SEQUENCE {
名前の                  可視文字列、
部屋番号                選択 {
    部屋                INTEGER、
    outPatient NULL -- 外来患者の場合 --
}
}

lastPatient PatientIdentifier ::= { name
"Jane Doe"、roomNumber
outPatient : NULL
}
```

または XML 値表記を使用します。

```
最後の患者 ::=
<PatientIdentifier>
  <name>Jane Doe</name>
  <roomNumber><outPatient/></roomNumber>
</PatientIdentifier>
```

G.2.10 シーケンスとシーケンス

G.2.10.1 sequence-of 型を使用して、型が同じで、数が大きいか予測不可能で、順序が重要な変数のコレクションをモデル化します。

例

```
NamesOfMemberNations ::= VisibleString の配列-- アルファ
ベット順

加盟国の最初の 2 つの名前 ::= {"オーストラリア", "オーストリア"}
```

または、オプションの識別子を使用して：

```
NamesOfMemberNations2 ::= memberNation VisibleString の配列-- アルファ
ベット順

firstTwo2 NamesOfMemberNations2 ::=
{memberNation "オーストラリア", memberNation "オーストリア"}
```

XML 値表記を使用すると、上記の 2 つの値は次のようになります。

```
最初の 2 ::=
<加盟国の名前>
  <VisibleString>オーストラリア</VisibleString>
```

```
    <VisibleString>オーストリア</VisibleString>
</加盟国の名前>

最初の 2 ::=
<加盟国の名前2>
    <memberNation>オーストラリア</memberNation>
    <memberNation>オーストリア</memberNation>
</加盟国の名前2>
```

G.2.10.2シーケンス型を使用して、型が同じで、数が既知で控えめで、順序が重要である変数のコレクションをモデル化します (コレクションの構成が 1 つのバージョンから変更される可能性が低い場合)。プロトコルの次へ。

例

```
役員の名前 ::= SEQUENCE {
        社長                    可視文字列、
        副社長                  可視文字列、
        秘書                    可視文字列}

acmeCorp 役員名 ::= {
        社長                    "ジェーン•ドウ"、
        副社長                  "ジョン•ドウ"、
        秘書                    「ジョー•ドゥ」 }
```

または XML 値表記を使用します。

```
アクメコープ ::=
<役員氏名>
    <社長>ジェーン•ドウ</社長>
     <副大統領>ジョン•ドゥ</副大統領>
     <秘書>ジョー•ドゥ</秘書>
</役員の名前>
```

G.2.10.3コレクションの構成が 1 つのバージョンから変更される可能性が低いという条件で、拡張不可能なシーケンス型を使用して、型が異なり、その数が既知で控えめで、順序が重要である変数のコレクションをモデル化します。プロトコルを次へ。

例

```
資格情報 ::= SEQUENCE {
        ユーザー名              可視文字列、
        パスワード              可視文字列、
        口座番号                    整数}
```

G.2.10.4拡張可能なシーケンス型を使用して、順序が重要で、現在その数がわかっていて控えめであるが、増加が予想される変数のコレクションをモデル化します。

例

```
Record ::= SEQUENCE {-- 「Record」を含むプロトコルの最初のバージョン
ユーザー名                  可視文字列、
パスワード                  可視文字列、
アカウント番号                  整数、
...、
...
}
```

見越して：

```
Record ::= SEQUENCE {-- 「Record」を含むプロトコルの 2 番目のバージョン
ユーザー名                  可視文字列、
パスワード                  可視文字列、
口座番号                        整数、
...、
[[2:                -- プロトコルバージョン2で追加された拡張機能
  最後にログインしました          GeneralizedTime オプション、
  分最終ログイン                整数
]]、
...
}
```

それ以降 (プロトコルのバージョン 3 ではRecord に追加は行われませんでした):

```
Record ::= SEQUENCE {-- 「Record」を含むプロトコルの第 3 バージョン
ユーザー名                  可視文字列、
パスワード                  可視文字列、
アカウント番号                  整数、
...、
[[2:                -- プロトコルバージョン2で追加された拡張機能
  最後にログインしました          GeneralizedTime オプション、
  分最終ログイン                整数
]]、
[[4:                -- プロトコルバージョン3で追加された拡張機能
        証明書のサ              証明書、
  ム ]]、                サムプリントはオプションです

...
}
```

G.2.11 集合と集合

G.2.11.1セット型を使用して、数が既知で控えめで順序が重要ではない変数のコレクションをモデル化します。自動タグ付けが有効でない場合は、以下に示すようにコンテキスト固有のタグを付けて各変数を識別します。 (自動タグ付けの場合、タグは必要ありません。)

例

```
ユーザー名 ::= SET {
        個人名組織名国名                [0] 可視文字列、
                                [1] 可視文字列、
                                [2] 可視文字列}

ユーザー ユーザー名 ::= {
        国名 個人名 組織            "ナイジェリア"、
        名                    「ジョナス•マルバ」、
                                「株式会社気象学」}
```

または XML 値表記を使用します。

```
ユーザー::=
<ユーザー名>
  <国名>ナイジェリア</国名>
  <個人名>ジョナス マルバ</個人名>
  <組織名>気象株式会社</組織名>
</ユーザー名>
```

G.2.11.2 OPTIONALを指定したセット型を使用して、数が既知で適度に小さく順序が重要ではない別の変数コレクションの (適切または不適切な) サブセットである変数のコレクションをモデル化します。自動タグ付けが有効でない場合は、以下に示すようにコンテキスト固有のタグを付けて各変数を識別します。 (自動タグ付けの場合、タグは必要ありません。)

例

```
ユーザー名 ::= SET {
個人名                        [0] 可視文字列、
組織名                        [1] VisibleString オプション
            -- デフォルトはローカル組織のもの --        、
countryName                   [2] VisibleString オプション
            -- デフォルトは現地の国名 -- }
```

G.2.11.3拡張可能なセット タイプを使用して、プロトコルのバージョンごとに構成が変わる可能性がある変数のコレクションをモデル化します。以下では、モジュール定義でAUTOMATIC TAGS が指定されていることを前提としています。

例

```
ユーザー名 ::= SET {
個人名              可視文字列、          -- 「UserName」の最初のバージョン
組織名                  VisibleString オプションの国名    、
                        VisibleString オプション、
...、
...
}
```

```
ユーザー ユーザー名 ::= { 個人名 "ジョナス マルバ" }
```

または XML 値表記を使用します。

```
ユーザー::=
<ユーザー名>
  <個人名>ジョナス マルバ</個人名>
</ユーザー名>
```

見越して：

```
ユーザー名 ::= SET { 個人名          -- 「UserName」の 2 番目のバージョン
組織名 国名                          可視文字列、
                                        VisibleString オプション、
                                        VisibleString オプション、
...、
[[2:                              -- プロトコルバージョン2で追加された拡張機能
  InternetEmailAddress VisibleString,
  ファクス番号                       VisibleString オプション
]]、
...
}

ユーザー ユーザー名 ::= {
個人名インターネッ                    「ジョナス•マルバ」、
ト電子メールアドレス                        「jonas@meteor.ngo.com」
}
```

または XML 値表記を使用します。

```
ユーザー::=
<ユーザー名>
  <個人名>ジョナス マルバ</個人名>
  <internetEmailAddress>jonas@meteor.ngo.com</internetEmailAddress>
</ユーザー名>
```

それ以降 (プロトコルのバージョン 3 と 4 ではUserName に追加は行われませんでした):

```
UserName ::= SET {-- 「UserName」を含むプロトコルの 5 番目のバージョン
個人名組織名国名                      可視文字列、
                                        VisibleString オプション、
                                        VisibleString オプション、
...、
[[2:                              -- バージョン2で追加された拡張機能
  InternetEmailAddress VisibleString,
  ファクス番号                       VisibleString オプション
]]、
[[5:                              -- バージョン5で追加された拡張機能
電話番号 ]],                          VisibleString オプション

...
}

user ユーザー名 ::= { 個人名
インターネット電子                    「ジョナス•マルバ」、
メールアドレス                          「jonas@meteor.ngo.com」
}
```

または XML 値表記を使用します。

```
ユーザー::=
<ユーザー名>
  <個人名>ジョナス マルバ</個人名>
  <internetEmailAddress>jonas@meteor.ngo.com</internetEmailAddress>
</ユーザー名>
```

G.2.11.4 型のセットを使用して、型が同じで順序が重要ではない変数のコレクションをモデル化します。

例

```
キーワード ::= VisibleString のセット-- 任意の順序

someASN1Keywords キーワード ::= {"INTEGER"、"BOOLEAN"、"REAL"}
```

または、オプションの識別子を使用して:

```
キーワード2 ::= キーワードVisibleStringのセット-- 任意の順序

someASN1キーワード2 キーワード2 ::= {キーワード "INTEGER", キーワード "BOOLEAN",
                キーワード「リアル」}
```

XML 値表記を使用すると、上記の 2 つの値は次のようになります。

```
いくつかのASN1キーワード ::=
<キーワード>
  <VisibleString>整数</VisibleString>
  <VisibleString>ブール値</VisibleString>
  <VisibleString>REAL</VisibleString>
</キーワード>

いくつかのASN1キーワード2 ::=
<キーワード2>
  <キーワード>整数</キーワード>
  <キーワード>ブール値</キーワード>
  <キーワード>本物</キーワード>
              </キーワード2>
```

G.2.12 タグ付き

AUTOMATIC TAGS構造が導入される前は、ASN.1 仕様にはタグが頻繁に含まれていました。次の節では、タグ付けが通常適用される方法について説明します。 AUTOMATIC TAGSの導入により、新しい ASN.1 仕様ではタグ表記を使用する必要がなくなりましたが、古い表記を変更する場合はタグに注意する必要があるかもしれません。 ASN.1 表記法の新規ユーザーは、表記法を読みやすくするため、自動タグを使用することをお勧めします。

G.2.12.1ユニバーサル クラス タグは、この推奨事項内でのみ使用されます。国際標準。 [UNIVERSAL 30] (たとえば) という表記は、「UsefulTypes」の定義の精度を高めるためにのみ提供されています (45.1 を参照)。
他の場所では使用しないでください。

G.2.12.2タグの使用で頻繁に見られるスタイルは、仕様全体でアプリケーション クラス タグを 1 回だけ割り当て、それを使用して仕様内で広範囲に分散して使用されるタイプを識別することです。アプリケーション クラス タグは、アプリケーションの最も外側のCHOICE内の型にタグを付けるために (1 回のみ) 頻繁に使用され、アプリケーション クラス タグによる個々のメッセージの識別を提供します。以下は前者の場合の使用例です。

例

```
ファイル名 ::= [アプリケーション 8] SEQUENCE {
        ディレクトリ名                  可視文字列、
        directoryRelativeFileName VisibleString}
```

上記の例では、デフォルトのエンコード参照が「空」またはTAG であることを前提としています。それ以外の場合、上記の例は次のように記述されます。

```
ファイル名 ::= [タグ: アプリケーション 8] SEQUENCE {
        ディレクトリ名                  可視文字列、
        directoryRelativeFileName VisibleString}
```

後続の例でも同様の変更が必要になります。

G.2.12.3コンテキスト固有のタグ付けは、 SET、 SEQUENCE、またはCHOICE のすべてのコンポーネントにアルゴリズム的な方法で適用されることがよくあります。ただし、自動タグ機能を使用すると、これが簡単に実行できることに注意してください。

例

```
顧客レコード ::= SET {
        名前                            [0] 可視文字列、
        郵送先住所アカウン              [1] 可視文字列、
        ト番号残高期限                  [2] 整数、
                                        [3] INTEGER -- セント単位 --}

CustomerAttribute ::= CHOICE {
        名前                            [0] 可視文字列、
        郵送先住所アカウン              [1] 可視文字列、
        ト番号残高期限                  [2] 整数、
                                        [3] INTEGER -- セント単位 --}
```

G.2.12.4プライベート クラスのタグ付けは、通常、国際標準化された仕様では使用されるべきではありません (ただし、これを禁止することはできません)。企業によって作成されたアプリケーションは通常、アプリケーションおよびコンテキスト固有のタグ クラスを使用します。ただし、企業固有の仕様が国際標準化された仕様を拡張しようとするケースが時折あります。この場合、プライベート クラス タグを使用すると、企業固有の仕様を国際標準化された仕様の変更から部分的に保護するという利点が得られる可能性があります。 。

例

```
AcmeBadgeNumber ::= [プライベート 2] 整数

バッジ番号 アクメバッジ番号 ::= 2345
```

または XML 値表記を使用します。

```
バッジ番号 ::= <AcmeBadgeNumber>2345</AcmeBadgeNumber>
```

G.2.12.5すべてのタグでのIMPLICITのテキスト使用は、通常、古い仕様でのみ見られます。明示的なタグ付けが使用される場合、暗黙的なタグ付けが使用される場合よりも BER が生成する表現はコンパクトではありません。 PER はどちらの場合でも同じコンパクトなエンコーディングを生成します。 BER と明示的なタグ付けを使用すると、エンコードされたデータ内の基礎となる型(INTEGER、 REAL、 BOOLEANなど) がよりわかりやすくなります。これらのガイドラインでは、合法である場合は常に、例で暗黙的なタグ付けを使用しています。これにより、エンコード規則によってはコンパクトな表現が得られる場合があり、これは一部のアプリケーションでは非常に望ましいことです。他のアプリケーションでは、コンパクトさは、たとえば、強力な型チェックを実行する機能ほど重要ではない場合があります。後者の場合、明示的なタグ付けを使用できます。

例

```
顧客レコード ::= SET {
    名前                                [0] 暗黙的な VisibleString、
    郵送先住所アカウント                [1] 暗黙的な VisibleString、
    番号残高期限                        [2] 暗黙的な整数、
                                        [3] 暗黙の整数-- セント単位 --}

CustomerAttribute ::= CHOICE {
    名前                                [0] 暗黙的な VisibleString、
    郵送先住所アカウント                [1] 暗黙的な VisibleString、
    番号残高期限                        [2] 暗黙的な整数、
                                        [3] 暗黙の整数-- セント単位 --}
```

G.2.12.6この勧告を参照する新しい ASN.1 仕様におけるタグの使用に関するガイダンス |国際標準は非常にシンプルです: タグを使用しないでください。モジュールヘッダーにAUTOMATIC TAGSを入れて、タグのことは忘れてください。新しいバージョンでSET、 SEQUENCE 、またはCHOICEに新しいコンポーネントを追加する必要がある場合は、それらを最後に追加します。

G.2.13 選択

G.2.13.1 CHOICEを使用して、数が既知で控えめな変数のコレクションから選択された変数をモデル化します。

例

```
ファイル識別子 ::= CHOICE {
    相対名 VisibleString、
        -- ファイル名 (例: 「MarchProgressReport」)
    絶対名 VisibleString、
        -- ファイル名とそれを含むディレクトリ名
        -- (例: "<Williams>MarchProgressReport")
    シリアル番号 INTEGER
        -- システムによって割り当てられたファイルの識別子 --}

ファイル ファイル識別子 ::= シリアル番号 : 106448503
```

または XML 値表記を使用します。

```
ファイル識別子 ::=
    <ファイル識別子>
      <シリアル番号>106448503</シリアル番号>
    </ファイル識別子>
```

G.2.13.2拡張可能なCHOICEを使用して、プロトコルのバージョンごとに構成が変わる可能性がある変数のコレクションから選択された変数をモデル化します。

例

```
ファイル識別子 ::= CHOICE {                                    -- FileIdentifier の最初のバージョン
```

```
                相対名 VisibleString、

                絶対名 VisibleString、

                ...、...
}

        fileId1 ファイル識別子 ::= 相対名 : "MarchProgressReport.doc"
```

または XML 値表記を使用します。

```
ファイルID1 ::=
<ファイル識別子>
   <relativeName>MarchProgressReport.doc</relativeName> </FileIdentifier>
```

見越して：

```
        FileIdentifier ::= CHOICE {相対名                    -- FileIdentifier の 2 番目のバージョン
        VisibleString,絶対名 VisibleString、

        ...、
        serialNumber INTEGER, -- バージョン 2 で追加された拡張機能
        ...
}

fileId1 ファイル識別子 ::= 相対名 : "MarchProgressReport.doc"

fileId2 ファイル識別子 ::= シリアル番号 : 214
```

または XML 値表記を使用します。

```
ファイルID1 ::=
<ファイル識別子>
   <relativeName>MarchProgressReport.doc</relativeName> </FileIdentifier>


ファイルID2 ::=
<ファイル識別子>
   <シリアル番号>214</シリアル番号>
</ファイル識別子>
```

そしてさらに後:

```
        FileIdentifier ::= CHOICE {相対名                    -- FileIdentifier の 3 番目のバージョン
        VisibleString,絶対名 VisibleString、

        ...、
        serialNumber INTEGER, -- バージョン 2 で追加された拡張機能[[
                                        -- バージョン 3 で追加された拡張機能VendorSpecific
          VendorExt、未確認 NULL ]]、


        ...
}

        fileId1 ファイル識別子 ::= 相対名 : "MarchProgressReport.doc"

        fileId2 ファイル識別子 ::= シリアル番号 : 214

        fileId3 ファイル識別子 ::= 不明 : NULL
```

または XML 値表記を使用します。

```
        ファイルID1 ::=
        <ファイル識別子>
           <relativeName>MarchProgressReport.doc</relativeName> </FileIdentifier>


        ファイルID2 ::=
        <ファイル識別子>
           <シリアル番号>214</シリアル番号>
        </ファイル識別子>

        ファイルID3 ::=
```

```
        <ファイル識別子>
          <未確認/>
        </ファイル識別子>
```

G.2.13.3将来的に複数のタイプが許可される可能性が想定される場合は、1 つのタイプのみの拡張可能なCHOICEを使用します。

例

```
ご挨拶 ::= CHOICE { ポストカー          -- 「Greeting」の最初のバージョン
ド                  可視文字列、
...、
...
}
```

見越して：

```
挨拶 ::= CHOICE { postCard              -- 「ご挨拶」第2弾
VisibleString,
...、
[[2:                                    -- バージョン2で追加された拡張機能
  オーディ          オーディオ、
  オ ビデ           ビデオ
オ ]]、
...
}
```

G.2.13.4選択値が別の選択値内にネストされている場合は、複数のコロンが必要です。

例

```
挨拶 ::= [アプリケーション 12] 選択 {
    ポストカード        可視文字列、
    録音                声 }

音声 ::= 選択 {
    英語                オクテット文字列、
    スワヒリ語          オクテット文字列 }

myGreeting 挨拶 ::= 録音 : 英語 : '019838547E0'H
```

または XML 値表記を使用します。

```
私の挨拶 ::=
<ご挨拶>
  <録音><英語>019838547E0</英語></録音>
</ご挨拶>
```

G.2.14 選択タイプ

G.2.14.1選択タイプを使用して、以前に定義されたCHOICE の特定の選択肢のタイプを持つ変数をモデル化します。

G.2.14.2次の定義を検討してください。

```
ファイル属性 ::= CHOICE {
    最後に使用された日付 INTEGER、
    ファイル名          可視文字列}
```

その場合、次の定義が可能です。

```
属性リスト ::= SEQUENCE {
    最初の属性の日付と最後に使用された日付 < FileAttribute,
    第二属性ファイル名 < FileAttribute }
```

可能な値の表記は次のとおりです。

```
listOfAttributes 属性リスト ::= {
    最初の属性 27、
    2 番目の属性 "PROGRAM" }
```

または XML 値表記を使用します。

```
属性のリスト ::=
<属性リスト>
  <first-attribute>27</first-attribute>
```

```
   <2 番目の属性>プログラム</2 番目の属性>
</属性リスト>
```

G.2.15 オブジェクトクラスのフィールドタイプ

G.2.15.1オブジェクト クラス フィールド タイプを使用して、情報オブジェクト クラスによって定義されたタイプを識別します（「Rec.2.15.1」を参照）。
ITU-T X.681 | ISO/IEC 8824-2)。たとえば、情報オブジェクト クラスATTRIBUTEのフィールドは、タイプAttribute の定義に使用できます。

例

```
属性 ::= クラス {
属性タイプ(&A),
&属性ID                    一意のオブジェクト識別子
    }

属性 ::= シーケンス {
属性ID ATTRIBUTE.&attributeId、attributeValue                    -- これは通常は制約されます。
ATTRIBUTE.&AttributeType                                        -- これは通常は制約されます。
    }
```

ATTRIBUTE.&attributeIdとATTRIBUTE.&AttributeType は両方とも、情報オブジェクト クラス (ATTRIBUTE) への参照によって定義された型であるという点で、オブジェクト クラスのフィールド型です。タイプATTRIBUTE.&attributeId は、 ATTRIBUTEでOBJECT IDENTIFIER として明示的に定義されているため、固定されています。 ただし、タイプ
ATTRIBUTE.&AttributeType は、情報オブジェクト クラス ATTRIBUTE の定義でタイプが固定されていないため、ASN.1 を使用して定義された任意のタイプの値を保持できます。任意の型の値を保持できるというこの特性を持つ表記法は「オープン型表記法」と呼ばれるため、
ATTRIBUTE.&AttributeTypeはオープン型です。

G.2.16 埋め込み型 pdv

G.2.16.1embedded -pdv 型を使用して、型が指定されていない変数、または型の指定に使用される表記法に制限なく他の場所で指定されている変数をモデル化します。

例

```
ファイルコンテンツ ::= 埋め込み PDV

DocumentList ::= ドキュメント埋め込み PDV のシーケンス
```

G.2.17 外部

外部タイプはembedded-pdvタイプに似ていますが、識別オプションが少なくなります。一般に、新しい仕様では、embedded-pdv の柔軟性が高く、一部のエンコード ルールがその値をより効率的にエンコードするため、embedded-pdv の使用が好まれます。

G.2.18 インスタンス

G.2.18.1オブジェクト識別子フィールドを含む型と、オブジェクト識別子によって型が決定されるオープン型値を指定するには、instance-of を使用します。タイプのインスタンスは、オブジェクト識別子の値とタイプの間の関連付けが、 TYPE-IDENTIFIERから派生したクラスの情報オブジェクトを使用して指定されている場合にのみ使用できます(Rec. ITU-T X.681 | ISO/IEC 8824 を参照)。 2、付録 A および付録 C)。

例

```
アクセス制御クラス ::= タイプ識別子

Get-Invoke ::= SEQUENCE {
オブジェクトクラス                オブジェクトクラス、
オブジェクトインスタンスオブジェクトインスタンス、
accessControl ACCESS-CONTROL-CLASS のインスタンス-- これは通常、
                                    -- 制約された。
属性ID }                          属性.&属性ID
```

Get-Invoke は次と同等になります。

```
Get-Invoke ::= SEQUENCE {
オブジェクトクラス                オブジェクトクラス、
オブジェクトインスタンスオブジェクトインスタンス、
accessControl [UNIVERSAL 8] 暗黙のシーケンス {
        type-id ACCESS-CONTROL-CLASS.&id、 -- 制約付き。          -- これは通常です

        価値                      [0] ACCESS-CONTROL-CLASS.&Type --通常、これは
```

```
                                        -- 制約された。
        }、
        属性ID ATTRIBUTE.&attributeId
}
```

情報オブジェクト セットを使用して制約されるまで、instance-of 型の真の有用性はわかりませんが、そのような例はこの推奨事項の範囲を超えます。国際標準。 「記録」を参照してください。 ITU-T X.682 |情報オブジェクトセットの定義については ISO/IEC 8824-3、および Rec. の付録 A。 ITU-T X.682 | ISO/IEC 8824-3 では、instance-of タイプを制約するために情報オブジェクト セットを使用する方法について説明します。

G.2.19 オブジェクト識別子

バイナリ エンコーディングで OID ツリーのノードのコンパクトな数値識別が必要な場合は、 OBJECT IDENTIFIERを使用します。

G.2.20 OID 国際化リソース識別子

ほとんどの Unicode 文字をすべて含む名前を使用する必要があり、文字エンコーディングが許容される場合は、 OID-IRIを使用します。OID-IRI値は、「oid」IRI/URI スキームを使用して IRI または URI としても使用できます (「oid」を参照)。
ITU-T Rec X.660 | ISO/IEC 9834-1 付属書 F)。

G.2.21 相対オブジェクト識別子

G.2.21.1相対オブジェクト識別子タイプを使用して、オブジェクト識別子の値の前半部分が既知であるコンテキストで、よりコンパクトな形式でオブジェクト識別子の値を送信します。次の 3 つの状況が発生する可能性があります。

a) オブジェクト識別子の値の前半部分は、特定の仕様に対して固定されています (これは業界固有の標準であり、すべての OID は標準化団体に割り当てられた OID を基準としています。この場合、次を使用します。

```
RELATIVE-OID --相対オブジェクト識別子の値は次のとおりです。
        -- {isoidentified-organization set(22)} に相対
```

b) オブジェクト識別子の値の前半部分は仕様時点で既知の値であることが多いですが、
場合によっては、より一般的な値になる場合があります。この場合、次を使用します。

```
CHOICE {a
RELATIVE-OID --値は {1 3 22} に対する相対値です --、
 b OBJECT IDENTIFIER --任意のオブジェクト識別子の値 --}
```

c) オブジェクト識別子の値の前半部分は通信時までわかりませんが、送信する必要がある多くの値に共通であることが多く、仕様の時点で既知の値になることが非常に多くあります。この場合、(たとえば) を使用します。

```
順序
{oid-root オブジェクト識別子のデフォルト {1 3 22},
 reloids 相対 OID の配列--oid-root に相対--}
```

G.3 値の表記とプロパティの設定（TIME型と有用な時刻型）

この節では、時間タイプの値の表記例を示します。同じ値の表記が有用な時間型に使用されますが、それらの型に存在する抽象値の表記に限定されます。各例では、通常の人間による表記法で時刻抽象値を指定し、それを含むものがあれば便利な時刻型を使用し、それ以外の場合は TIME型を使用して、その値に対する値の割り当てを行います。次のコメントは、同様の抽象値をすべて含むTIME型のサブタイプを定義するために必要な設定を示しています。

G.3.1 日付

例

暦日 – 1985 年 4 月 12 日:

```
date1 DATE ::= "1985-04-12" -- Basic=Date Date=YMD Year=Basic
```

基準日 – 1985 年 4 月 12 日:

```
date2 TIME ::= "1985-102" --基本=日付 日付=YD 年=基本
```

曜日 – 1985 年 4 月 12 日金曜日:

```
date3 TIME ::= "1985-W15-5" --基本=日付 日付=YWD 年=基本
```

暦週 – 1985 年の第 15 週:

date4 TIME ::= "1985-W15" --基本=日付 日付=YW 年=基本

暦月 – 1985 年 4 月:

date5 TIME ::= "1985-04" -- Basic=Date Date=YM Year=Basic

暦年 – 1985:

date6 TIME ::= "1985" --基本=日付 日付=Y 年=基本

暦日 – 11985 年 4 月 12 日:

date7 TIME ::= "+11985-04-12" -- Basic=Date Date=YMD Year=L5

西暦0000年前2年目の4月12日：

date8 TIME ::= "-0002-04-12" -- Basic=Date Date=YMD Year=Negative

20世紀:

date9 TIME ::= "19C" --基本=日付 日付=C 年=基本

G.3.2 時刻

例

15 時間を経過した 27 分 46 秒:

time1 時刻 ::= "15:27:46"
-- Basic=Time Time=HMS Local-or-UTC=L

最も近い分まで:

time2 時刻 ::= "15:28"
-- Basic=Time Time=HM Local-or-UTC=L

カンマを使用した小数点以下の現地時刻 - 15 時間を経過した 27 分 35 秒半:

time3 時刻 ::= "15:27:35,5"
-- Basic=Time Time=HMSF1 Local-or-UTC=L

UTC – 23 時間経過 20 分 30 秒:

time4 時刻 ::= "23:20:30Z"
-- Basic=Time Time=HMS Local-or-UTC=Z

最も近い時間まで:

time5 時間 ::= "23Z"
-- Basic=Time Time=H Local-or-UTC=Z

現地時間と UTC との差 - ジュネーブ現地時間の 15 時間を 27 分 46 秒 (UTC より 1 時間進んでいます):

time6 時刻 ::= "15:27:46+01:00"
-- Basic=Time Time=HMS Local-or-UTC=LD

同じ抽象値の代替値表記:

time7 時刻 ::= "15:27:46+01"
-- Basic=Time Time=HMS Local-or-UTC=LD

ニューヨーク現地時間 15 時間経過 27 分 46 秒 (UTC から 5 時間遅れ):

time8 時刻 ::= "15:27:46-05:00"
-- Basic=Time Time=HMS Local-or-UTC=LD

G.3.3 日付と時刻

例

カレンダーの日付と現地時刻の組み合わせ:

日付-時刻1 日付-時刻 ::= "1985-04-12T10:15:30"
-- Basic=日付-時刻 Date=YMD Year=Basic Time=HMS
-- ローカルまたは UTC=L

カレンダーの日付と現地時刻と時差を組み合わせたもの。現地時間は1日の01:30です。
1985年4月。その場所の UTC 時間は1985 年 3 月31 日の 23:30 です。

```
日時2 TIME ::= "1985-04-01T01:30:00+02.00"
-- Basic=Date-Time Date=YMD Time=HMS Local-or-UTC=LD
```

日付と UTC の組み合わせ:

```
日時3 TIME ::= "1985-102T23:50:30Z"
-- Basic=日付-時刻 Date=YD Year=Basic Time=HMS Local-or-UTC=Z
```

曜日と現地時刻の組み合わせ:

```
日時4 TIME ::= "1985-W14-5T23:50:30"
-- Basic=日付-時刻 Date=YWD Year=Basic Time=HMS
-- ローカルまたは UTC=L
```

G.3.4 時間間隔

例

1985 年 4 月 12 日の 23 時過ぎ 20 分 50 秒から始まり、1985 年 6 月 25 日の 10 時過ぎ 30 分で終わる時間間隔:

```
間隔1 時間 ::= "1985-04-12T23:20:50/1985-06-25T10:30:00"
-- Basic=Interval Interval-type=SE SE-point=Date-Time
-- 日付=YMD 年=基本時刻=HMS ローカルまたは UTC=L
```

1985 年 4 月 12 日の現地時間の 12 時間過ぎ 30 分 (UTC 時間では 10 時間過ぎの 30 分) から始まり、同じ時差で 4 月 12 日の 13 時間過ぎ 30 分で終わる時間間隔 (これは必須ではありません)。 :

```
間隔 2 時間 ::= "1985-04-12T12:30:00+02:00/1985-04-12T13:30:00+02:00"
-- Basic=Interval Interval-type=SE SE-point=Date-Time
-- 日付=YMD 年=基本時刻=HMS ローカルまたは UTC=L
```

2 番目の時間差を省略した、同じ抽象値の代替値表記:

```
間隔 3 時間 ::= "1985-04-12T12:30:00+02:00/1985-04-12T13:30:00"
-- Basic=Interval Interval-type=SE SE-point=Date-Time
-- 日付=YMD 年=基本時刻=HMS ローカルまたは UTC=L
```

1985 年 4 月 12 日に始まり、1985 年 6 月 25 日に終わる時間間隔:

```
間隔4 時刻 ::= "1985-04-12/1985-06-25"
-- Basic=Interval Interval-type=SE SE-point=Date
-- 日付=YMD 年=基本
```

2 年、10 か月、15 日、10 時間、20 分、30 秒の時間間隔:

```
継続時間1 継続時間 ::= "P2Y10M15DT10H20M30S"
-- 基本=間隔 間隔タイプ=D
```

1 年 6 か月の間隔:

```
持続時間2 持続時間 ::= "P1Y6M"
-- 基本=間隔 間隔タイプ=D
```

72 時間の時間間隔:

```
持続時間3 持続時間 ::= "PT72H"
-- 基本=間隔 間隔タイプ=D
```

1985 年 4 月 12 日の 23 時 20 分に始まる、1 年 2 か月と 15 日と 12 時間の時間間隔:

```
間隔5 時間 ::= "1985-04-12T23:20:00/P1Y2M15DT12H"
-- Basic=Interval Interval-type=SD SE-point=Date-Time
-- 日付=YMD 年=基本時刻=HMS ローカルまたは UTC=L
```

1985 年 4 月 12 日の 23 時 20 分に終了する、1 年 2 か月と 15 日と 12 時間の時間間隔:

```
間隔6 時間 ::= "P1Y2M15DT12H/1985-04-12T23:20:00"
-- Basic=Interval Interval-type=DE SE-point=Date-Time
-- 日付=YMD 年=基本時刻=HMS ローカルまたは UTC=L
```

G.3.5 定期的な間隔

例

2 年、10 か月、15 日、10 時間、20 分、30 秒の間隔で 15 回繰り返します。

rec-int1 時間 ::= "R15/P2Y10M15DT10H20M30S"
-- Basic=Rec-Interval Recurrence=R2 Interval-type=D

2 年、15 日、10 時間、20 分、30 秒の時間間隔での無制限の繰り返し回数:

rec-int2 時間 ::= "R/P2Y15DT10H20M30S"
-- Basic=Rec-Interval Recurrence=Unlimited Interval-type=D

1 年 6 か月の間隔で 2 回繰り返されました。

rec-int3 時間 ::= "R2/P1Y6M"
-- Basic=Rec-Interval Recurrence=R1 Interval-type=D

1 年、2 か月、15 日、12 時間の時間間隔の無制限の発生回数。そのうちの最後の発生は 1985 年 4 月 12 日の 23 時間経過 20 分 50 秒に終了します。

rec-int4 時刻 ::= "R/P1Y2M15DT12H/1985-04-12T23:20:50"
-- Basic=Rec-Interval Recurrence=Unlimited Interval-type=DE
-- SE-point=Date-Time Date=YMD Year=Basic Time=HMS
-- ローカルまたは UTC=L

G.4 抽象構文の識別

G.4.1プロトコルは、単一の ASN.1 タイプ (通常は選択タイプ) のそれぞれの値にセマンティクスを関連付けることによって定義されるのが一般的です。 (この ASN.1 型は、非公式に「アプリケーションのトップレベル型」と呼ばれることもあります。) この抽象値のセットは、正式にはアプリケーションの抽象構文と呼ばれます。抽象構文は、ASN.1 型のオブジェクト識別子の抽象構文名を与えることで識別できます。

G.4.2抽象構文へのオブジェクト識別子の割り当ては、Rec. で定義されている組み込み情報オブジェクト クラスABSTRACT-SYNTAXを使用して行うことができます。 ITU-T X.681 | ISO/IEC 8824-2。これは、アプリケーションのトップレベルのタイプを明確に識別するのにも役立ちます。

G.4.3以下は、アプリケーション仕様に表示されるテキストの例です。

例

```
アプリケーション ASN1 の定義 ::=
始める
アプリケーション PDU をエクスポートします。

    アプリケーション-PDU ::= CHOICE { 接続-
        pdu ..... 、データ-pdu
        CHOICE {
            ……、
            ……

        }、
        ……

    }
    ……
終わり

抽象構文モジュールの定義 ::=
始める
アプリケーション ASN1 からアプリケーション PDU をインポートします。

    -- このアプリケーションは次の抽象構文を定義します。

    抽象構文 ABSTRACT-SYNTAX ::=
        { アプリケーション PDU の識別者
            アプリケーション抽象構文オブジェクト ID }

    application-abstract-syntax-object-id オブジェクト識別子 ::= {joint-iso-itu-t asn1(1)
        example(123) application-abstract-syntax(3) }

-- 対応するオブジェクト記述子は次のとおりです。
```

```
        アプリケーション抽象構文記述子 ObjectDescriptor ::=
                「アプリケーションの抽象構文の例」

    -- ASN.1 オブジェクト識別子とオブジェクト記述子の値:
                        -- エンコード ルール オブジェクト識別子
                        -- エンコード ルール オブジェクト記述子
    -- Rec のエンコード ルールに割り当てられます。 ITU-T X.690 | ISO/IEC 8825-1
    -- そして Rec. ITU-T X.691 | ISO/IEC 8825-2を転送として使用可能
    -- この転送構文と組み合わせた構文識別子。
    終わり
```

G.4.4インターワーキングを確保するために、標準では必須の転送構文 (通常、Rec. ITU-T X.690 | ISO/IEC 8825-1 または Rec. ITU-T のエンコード規則で定義されているものの 1 つ) を追加で識別する場合があります。 X.691 | ISO/IEC 8825-2 または Rec. ITU-T X.692 | ISO/IEC 8825-3)。

G.5 サブタイプ

G.5.1サブタイプを使用して、特定の状況で許可される既存のタイプの値を制限します。

例

```
AtomicNumber ::= INTEGER (1..104)

TouchToneString ::= IA5String (FROM
            ("0123456789" | "*" | "#")) (SIZE (1..63))

パラメータリスト ::= パラメータのサイズ (1..63) を設定

SmallPrime ::= INTEGER (2|3|5|7|11|13|17|19|23|29)
```

G.5.2拡張可能なサブタイプ制約を使用して、許可される値のセットが小さく、明確に定義されているが、増加が予想されるINTEGER型をモデル化します。

例

```
SmallPrime ::= INTEGER (2 | 3, ...)                    -- SmallPrime の最初のバージョン
```

見越して：

```
SmallPrime ::= INTEGER (2 | 3, ..., 5 | 7 | 11)
-- SmallPrime の 2 番目のバージョン
```

そしてさらに後:

```
SmallPrime ::= INTEGER (2 | 3, ..., 5 | 7 | 11 | 13 | 17 | 19)
-- SmallPrime の 3 番目のバージョン
```

注 – 特定のタイプの場合、一部のエンコード ルール (PER など) は、サブタイプ制約拡張のルート値 (つまり、 「...」の前に表示される値)に対しては高度に最適化されたエンコードを提供し、サブタイプ制約拡張の追加値に対してはあまり最適化されていないエンコードを提供します (一方、他の一部のエンコード ルール (BER など) では、サブタイプ制約はエンコードに影響を与えません。

G.5.3 2 つ以上の関連する型に重要な共通点がある場合は、それらの共通の親を型として明示的に定義し、個々の型にサブタイプを使用することを検討してください。このアプローチは、関係と共通性を明確にし、タイプが進化するにつれてこれが継続することを奨励します (ただし、強制するものではありません)。したがって、これらの型の値の処理に対する共通の実装アプローチの使用が容易になります。

例

```
エンベロープ ::= SET {
                タイプA タイプA、
                typeB TypeB オプション、
                タイプC タイプC オプション}
                -- 共通の親

ABEnvelope ::= エンベロープ (コンポーネント付き)
                {...、
                    typeB あり、typeC なし})
                -- ここで、typeB は常に出現する必要があり、typeC は出現してはならない

ACEnvelope ::= エンベロープ (コンポーネント付き)
                {...、
                    typeB なし、typeC あり})
```

```
                -- ここで、typeC は常に出現する必要があり、typeB は出現してはならない
```

後者の定義は次のように表現することもできます。

```
ABEnvelope ::= エンベロープ (WITH COMPONENTS {typeA, typeB})

ACEnvelope ::= エンベロープ (WITH COMPONENTS {typeA, typeC})
```

選択肢間の選択は、親タイプのコンポーネントの数、オプションのコンポーネントの数、個々のタイプ間の違いの程度、および考えられる進化戦略などの要因に基づいて行われます。

G.5.4サブタイプを使用して値を部分的に定義します。たとえば、テストが PDU の一部のコンポーネントのみに関係する適合性テストでテストされるプロトコル データ ユニットなどです。

例

与えられる:

```
PDU ::= セット
  {アルファ          整数、
   ベータ            IA5文字列オプション、
   ガンマデ          パラメータのシーケンス、
   ルタ              ブール値}
```

次に、ブール値が false で整数が負であることを必要とするテストを作成するには、次のように記述します。

```
TestPDU ::= PDU (コンポーネント付き)
    {...、
     デルタ (FALSE)、
     アルファ (MIN..<0)})
```

さらに、 IA5Stringベータ版が存在し、長さが 5 文字または 12 文字の場合は、次のように記述します。

```
さらにTestPDU ::= TestPDU (WITH COMPONENTS {... , beta (SIZE (5|12)) PRESENT } )
```

G.5.5汎用データ型がSEQUENCE OF として定義されている場合は、サブタイプを使用して汎用型の制限されたサブタイプを定義します。

例

```
テキストブロック ::= VisibleString のシーケンス

アドレス ::= テキストブロック (SIZE (1..6)) (WITH COMPONENT (SIZE (1..32)))
```

G.5.6汎用データ型がCHOICE として定義されている場合は、サブタイプを使用して汎用型の制限されたサブタイプを定義します。

例

```
Z ::= 選択肢 {
        腹              あ、
        筋              B、
        c               C、
        d               D、
        e               E
}

V ::= Z (WITH COMPONENTS { ..., a ABSENT, b ABSENT })
                                                    -- 'a' と 'b' は存在しない必要があります。 --
                                                    値には 'c'、'd'、または 'e' のいずれかが存在する可能性があります。

W ::= Z (WITH COMPONENTS { ..., --     プレゼント })                    -- 'a' のみが存在できます
                        (51.8.10.2 を参照)。

X ::= Z (WITH COMPONENTS { a PRESENT }) --                              -- 'a' のみが存在できます
                        (51.8.10.2 を参照)。

Y ::= Z (コンポーネント { a 不在、 b、 c } 付き)
                -- 'a'、'd'、および 'e' は存在しない必要があり
                ます。 -- 値には 'b' または 'c' が存在しても構いません。
```

注 – WとX は意味的に同一です。

G.5.7含まれるサブタイプを使用して、既存のサブタイプから新しいサブタイプを形成します。

例

```
月 ::= 列挙 {
                1月 2月 3          (1)、
                月 4月 5月         (2)、
                6月 7月            (3)、
                8月 9              (4)、
                月 10              (5)、
                月 11              (6)、
                月                 （7）、
                                   (8)、
                                   (9)、
                                   (10)、
                                   (11)、
                12月               (12) }

第 1 四半期 ::= 月 (1 月 | 2 月 | 3 月)

第 2 四半期 ::= 月 (4 月 | 5 月 | 6 月)

第 3 四半期 ::= 月 (7 月 | 8 月 | 9 月)

第 4 四半期 ::= 月 (10 月 | 11 月 | 12 月)

上半期 ::= 月 ( 第 1 四半期 | 第 2 四半期 )

下半期 ::= 月 ( 第 3 四半期 | 第 4 四半期 )
```

G.5.8時間タイプのサブタイプの例は 38.4 にあり、いくつかの便利な設定が G.3 のコメントに示されています。追加の例がコメント付きで続きます。サブタイプのすべての例は有用な時間型にも適用できますが、それらの型にすでに存在する抽象値のみが選択されることに注意してください。この表記法の主な用途は、有用な時間タイプのバリエーションを提供することです。

例

```
私の日付 ::= 時間
(設定「基本=日付年=基本日付=YD」)
-- 年と日を使用する日付型

私の日付1 ::= 時間
(設定「基本=日付年=基本日付=YD」)
("2000-001" .. < "2011-001")
-- に制限された年と日を使用する日付タイプ。
-- 西暦 2000 年 1 月 1 日から 2010 年 12 月 31 日までの期間。


私の日付2 ::= 時間
("2000-001" .. < "2011-001")
-- My-Date1 と同じ日付タイプですが、おそらくこれよりも小さいです。
-- 人間のユーザーには明らかです。プロパティ設定に依存します
-- 値の表記から推定されます (付録 K を参照)。

私の違法な日付 ::= TIME
("1500-01" .. < "2011-01")
-- 下限は予測日、上限は
-- は基本的な日付であるため、同じプロパティはありません。
-- そしてこれは違法です。

私の時刻-1 ::= TIME
(設定「基本=時間 時間=HMS ローカルまたは UTC=L」
 午前0時=開始")
-- これは TIME-OF-DAY と同じですが、終了時の午前 0 時です。
-- 日は除外され、午前 0 時のみが表されます
-- 値表記「00:00:00」による。

私の時刻-2 ::= TIME
(設定「基本=時間 時間=HMS ローカルまたは UTC=L」
 深夜=終わり」)
-- これは TIME-OF-DAY と同じですが、開始時の午前 0 時です。
-- 日は除外され、午前 0 時のみが表されます
-- 値表記「24:00:00」による。

私の時刻-3 ::= TIME
```

(設定 "Basic=Time Time=HMS Local-or-UTC=Z")
-- これは TIME-OF-DAY と同じですが、時刻は現地時間ではなく UTC で
す。

付録 H

ASN.1文字列に関するチュートリアル付録

(この附属書は、この勧告 | 国際規格の不可欠な部分を形成するものではありません。)

H.1 ASN.1 での文字列のサポート

H.1.1 ASN.1 には 4 つのグループの文字列サポートがあります。 4 つのグループは次のとおりです。

a)エスケープ シーケンスで使用される ISO International Register of Coded Character Sets (つまり、ISO/IEC 646 の構造に基づく) および関連する International Register of Coded Character Sets に基づく文字列タイプ。VisibleStringタイプによって提供されます。 、 IA5String、 TeletexString、 VideotexString、 GraphicString、およびGeneralString。

b) ISO/IEC 10646 に基づいた文字列型。ISO/IEC 10646 で定義されたサブセットを使用してUniversalString、 UTF8String 、またはBMPString 型をサブセット化するか、名前付き文字を使用することによって提供されます。

c) この推奨事項で指定されている文字の単純な小さなコレクションを提供する文字列タイプ
日付 |国際規格であり、特殊な使用を目的としています。これらは、 NumericString型とPrintableString型です。

d) CHARACTER STRING型を使用し、使用する文字セットのネゴシエーション (または使用される文字セットのアナウンス) を行う。これにより、エスケープ シーケンス、 ISO/IEC 7350、ISO/IEC 10646、およびプライベート コレクションで使用される ISO 国際コード化文字セット登録の文字およびエンコーディングを含む、オブジェクト識別子が割り当てられた文字およびエンコーディングのコレクションを実装で使用することが許可されます。文字とエンコーディングの種類 (プロファイルは、使用される文字セット (文字抽象構文) に要件または制限を課す場合があります)。

H.2 UniversalString、UTF8String、および BMPString 型

H.2.1 UniversalStringおよびUTF8String型は、ISO/IEC 10646 の任意の文字を運びます。 ISO/IEC 10646 の文字セットは、一般に意味のある適合性を要求するには大きすぎるため、通常は、次の標準コレクションの組み合わせにサブセット化する必要があります。 ISO/IEC 10646 の付録 A の文字。

H.2.2 BMPString型には、ISO/IEC 10646 の基本多言語プランの任意の文字が含まれます。基本多言語プレーンは、通常、ISO/IEC 10646 の付録 A にある標準の文字コレクションの組み合わせにサブセットされます。

H.2.3 ISO/IEC 10646 の付録 A で定義されたコレクションについては、組み込み ASN.1 モジュールASN1-CHARACTER-MODULEで定義された型参照があります(条項 42 を参照)。 「サブタイプ制約」メカニズムにより、既存のサブタイプの組み合わせであるUniversalStringの新しいサブタイプを定義できます。

H.2.4 ASN1-CHARACTER-MODULEで定義された型参照と、それに対応する ISO/IEC 10646 コレクション名の例は次のとおりです。

ベーシックラテン語	ベーシックラテン語
Latin-1サプリメント	ラテン-1 サプリメント
ラテン拡張-a	拡張ラテン語-A
ラテン拡張-b	拡張ラテン語-B
IpaExtensions	IPA 拡張機能
SpacingModifierLetters間隔修飾文字	
分音記号の結合分音記号の結合	

H.2.5 ISO/IEC 10646 は 3 つの「実装レベル」を規定しており、ISO/IEC 10646 のすべての使用において実装レベルを指定することを要求しています。

実装レベルは、文字レパートリー内の文字の結合をサポートする範囲に関係するため、ASN.1 の用語では、 UniversalStringおよびBMPStringの制限付き文字列タイプのサブセットを定義します。

実装レベル 1 では、文字の結合は許可されず、通常、ASN.1 文字列内の抽象文字と文字列の物理的表現内の印刷文字との間には 1 対 1 の対応関係があります。

実装レベル 2 では、特定の結合文字 (ISO/IEC 10646、付録 B にリストされている) が使用可能ですが、使用が禁止されている文字もあります。

実装レベル 3 では、結合文字の使用に制限はありません。

H.2.6 BMPStringまたはUniversalString は、次のようにサブタイプ表記を使用して、すべての制御関数を除外するように制限できます。

```
VanillaBMPString ::= BMPString (FROM (BMPString(SIZE(1)) EXCEPT)
                                    ({0,0,0,0}..{0,0,0,31} |
                                        {0,0,0,128}..{0,0,0,159})))
```

または同等の意味で:

```
        C0 ::= BMPString (FROM ({0,0,0,0} .. {0,0,0,31})) -- C0 制御関数
C1 ::= BMPString (FROM ({0,0,0,128} .. {0,0,0,159}))                -- C1制御機能
VanillaBMPString ::= BMPString (FROM (BMPString(SIZE(1)) EXCEPT (C0 | C1)))
```

H.3 ISO/IEC 10646 適合要件について

ASN.1 型定義でUniversalString、 BMPString 、またはUTF8String (またはこれらのサブタイプ)を使用するには、ISO/IEC 10646 の適合要件に対処する必要があります。

これらの適合要件は、そのような ASN.1 タイプを使用する標準 (X 氏) の実装者が、標準 X の実装のために採用された ISO/IEC 10646 のサブセットとそのレベルの記述を (プロトコル実装適合性宣言で) 提供することを要求します。 (文字の結合のサポート) の実装。

仕様でUniversalString、 UTF8String 、またはBMPStringの ASN.1 サブタイプを使用するには、実装がその ASN.1 サブタイプに含まれるすべての ISO/IEC 10646 文字をサポートし、したがって (少なくとも) それらの文字が存在する必要があります。実装のために採用されたサブセット内。また、そのようなすべての ASN.1 サブタイプに対して、指定されたレベルがサポートされることも要件です。

注 – ASN.1 仕様 (抽象構文のパラメータと例外仕様がない場合) は、送信できる文字の (最大) セットと受信時に処理する必要がある文字の (最小) セットの両方を決定します。 ISO/IEC 10646 の採用されたセットでは、このセットを超える文字は送信されず、このセット内のすべての文字が受信時にサポートされることが要求されます。したがって、採用されるセットは、ASN.1 仕様で許可されているすべての文字のセットである必要があります。抽象構文のパラメータが存在する場合については後述する。

H.4 ISO/IEC 10646 準拠に関する ASN.1 ユーザーへの推奨事項

ASN.1 のユーザーは、標準の要件が満たされる場合、採用される実装のサブセット (および必要な実装レベル) を形成する ISO/IEC 10646 文字のセットを明確にする必要があります。

これは、標準に必要なすべての文字を含むUniversalString、 UTF8String 、またはBMPStringの ASN.1 サブタイプを定義し、必要に応じてレベル 1またはレベル 2に制限することで簡単に実行できます。この型の便利な名前はISO-10646-Stringです。

例

```
ISO-10646-文字列 ::= BMPString
(FROM (レベル 2 交差点 (基本ラテン語 UNION ヘブライ語拡張 UNION ひらがな)))
        -- これは、文字の最小セットを定義するタイプです。
        -- この標準の実装に採用されたサブセット。の
        -- 実装レベルは少なくともレベル 2 である必要があります。
```

OSI 環境では、OSI プロトコル実装適合性ステートメントには、ISO/IEC 10646 の採用サブセットがISO-10646で定義された限定されたサブセット (およびレベル) であるという単純なステートメントが含まれます。
String、およびISO-10646-String (おそらくサブタイプ化) は、ISO/IEC 10646 文字列が含まれる規格全体で使用されます。

適合性宣言の例

ISO/IEC 10646 の採用されたサブセットは、実装レベル 2 のモジュール <your module name fits here> で定義されたASN.1 タイプISO-10646-Stringのすべての文字で構成される限定されたサブセットです。

プロトコルでの使用例

```
メッセージ ::= シーケンス {
最初のフィールド ISO-10646-String、 -- 採用されたすべての文字
                                    -- サブセットが表示される場合があります
2 番目のフィールド ISO-10646 文字列
        (FROM (latinSmallLetterA .. latinSmallLetterZ))、 -- 小文字
```

```
                                        -- ラテン文字のみ
    サードフィールド        ISO-10646-文字列
        (FROM (桁ゼロ .. 桁九))                    -- 数字のみ

    }
```

H.5 抽象構文のパラメータとしてサブセットを採用

ISO/IEC 10646 では、採用される実装のサブセットとレベルを明示的に定義することが求められています。 ASN.1 ユーザーが、定義されている標準の一部で ISO/IEC 10646 文字の範囲を制限したくない場合、これは、 (たとえば) ISO-10646-String をUniversalString やBMPStringのサブタイプとして定義することで表現できます。または、抽象構文のパラメーターとして残されるImplementorsSubsetで構成される (またはImplementorsSubset を含む) サブタイプ制約を持つUTF8String 。

ASN.1 のユーザーは、この場合、適合する送信者が、受信者の (実装に依存する) 採用されたサブセットまたはレベルの範囲外にあるため、受信者が処理できない文字を適合する受信者に送信する可能性があることを警告し、推奨されます。この場合、 ISO-10646-Stringの定義に例外処理仕様を含める必要があります。

例

```
ISO-10646-String {UniversalString : ImplementorsSubset,ImplementationLevel} ::=
        UniversalString (FROM((ImplementorsSubset UNION BasicLatin)
                INTERSECTION 実装レベル) !characterSet問題)
-- ISO/IEC 10646 の採用サブセットには「BasicLatin」が含まれますが、
-- で指定された追加の文字も含めることができます。
-- 「ImplementorsSubset」。抽象構文のパラメータです。
-- 「ImplementationLevel」、これは要約のパラメータです
-- 構文は実装レベルを定義します。適合する受信機は、
-- 採用されたサブセット以外の文字を受け取る準備ができており、
-- 実装レベル。この場合、で指定された例外処理は、
-- 「characterSet問題」の句<ここに句番号を追加>は
-- 呼び出されました。これは、適合するプログラムによっては決して呼び出すことができないことに注意してください。
-- 通信のインスタンスで実際に使用される文字の場合は受信者
-- 「BasicLatin」に制限されます。

My-Level2-String ::= ISO-10646-String { { HebrewExtended UNION ひらがな }, Level2 }
```

H.6 CHARACTER STRING 型

H.6.1 CHARACTER STRINGタイプは、文字セットとエンコード方法の選択において完全な柔軟性を提供します。

注 – 単一の接続でエンドツーエンドのデータ転送 (中継なし) が提供され、OSI プロトコルが使用されている場合、使用する文字セットとそのエンコーディングのネゴシエーションは、OSI の定義の一部として実行できます。文字抽象構文のプレゼンテーション コンテキスト。それ以外の場合、抽象文字構文と転送文字構文 (文字レパートリーとエンコーディング) は、オブジェクト識別子の値のペアによって通知されます。

H.6.2形式的に言えば、文字抽象構文は、取り得る値にいくつかの制限を加えた通常の抽象構文です (値はすべて文字列であり、実際、文字の集合から形成されたすべての文字列です)。したがって、文字抽象および転送構文に対するオブジェクト識別子の値の割り当ては、通常の方法で実行されます。

H.6.3 CHARACTER STRINGのエンコーディングは、使用されている文字レパートリー (つまり、文字セットとエンコーディング) の抽象および転送構文を通知します。 OSI 環境では、これら両方の構文のネゴシエーションが可能です。

H.6.4文字抽象構文 (および対応する文字転送構文) は、多くの ITU-T 勧告および国際標準で定義されており、追加の文字抽象構文 (および/または文字転送構文) は、割り当てが可能な組織によって定義できます。オブジェクト識別子。

H.6.5 ISO/IEC 10646 では、サブセット (BASIC LATIN、BASIC SYMBOLS など) に対して定義された文字のコレクションごとに、文字のコレクション全体に対して定義された文字抽象構文 (および割り当てられたオブジェクト識別子) があります。そして、定義された文字のコレクションのあらゆる可能な組み合わせに対して。 ISO/IEC 10646 には、さまざまなオプション (特に 16 ビットと 32 ビット) を識別するために定義された 2 つの文字転送構文もあります。

附属書I

型拡張の ASN.1 モデルに関するチュートリアルの付録

(この附属書は、この勧告 | 国際規格の不可欠な部分を形成するものではありません。)

I.1 概要

I.1.1 ASN.1 タイプは、拡張機能の追加と呼ばれる一連の拡張機能によって、拡張ルートタイプから時間の経過とともに進化する場合があります。

I.1.2特定の実装で利用可能な ASN.1 タイプは、拡張ルート タイプである場合もあれば、拡張ルート タイプに 1 つ以上の拡張機能を追加したものである場合もあります。拡張機能追加を含む各 ASN.1 タイプには、以前に定義されたすべての拡張機能追加も含まれます。

I.1.3このシリーズの ASN.1 型定義は拡張子関連であると言われており(「拡張子関連」のより正確な定義については 3.8.38 を参照)、拡張子関連型をエンコードするにはエンコード ルールが必要です。これにより、2 つのシステムが拡張関連の 2 つの異なるタイプを使用している場合、2 つのシステム間の送信では、2 つのシステムに共通する拡張関連タイプの部分の情報コンテンツが正常に転送されます。また、同じ転送構文が使用される場合、両方のシステムに共通ではない部分を区切って、後続の送信時に (おそらくサードパーティに) 再送信できることも必要です。

注 – 送信者は、一連の拡張機能の追加の前または後のタイプを使用している可能性があります。

I.1.4ルート型に漸進的に追加して得られる型の系列を拡張系列と呼びます。エンコード ルールで拡張関連の型 (回線により多くのビットが必要になる場合がある) の送信を適切に行うためには、そのような型 (拡張ルートの型を含む) に構文的にフラグを付ける必要があります。フラグは省略記号(...) であり、拡張マーカーと呼ばれます。

例

```
拡張ルートの種類                    1回目の延長                2回目の延長                3回目の延長

A ::= シーケンス { A ::= シーケンス { A ::= シーケンス { A ::= シーケンス {
整数、整数、整数、整数、
  …                …、              …、                …、
}                  b ブール値、b ブール値、b ブール値、
                   c 整数 c 整数、c 整数、
                 } d シーケンス { d シーケンス {
                                    e 整数、                  e 整数、
                                      …、                      …、
                                      …、                  g ブール型オプション、
                                    f IA5文字列 } }        h BMPString、
                                                              …、
                                                            f IA5文字列
                                                          }
                                                        }
```

ーは、 シーケンス、セット、選択タイプのすべての拡張追加は、拡張マーカーのペアの間に挿入されます。 I.1.5の単一の拡張マーカ(拡張ルート タイプで) タイプの最後の項目として表示される場合に許可されます。この場合、一致する拡張マーカーはタイプの閉じ括弧の直前に存在すると想定されます。このような場合、すべての拡張機能の追加は型の最後に挿入されます。

I.1.6拡張マーカーを持つ型は、拡張マーカーを持たない型の中にネストすることも、拡張ルートの型内にネストすることも、拡張追加型にネストすることもできます。このような場合、一連の拡張子は独立して処理され、拡張子マーカーを持つネストされた型は、ネストされている型に影響を与えません。特定のコンストラクト内に出現できる拡張挿入ポイントは1 つだけ(単一の拡張マーカーが使用されている場合はタイプの末尾、または 1 組の拡張マーカーが使用されている場合は 2 番目の拡張マーカーの直前) です。

I.1.7拡張シリーズの新しい拡張追加は、単一の拡張追加グループに関して定義されます。
("[[" "]]"内にネストされた 1 つ以上の型) 、または拡張子の挿入ポイントに追加された単一の型。次の例では、最初の拡張は、型 A の値にbとcの両方が存在するか、両方とも存在しない必要がある拡張追加グループを定義します。 2 番目の拡張は、単一のコンポーネント型d を定義します。これは、型 A の値に存在しなくても構いません。タイプA。3 番目の拡張は、タイプAの値にhが存在する必要がある拡張追加グループを定義します。
新しく追加された拡張機能追加グループが値に存在する場合はいつでも。

例

```
拡張ルートの種類          1回目の延長 2回目の延長 3回目の延長
```

```
A ::= シーケンス { A ::= シーケンス { A ::= シーケンス { A ::= シーケンス {
整数、整数、整数、整数、
  ...                ...、                ...、                ...、
}                    [[ [[ b ブール値、b                    [[
                      ブール値、b ブール値、
                      c 整数 c 整数 c 整数
                    ]] ]]、 ]]、
                  } d シーケンス { d シーケンス {
                                    e 整数、e 整数、
                                      ...、                ...、
                                      ...、                [[
                                    f IA5String g ブール型オプション、
                                   h BMPString
                                  } ]]、
                                                          ...、
                                                        f IA5文字列
                                                      }
                                                    }
```

I.1.8バージョン番号をバージョン括弧に追加することもできますが、それがモジュール内のすべての括弧に存在し、モジュール内のすべての拡張機能がバージョン括弧内にある場合に限ります。バージョン番号を使用することをお勧めします。数字とバージョン括弧を省略できるのは、歴史的な理由によるものです。 (この勧告 | 国際標準の以前のバージョンでは、バージョン括弧とバージョン番号は許可されていませんでした。) (I.3 も参照)

I.1.9通常、拡張機能は時間をかけて追加されますが、基礎となる ASN.1 モデルと仕様には時間がかかりません。拡張機能の追加によって一方が他方から「成長」できる場合、2 つのタイプは拡張機能に関連しています。つまり、一方には他方のすべてのコンポーネントが含まれています。逆方向に「成長」する必要があるタイプもあるかもしれません (これは可能性は低いですが)。時間の経過とともに、型は多くの拡張機能の追加から始まり、徐々に削除される可能性さえあります。 ASN.1 とそのエンコード ルールが考慮するのは、型仕様のペアが拡張機能に関連しているかどうかだけです。そうである場合、すべてのASN.1 エンコードルールによってユーザー間のインターワーキングが保証されます。

I.1.10型から始めて、後でそれを拡張する必要がある場合に、以前のバージョンの実装と相互作用するかどうかを決定します。その場合は、拡張子マーカーを含めます。その後、以前のシステムで定義された拡張値の処理を使用して、後の拡張機能を型に追加できます。ただし、以前は拡張マーカーがなかった型に拡張マーカーを追加すると (または拡張マーカーを削除すると)、相互作用が妨げられる可能性があることに注意することが重要です。

注 – ECN を使用すると、バージョン 1 では拡張子マーカーがなかった場所にバージョン 2 で拡張子を追加し、バージョン 1 と 2 の間の相互作用を維持できる可能性があります。

I.1.11表 I.1 は、ASN.1 拡張シリーズの拡張ルート タイプを形成できる ASN.1 タイプと、そのタイプに許可される単一の拡張追加の性質を示します (もちろん、複数の拡張追加を追加することもできます)。連続して作成するか、拡張グループとして一緒に作成します)。

表 I.1 – 拡張機能の追加

拡張ルートの種類	拡張子追加の性質
列挙された	「AdditionalEnumeration」の末尾に、すでに追加されている列挙値よりも大きい列挙値を含む、さらに 1 つの列挙体を追加します。
シーケンスとセット	単一のタイプまたは拡張子追加グループを「ExtensionAdditionList」の末尾に追加します。 拡張機能の追加である (拡張機能の追加グループに含まれていない)「ComponentType」には、 OPTIONALまたはDEFAULTのマークを付ける必要はありませんが、そうされることがよくあります。
選択	「ExtensionAdditionAlternativesList」の末尾に 1 つの「NamedType」を追加します。
制約の表記法	「ElementSetSpecs」表記に 1 つの「AdditionalElementSetSpec」を追加。

I.2 バージョン番号の意味

ECN バージョン番号は、BER または PER エンコーディングでは使用されません。 ECN エンコーディングでの使用 (存在する場合) が決定されます。仕様によるI.2.1 。

I.2.2 バージョン番号は、個々のタイプではなく、完全な PDU をデコードする手段に関連する場合に最も役立ちます。タイプが複数のプロトコルのコンポーネントとして使用され、さまざまな完全な PDU に貢献する場合、そのタイプに追加するには、通常、そのタイプが貢献するすべての PDU のバージョン番号をインクリメントする必要があります。

I.2.3展開されたシステム間の相互作用を提供するために使用される場合、展開されたシステムが特定のバージョン番号を持つすべての拡張機能追加グループの構文とセマンティクスを認識できるように、バージョン番号を拡張機能追加グループで使用する必要があります (バージョン番号がどこにあるかは関係ありません)。プロトコル内に表示されます)、および以前のバージョン番号を持つすべての拡張機能追加グループの。 ECN 指定子は通常、この原則に従ってバージョン番号が (ECN が適用される型のすべての部分に) 割り当てられていると想定します。

I.3 エンコードルールの要件

I.3.1抽象構文は、拡張可能な型である単一の ASN.1 型の値として定義できます。これには、extension-additions の追加または削除によって取得できるすべての値が含まれます。このような抽象構文を拡張関連抽象構文と呼びます。

I.3.2拡張関連の抽象構文用の整形式のエンコード規則のセットは、I.3.3 から I.3.5 に記載されている追加要件を満たします。

注 – すべての ASN.1 エンコード規則はこれらの要件を満たしています。

I.3.3抽象値を転送用のエンコードに変換する手順、および受信したエンコードを抽象値に変換する手順の定義は、送信者と受信者が同一ではないが拡張された抽象構文を使用している可能性を認識しなければならない。 -関連している。

I.3.4この場合、エンコード規則は、送信者の型仕様が受信者のものよりも拡張シリーズの中で早い場合、送信者の値は、受信者が次のことを判断できる方法で転送されることを保証するものとする。拡張機能の追加は存在しません。

I.3.5エンコード規則は、送信者の型指定が受信者の拡張シリーズよりも後の場合、その型の値の受信者への転送が可能であることを保証しなければならない。

I.4 (拡張可能な) 制約の組み合わせ

I.4.1 モデル

I.4.1.1制約を適用するための基本的な ASN.1 モデルは単純です。型は抽象値のセットであり、それに適用される制約はそれらの抽象値のサブセットを選択します。制約のない型が拡張可能でない場合、適用された制約が拡張可能であると定義されている場合に限り、結果の型は拡張可能であると定義されます。

I.4.1.2この単純なケースでも、明確にするべき特徴が 1 つあります。それは、拡張機能の追加が決してできない場合でも、型は形式的に拡張可能である可能性があるということです。考慮する：

A ::= INTEGER (MIN .. MAX, ... , 1..10)

この付録の多くの例と同様、これは誰も書かないものですが、ASN.1 標準は単純かつ一般的なままになっており、したがってこの例は合法的な ASN.1 であるため、ツール ベンダーはコードを作成する必要があります。
この例では、A は形式的には拡張可能なINTEGER であり、ルートに整数値の全範囲が含まれます。

I.4.1.3複雑さは、次の 3 つの主な原因から生じます。

– すでに拡張可能な制約が適用されている型 (シリアル) への制約の適用。
制約の適用 - I.4.2 を参照)。

– UNION 、 INTERSECTION 、 EXCEPT （集合算術演算)を使用した拡張可能な制約の組み合わせ
– I.4.3を参照)。

– 型参照が拡張可能な型への参照を解除するときの、制約の集合算術における型参照 (包含されるサブタイプ) の使用 (おそらく実際の拡張機能の追加を伴う – I.4.4 を参照)。

I.4.2 制約の連続適用

I.4.2.1制約の連続適用は、型が (型参照への代入で) 制約され、その後その型参照がさらに制約が適用されて使用されるときに発生します。

I.4.2.2また、それほど一般的ではありませんが、型に複数の制約が連続的に直接適用されている場合にも発生することがあります。この後者の形式は、(説明を簡単にするために) この付録の例の多くで使用されますが、型参照が 2 つ (またはそれ以上) の制約をリンクする場合は、実際の仕様でシリアル アプリケーションが通常発生する形式です。

I.4.2.3制約の連続適用には 2 つの重要なポイントがあります。

– 制約付きタイプが拡張可能 (およびおそらく拡張可能) の場合、その後さらに制約が連続して適用されると、「拡張可能」フラグとすべての拡張機能の追加は破棄されます。制約付き型 (および拡張機能の追加) の拡張性は、適用される最後の制約のみに依存します。

さらに制約されている型 (親型) のルート内の値のみを参照できます。
結果の型のルートまたは拡張機能の追加に含まれる値は、親型のルートにある値のみにすることができます。

– 制約の逐次適用は、(複雑な場合には) 拡張性が関係しない場合でも、算術積集合と同じではありません。第一に、 MINとMAXが解釈される環境、第二に、2 番目の制約で参照できる抽象値は、シリアル アプリケーションでは、2 つの制約が共通の親からの値の交差として指定されている状況とは大きく異なります。

注 –整数型の制約で20..28などの範囲を使用することは、 20と28の両方が親型 (のルート) にある場合 (その場合に限り) 正当ですが、この範囲指定で参照される値は親 (のルート) にあるもののみです。したがって、親が値24と25 を除外するようにすでに制約されている場合、範囲20..28は20 ～ 23と26 ～ 28のみを参照します。

ここではいくつかの例を示します。

```
A1 ::= 整数 (1..32, ..., 33..128)
        -- A1 は拡張可能で、1 ～ 128、1 ～ 32 の値が含まれます。
        -- ルートに、拡張子として 33 ～ 128 を追加します。

B1 ::= A1 (1..128)
        -- または同等の

B1 ::= INTEGER (1..32, ... , 33..128) (1..128)
        -- 128 が親にないため、これらは不正です。
        -- さらに制限されると拡張機能の追加が失われます

B2 ::= A1 (1..16)
        -- これは合法です。 B2 には拡張性がなく、1 ～ 16 が含まれます。

A2 ::= INTEGER (1..32) (MIN .. 63)
        -- MIN は 1、63 は不正です

A3 ::= INTEGER ( (1..32) INTERSECTION (MIN..63) )
        -- これは合法です。 MIN はマイナス無限大で、A3 には 1 ～ 32 が含まれます
```

I.4.3 集合演算の使用

I.4.3.1結果は大部分が直観的であり、交差、和集合、および集合の差(例外) に関する通常の数学的規則に従います。特に、交差と結合は両方とも可換です。つまり、次のようになります。

(<値の一部のセット 1> INTERSECTION <値の一部のセット 2>)

と同じです
(<値のセット 2> INTERSECTION <値のセット 1>)

UNIONの場合も同様です。

I.4.3.2可換性は、どのような値のセットが拡張可能であっても、またどのような拡張追加が存在しても、真です。

I.4.3.3交差により拡張加算値が発生できない場合、誤解が生じる可能性があります。
これは INTEGER (MIN..MAX, ...) の場合と似ています。

I.4.3.4例:

```
A ::= INTEGER (1..256, ... , B) (1..256)
-- A には、値に関係なく、常に値 1..256 (のみ) が含まれます。
-- B には次の内容が含まれます
```

I.4.3.5さらに制約されると親は拡張性と拡張加算を失い、含まれるサブタイプは拡張性と拡張加算を失うが、集合算術で直接指定された値のセットは拡張性も拡張加算も失われないことを覚えておくことも重要です。 。

I.4.3.6集合算術によって生成される値の集合の拡張性に関する規則は、50.4 および 50.5 に明確に記載されており、集合算術が実際の拡張加算を可能にするかどうかには依存しません。

I.4.3.7完全を期すためにここでルールを要約します。E は「拡張可能」フラグが設定された値のセットを示し、 Nは形式的には拡張不可能な値のセットを示します。各セットのルートの値はR で示され、拡張の追加 (存在する場合) はX で示され、結果の内容がそれぞれの場合に示されます。

注 1 – この付録の目的と説明の簡略化のため、値のセットが拡張できない場合、そのすべての値をルート値として説明します。

注 2 – シリアルに適用される制約で使用される値の結果セットのルートが空の場合、これは不正な仕様です。

注 3 – 以下では冗長になるのを避けるため、より正確な「拡張機能の追加」の代わりに「拡張機能」を使用します。

I.4.3.8ルールは次のとおりです。

N1 交差点 N2 => N
ルート: R1 交差点 R2
N1 交差点 E2 => E
ルート: R1 INTERSECTION R2、拡張子: R1 INTERSECTION X2
E1 交差点 E2 => E
ルート: R1 INTERSECTION R2、拡張子: ((R1 UNION X1)
交差点
(R2ユニオンX2))
を除外する
(R1交差点R2)
N1 ユニオン N2 => N
ルート: R1 UNION R2
N1 ユニオン E2 => E
ルート: R1 UNION R2、拡張子: X2
E1 ユニオン E2 => E
ルート: R1 UNION R2、拡張子: (R1 UNION X1 UNION R2 UNION X2)
を除外する
(R1ユニオンR2)
N1 N2 を除く => N
ルート: R2 を除く R1
N1 E2 を除く => N
ルート: R2 を除く R1
E1 N2 を除く => E
ルート: R2 を除く R1、拡張子: (R2 を除く X1)
を除外する
(R1は除く、R2)
E1 E2 を除く => E
ルート: R1 EXCEPT R2、拡張子: (X1 EXCEPT (R2 UNION X2))
を除外する
(R1は除く、R2)
N1 ... N2 => E ルー
ト: R1、拡張子: R2 を除く R1 E1 ... N2 => E ルート:
R1、拡張子: X1
UNION R2 例外 R1

N1 ... E2 => E ル
ート: R1、拡張: R2 UNION X2 EXCEPT R1

E1 ... E2 => E
ルート: R1、拡張子: X1 UNION R2 UNION E2
を除外する
R1

注 – 拡張可能な値のセットに対する集合算術の結果に実際の拡張加算がないか、実際の拡張加算がまったくない場合でも (拡張可能な入力にどのような拡張加算が追加されても)、結果は依然として次のように正式に定義されます。上記の結果Eについては拡張可能です。

I.4.4 含まれるサブタイプ表記の使用

含まれるサブタイプは拡張可能である場合とそうでない場合がありますが、セット算術で使用される場合は常に拡張不可能として扱われ、その拡張機能の追加はすべて破棄されます。

付録 J

TIME タイプに関するチュートリアルの付録

(この附属書は、この勧告 | 国際規格の不可欠な部分を形成するものではありません。)

J.1 時刻と日付の ASN.1 タイプのコレクション

J.1.1歴史的に、ASN.1 は独自の時刻タイプであるUTCTime を「UsefulType」として定義していました。これは、それがVisibleStringタイプの内容の指定に基づいていたためです。その後、 4 桁の年を許可するGeneralizedTimeが追加され、VisibleString の内容を指定するために、ISO 8601 の最初の (1988 年) バージョンの前身である一連の標準を参照して定義されました。 (もう 1 つの「UsefulType」は、 GraphicStringの内容を指定することによって定義され、 ObjectDescriptorでした。)従来、いわゆる「UsefulType」は、型参照名に大文字と小文字を混合して使用してきました。大文字のみを使用する組み込み ASN.1 タイプ。

ただし、有用な型には独自のUNIVERSALクラス タグがあり、エンコーディング ルール仕様で独立して参照できます。

J.1.2これらの型(UTCTime、 GeneralizedTime 、 ObjectDescriptor)は間違いなく便利ですが、「UsefulType」という用語によって他の型から分離されているのは、単にこれらの型が他の観点から定義されているためです。
文字列 - 型)、およびその型参照名に大文字と小文字を混在して使用することは、歴史的な偶然であると認識されることが増えています。

J.1.3 ISO 8601 の 2004 バージョンをサポートするための時間タイプの導入により、主要な時間タイプ(TIME)が必要であるが、一般的に役立つ多くの時間タイプ(DATE、 TIME-OF-DAY、 DATE-TIMEおよびDURATION) は、基本的なTIME型 (ASN.1 サブタイプ表記を使用)のサブセットとして定義される必要がありました。これらは新しい予約語であるため、下位互換性の問題を最小限に抑えるために、これらを「有用な時刻タイプ」と呼び、すべて大文字の名前を付けることが決定されました。これらはすべて、 TIMEタイプのタグとは異なる個別のUNIVERSALクラス タグを持ち(最適化された BER エンコーディングを有効にするため)、すべてプロダクション "BuiltinType" (17.2 を参照) の下にリストされます。

J.2 ISO 8601 の主要な概念

J.2.1 ISO 8601 は、時刻の識別とその文字表現に関する決定的なリファレンスを提供します。これは、時間関連の概念と、ASN.1 値表記および基本エンコーディング ルール (BER) で使用される実際の表現の両方の観点から、 ASN.1 TIME 型の仕様の基礎を形成します。

J.2.2 ISO 8601 は、1582 年に導入されたグレゴリオ暦と、共通 (非うるう) の定義に通常の規則を使用して、グレゴリオ暦を 1582 年から時間を遡って順次拡張する、いわゆる予期的グレゴリオ暦に完全に基づいています。) 年と閏年。一般に、予期的グレゴリオ暦を使用して指定された日付から、ユリウス暦を使用して西暦または紀元前の日付を決定する簡単な方法はありません。特に、西暦 1 年は予期的グレゴリオ暦 1 年とほぼ (ただし正確ではありません) 一致します。そして、紀元前 1 年 (前年) は、予兆期のグレゴリオ暦 0 年とほぼ (正確ではありませんが) 一致しています。

J.2.3 ISO 8601 の主要な定義と概念には、時間軸の複数の時間スケールの概念が含まれています。各時間スケールは、時間軸上の順序付けられたマークのセットで構成されます。各マークは時刻 (瞬間)を表します。

J.2.4 ISO 8601 では 3 つの主要な時間スケールが定義されています。

J.2.4.1 1 つ目は、カレンダー日付時間スケールと呼ばれます。これには、暦年、暦月、および暦月内の日の序数に対応するマークが付いています (日には、月に応じて 01 から 28、29、30、または 31 の番号が付けられます)。

J.2.4.2 2 番目は順序日付時間スケールと呼ばれます。これには、暦年と、その暦年の中の日の序数に対応するマークが付いています (日には、1 月 1 日の 001 から、年に応じて 365 または 366 までの番号が付けられます)。

J.2.4.3 3 番目は、週の日付タイム スケールと呼ばれます。これには、暦年、その暦年の週の序数、その週内の日の序数 (1 日が月曜日) に対応するマークが付いています。週には 01 から 52 または 53 (年に応じて) の番号が付けられ、第 01 週は 1 月 4 日を含む週として定義され、前年の最終週はその前の週として定義されます (そのため、年によっては53 週が含まれます)。

J.2.5各時間スケールの日のマークの間には、時、分、秒のマークがあります。ただし、時間軸は時間の瞬間の連続体であり、3 つの時間スケールすべてにも、時間軸上のあらゆる場所に密集したマークが含まれています。

注 – これを別の言い方で表現すると、任意の 2 つのマークの間には、無限に多くの他のマークが存在し、それぞれが秒の小数部分を任意の精度で識別するということです。

J.2.6カレンダーの日付時間スケールのバリエーションは、秒が表現されないが、各分時間の間に、その分の小数部分を任意の精度で表す無限に多くの他のマークが存在する時間スケールです。時間の小数部分についても同様です。

注 – ISO 8601 には、1 日の小数またはそれ以上の時間単位を使用して時刻を指定するという概念はありません。ただし、期間の指定には、年、月、週、または日の小数を使用できます。

J.2.7有理数 1/60 には終端の小数表現がないため、秒を使用する時間スケール上の時間の中には、いかなる有限表現においても、分数を使用する時間スケール上の時間として表現できないものがあります。

J.2.8同様に、どの年が閏年であるかが分からなければ、ある時間スケールのどの日のマークが別の時間スケールの日のマークに対応するかを識別することはできません。同様の問題は、開始点と終了点、または開始点と継続時間 (秒単位など)、または継続時間と終了点に基づくスケールを使用して時間間隔を識別するうるう秒でも発生します。

J.2.9 ISO 8601 は、さまざまな精度でのマークの識別の概念も認めています。したがって、任意のタイム スケールで同時に異なるマークが存在する可能性があり、1 つは (たとえば) 3.100 秒として指定し、もう 1 つは 3.1 秒として指定します。

注 – 時間タイプ(UTCTimeおよびGeneralizedTime)に関する以前の ASN.1 作業では、異なる精度で表現された同じ時刻に対して個別の抽象値を持つ問題は解決されていませんでした。 TIME型の場合、精度は異なるが同時に配置された時間軸上のマークを個別の抽象値としてしっかりと認識します。したがって、3.100 で表される抽象値は、3.1 で表される抽象値とは異なり、異なるアプリケーション セマンティクスを持つ可能性があります。

J.2.10使用される精度の管理、および ISO 8601 のその他の側面の管理は、ISO 8601 に「相互合意による」と記載されています。一般に、ISO 8601 が相互合意を必要とする領域を特定する場合、アプリケーション設計者が ASN.1 タイプ定義で前提とされる相互合意を指定するための表記が ASN.1 で提供されます。これは、各時間抽象値に関連付けられた時間プロパティを使用して、 TIME型の複数の無限の抽象値のサブセットを選択することによって行われます。

J.2.11 ISO 8601 は時差の概念を認めています。これは、特定のワールド タイム ゾーンの現地時刻と UTC の差です。異なるワールドタイム ゾーンの時差に関する合意や記録を行う国際的な権威はありません。これは地方行政の問題ですが、英国航海暦局は、世界各地に現在割り当てられている時差に関する信頼できる記録を維持しようとしています。 2005 年と同様に、時差はさまざまな地方自治体によって -12 から +14 の範囲で定義されています。将来の変更に備えて、ASN.1 は –15 ～ +16 (のみ) の範囲の時差をサポートします。

J.3 TIME型の抽象値

J.3.1 各時間スケール上の各マークは、それぞれの精度で、 TIMEタイプの個別の抽象値として識別されます。
は、すべての ASN.1 エンコーディング ルールで独自の ASN.1 値表記と独自のエンコーディングを備えています。

J.3.2 ISO 8601 は主に時刻の識別に関係していますが、日付のみ、時刻のみ、および日付と時刻の識別を区別しています。これらのさまざまな識別により、 TIME型に個別の抽象値も生成されます。

J.3.3時刻の識別では、現地時刻または UTC、またはその両方を使用できます。繰り返しになりますが、使用されている時間スケールが異なるため、これらの異なる識別は、異なる関連性のない抽象値を生成します (また、通常は異なるアプリケーション セマンティクスを保持します)。

J.3.4 ISO 8601 のもう 1 つの特徴は、開始点と終了点 (さまざまな時間スケールのいずれかを使用して識別できる)、期間、または開始点または終了点のいずれかを使用して時間間隔を識別することです。
繰り返しになりますが、これらは時間を表す抽象値とは異なる 4 つの主要な抽象値セットを提供しますが、時間間隔の開始点と終了点の仕様に使用される抽象値に応じて、時間間隔の主要なセットの多くのサブセットを備えています。 、または期間の指定に使用される時間コンポーネント。

注 – ASN.1 では、時間間隔の開始点と終了点に異なる時間スケールを使用することはできません。これは仕様を簡素化するためのものであり、アプリケーション設計者にとって問題になることはありません。

J.3.5 最後に、ISO 8601 には、繰り返しの時間間隔を指定するという概念があります。定期的な時間間隔のマッピング
時間間隔や時間を表す値とは異なる抽象値。

J.4 時間抽象値の時間プロパティ

J.4.1共通の時間プロパティを持つ時間抽象値のセットを識別することが可能です。一部の時間プロパティ (時間、時間間隔、または繰り返しの時間間隔など) は、すべての時間抽象値に適用されます。時間間隔が開始点と継続時間として表現されるか、または継続時間と終了点によって表現されるか (たとえば) などの他の時間プロパティは、時間間隔である時間抽象値にのみ適用されます。同様に、時間抽象値が現地時刻、UTC、またはその両方を表すかどうかを示す時間プロパティは、時刻に関連するコンポーネントが少なくとも 1 つある時間抽象値にのみ適用されます。

J.4.2条項 38.2 および表 6 は、時間抽象値に関連付けることができる時間プロパティの完全なセットと、それらの時間プロパティごとに可能な設定を指定します。時間プロパティの設定をサブタイプ表記で使用して、 TIMEタイプのサブセットを選択できます。そのすべての抽象値は、特定の時間プロパティに対して同じ設定を持ちます。

注 – 「抽象値」という用語での「値」の使用との混同を避けるために、「時間プロパティの値」ではなく「時間プロパティの設定」という用語が使用されています。

J.4.3時間抽象値上の一部の時間プロパティの存在は、上で概説したように、他の時間プロパティの設定に依存します。

J.4.4サブタイプ表記では、 TIME型の抽象値は、時間プロパティと設定のペアのリストを使用して指定されます。指定できる時間プロパティと設定の組み合わせには制限があります (51.10.6 を参照) が、プロパティと設定のペアの順序は重要ではありません (ただし、J.4.5 を参照)。 TIME型の抽象値は、それに適用できるリストされたすべてのプロパティに対して指定された設定がある場合にのみ、サブタイプに含まれます。いつものように、型参照に割り当てられた抽象値の結果セットが空である場合、これは不正な ASN.1 仕様です (ただし、空のセットは集合演算では禁止されていません)。

J.4.5人間の読者にわかりやすくし、エラーを避けるために、プロパティと設定のペアの指定順序は、主要なプロパティ ("Basic=Date-Time" など) からその他のプロパティに進むことが推奨されますが、必須ではありません。詳細なプロパティ (「TIME=HMS」など)。これは通常、表 6 の順序で時間プロパティと設定のペアを指定することを意味します。この規則は、この推奨事項のすべての例で使用されます。国際規格 (38.4、G.3、および G.5.8 を参照)。

J.4.6間隔の指定と値の範囲を使用したサブタイプの場合、 TIME型の抽象値に順序関係があることが重要です。一般に、時間を表す抽象値の間には順序関係があります。
(時間軸上の位置に基づいて) すべてが同じ時間プロパティ設定を持っている場合に限ります。同様に、2 つの時間抽象値の間の秒数は、一般に、それらの値が同じプロパティ設定を持つ場合にのみ決定できます。これは、一部のサブタイプ表記と間隔指定で使用される時間は、同じプロパティ設定を持つ必要があることを意味します。順序関係は、期間が同じ精度を持ち、単一の時間コンポーネントのみが異なる場合にのみ、期間に対して定義されます (51.11 を参照)。

J.5 値の表記

J.5.1 抽象値の値表記 (およびエンコーディング) は、それに関連付けられた時間プロパティとその値に依存します。
設定。値の表記は 38.3 で規定されています。

J.5.2 ISO 8601 は一般に、時間スケールでマークを識別するために、基本形式と拡張形式と呼ばれる 2 つの別個の (文字ベースの) 表現を指定します。

J.5.3一般に、基本形式は単純な数字の文字列であり、ISO 8601 によって暗示される抽象値の一般的に有用なサブセット内で明確な表現を提供するために必要な場合にのみ数字以外の区切り文字 (小数点区切り文字など) が使用されます。この形式はASN.1 値表記では使用されません。

注 – たとえば、基本形式では、文字列2020 は2020年と午後 8 時 20 分の両方を表します。

J.5.4拡張形式には、人間のユーザーにとって表現を読みやすくするために設計された追加の非数字区切り文字が含まれており、一般に (ただし 1 つの例外を除き、以下の注 1 を参照) ISO 8601 によって暗示されるすべての抽象値に対して明確です。この形式は、ISO 8601 によってプレーン テキストでの使用が推奨されています。

注 1 – 例外は、世紀を表す 4 桁の日付表現であり、年を表す 4 桁と混同される可能性があります。 ASN.1 値表記では、あいまいさを解決するために、2 桁表記を含むすべての世紀表記に対してCが世紀表記に追加されます。

注 2 – たとえば、拡張形式では、 2020年は2020で表されますが、午後 8 時 20 分の時刻は20:20 で表されます。

J.5.5拡張形式 (何世紀にもわたって大文字のCが追加された) を使用すると、表現されている抽象値のプロパティ設定を値の表記から決定できるようになり、その表記は、次のことだけを知って、抽象値を明確に識別できます。その型はTIME型です。

J.5.6基本形式では、表現のあいまいさを解決するために、表現される抽象値のプロパティ設定の一部についての知識が必要ですが、ASN.1 では使用されません。

J.5.7基本的な ASN.1 値の表記法と XML 値の表記法 (38.3 で規定)、および Rec. で規定された XML エンコーディング規則ITU-T X.693 | ISO/IEC 8825-4、ISO 8601 拡張フォーマットを使用します。 Rec. で指定されている基本的なエンコーディング ルール。 ITU-T X.690 | ISO/IEC 8825-1 は ISO 8601 拡張形式を使用します (ただし、期間を表す P、時刻のコロン、日付のハイフンなど、一部の指定子と区切り文字が削除されています)。異なる ASN.1 タグが有用な型に割り当てられているため、BER は、有用な時間型のエンコードにおけるあいまいさを解決するために必要なプロパティ設定を識別できます。 Rec. で指定されているパックされたエンコーディング ルール。 ITU-T X.691 | ISO/IEC 8825-2 は、ISO 8601 とは無関係 (および ISO 8601 の範囲外) のバイナリ エンコーディングを使用します。PER エンコーディングは、日付、時刻、期間を非常にコンパクトに表現します (通常、日付は 17 ビット、時刻は 15 ビット) 、日付/時刻の場合は 32 ビット (4 オクテット)、期間の場合は 16 ビット未満であることがよくあります)。

J.6 ASN.1 サブタイプ表記の使用

J.6.1 このタイプでは、6 つの形式のサブタイプ表記 (制限された場合の内部サブタイプを追加 – J.6.8 を参照) が許可されます。
(第 51 条および表 12 を参照)。

注 – TIME型のサブタイプ表記の例と役立つ時間型は、38.4、G.3、および G.5.8 にあります。

J.6.2プロパティ設定のサブタイプ表記により、リストされた 1 つ以上の時間プロパティの特定の設定を持つすべての抽象値を選択できます。これは、アプリケーション用に追加のカスタマイズされた時間タイプを作成する通常の手段であり、J.7 で詳しく説明されています。

J.6.3 単一値のサブタイプは許可されていますが、一般的に役立つとは期待されていません。

ことが 含まれるサブタイプは許可されており、アプリケーション設計者によるカスタマイズされたJ.6.4時間タイプの仕様で一般的に使用される期待されます。

J.6.5期間範囲のサブタイプ (順序付けされた期間のペアを含む) を適用できます。これらは、期間として指定された時間間隔である抽象値のみを親タイプから選択し、期間を指定された範囲に制限します (51.11 を参照)。

J.6.6時点範囲サブタイプ (順序付けられた時間のペアを含む) を適用できます。これらは、時点範囲の両端と同じプロパティ設定を持つ時刻 (同じプロパティ設定を持つ必要がある) を持つ抽象値のみを親型から選択し、時刻が指定された範囲内に収まるように制限します。

注 – このサブタイプ制約は、時間の値の範囲を制限するものであり、時間間隔である単一の時間抽象値を識別するための値表記の直接使用とはまったく別のものです。

J.6.7反復範囲サブタイプ (整数の順序付きペアを含む) を適用できます。これらは、親タイプから反復的な時間間隔である抽象値のみを選択し、反復回数の指定に使用できる桁数を制限します。

J.6.8 内部サブタイプは、時間型がすでに期間に制限されている場合に適用できます (通常は、
DURATION有効時間タイプ)。これにより、期間指定の形式に制限を設けることができます。

J.6.9他の形式のサブタイプ制約は許可されません。

J.7 プロパティ設定のサブタイプ表記

J.7.1特定の時間プロパティに対して同じ設定を持つ時間抽象値は、時間型の自然なサブセットを形成します。プロパティ設定のサブタイプ表記を使用すると、1 つ以上の時間プロパティの設定をリストすることで抽象値を選択できます。抽象値は、すべてのサブタイプに対して指定された設定がある場合にのみ、結果のサブタイプに含まれます。
それに適用できるリストされたプロパティ。

例: 次の表記法を使用して、4 桁の年、年間の週、および日を使用して指定される、日付のみであるすべての抽象値を含む時間サブタイプを定義できます。

私の時間 ::= TIME (設定 "基本=日付 日付=YWD 年=基本")

サブタイプ表記のより広範な例は、38.4、G.3、および G.5.8 に示されています。

J.7.2 ASN.1 集合算術 (または単に複数の制約の適用) を通常の方法で使用して、時間サブタイプの組み合わせ (INTERSECTION、 UNION 、およびEXCEPT を使用)を定義し、特定のアプリケーションでの使用に適した型を生成できます。 。

J.7.3時間プロパティ、その名前、可能な設定、およびこの設定を持つ抽象値は、表 6 に指定されています。

J.7.4少数の有用な時間サブタイプは、プロパティ設定サブタイプ表記法 (38.4 を参照) を使用して指定され、わかりやすい名前が付けられています。これらの便利なサブタイプ(DATE、 TIME-OF-DAY、 DATE-TIME、およびDURATION)は、多くのアプリケーションに十分であることが期待されます。一般ユーティリティの定義された時間タイプのより広範なセットは、付録 B の ASN.1 定義時間タイプ モジュールで指定されています。これらのタイプは、直接インポートして使用することも、アプリケーション固有の時間タイプを定義するために使用することもできます。これらは、ISO 8601 の全機能をサポートします。さらに、必要に応じて、設計者は、プロパティ設定サブタイプ表記を使用して、 TIMEタイプまたは有用な時間タイプまたは定義済みの時間タイプのサブタイプとして追加のタイプを定義できます。これらのタイプは、ASN.1 集合演算を使用してさらに組み合わせることができます。

注 – BER での効率的なエンコードを可能にするために、有用な時間型には、 TIME型とは異なるASN.1 UNIVERSALクラス タグが与えられています。これらは、サブタイプではなく独立したタイプとしてみなされる必要がありますが、親タイプが TIME の場合は、含まれるサブタイプ制約で使用することもできます。

附属書K

TIME型の値表記の解析

(この附属書は、この勧告 | 国際規格の不可欠な部分を形成するものではありません。)

K.1 一般的な

K.1.1この勧告の本文 |国際標準は、特定の時間プロパティを持つ抽象値の値表記を指定します。

K.1.2この値表記のすべてのインスタンスは、時間型の単一の抽象値とそのプロパティを明確に識別します。

K.1.3この有益な付録では、値表記のインスタンスによって表される抽象値の時間プロパティ設定を決定するための 1 つの可能なアルゴリズムについて説明します。代替の (そしておそらくより優れた) アルゴリズムが多数あり、この付録は単にそのようなアルゴリズムが存在することを示すために提供されています。

注 – このアルゴリズムをランダムな文字列に適用すると、その文字列は抽象値のみを表すことができることが識別されます。指定された一連の時間プロパティ設定。次に、表記を受け入れて抽象値を識別する前に、文字列の構文がこれらのプロパティ設定を使用して抽象値に必要な構文に準拠していることを確認する必要があります。

K.1.4 2 つの抽象値が同じプロパティ設定を持つ場合、それらの値の表記は、以下の例外を除いて、表記に存在する数字の値が異なるだけです。

a) 期間の抽象値には、時間単位に応じていくつかの異なる表現があります。
使用されているかどうか、および小数が使用されているかどうか。

b) カンマまたはピリオドのいずれかを小数点区切り文字として使用できます。

c) 時間の整数である時差成分は、時間のみで表すことも、時と分で表すこともできます。

d) UTC を表す間隔の終了点は、時間が異なる場合には時差成分を省略することができます。
差はスタート地点のタイム差と同じです。

e) 時間差成分にプラスまたはマイナスがあるだけで異なる抽象値にもかかわらず、同じ時間特性を持ちます。

K.1.5 2 つの文字列が、文字列内に存在する数字の実際の値のみが異なる場合、それらは同じプロパティ設定を持ちますが、例外として、0000 から 1581 の範囲の年の日付の使用は、日付がプロパティ設定を持つことを意味します。 「年=基本」ではなく「年=プロレプティック」です(世紀表記も同様)。

K.2 完全な文字列を分析する

K.2.1文字列がラテン大文字 R で始まる場合、その文字列のプロパティ設定は「Basic=Rec-Interval」になります。 「 R 」の後にはいくつかの繰り返し (無制限の場合は空の文字列) が続き、その後に SOLIDUS (「/」) が続きます。
文字列の「R」の後と「/」の前の部分が空の場合、プロパティ設定は「Recurrence=Unlimited」になります。それ以外の場合、文字列の「R」の後および「/」の前の部分の桁数が1、2、3 などの場合、そのプロパティ設定は「Recurrence=R1」、 「Recurrence=R2」、 「Recurrence=R3」など。文字列の残りの部分は、追加のプロパティ設定を決定するために、間隔を含む文字列として分析できます (K.3 を参照)。

K.2.2それ以外の場合、文字列に固相線 (「/」) が含まれる場合、その文字列には「Basic=Interval」というプロパティ設定があり、追加のプロパティ設定を決定するために間隔を含む文字列として分析できます (K.3 を参照)。 。

K.2.3それ以外の場合、文字列がラテン大文字 P で始まる場合は、 「Basic=Interval」および「Interval-type=D」の設定があり、プロパティの分析が完了します。

K.2.4それ以外の場合、文字列にラテン大文字の T が含まれている場合、プロパティ設定 「Basic=Date-Time」があり、文字列の「T」より前の部分は日付を含む文字列として分析できます(K.4 を参照)、 「T」に続く部分は時間を含む文字列として分析され (K.7 を参照)、さらなるプロパティ設定を決定できます。

K.2.5それ以外の場合、文字列がラテン大文字 C で終わる場合、その文字列には「Basic=Date」および「Date=C」というプロパティ設定があり、世紀を含む文字列として分析できます (K.6 を参照)。さらにプロパティ設定を決定します。

K.2.6それ以外の場合、文字列にコロン (「:」) が含まれているか、文字数が 4 文字未満であるか、文字列が 4 文字以上で 3 文字目がプラス (「+」) またはハイフンマイナス (「-」) である場合)、プロパティ設定 「Basic=Time」を持ち、時刻を含む文字列 (K.7 を参照) として分析して、さらなるプロパティ設定を決定できます。

K.2.7それ以外の場合は、 「Basic=Date」というプロパティ設定があり、日付を含む文字列として分析して (K.4 を参照)、さらなるプロパティ設定を決定できます。

K.3 インターバルを含む文字列の解析

K.3.1 文字列がラテン大文字 P で始まり、SOLIDUS (「/」) を含まない場合は、
プロパティ設定 「Interval-type=D」、プロパティの分析が完了しました。

K.3.2文字列に SOLIDUS が含まれる場合、文字列の SOLIDUS の前の部分または SOLIDUS の後の部分がラテン大文字 P で始まるか、どちらも "P" で始まりません (両方ではありません)。もし：

a) SOLIDUS の前の部分は 「P」で始まり、プロパティ設定 「Interval-type=DE」を持ち、この K.3 の後続のサブ条項で指定されているように SOLIDUS の後の部分を分析することによってさらなるプロパティを持つ可能性があります。

b) SOLIDUS の後の部分は 「P」で始まり、プロパティ設定 「Interval-type=SD」を持ち、この K.3 の後続のサブ条項で指定されているように SOLIDUS より前の部分を分析することによってさらなるプロパティを持つ可能性があります。

c) どちらの部分も 「P」で始まっていない場合は、 「Interval-type=SE」というプロパティ設定があり、この K.3 の後続の節で指定されているように SOLIDUS より前の部分を分析することによって、さらなるプロパティを持つことができます。

注 – ASN.1 には、終了ポイントが開始ポイントと同じ時間プロパティ設定を持つという要件があります (ISO 8601 にはありません)。これは、プロパティ設定を決定する際に文字列の終点部分を分析する必要がないことを意味します。ただし、始点の時差と同じであれば、時差成分を省略した終点の表現も認められることに注意してください。エンドポイントの抽象値を決定する際には、これを考慮する必要があります。

K.3.3その部分にラテン大文字の T が含まれる場合、プロパティ設定 「SE-point=Date-Time」があり、 「T」より前の部分は日付を含む文字列として解析できます (K.4 を参照)、 「T」に続く部分は時間を含む文字列として分析され (K.7 を参照)、さらなるプロパティ設定を決定できます。

K.3.4その部分がラテン大文字 C で終わる場合、プロパティ設定 「SE-point=Date Date=C」があり、世紀を含む文字列として分析できます (K.6 を参照)。さらなるプロパティを決定します。設定。

K.3.5それ以外の場合、その部分にコロン (「:」) が含まれている場合、その部分には 「SE-point=Time」というプロパティ設定があり、時間を含む文字列として分析できます (K.7 を参照)。さらにプロパティの設定を行います。

K.3.6それ以外の場合、プロパティ設定 「SE-point=Date」があり、日付を含む文字列として分析され (K.4 を参照)、さらなるプロパティ設定を決定できます。

K.4 日付を含む文字列の解析

K.4.1 文字列がラテン大文字 C で終わる場合、プロパティ設定 「Date=C」と残りの部分が設定されます。
文字列は、世紀を含む文字列として分析できます (K.6 を参照)。

K.4.2文字列がハイフンマイナス (「-」) で始まる場合、これはこの節 K.4 の残りの部分の分析では無視される必要があります。

注 – この場合、ハイフンは区切り文字ではなくマイナス記号を表します。

K.4.3それ以外の場合、文字列には 0、1、または 2 つのハイフン-マイナス (「-」) 文字が含まれ、最後の 2 つのケースでは、ラテン大文字 W が含まれる場合と含まれない場合があります。

K.4.4 文字列にラテン大文字の W が含まれていない場合、次のような場合:

a) 文字列にはハイフン-マイナス文字が含まれておらず、プロパティ設定 「Date=Y」があり、年を含む文字列 (K.5 を参照) として分析して、さらなるプロパティ設定を決定できます。

b) 文字列には 1 つのハイフンマイナス文字が含まれており、プロパティ設定 「Date=YM」があります。ハイフンマイナスの後に月の 2 桁が続き、ハイフンマイナスの前の部分は年を含む文字列として分析され (K.5 を参照)、さらなるプロパティ設定を決定できます。

c) 文字列には 2 つのハイフン-マイナス文字が含まれており、プロパティ設定 「Date=YMD」があります。最初のハイフンマイナスの後に月の 2 桁が続き、2 番目のハイフンマイナスの後に日の 2 桁が続きます。

HYPHEN-MINUS、および最初の HYPHEN-MINUS より前の文字列の部分は、年を含む文字列 (K.5 を参照) として分析して、さらなるプロパティ設定を決定できます。

K.4.5 文字列にラテン大文字の W が含まれている場合、次のようになります。

a) 文字列には 1 つのハイフンマイナス文字が含まれており、プロパティ設定「Date=YW」があります。ハイフンマイナスの後には週を表す 2 桁が続き、ハイフンマイナスの前の部分は年を含む文字列として分析され (K.5 を参照)、さらなるプロパティ設定を決定できます。

b) 文字列に 2 つのハイフン-マイナス文字が含まれている場合、プロパティ設定は「Date=YWD」になります。最初のハイフンマイナスの後には週を表す 2 桁が続き、2 番目のハイフンマイナスの後には日を表す 1 桁が続き、最初のハイフンマイナスより前の文字列の部分は、年を含む文字列として分析できます (K を参照) .5) を使用して、さらなるプロパティ設定を決定します。

K.5 年を含む文字列の分析

K.5.1 文字列がハイフンマイナス (「-」) 文字で始まり、5 文字である場合、その文字列のプロパティ設定は「年=負」となり、プロパティの分析が完了します。

K.5.2それ以外の場合、文字列が 4 文字を超え、ハイフンマイナス (「-」) で始まらない場合
文字の場合、それぞれ 5、6、7 などに等しい文字数に対して、プロパティ設定「年 = L5」、 「年 = L6 」、「年 = L7 」などがあります。

K.5.3それ以外の場合、文字列が 4 文字を超え、ハイフンマイナス (「-」) 文字で始まる場合、その文字列のプロパティ設定は「年 = L5」、「年 = L6」、 「年 = L7」、など、それぞれ 6、7、8 などの文字数に相当します。

K.5.4それ以外の場合、4 桁の文字列の値が 1582 より小さい場合、プロパティ設定「年 = プロレプティック」があり、プロパティの分析が完了します。

K.5.5それ以外の場合は、プロパティ設定「年=基本」があり、プロパティの分析が完了します。

K.6 世紀を含む文字列の分析

K.6.1文字列がハイフンマイナス (「-」) 文字で始まり 3 文字である場合、その文字列には「年 = 負」というプロパティ設定があり、プロパティの分析が完了します。

K.6.2それ以外の場合、文字列が 3 文字を超え、ハイフンマイナス (「-」) 文字で始まらない場合、その文字列のプロパティ設定は「年 = L5」、 「年 = L6」、 「年 = L7」になります。 "など、それぞれ 3、4、5 などの文字数に相当します。

K.6.3それ以外の場合、文字列が 3 文字以上でハイフンマイナス (「-」) 文字で始まる場合、その文字列のプロパティ設定は「年 = L5」、「年 = L6」、 「年 = L7」、など、それぞれ 4、5、6 などの文字数に相当します。

K.6.4それ以外の場合、2 桁の文字列の値が 15 未満の場合、プロパティ設定「年 = プロレプティック」があり、プロパティの分析が完了します。

K.6.5それ以外の場合は、プロパティ設定「年=基本」があり、プロパティの分析が完了します。

K.7 時刻を含む文字列の解析

K.7.1文字列がラテン大文字 Z で終わる場合、プロパティ設定「Local-or-UTC=Z」があり、文字列の「Z」より前の部分は単純な時刻を含む文字列として分析できます。 (K.8 を参照)、さらなるプロパティ設定を決定します。

K.7.2それ以外の場合、文字列にプラス ("+") またはマイナス ("-") が含まれる場合、その文字列には次のプロパティがあります。
「Local-or-UTC=LD」 、およびプラスまたはマイナスの前の文字列の部分は、さらなるプロパティ設定を決定するための単純な時間 (K.8 を参照) として分析できます。

注 – プロパティ設定を決定するために、プラスまたはマイナスに続く文字列部分 (時間差) の分析は必要ありません。

K.7.3それ以外の場合、プロパティ「Local-or-UTC=L」があり、さらなるプロパティ設定を決定するための単純な時間 (K.8 を参照) として分析できます。

K.8 単純な時刻を含む文字列の分析

K.8.1文字列には 0、1、または 2 つのコロン (:) が含まれ、ピリオド (".") またはカンマ (",") である小数点記号が含まれる場合があります。

K.8.2文字列に小数点記号が含まれていない場合は、次のとおりです。

a) 文字列にはコロンが含まれておらず、プロパティ設定「Time=H」があり、分析が完了しています。
プロパティ;

b) 文字列にはコロンが 1 つ含まれており、プロパティ設定「Time=HM」があり、分析が完了します。
プロパティ;

c) 文字列には 2 つのコロンが含まれており、プロパティ設定「Time=HMS」があり、分析が完了します。
プロパティ。

K.8.3 文字列に小数点記号が含まれている場合は、次のようになります。

a) 文字列にはコロンが含まれておらず、小数点以下の桁数が1、2 、 3 など、それぞれ特性の分析が完了します。

b) 文字列にはコロンが 1 つ含まれており、小数点以下の桁数が 1、2、3 の場合、 「Time=HMF1」、 「Time=HMF2」、 「Time= HMF3」などのプロパティ設定が含まれます。など、それぞれ特性の分析を完了します。

c) 文字列に 2 つのコロンが含まれており、小数点以下の桁数が 1、2、3 の場合、プロパティ設定は「Time=HMSF1」、 「Time=HMSF2」、 「Time=HMSF3」などとなります。 、など、それぞれ、プロパティの分析が完了します。

付録L

ASN.1 表記の概要(この付録は、この勧告 | 国際標準の不可欠な部分を形成しません。)

次の語彙項目が第 12 項で定義されています。

タイプリファレンス

識別子

値参照

モジュール参照

コメント

空の

番号

実数

bstring

xmlbstring

hstring

xmlhstring

cstring

xmlcstring

単純な文字列

文字列

xmltstring

psname

「::=」

「...」

「...」

「[[」

「]]」

エンコーディングリファレンス

integerUnicodeLabel 非整

数UnicodeLabel

「</」

"/>"

"真実"

拡張された真

"間違い"

拡張-偽

「な〜ん」

「INF」

xmlasn1タイプ名

「{」

"}"

「<」

>"

「、」

「。」

「/」

"("

")"

「[」

「]」

「-」（ハイフンマイナス）

「:」

「=」

""" （クォーテーションマーク）

「'」（アポストロフィー）

" " （空間）

";"

「@」

"|"

「！」

「`」

不在

抽象構文

全て

応用

自動

始める

少し

BMP文字列

ブール値

による

キャラクター

選択

クラス

成分

コンポーネント

拘束された

含む

日付

日付時刻

デフォルト

定義

間隔

埋め込み

エンコード済み

エンコーディング制御

終わり

列挙された

を除外する

明示的

輸出

拡張性

外部の

間違い

から

一般化された時間

一般文字列

グラフィック文字列

IA5文字列

識別子

暗黙

暗黙的

輸入

含まれるもの

実例

説明書

整数

交差点

ISO646文字列

マックス

最小

マイナス無限大

数字ではありません

ヌル

数値文字列

物体

オブジェクト記述子

オクテット

の

OID-IRI

オプション

パターン

PDV

プラスインフィニティ

現在

印刷可能な文字列

プライベート

本物

相対OID

相対OID-IRI

順序

セット

設定

サイズ

弦

構文

T61ストリング

タグ

テレテックスストリング

時間

時刻

真実

タイプ識別子

連合

個性的

ユニバーサル

ユニバーサル文字列

UTC時間

UTF8文字列

ビデオテックス文字列

可視文字列

と

この勧告では次の作品が使用されています。国際標準。上記の語彙項目を終端記号として使用します。

```
モジュール定義 ::=
        モジュール識別子
        定義
        エンコーディング参照デフォルト
        タグデフォルト
        拡張子デフォルト
        「::=」
        始める
        モジュール本体
        エンコーディング制御セクション
        終わり

モジュール識別子 ::= モジュ
        ール参照
        決定的な識別

確定的な識別 ::=
        決定的なOID
|決定版OIDとIRI |空の


決定的なOID ::=
        "{" DefinitiveObjIdComponentList "}"

決定的なOIDとIRI ::=
                決定的なOID
        IRI値

DefinitiveObjIdComponentList ::=
        DefinitiveObjIdComponent
| DefinitiveObjIdComponent DefinitiveObjIdComponentList

DefinitiveObjIdComponent ::=
        お名前フォーム
|決定的な番号フォーム
|決定的な名前と番号フォーム

DefinitiveNumberForm ::= 数値

DefinitiveNameAndNumberForm ::= 識別子"(" DefinitiveNumberForm ")"

エンコーディング参照デフォルト ::=
                エンコーディングリファレンス_空の


タグデフォルト::=
        明示的なタグ
|暗黙のタグ
|自動タグ|空の


拡張子のデフォルト::=
        拡張性の暗示
|空の

モジュール本体 ::=
        エクスポート インポート AssignmentList |空
の

エクスポート ::=
        エクスポートシンボルエクスポート ";"
|すべての「;」をエクスポートし
ます|空の

エクスポートされたシンボル ::=
        シンボルリスト
```

```
|空の

インポート ::=
        IMPORTSシンボルインポートされた ";" |
空の

インポートされたシンボル ::=
        モジュールリストからのシンボル |
空の

モジュールリストからのシンボル ::=
        モジュールからのシンボル
|モジュールからのシンボルリストモジュールからのシンボル

モジュールからのシンボル ::=
        GlobalModuleReference 選択オプションからのSymbolList

選択オプション ::= 空

| 「後継者」とともに
| 「子孫」とともに

GlobalModuleReference ::= モジュ
        ール参照 AssignedIdentifier

割り当てられた識別子 ::=
        オブジェクト識別子の値
|定義された値 |空の


シンボルリスト ::=
        シンボル
| SymbolList "," シンボル

記号::=
        参照
|パラメータ化された参照

参照::=
        タイプ参照値参照
        オブジェクトクラス
        参照オブジェクト参照オブ
        ジェクトセット参照


割り当てリスト ::=
        割り当て
|       割り当てリストの割り当て

割り当て ::=
        タイプの割り当て
|値の割り当て
| XML値の割り当て
| ValueSetTypeAssignment
|オブジェクトクラスの割り当て
|オブジェクトの割り当て
|オブジェクトセットの割り当て
|パラメータ化された割り当て

定義されたタイプ::=
        外部タイプ参照 |タイプリファレ
ンス
|パラメータ化されたタイプ
|パラメータ化された値セットタイプ

定義された値 ::=
        外部値参照
|       値参照
```

```
        |パラメータ化された値

        NonParameterizedTypeName ::=
            外部タイプ参照 |タイプリファレン
        ス | xmlasn1タイプ名

        外部タイプ参照 ::= モジュール参
            照"."タイプリファレン
            ス

        外部値参照 ::= モジュール参照"."値参
            照


    絶対参照 ::=
                「@「モジュール識別子」。」

                    アイテムスペック

        アイテムスペッ
            ク ::= タイプリファ
        レンス |アイテムID 「.」コンポーネントID

        アイテムID ::= アイテムスペック

        コンポーネント ID ::= 識
            別子 |番号 |
        「*」

        TypeAssignment ::=
            typereference
            "::="
            タイプ

        値の割り当て ::= 値参照

                「::=」と
                入力します
                価値

        XMLValueAssignment ::= 値参
            照"::="


            XMLTypedValue

        XMLTypedValue ::=
            "<" & NonParameterizedTypeName ">"
            XML値
            "</" & NonParameterizedTypeName ">"
        | "<" & NonParameterizedTypeName "/>"

        ValueSetTypeAssignment ::= タイ
            プリファレンス
            「::=」と
            入力します
            値セット

        値セット ::= "{"要素セット仕様"}"

        タイプ ::= ビルトインタイプ |参照型 |制約されたタイプ

        ビルトインタイプ::=
            ビット文字列タイプ
        |ブール型
```

|文字列タイプ
|選択タイプ
|日付の種類
|日付時刻タイプ
|期間タイプ
|埋め込みPDVタイプ
|列挙型
|外部タイプ
|インスタンスのタイプ |
整数型
| IRIタイプ
|ヌルタイプ |
オブジェクトクラスフィールドタイプ |
オブジェクト識別子タイプ |
オクテット文字列型
|リアルタイプ |
相対IRIタイプ
|相対OIDタイプ
|シーケンスタイプ
|タイプのシーケンス
|セットタイプ
|タイプのセット
|接頭辞付きタイプ
|時間の種類
|時刻タイプ

参照型 ::=
定義されたタイプ
|便利なタイプ |
選択タイプ
| TypeFromObject
|オブジェクトからの値セット

NamedType ::= 識別子のタイプ

値::=
ビルトインバリュー
|参照値 |
オブジェクトクラスフィールド値

XML値 ::=
XML組み込み値
| XMLオブジェクトクラスフィールド値

ビルトイン値 ::=
ビット文字列値
|ブール値
|文字列値
|選択値
|埋め込みPDV値
|列挙値
|外部値
|値のインスタンス |
整数値
| IRI値
|ヌル値 |
オブジェクト識別子値 |
オクテット文字列値
|リアルバリュー |
相対IRI値
|相対OID値
|シーケンス値
|値の順序
|値の設定
|値のセット

```
|       接頭辞付きの値
|時間値

XMLBuiltinValue ::=
        XMLBitStringValue
| XMLブール値
| XML文字列値
| XMLChoiceValue
| XMLEmbeddedPDVValue
| XML列挙値
| XML外部値
| XMLインスタンスの値
| XML整数値
| XMLIRI値
| XMLNull値
| XMLObjectIdentifierValue
| XMLOctetStringValue
| XMLRealValue
| XMLRelativeIRI値
| XML相対OID値
| XMLシーケンス値
| XML値のシーケンス
| XMLSetValue
| XMLSetOfValue
| XMLPrefixedValue
| XMLTimeValue

参照値 ::=
        定義された値
|オブジェクトからの値

NamedValue ::= 識別子の値

XMLNamedValue ::= "<" & 識別子 ">" XMLValue "</" & 識別子 ">"

ブール型::=BOOLEAN

ブール値::= TRUE |間違い

XMLブール値::=
        空の要素ブール値
|テキストブール値

EmptyElementBoolean ::= "<"
        & "true" "/>" | 「<」&
「false」「/>」

テキストブール値                ::=
        拡張された真
|       拡張-偽

整数型::=
        整数
| INTEGER "{" NamedNumberList "}"

名前付き番号リスト ::=
        名前付き番号
| NamedNumberList "," NamedNumber

NamedNumber ::= 識別
        子"(" SignedNumber ")"識別子"(" DefinedValue
|       ")"

SignedNumber ::= 数値

|       "-" 番号
```

```
整数値::=
        SignedNumber 識
|       別子

XMLIntegerValue ::=
        XMLSignedNumber
|空の要素整数 |
        テキスト整数

XMLSignedNumber ::= 数値

|       "-" & 番号

空の要素整数 ::= | 「<」&識別子「/
>」

TextInteger ::= 識別
        子

列挙型::=
        ENUMERATED "{"列挙型"}"

列挙型::=
        ルート列挙
| RootEnumeration "," "..." RootEnumeration "," "..."
ExceptionSpec "," AdditionalEnumeration

RootEnumeration ::= 列挙型

AdditionalEnumeration ::= 列挙型

列挙 ::= 列挙項目 | EnumerationItem "," 列挙型

EnumerationItem ::= 識別子 |名前付き番号

EnumeratedValue ::= 識別子

XMLEnumeratedValue ::=
        空の要素列挙型
|テキスト列挙型

EmptyElementEnumulated ::= "<" & 識別子 "/>"

TextEnumulated ::= 識別子

リアルタイプ ::=            本物

リアルバリュー::=
        数値実数値
|       スペシャルリアルバリュー

数値実数値 ::=
        実数
|       "-" 実数
|シーケンス値

特別な実値 ::=
        プラスインフィニティ
|マイナス無限大
|数字ではありません

XMLRealValue ::=
        XML数値実数値 | XMLSpecialRealValue

XMLNumericRealValue ::= 実数 "-"
        & 実数
|

XMLSpecialRealValue ::=
        空の要素実数
|テキストリアル
```

```
空の要素実数 ::=
        "<" & PLUS-INFINITY "/>"
| "<" &マイナス無限大"/>"
| "<" &非数字"/>"

テキストリアル ::=
        「INF」
|       「-」と「INF」
| 「な〜ん」

ビット文字列タイプ ::=
        ビット文字列
|ビット文字列"{" NamedBitList "}"

名前付きビットリスト ::=
        名前付きビット
| NamedBitList "," NamedBit

名前ビット ::=
        識別子"("数値")"識別子"("
|       DefinedValue ")"

ビットストリング値 ::=
        bstring
| hstring
| "{"識別子リスト"}" | 「{」 「}」

|値を含む

IdentifierList ::= 識別子

|       IdentifierList "," 識別子

XMLBitStringValue ::=
        XMLTypedValue
        xmlbstring
        XMLIdentifierList
| | |空の

XMLIdentifierList ::=
        空の要素リスト
|       テキストリスト

EmptyElementList ::= "<" & 識
        別子 "/>"
| EmptyElementList "<" & 識別子 "/>"

TextList ::= 識別
        子
|       TextList 識別子

オクテット文字列タイプ ::=オクテット文字列

オクテット文字列値 ::=
        bstring
| hstring
|値を含む

XMLOctetStringValue ::=
        XMLTypedValue
|       xmlhstring

NullType ::= NULL

Null値 ::= NULL

XMLNullValue ::= 空
```

```
シーケンスタイプ::=
        順序"{" "}"
| SEQUENCE "{" ExtensionAndException OptionalExtensionMarker "}"
| SEQUENCE "{" ComponentTypeLists "}"

ExtensionAndException ::= "..." | 「...」例外仕様

OptionalExtensionMarker ::= "," "..." |空の

コンポーネントタイプリスト ::=
        RootComponentTypeList
        | RootComponentTypeList "," ExtensionAndException ExtensionAdditions
                オプションの拡張マーカー
| RootComponentTypeList "," ExtensionAndException ExtensionAdditions
                ExtensionEndMarker "," RootComponentTypeList
| ExtensionAndException ExtensionAdditions ExtensionEndMarker ","
                RootComponentTypeList
| ExtensionAndException ExtensionAdditions OptionalExtensionMarker

ルートコンポーネントタイプリスト ::= コンポーネントタイプリスト

ExtensionEndMarker ::= "," "..."

拡張機能の追加 ::=
        "," 拡張機能追加リスト |空の


拡張機能追加リスト ::=
        拡張子追加
|       ExtensionAdditionList "," ExtensionAddition

拡張機能の追加 ::=
        コンポーネントの種類
|拡張機能追加グループ

ExtensionAdditionGroup ::= "[[" VersionNumber ComponentTypeList "]]"

バージョン番号 ::= 空 |番号"："

コンポーネントタイプリスト ::=
        コンポーネントの種類
| ComponentTypeList "," コンポーネントタイプ

コンポーネントタイプ::=
        名前付きタイプ
| NamedTypeオプション
| NamedType のデフォルト値
|タイプのコンポーネント

シーケンス値 ::=
        "{"コンポーネント値リスト"}" | 「{」
「}」

コンポーネント値リスト ::=
        名前付き値
| ComponentValueList "," NamedValue

XMLシーケンス値 ::=
        XMLコンポーネント値リスト |空の


XMLコンポーネント値リスト ::=
        XMLNamedValue
| XMLComponentValueList XMLNamedValue

SequenceOfType ::=タイプのシーケンス| NamedTypeのシーケンス

値の順序 ::=
        "{"値リスト"}"
```

| "{" NamedValueList "}" | 「{」 「}」

値リスト ::=
価値
| ValueList "," 値

名前付き値リスト ::=
名前付き値
| NamedValueList "," NamedValue

XMLシーケンス値::=
XMLValueList
| XMLDelimitedItemList |空の

XMLValueList ::=
XMLValueOrEmpty
| XMLValueOrEmpty XMLValueList

XMLValueOrEmpty ::=
XML値
| "<" & NonParameterizedTypeName "/>"

XMLDelimitedItemList ::=
XMLDelimitedItem
| XMLDelimitedItem XMLDelimitedItemList

XMLDelimitedItem ::=
"<" & NonParameterizedTypeName ">" XMLValue
"</" & NonParameterizedTypeName ">" | "<" & 識別
子 ">" XMLValue "</" & 識別子 ">"

セットタイプ::=
セット"{" "}"
| SET "{" ExtensionAndException OptionalExtensionMarker "}"
| SET "{" ComponentTypeLists "}"

値を設定::=
"{"コンポーネント値リスト"}" | 「{」
「}」

XMLSetValue ::=
XMLコンポーネント値リスト |空の

タイプのセット::=
タイプのセット
| NamedTypeのセット

値の設定::=
"{"値リスト"}"
| "{" NamedValueList "}" | 「{」 「}」

XMLSetOfValue ::=
XMLValueList
| XMLDelimitedItemList |空の

ChoiceType ::= CHOICE "{" AlternativeTypeLists "}"

代替タイプリスト ::=
RootAlternativeTypeList
| RootAlternativeTypeList ","
ExtensionAndException ExtensionAdditionAlternatives
オプションの拡張マーカー

RootAlternativeTypeList ::= AlternativeTypeList

```
ExtensionAdditionAlternatives ::=
        "," ExtensionAdditionAlternativesList |空の


ExtensionAdditionAlternativesList ::=
        拡張機能追加代替
|       ExtensionAdditionAlternativesList "," ExtensionAdditionAlternative

ExtensionAdditionAlternative ::=
        ExtensionAdditionAlternativesGroup
|名前付きタイプ

ExtensionAdditionAlternativesGroup ::=
        "[[" VersionNumber AlternativeTypeList "]]"

AlternativeTypeList ::=
        名前付きタイプ
|       AlternativeTypeList "," NamedType

ChoiceValue ::= 識別子":"値

XMLChoiceValue ::= "<" & 識別子 ">" XMLValue "</" & 識別子 ">"

SelectionType ::= 識別子 "<" タイプ

PrefixedType ::=
        タグ付きタイプ
| EncodingPrefixedType

PrefixedValue ::= 値

XMLPrefixedValue ::= XMLValue

タグ付きタイプ ::=
        タグの種類
|タグの暗黙的なタイプ
|タグの明示的なタイプ

タグ ::= "[" EncodingReference クラス ClassNumber "]"

EncodingReference ::= エンコー
        ディング参照 ":" |空の


クラス番号 ::= 数値


|定義された値

クラス ::=
        ユニバーサル

|応用
|プライベート|空の


EncodingPrefixedType ::=
エンコーディングプレフィックスタイプ

エンコーディングプレフィックス ::=
"[" エンコーディング参照エンコーディング命令 "]"

オブジェクト識別子タイプ ::=
        オブジェクト識別子


オブジェクト識別子値 ::=
        "{" ObjIdComponentsList "}"
| "{" DefinedValue ObjIdComponentsList "}"

ObjIdComponentsList ::=
        ObjIdコンポーネント
| ObjIdComponentsObjIdComponentsList
```

```
ObjIdコンポーネント ::=
        お名前フォーム
|ナンバーフォーム
|名前と番号フォーム
|定義された値

名前フォーム ::= 識別子

NumberForm ::= 数値 |定義された値

NameAndNumberForm ::= 識別子"("
        NumberForm ")"

XMLObjectIdentifierValue ::=
        XMLObjIdComponentList

XMLObjIdComponentList ::= XMLObjIdコ
        ンポーネント |
XMLObjIdComponent & "." & XMLObjIdComponentList

XMLObjIdコンポーネント ::=
        お名前フォーム
| XML番号フォーム
| XML名前と番号フォーム

XMLNumberForm ::= 数値

XMLNameAndNumberForm ::= 識別子 & "("
        & XMLNumberForm & ")"

RelativeOIDType ::= RELATIVE-OID

相対OID値::=
        "{" RelativeOIDComponentsList "}"

RelativeOIDComponentsList ::=
        相対OIDコンポーネント
| RelativeOIDComponents RelativeOIDComponentsList

相対OIDコンポーネント ::=
        ナンバーフォーム
|名前と番号フォーム
|定義された値

XMLRelativeOID値 ::=
        XMLRelativeOIDComponentList

XMLRelativeOIDComponentList ::= XMLRelativeOID
        コンポーネント | XMLRelativeOIDComponent
& "." & XMLRelativeOIDComponentList

XMLRelativeOIDコンポーネント ::=
        XML番号フォーム
| XML名前と番号フォーム

IRIタイプ ::= OID-IRI

IRI値::=
        「」
        最初のアーク識別子
        後続のアーク識別子
        「」

FirstArcIdentifier ::=
        "/" ArcIdentifier

後続のArcIdentifier ::=
        "/" ArcIdentifier 後続の ArcIdentifier |空の
```

ArcIdentifier ::=
integerUnicodeLabel 非
| 整数 UnicodeLabel

XMLIRI値 ::=
最初のアーク識別子
後続のアーク識別子

RelativeIRIType ::= RELATIVE-OID-IRI

相対IRI値 ::=
「」
FirstRelativeArcIdentifier
後続のアーク識別子
「」

FirstRelativeArcIdentifier ::=
アーク識別子

XMLRelativeIRI値 ::=
FirstRelativeArcIdentifier
後続のアーク識別子

EmbeddedPDVType ::=埋め込み PDV

EmbeddedPDVValue ::= SequenceValue

XMLEmbeddedPDVValue ::= XMLSequenceValue

外部タイプ ::=外部

外部値 ::= シーケンス値

XML外部値 ::= XMLシーケンス値

時間タイプ ::= TIME

時間値 ::= 文字列

XMLTimeValue ::= xmltstring

DateType ::=日付

TimeOfDayType ::=時刻

DateTimeType ::=日付-時刻

期間タイプ ::=期間

文字列タイプ ::=
RestrictedCharacterStringType
| 無制限の文字列タイプ

文字列値 ::=
制限された文字列値
| 無制限の文字列値

XML文字列値 ::=
XMLRestrictedCharacterStringValue
| XMLUnrestrictedCharacterStringValue

RestrictedCharacterStringType ::=
BMP文字列
|一般文字列
|グラフィック文字列
| IA5文字列
| ISO646文字列
|数値文字列
|印刷可能な文字列
|テレテックスストリング
| T61ストリング

|ユニバーサル文字列
| UTF8文字列
|ビデオテックス文字列
|可視文字列

RestrictedCharacterStringValue ::= cstring
|
文字列リスト
|クワドルプル
|タプル

CharacterStringList ::= "{" CharSyms "}"

チャーシムズ ::=
文字定義
| CharSyms "," CharsDefn

文字定義::=
cstring
|クワドルプル
|タプル
|定義された値

4 倍::= "{"グループ "、" 平面 "、" 行 "、" セル"}"

グループ ::= 番号

平面 ::= 数値

行 ::= 番号

細胞 ::= 数値

タプル ::= "{" TableColumn "," TableRow "}"

テーブル列 ::= 数値

テーブル行 ::= 数値

XMLRestrictedCharacterStringValue ::= xmlcstring

UnrestrictedCharacterStringType ::=文字列

UnrestrictedCharacterStringValue ::= シーケンス値

XMLUnrestrictedCharacterStringValue ::= XMLSequenceValue

UsefulType ::= タイプリファレンス

41.1 では次の文字列型が定義されています。

UTF8文字列	グラフィック文字列
数値文字列	可視文字列
印刷可能な文字列	ISO646文字列
テレテックスストリング	一般文字列
T61ストリング	ユニバーサル文字列
ビデオテックス文字列	BMP文字列
IA5文字列	

次の便利なタイプが第 46 節から第 48 節で定義されています。

一般化された時間

UTC時間

オブジェクト記述子

次の演出が第 49 条から第 51 条で使用されています。

```
制約タイプ ::=
        型制約
|制約付きタイプ

制約付きタイプ::=
        SET制約OFタイプ
| SETサイズ拘束OFタイプ
| SEQUENCE制約OFタイプ
| SEQUENCEサイズ制約OFタイプ
| NamedTypeのSET制約
| NamedTypeのSizeConstraintを設定する
| NamedTypeのSEQUENCE制約
| NamedTypeのシーケンスサイズ制約

制約 ::= "("制約仕様例外仕様")"

制約仕様 ::= サブタイプ制約 |
                                一般的な制約

サブタイプ制約 ::= 要素セット仕様

要素セット仕様 ::= ルート要
        素セット仕様 |
RootElementSetSpec "," "..."
| RootElementSetSpec "," "..." "," AdditionalElementSetSpec

ルート要素セット仕様 ::= 要素セット仕様

追加要素セット仕様 ::= 要素セット仕様

ElementSetSpec ::= 共用体
|すべての除外事項

ユニオン ::= 交差
| UElems ユニオンマーク交差点

UElems ::= ユニオン

交差点 ::= IntersectionElements
| IElems IntersectionMark IntersectionElements

IElems ::= 交差

IntersectionElements ::= 要素 |要素の除外

要素 ::= 要素

除外 ::= EXCEPT要素

ユニオンマーク ::= "|"                  |           連合

交差点マーク ::=                "^" |           交差点

要素::=
        サブタイプ要素
|オブジェクトセット要素
| "(" ElementSetSpec ")"

サブタイプ要素 ::=
        単一値
|含まれるサブタイプ
|値の範囲
|許可されたアルファベット |
        サイズ制約
|タイプ制約
|インナータイプ制約 |
        パターン制約 |
        プロパティ設定
```

```
|期間範囲
|時間点範囲
|繰り返し範囲

単一値 ::= 値

ContainedSubtype ::= タイプを含む

含まれます ::=含まれます|空の

ValueRange ::= LowerEndpoint ".." UpperEndpoint

下端点 ::= 下端値 | LowerEndValue "<"

アッパーエンドポイント ::= アッパーエンド値 | "<" 上限値

LowerEndValue ::= 値 |最小

上限値 ::= 値 |マックス

サイズ制約 ::=サイズ制約

TypeConstraint ::= タイプ

許可されたアルファベット ::= FROM制約

InnerTypeConstraints ::=
        WITH COMPONENT SingleTypeConstraint
|コンポーネント付き

SingleTypeConstraint::= 制約

MultipleTypeConstraints ::=
        フルスペック
|       部分仕様

フルスペック ::= "{" TypeConstraints "}"

PartialSpec ::= "{" "..." "," TypeConstraints "}"

タイプ制約::=
        名前付き制約
| NamedConstraint "," TypeConstraints

NamedConstraint ::= 識
        別子 ComponentConstraint

ComponentConstraint ::= ValueConstraint PresenceConstraint

値制約 ::= 制約 |空の

プレゼンス制約 ::=現在|不在|オプション|空の

PatternConstraint ::= PATTERN値

PropertySettings ::= SETTINGS単純文字列

プロパティ設定リスト ::=
        プロパティと設定ペア
|       PropertySettingsList PropertyAndSettingPair

PropertyAndSettingPair ::= プロパティ名 "= 設定名

プロパティ名 ::= psname

設定名 ::= psname

期間範囲 ::= 値範囲

TimePointRange ::= ValueRange

RecurrenceRange ::= ValueRange
```

例外仕様 ::= "!"例外識別 |空の

例外識別 ::=
署名付き番号
|定義された値
| 「:」値を入力します

一連の ITU-T 勧告

シリーズ A ITU-T の作業の構成

シリーズ D 関税と会計原則、国際電気通信/ICT 経済と政策
問題

シリーズ E 全体的なネットワーク運用、電話サービス、サービス運用および人的要因

シリーズF 電話以外の電気通信サービス

シリーズ G 伝送システムとメディア、デジタル システムとネットワーク

シリーズ H オーディオビジュアルおよびマルチメディア システム

シリーズI 統合サービスデジタルネットワーク

シリーズJ ケーブル ネットワークとテレビ、音声プログラム、その他のマルチメディア信号の送信

シリーズ K 干渉に対する保護

シリーズ L 環境と ICT、気候変動、電子廃棄物、エネルギー効率。屋外プラントのケーブルおよびその他の要素の建設、設置および保護

シリーズ M TMN およびネットワーク メンテナンスを含む通信管理

シリーズ N メンテナンス: 国際音声番組およびテレビ送信回線

シリーズO 測定器仕様

シリーズP 電話伝送品質、電話設備、ローカル回線ネットワーク

シリーズ Q スイッチングとシグナリング、および関連する測定とテスト

シリーズR 電信伝送

シリーズS 電信サービス端末装置

テレマティック サービス用シリーズ T 端末

シリーズ U 電信交換機

シリーズ V 電話網を介したデータ通信

シリーズ X データ ネットワーク、オープン システム通信およびセキュリティ

シリーズ Y グローバル情報インフラストラクチャ、インターネット プロトコルの側面、次世代ネットワーク、モノのインターネット、スマート シティ

シリーズ Z 言語と通信システムの一般的なソフトウェア側面

スイスで印刷
ジュネーブ、2021