



拡張可能なマーク付け言語 (XML) 1.0

W3C勧告 10日 2月 1998年

本版

<http://www.w3.org/TR/1998/REC-xml-980210>
<http://www.w3.org/TR/1998/REC-xml-980210.xml>
<http://www.w3.org/TR/1998/REC-xml-980210.html>
<http://www.w3.org/TR/1998/REC-xml-980210.pdf>
<http://www.w3.org/TR/1998/REC-xml-980210.ps>

最新版

<http://www.w3.org/TR/REC-xml>

旧版

<http://www.w3.org/TR/PR-xml-971208>

作成者及び寄稿者

Tim Bray (Textuality and Netscape) <tbray@textuality.com>
Jean Paoli (Microsoft) <jeanpa@microsoft.com>
C. M. Sperberg-McQueen (University of Illinois at Chicago) <cmsmq@uic.edu>

Copyright © 1998年 W3C[®] (MIT, INRIA, Keio), All Rights Reserved.
本文書にはW3Cの免責，商標，文書使用及びソフトウェア使用許諾の規定が適用される。

概要

拡張可能なマーク付け言語(XML)はSGMLのサブセットであって、この標準情報(TR)で、そのすべてを規定する。XMLの目標は、現在のHTMLと同様に、任意の文書型を持つSGML文書をWWW上で配布、受信及び処理できることとする。XMLは実装が容易であって、SGML及びHTMLのどちらに対しても相互運用性を保つ設計がなされている。

本文書の状態

この標準情報(TR)の原勧告はW3Cメンバ及び作業グループによってレビューされ、W3C勧告として技術統括責任者によって承認された。この標準情報(TR)は、安定したものであって、参考

資料として使用してもよく、他の文書から規定の参考資料として引用してもよい。勧告作成において、W3Cは、この規定を広く知らせ、普及させる役割をもつ。これによってWWWの機能と相互運用性が高まる。

この標準情報(TR)は、現在広範囲に使用されている国際的なテキスト処理の標準(Standard Generalized Markup Language, ISO 8879:1986に修正を加えたもの)をWWW上で使用するためにサブセット化した構文を規定する。この標準情報(TR)は、XML活動を通じて作成されたもので、この活動の詳細は <http://www.w3.org/XML> に示されている。現在のW3C勧告のリスト及び他の技術文書は<http://www.w3.org/TR>に示されている。

この標準情報(TR)では、[Berners-Lee et al.]に定義する URI(Uniform Resource Identifier)を使用する。URIの制定作業は進行中であって、[IETF RFC1738]及び[IETF RFC1808]を更新する予定となっている。

この標準情報(TR)の原勧告の正誤表は <http://www.w3.org/XML/xml-19980210-errata> において入手できる。

この標準情報(TR)の原勧告に誤りがあれば xml-editor@w3.org に報告されたい。

目次

| | |
|-----------------------------|----|
| 1. 一般事項 | 1 |
| 1.1. 経緯及び目標 | 1 |
| 1.2. 定義 | 2 |
| 2. 文書 | 3 |
| 2.1. 整形形式のXML文書 | 3 |
| 2.2. 文字 | 4 |
| 2.3. 共通の構文構成子 | 4 |
| 2.4. 文字データ及びマーク付け | 5 |
| 2.5. コメント | 5 |
| 2.6. 処理命令 | 6 |
| 2.7. CDATAセクション | 6 |
| 2.8. 前書き及び文書型宣言 | 6 |
| 2.9. スタンドアロン文書宣言 | 9 |
| 2.10. 空白の取扱い | 9 |
| 2.11. 行末の取扱い | 10 |
| 2.12. 言語識別 | 10 |
| 3. 論理構造 | 11 |
| 3.1. 開始タグ、終了タグ及び空要素タグ | 12 |
| 3.2. 要素型宣言 | 13 |
| 3.2.1. 要素内容 | 14 |
| 3.2.2. 混合内容 | 15 |
| 3.3. 属性リスト宣言 | 15 |
| 3.3.1. 属性の型 | 16 |
| 3.3.2. 属性のデフォルト | 17 |
| 3.3.3. 属性値の正規化 | 18 |
| 3.4. 条件付きセクション | 18 |
| 4. 物理構造 | 19 |
| 4.1. 文字参照及び実体参照 | 20 |
| 4.2. 実体宣言 | 21 |
| 4.2.1. 内部実体 | 22 |
| 4.2.2. 外部実体 | 22 |
| 4.3. 解析対象実体 | 23 |
| 4.3.1. テキスト宣言 | 23 |
| 4.3.2. 整形形式の解析対象実体 | 23 |
| 4.3.3. 実体における文字符号化 | 23 |

| | |
|------------------------------------|----|
| 4.4. XMLプロセサによる実体及び参照の扱い | 24 |
| 4.4.1. “認識しない” | 25 |
| 4.4.2. “取込み” | 25 |
| 4.4.3. “検証のために取込み” | 25 |
| 4.4.4. “禁止” | 26 |
| 4.4.5. リテラル内での取込み | 26 |
| 4.4.6. “通知” | 26 |
| 4.4.7. “処理しない” | 26 |
| 4.4.8. “PEとして取込み” | 27 |
| 4.5. 内部実体置換テキストの構築 | 27 |
| 4.6. 定義済み実体 | 27 |
| 4.7. 記法宣言 | 28 |
| 4.8. 文書実体 | 28 |
| 5. 適合性 | 28 |
| 5.1. 妥当性を検証するプロセサ及び検証しないプロセサ | 28 |
| 5.2. XMLプロセサの使用 | 29 |
| 6. 表記法 | 29 |
| 附属書 | |
| A. 参考文献 | 31 |
| A.1. 規定の参考文献 | 31 |
| A.2. 他の参考文献 | 31 |
| B. 文字クラス | 32 |
| C. XML及びSGML【非準】 | 35 |
| D. 実体参照及び文字参照の展開【非準】 | 35 |
| E. 決定的内容モデル【非準】 | 36 |
| F. 文字符号化方式の自動検出【非準】 | 37 |
| G. W3C XML作業グループ【非準】 | 39 |

1. 一般事項

拡張可能なマーク付け言語XML(eXtensible Markup Language)は、XML文書というデータオブジェクトのクラスを規定し、XML文書进行处理するプログラムの動作の一部を規定する。XMLは、SGML(Standard Generalized Markup Language)[ISO 8879]の制限したサブセットとする。XML文書は、必ずSGML規格に適合する。

XML文書は**実体**という記憶単位から成り、実体は構文解析されるデータ又は構文解析されないデータから成る。構文解析されるデータは、**文字**から成り、その一部は**文字データ**を構成し、一部は**マーク付け**を構成する。マーク付けは、文書の記憶レイアウト及び論理構造を記述する符号とする。XMLは、記憶レイアウト及び論理構造についての制約条件を記述する機構を提供する。

XMLプロセサ というソフトウェアモジュールは、XML文書を読み込み、その内容及び構造へのアクセスを提供するために用いる。XMLプロセサは、他のモジュールのために動作することを前提としており、そのモジュールをアプリケーションという。この標準情報(TR)は、XMLプロセサに要求される振舞いを規定する。つまり、XMLデータの読み込み方法を規定し、アプリケーションに提供する情報を規定する。

1.1. 経緯及び目標

1996年にWorld Wide Web Consortium(W3C)の中に設立されたXML作業グループ(以前は、SGML編集レビュー委員会と呼ばれた。)がXMLを開発した。この作業グループの議長を、Sun MicrosystemsのJon Bosakが務めた。W3Cが組織し、以前はSGML作業グループと呼ばれたXML SIG(Special Interest Group)も、XMLの制定に非常に活発に参画した。XML作業グループのメンバーを附属書Gに示す。Dan Connollyは、作業グループとW3Cとの調整役を務めた。

XMLの設計目標を次に示す。

1. XMLは、インターネット上でそのまま使用できる。
2. XMLは、広範囲のアプリケーションを支援する。
3. XMLは、SGMLと互換性をもつ。
4. XML文書进行处理するプログラムは容易に書ける。
5. XMLでは、オプションの機能はできるだけ少なくし、理想的には一つも存在しない。
6. XML文書は、人間にとって読みやすく、十分に理解しやすい。
7. XMLの設計は、すみやかに行う。
8. XMLの設計は、厳密で、しかも簡潔なものとする。
9. XML文書は、容易に作成できる。
10. XMLでは、マーク付けの数を減らすことは重要ではない。

XML第1.0版を理解し、それを処理する計算機プログラムを書くために十分な情報は、この標準情報(TR)、関連する規格など(文字についてはUnicode及びISO/IEC 10646、言語識別タグについてはIETF RFC 1766、言語コードについてはISO 639、並びに国コードについてはISO 3166。)によってすべて示す。

この版のXMLの規定は、テキスト及び法律上の注意を一切改変しない限り、自由に配布してもよい。

1.2. 定義

XML文書を規定するために使用する用語は、この標準情報(TR)内で定義する。次に示す語句は、それらの用語を定義するため、及びXMLプロセサの動きを規定するために使用する。

1.2.1 してもよい(may)

適合する文書又はXMLプロセサは、記述されたとおりに動作してもよいが、そのとおりにする必要はない。

1.2.2 しなければならない(must)

適合する文書又はXMLプロセサは、記述されたとおりに動作することが要求される。そうでなければ、エラーとする。

1.2.3 エラー(error)

この標準情報(TR)が定める規則に対する違反。結果は定義しない。適合するソフトウェアは、エラーを検出して報告してもよく、エラーから回復してもよい。

1.2.4 致命的エラー(fatal error)

適合するXMLプロセサが検出し、アプリケーションに報告しなければならないエラー。プロセサは、致命的エラーを発見したあとも、それ以降のエラーを探すためにデータ処理を続行し、見つかったエラーをアプリケーションに報告してもよい。エラー訂正をサポートするために、プロセサは、処理していないデータ(文字データ及びマーク付けの混在したもの。)を文書から取り出し、アプリケーションに渡してもよい。しかし、プロセサは、致命的エラーを一度でも検出したなら通常の処理を続行してはならない。つまり、プロセサは、文字データ及び文書の論理構造についての情報を、通常の方法でアプリケーションに渡し続けてはならない。

1.2.5 ユーザのオプション指定によっては(at user option)

適合するソフトウェアは、記述されたとおりに振る舞ってもよい(may)、又は振る舞わなくてはならない(must)(文章中の助動詞による。)。そのとおりに振る舞う場合は、記述された振舞いを選択又は拒否する手段をユーザに提供しなければならない。

1.2.6 妥当性制約(validity constraint)

すべての**妥当な** XML文書に適用する規則。妥当性制約の違反は、エラーとする。ユーザのオプション指定によっては、**検証を行うXMLプロセサ**は、このエラーを報告しなければならない。

1.2.7 整形式制約(well-formedness constraint)

すべての**整形式**のXML文書に適用する規則。整形式制約の違反は**致命的エラー**とする。

1.2.8 マッチ(match)

a) 文字列又は名前のマッチ 比較する二つの文字列又は名前は、同一でなければならない。ISO/IEC 10646において、複数の表現が可能な文字 [例えば、合成形式及び基底文字+発音符(ダイアクリティカルマーク)形式] は、どちらの文字列も同じ表現のときに限り、マッチする。ユーザのオプション指定によっては、プロセサは、その文字を標準形に正規化してもよい。比較のとき、大文字と小文字との区別をする。
b) 文字列と文法中の規則とのマッチ ある生成規則から生成する言語に、ある文字列が属するとき、この文字列は、この生成規則にマッチするという。
c) 内容と内容モデルとのマッチある要素が、制約**"要素の妥当性"**に示す意味で適合するとき、この要素は、その宣言にマッチするという。

1.2.9 互換性のためには(for compatibility)

XMLの機能であって、XMLがSGMLと互換であることを保証するためだけに導入されるもの。

1.2.10 相互運用性のためには(for interoperability)

拘束力をもたない推奨事項。ISO 8879へのWebSGML適用附属書以前から存在するSGMLプロセッサが、XML文書进行处理できる可能性を高めるために取り入れるもの。

2. 文書

この標準情報(TR)で定義する意味で、**整形形式**のデータオブジェクトをXML文書という。整形形式のXML文書が、ある制約条件を満足すれば、**妥当なXML文書**という。

XML文書は、論理構造及び物理構造をもつ。物理的には、文書は、**実体**という単位からなる。実体が他の実体を**参照**すれば、参照された実体も文書の一部になる。文書は、“ルート”すなわち**文書実体**から始まる。論理的には、文書は、宣言、要素、コメント、文字参照及び処理命令を含み、これらすべては、文書内で明示的なマーク付けによって示す。論理構造及び物理構造は、23ページの§ 4.3.2 **整形形式の解析対象実体**に示すとおり、厳密に入れ子になっていなければならない。

2.1. 整形形式のXML文書

あるテキストオブジェクトが、次の条件を満たすとき、そのテキストオブジェクトを整形形式のXML文書と呼ぶ。

- a) 全体として、**document**というラベルをもつ生成規則にマッチする。
- b) この標準情報(TR)で定義するすべての整形形式制約に従う。
- c) 文書内で直接的または間接的に参照されるそれぞれの**解析対象実体**が**整形形式**となる。

[1] `document ::= prolog element Misc*`

document生成規則にマッチするとは、次を意味する。

- a) 一つ以上の**要素**を含む。
- b) ルート又は文書要素という要素が一つだけ存在し、これは、他の要素の**内容**に含まれない。これ以外のすべての要素は、その開始タグが他の要素の内容に含まれれば、対応する終了タグも同じ要素の内容に含まれる。つまり、要素は、開始タグ及び終了タグによって区切られ、入れ子構造をなす。

これらの結果として、文書内のどの非ルート要素Cに対しても、ある他の要素Pが存在し、Cは、Pの内容に含まれるが、Pの内容に含まれる他の要素に含まれることはない。このとき、PをCの親といい、CをPの子という。

2.2. 文字

解析対象実体は、テキスト(文字の並びであって、マーク付け又は文字データを表してもよい。)を含む。文字は、テキストの最小単位であって、[ISO/IEC 10646]に規定されている。使用できる文字は、タブ、改行、復帰及び(Unicode及びISO/IEC 10646に規定する)図形文字とする。[Unicode]の6.8節で定義される互換性文字は使用を避けることが望ましい。

```
[2] Char ::= #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD]
        | [#x10000-#x10FFFF] /* 任意のUnicode文字。ただし、サロゲートブロック、FFFE及びFFFFは除く。 */
```

文字番号をビットパターンに符号化する機構は、実体ごとに違ってよい。すべてのXMLプロセッサは、ISO/IEC 10646のUTF-8符号化方式及びUTF-16符号化方式を受け付けなければならない。二つのどちらが用いられているかを明示するための機構、及び他の符号化方式を利用するための機構は、23ページの § 4.3.3 実体における文字符号化に記述する。

2.3. 共通の構文構成子


2.3では、文法内で広く使用するいくつかの記号を定義する。

S (空白)は、一つ以上のスペース文字(#x20)、復帰、改行又はタブから成る。

```
[3] S ::= (#x20 | #x9 | #xD | #xA)+
```

便宜上、文字を、字、数字又は他の文字に分類する。字は、アルファベットの、若しくは表音的である基底文字(一つ以上の結合文字が、後に続くこともある。)、又は統合漢字から成る。各クラスにおける実際の文字についての完全な定義は、32ページの附属書 B 文字クラスに示す。

Nameは、字又はいくつかの区切り文字の一つで始まり、その後には字、数字、ハイフン、下線、コロン又はピリオドが続く。これらの文字を名前文字という。文字列"xml"で始まる名前、又は正規表現((('X'|'x') ('M'|'m') ('L'|'l'))にマッチする任意の文字列で始まる名前は、この標準情報(TR)の現在の版又は将来の版での標準化のために予約する。

 XMLの名前の中のコロンは、名前空間での実験のために予約する。コロンの意味は、将来のある時点で標準化するものとし、そのときには、実験的な目的でコロンを使用する文書を更新する必要がある可能性がある。XMLで採用する名前空間の機構が、区切り子として実際にコロンを使用するという保証はない。事実上、これは、名前空間の実験の一つとして以外には、XMLの名前の中でコロンを使用しないほうがよいことを意味する。しかし、XMLプロセッサは、名前文字としてコロンを受け付けることが望ましい。

Nmtoken(名前トークン)は、名前文字の列とする。

```
[4] NameChar ::= Letter | Digit | ':' | '-' | '_' | '.' | CombiningChar |
        Extender
[5] Name ::= (Letter | '_' | ':') (NameChar)*
[6] Names ::= Name (S Name)*
[7] Nmtoken ::= (NameChar)+
[8] Nmtokens ::= Nmtoken (S Nmtoken)*
```


リテラルデータは、引用符で囲まれた文字列とし、その列の区切り子として使用する引用符は含まない。リテラルは、内部実体(EntityValue)、属性値(AttValue)、外部識別子(SystemLiteral)の内容の指定に使用する。SystemLiteralはマーク付けの走査を行わずに解析できることに注意せよ。

- ```
[9] EntityValue ::= '"' ([^%&"] | PEReference | Reference)* '"'
 | "'" ([^%&'] | PEReference | Reference)* "'"

[10] AttValue ::= '"' ([^<&"] | Reference)* '"'
 | "'" ([^<&'] | Reference)* "'"

[11] SystemLiteral ::= ('"' [^"]* '"') | ('"' [^"]* '"')

[12] PubidLiteral ::= '"' PubidChar* '"' | "'" (PubidChar - "'")* "'"

[13] PubidChar ::= #x20 | #xD | #xA | [a-zA-Z0-9] | [-'()+,./:=?;!*#@$_%]
```

## 2.4. 文字データ及びマーク付け

テキストは、文字データ及びマーク付けから成る。マーク付けは、開始タグ、終了タグ、空要素タグ、実体参照、文字参照、コメント、CDATAセクションの区切り子、文書型宣言及び処理命令の形をとる。

マーク付けではないすべてのテキストは、文書の文字データを構成する。

アンド記号(&)及び不等号(小なり) (<)は、マーク付けの区切り子として、又はコメント、処理命令若しくはCDATAセクション内で使用する場合にだけ、そのままの形で出現してよい。これらの文字は、内部実体宣言のリテラル実体値内に記述してもよい。詳しくは、23ページの§ 4.3.2 整形式の解析対象実体を参照。これらの文字が他の部分で必要な場合、番号による文字参照又は文字列"&amp;"及び文字列"&lt;"を使用して別扱いしなければならない。不等号(大なり) (>)は、文字列"&gt;"を使用して表現してもよい。内容の中で"]]>"を使用するときは、それが、CDATAセクションの終了をマーク付けしない限り、互換性のため、"&gt;"又は文字参照を使用して別扱いしなければならない。

要素の内容では、文字データは、いかなるマーク付けの開始区切り子を含まない任意の文字列とする。CDATAセクションでは、文字データとは、CDATAセクションの終了区切り子"]]>"を含まない任意の文字列とする。

属性値が一重引用符及び二重引用符を含むためには、アポストロフィ又は一重引用符(')は、"&apos;"として表現し、二重引用符(")は、"&quot;"として表現する。

- ```
[14] CharData ::= [^<&]* - ([^<&]* ']'> [^<&]*
```

2.5. コメント

コメントは、他のマーク付けの外ならば、文書のどこに現れてもよい。さらに、文書型宣言の中で、文法が許す場所に現れてもよい。コメントは、文書の文字データの一部ではない。XMLプロセサは、アプリケーションがコメントのテキストを取り出すことを可能としてもよいが、そうしなくともよい。互換性のためには、文字列 "--" (二連ハイフン) は、コメント内で現れてはならない。

- ```
[15] Comment ::= '<!--' ((Char - '-') | ('-' (Char - '-')))* '-->'
```

コメントの例を次に示す。

```
<!-- declarations for <head> & <body> -->
```

## 2.6. 処理命令

処理命令(PI)によって、アプリケーションのための命令を文書に入れることができる。

```
[16] PI ::= '<?' PITarget (S (Char* - (Char* '?'>' Char*)))? '?'
[17] PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
```

PIは、文書の文字データの一部ではないが、アプリケーションに渡されなければならない。PIは、命令が渡されるアプリケーションを特定するために使用するターゲット(PITarget)で始まる。ターゲット名"XML", "xml"などは、この標準情報(TR)の現在の版又は将来の版の標準化のために予約する。XMLの記法機構を、PIのターゲットを宣言するために使用してもよい。

## 2.7. CDATAセクション

CDATAセクションは、文字データが出現するところであれば、どこに出現してもよい。これは、CDATAセクションで囲まなければマーク付けとして認識されてしまう文字を含むテキストを別扱いするのに使用する。CDATAセクションは、文字列"<![CDATA["で始まり、文字列"]]>"で終わる。

```
[18] CDsect ::= CDStart CData CEnd
[19] CDStart ::= '<![CDATA['
[20] CData ::= (Char* - (Char* ']]>' Char*))
[21] CEnd ::= ']]>'
```

CDATAセクション内では、文字列CEndだけをマーク付けとして認識するので、不等号(小なり)及びアンド記号は、そのままの形で出現してよい。"&lt;"及び"&amp;"を使用して別扱いする必要はない。CDATAセクションは入れ子にはできない。

"<greeting>"及び"</greeting>"を、マーク付けではなく、文字データとして認識するCDATAセクションの例を次に示す。

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

## 2.8. 前書き及び文書型宣言

XML文書は、使用するXMLの版を指定するXML宣言で始めてもよく、又そうするのが望ましい。例えば、次に示す完全なXML文書は、整形形式であるが妥当ではない。

```
<?xml version="1.0"?>
<greeting>Hello, world!</greeting>
```

次の文書も同様とする。

```
<greeting>Hello, world!</greeting>
```

この標準情報(TR)のこの版に適合することを示すためには、版番号"1.0"を使用しなければならない。ある文書が、この標準情報(TR)のこの版に適合しないとき、値"1.0"を使用するのは、エラーとする。この標準情報(TR)の今後の版に"1.0"以外の値を付与することが、XML作業グ

ループの意図だが、XMLの将来の版を作成することを確約するわけではなく、作成したとしても番号付けについて、特定の方法を使用することを確約するわけでもない。将来の版を作成する可能性があるため、必要な場合に自動的な版の認識を可能とするため、この構成子を提供する。プロセサは、それがサポートしていない版番号のついた文書を受け取ったならエラーを通知してもよい。

XML文書内のマーク付けの機能は、記憶構造及び論理構造を記述すること、並びに属性及び属性値の対を論理構造に関連づけることにある。XMLは、論理構造についての制約条件を定義するため、及びあらかじめ定義された記憶単位を使用するための機構として**文書型宣言**を提供する。妥当なXML文書とは、文書型宣言をもち、その文書型宣言に示す制約条件を満たすXML文書とする。

文書型宣言は、文書の最初の**要素**の前に現れなければならない。

- Ⓐ prolog ::= XMLDecl? Misc\* (doctypedecl Misc\*)?
- Ⓑ XMLDecl ::= '<?xml' VersionInfo EncodingDecl? SDDDecl? S? '?'>'
- Ⓒ VersionInfo ::= S 'version' Eq (' VersionNum ' | " VersionNum ")
- Ⓓ Eq ::= S? '=' S?
- Ⓔ VersionNum ::= ([a-zA-Z0-9\_.:] | '-')+
- Ⓕ Misc ::= Comment | PI | S

XMLの文書型宣言は、ある文書クラスのための文法を記述する**マーク付け宣言**を含むか、参照する。この文法を、文書型定義又はDTDという。文書型宣言は、マーク付け宣言を含んだ外部サブセット(特別な種類の**外部実体**)を参照することができ、又は内部サブセットにマーク付け宣言を直接含むこともできる。外部サブセットと内部サブセットの両方を使うこともできる。ある文書のDTDは、両方のサブセットをまとめたものとして構成される。

マーク付け宣言は、**要素型宣言**、**属性リスト宣言**、**実体宣言**又は**記法宣言**とする。次に示す整形形式制約及び妥当性制約に規定する通り、これらの宣言は、**パラメタ実体内**に全体又は一部が含まれてもよい。詳しい規定は、19ページの§4 **物理構造**を参照のこと。

- Ⓖ doctypedecl ::= '<!DOCTYPE' S Name (S ExternalID)? S? (' (' markupdecl | PEReference | S)\* ')' S?)? '>'
- Ⓗ markupdecl ::= elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment

マーク付け宣言は、その全体又は一部を、**パラメタ実体の置換テキスト**で構成してもよい。各々の非終端記号 (**elementdecl**、**AttlistDecl**など) のための生成規則は、この標準情報(TR)の3.2節、3.3節、及び4.2節にあり、すべてのパラメタ実体を**取り込んだ**後の宣言について記述する。

#### 【妥当性制約】ルート要素型

文書型宣言における**Name**は、**ルート要素**の型とマッチしなければならない。

### 【妥当性制約】宣言及びパラメタ実体が厳密に入れ子をなすこと

パラメタ実体の置換テキストは、マーク付け宣言内において、厳密に入れ子になっていなければならない。つまり、マーク付け宣言(markupdecl)の最初又は最後の文字が、パラメタ実体参照の指し示す置換テキストに含まれれば、両方とも同じ置換テキストに含まなければならない。

### 【整形形式制約】内部サブセット内のパラメタ実体

DTDの内部サブセットでは、パラメタ実体参照は、マーク付け宣言が出現可能な場所だけに出現できる。マーク付け宣言の一部としては出現できない。この制約は、外部パラメタ実体又は外部サブセットでの参照には適用しない。

内部サブセットのときと同様に、外部サブセットと、DTDにおいて参照する任意の外部パラメタ実体とは、非終端記号markupdeclによって許される型の一連の完全なマーク付け宣言で構成されなければならない。マーク付け宣言の間には、空白又はパラメタ実体参照を置いてもよい。外部サブセット又は外部パラメタ実体の内容の一部は、条件付きセクションを使用して無視してもよいが、内部サブセットではこれは許されない。

β0 extSubset ::= TextDecl? extSubsetDecl

β1 extSubsetDecl ::= ( markupdecl | conditionalSect | PEReference | S )\*

外部サブセット及び外部パラメタ実体は、その中では、パラメタ実体参照がマーク付け宣言の間だけでなく、マーク付け宣言の内でも認められる、という点でも内部サブセットとは異なる。

文書型宣言付きのXML文書の例を、次に示す。

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>
```

システム識別子 "hello.dtd"が、文書のDTDのURIとなる。

次の例の通り、宣言を局所的に与えることもできる。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
 <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

外部サブセット及び内部サブセットの両方を使用するときは、内部サブセットが外部サブセットより先に出現したと見なす。これは、内部サブセットの実体及び属性リスト宣言が、外部サブセットの実体及び属性リスト宣言に優先するという効果をもたらす。

## 2.9. スタンドアロン文書宣言

XMLプロセサは、アプリケーションに文書の内容を渡すが、マーク付け宣言は、この内容に影響を与えることがある。例えば、属性のデフォルト値及び実体宣言は影響を与える。スタンドアロン文書宣言は、XML宣言の一部として出現することができ、影響を与えるマーク付け宣言が文書実体の外部に出現するかどうかを示す。

```

[例] SDDDecl ::= S 'standalone' Eq (("'" ('yes' | 'no') "'") | ("'" ('yes' | 'no') "'"))

```

スタンドアロン文書宣言においては、"yes"の値は、文書実体の外部に（DTDの外部サブセット内に、又は内部サブセットから参照される外部パラメタ実体内に）、XMLプロセサからアプリケーションへと渡される情報に影響するマーク付け宣言が存在しないことを意味する。"no"の値は、その外部マーク付け宣言が存在するか、又は存在する可能性があることを意味する。スタンドアロン文書宣言は、その宣言が文書外部に存在するかどうかを示すだけに注意すること。外部実体への参照が文書内に存在していても、その実体が内部的に宣言されているときは、文書のスタンドアロンの値には影響を与えない。

外部にマーク付け宣言が存在しなければ、スタンドアロン文書宣言は意味をもたない。外部にマーク付け宣言は存在するが、スタンドアロン文書宣言が存在しない場合は、"no"の値が設定されているものとする。

XML文書で `standalone="no"` が設定されているものは、あるアルゴリズムで `standalone="yes"` であるような文書に変換でき、変換後の文書のほうがネットワークによる配信には望ましいかもしれない。

### 【妥当性制約】スタンドアロン文書宣言

スタンドアロン文書宣言は、何らかの外部マーク付け宣言が次のいずれかを宣言しているときは、値 "no" を取らなければならない。

- a) デフォルト値付きの属性であって、この属性が適用される要素が、属性値を指定せずに文書内に現れるもの。
- b) amp, lt, gt, apos, quot 以外の実体であって、その実体に対する参照が文書内に出現するもの。
- c) 値が正規化の対象となる属性であって、正規化の結果として変化する値が文書内で属性に指定されているもの。
- d) 要素内容をもつ要素型であって、空白がその要素型のいずれかのインスタンス内に直接現れるもの。

スタンドアロン文書宣言付きのXML宣言の例を、次に示す。

```
<?xml version="1.0" standalone='yes'?>
```

## 2.10. 空白の取扱い

XML文書を編集するときは、マーク付けを目立たせ読みやすくするために、“空白”(スペース、タブ及び空白行。この標準情報(TR)では、非終端記号のSで表す。)を使うと便利ことが多い。これらの空白は、配布する版の文書の一部に含めることを普通は意図していない。しかし、“意味のある”空白であって、配布する版に保持されなければならないものも多い。例えば、詩及びソースコードにおける空白がこれにあたる。

XMLプロセサは、文書内のマーク付け以外のすべての文字を、変更せずにそのままアプリケーションに渡さなければならない。妥当性を検証するXMLプロセサは、これらの文字の中でどの文字が要素内容に出現する空白を構成するかをアプリケーション側に伝えなければならない。

"xml:space"という特別な属性を要素に加えることによって、アプリケーションはこの要素の中の空白を保存することが望ましいという意図を示してもよい。妥当な文書では、この属性を使用する場合は、他の属性と同じように宣言しなければならない。宣言するときは、取り得る値を"default"及び"preserve"だけとする列挙型でなければならない。例を次に示す。

```
<!ATTLIST poem xml:space (default|preserve) 'preserve'>
```

値"default"は、アプリケーションのデフォルトの空白処理モードを、その要素に適用可能とすることを意味する。値"preserve"は、アプリケーションがすべての空白を保存することを意味する。この宣言の意図は、"xml:space"属性の別の指定で上書きしない限り、要素の内容に現れるすべての要素に適用すると解釈する。

文書のルート要素については、この属性の値を指定するか、又はこの属性のデフォルト値がある場合を除いては、アプリケーションによる空白の取扱いについて、いかなる意図も示さないと解釈する。

## 2.11. 行末の取扱い

XMLの構文解析対象実体は、通常コンピュータのファイル内に保存され、編集の便宜のために複数の行に分けることが多い。これらの行は、普通は、carriage-return (#xD)コード及びline-feed (#xA)コードの何らかの組合せによって分けられる。

アプリケーションの処理を簡単にするため、外部解析対象実体又は内部解析対象実体のリテラル実体値が、"#xD#xA"の2文字の連続とするリテラル又は#xDの単独のリテラルを含む場合に、XMLプロセサは、アプリケーションに単一の文字#xAだけを渡さなければならない(この処理は、入力内に存在する改行コードを構文解析の前に正規化することによって、容易に実現できる。)

## 2.12. 言語識別

文書処理においては、その文書の中身がどんな自然言語又は形式言語で書かれているか明示することが、役に立つことが多い。XML文書内の要素のもつ内容又は属性値において使用する言語を指定するために、"xml:lang"という名前の特別な属性を、文書内に挿入してもよい。妥当な文書においてこの属性を使用する場合は、他の属性と同様に宣言されなくてはならない。属性の値は、[IETF RFC 1766]RFC1766：言語識別のためのタグによって規定される言語識別コードに従う。

- Ⓕ LanguageID ::= Langcode ('-' Subcode)\*
- Ⓖ Langcode ::= ISO639Code | IanaCode | UserCode
- Ⓖ ISO639Code ::= ([a-z] | [A-Z]) ([a-z] | [A-Z])
- Ⓖ IanaCode ::= ('i' | 'l') '-' ([a-z] | [A-Z])+
- Ⓖ UserCode ::= ('x' | 'X') '-' ([a-z] | [A-Z])+
- Ⓖ Subcode ::= ([a-z] | [A-Z])+

Langcodeは、次のどれでもよい。

- a) [ISO 639]“言語の名前表現のためのコード”で規定される2文字の言語コード
- b) Internet Assigned Numbers Authority [IANA]で登録されている言語コード。これは、先頭が“i-” (又は“I-”)で始まる。
- c) ユーザによって定められた言語コード，又は私的な使用のために複数の団体間が取り決めたコード。これらは、今後IANAにおいて標準化又は登録されるコードとの競合を避けるために、先頭を“x-”又は“X-”で始める。

**Subcode**は複数回使ってもよい。最初のサブコードが存在し、その内容が二つの文字から成るときは、[ISO 3166]ISO3166の“国名を表すコード(国コード)”でなければならない。最初のサブコードが3文字以上から成るときは、**Langcode**の先頭が“C”“x-”又は“X-”で始まらない限り、指定した言語に対するサブコードとし、IANAに登録されたものでなければならない。

言語コードは、小文字で表記する慣行があり、国コードは(存在するならば)大文字での表記する慣行がある。しかし、XML文書内における他の名前とは異なり、これらの値については、大文字及び小文字の区別をしないことに注意すること。

例を次に示す。

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
<p xml:lang="en-GB">What colour is it?</p>
<p xml:lang="en-US">What color is it?</p>
<sp who="Faust" desc='leise' xml:lang="de">
 <l>Habe nun, ach! Philosophie,</l>
 <l>Juristerei, und Medizin</l>
 <l>und leider auch Theologie</l>
 <l>durchaus studiert mit heißem Bemüh'n.</l>
</sp>
```

xml:langで宣言した意図は、この要素の内容の中に現れる他の要素のxml:lang属性で上書きされない限り、指定した要素のすべての属性と内容に適用される。

xml:langを次に示す通り、簡単に(デフォルト値を与えずに)宣言してもよい。

```
xml:lang NMTOKEN #IMPLIED
```

必要ならば、特定のデフォルト値を与えてもよい。英語を母語とする学生用のフランス語の詩集では、説明及び注を英語で記述すれば、xml:lang属性を次のとおりに宣言することとなる。

```
<!ATTLIST poem xml:lang NMTOKEN 'fr'>
<!ATTLIST gloss xml:lang NMTOKEN 'en'>
<!ATTLIST note xml:lang NMTOKEN 'en'>
```

### 3. 論理構造

いかなるXML文書も、一つ以上の要素を含む。要素の境界は、**開始タグ**及び**終了タグ**によって区切る。要素が**空要素**のときは、**空要素タグ**で示す。各々の要素は型をもつ。要素型は名前(共通識別子(generic identifier)又はGIと呼ぶことがある。)によって特定される。要素はいくつかの属性をもつことができる。属性は**名前**及び**値**をもつ。

☞ element ::= EmptyElemTag

## | STag content ETag

この標準情報(TR)は、要素型及び属性の意味、使用方法、又は(構文に関するものを除き)名前に制約を与えない。ただし、正規表現(( 'X' | 'x' ) ( 'M' | 'm' ) ( 'L' | 'l' ))にマッチする文字列で始まる名前は、この版又は今後の版のこの標準情報(TR)での標準化のために予約する。

**【整形制約】要素型のマッチ**

要素の終了タグの**名前**は、その要素の開始タグにおける要素型(の名前)とマッチしなければならない。

**【妥当性制約】要素の妥当性**

要素が妥当とは、その要素型(の名前)とマッチする**Name**をもつ宣言( `elementdecl` にマッチするもの)が存在し、さらに次のいずれかの条件を満たす場合とする。

1. a) 宣言が `EMPTY` にマッチし、要素が **内容** をもたない。
2. b) 宣言が `children` にマッチし、要素の**子要素**の並びが、内容モデル中の正規表現によって生成される言語に属する。子要素の間に空白(非終端記号 `S` にマッチする文字の並び)があってもよい。
3. c) 宣言が `Mixed` にマッチし、要素の内容が**文字データ**及び**子要素**からなる。子要素の要素型は、要素の内容モデルに出現する名前にマッチする。
4. d) 宣言が `ANY` にマッチし、どの**子要素**の要素型も宣言されている。

### 3.1. 開始タグ、終了タグ及び空要素タグ

空でない任意のXML要素の始まりは、開始タグによってマーク付けする。

[40] STag ::= '<' Name (S Attribute)\* S? '>'

[41] Attribute ::= Name Eq AttValue

開始タグ及び終了タグ内の**Name**は、要素の型を表わす。**Name**及び**AttValue**の対を要素の属性指定といい、個々の対における**Name**を属性名といい、**AttValue**の内容(区切り子'又は"の間のテキスト)を属性値という。

**【整形制約】属性指定の一意性**

開始タグ又は空要素タグでは、同一の属性名が二回以上出現してはならない。

**【妥当性制約】属性値の型**

属性は宣言されていなければならない。属性値の型は、その属性に対して宣言した型でなければならない(属性の型については、15ページの§ 3.3 **属性リスト宣言**を参照。)

**【整形制約】外部実体への参照がないこと**

属性値には、外部実体への直接的又は間接的な参照を含むことはできない。



## 【整形制約】属性値に&lt;を含まないこと

属性値内で直接的又は間接的に参照する実体("&lt;"を除く。)の置換テキストには、<を含んではならない。

開始タグの例を次に示す。

```
<termdef id="dt-dog" term="dog">
```

開始タグで始まる要素の終わりは、終了タグでマーク付けしなければならない。この終了タグは、対応する開始タグの要素型と同じ名前をもつ。

```
[42] ETag ::= '<' Name S? '>'
```

終了タグの例を、次に示す。

```
</termdef>
```

要素の開始タグと終了タグとの間のテキストをその要素の内容という。

```
[43] content ::= (element | CharData | Reference | CDsect | PI |
 Comment)*
```

要素が空のとき、その要素は、直後に終了タグをもつ開始タグ又は空要素タグで表現しなければならない。空要素タグは、次の特別な形式をとる。

```
[44] EmptyElemTag ::= '<' Name (S Attribute)* S? '/>'
```

空要素タグは、内容をもたない任意の要素の表現に利用できる。空要素タグで表現する要素を、キーワードEMPTYを用いて宣言しなくてもよい。相互運用性のためには、空要素タグは、EMPTYとして宣言された要素には必ず使用しなければならず、またこれ以外の要素には使用しない。

空要素の例を、次に示す。

```
<IMG align="left"
 src="http://www.w3.org/Icons/WWW/w3c_home" />

</br>


```

## 3.2. 要素型宣言

妥当性を保証するため、要素型宣言及び属性リスト宣言を用いてXML文書の要素の構造に制約を加えることができる。要素型宣言は、要素の内容についての制約とする。

要素型宣言は、要素の子として出現可能な要素型について、制約を加えることが多い。ユーザのオプション指定によっては、要素型宣言をもたない要素型が他の要素型宣言によって参照されれば、XMLプロセッサは警告を出してもよい。しかし、これはエラーとはしない。

要素型宣言は次の形式をとる。

```
[45] elementdecl ::= '<IELEMENT' S Name S contentspec S? '>'
```

```
[46] contentspec ::= 'EMPTY' | 'ANY' | Mixed | children
```

ここで、[Name](#)は宣言されている要素の型を示す。

【妥当性制約】要素型宣言の一意性  
一つの要素型を二回以上宣言できない。

要素型宣言の例を次に示す。

```
<!ELEMENT br EMPTY>
<!ELEMENT p (#PCDATA|emph)* >
<!ELEMENT %name.para; %content.para; >
<!ELEMENT container ANY>
```

### 3.2.1. 要素内容

ある型<sup>1</sup>の要素が子要素だけを必ず含み(文字データを含まない。), それらの間には空白(非終端記号Sにマッチする文字)だけしか現れないとき, その要素型は, 要素内容をもつという。この場合, 内容モデルが制約となる。内容モデルは, 子要素の型及び子要素の出現順序を制御する簡単な文法とする。この文法は, 内容素子(cp)から成る。内容素子は, 名前, 内容素子の選択リスト又は内容素子の列リストから構成される。

```
[47] children ::= (choice | seq) ('?' | '*' | '+')?
[48] cp ::= (Name | choice | seq) ('?' | '*' | '+')?
[49] choice ::= '(' S? cp (S? '|' S? cp)* S? ')'
[50] seq ::= '(' S? cp (S? ',' S? cp)* S? ')'
```

ここで、[Name](#)は、子として出現してよい要素の型を示す。この文法で選択リストが現れる位置では、選択リスト内のいずれの内容素子も要素内容の中に現れてよい。列リストに現れる内容素子は、リストで指定する順番のとおり、要素内容に現れなければならない。名前又はリストの後に出現するオプションの文字は、リスト内の要素又は内容素子が、1回以上任意の回数(+), 0回以上任意の回数(\*)又は0回若しくは1回(?)出現可能なことを規定する。この演算子がない場合は要素又は内容素子が正確に1度だけ現われなくてはならないことを意味する。ここで示す構文及び意味は、この標準情報(TR)における生成規則で用いるものと同一とする。

要素の内容が内容モデルにマッチするのは、列、選択及び繰返し演算子に従って、内容の中の要素と内容モデル内の要素型とをマッチさせながら、内容モデル内の一つのパスをたどれるときに限る。[互換性のため](#)、文書内の要素が、内容モデルにおける要素型の複数の出現位置とマッチすることは、エラーとする。詳細な規定については、36ページの[附属書 E 決定的内容モデル](#)を参照。

【妥当性制約】グループ及びパラメタ実体が厳密な入れ子をなしていること

パラメタ実体の置換テキストは、かっこで囲まれたグループによって、厳密な入れ子を構成しなければならない。つまり、[選択](#)、[列](#)又は[混在部品](#)に、開きかっこ又は閉じかっこのいずれか一方がパラメタ実体の置換テキストに含まれれば、他方も同じ置換テキストに含まなければならない。

[相互運用性のため](#)には、パラメタ実体参照が [選択](#)、[列](#)又は [混在内容](#)内容に含まれれば、その置換テキストは空でないことが望ましく、置換テキストの先頭及び末尾の空白でない文字は、コネクタ(|又は,)でない方がよい。

要素内容モデルのいくつかの例を次に示す。

```
<!ELEMENT spec (front, body, back?)>
<!ELEMENT div1 (head, (p | list | note)*, div2*)>
<!ELEMENT dictionary-body (%div.mix; | %dict.mix;)*>
```

### 3.2.2. 混合内容

ある要素型の要素内に、子要素に混在して文字データが含まれる可能性があるとき、その要素型は、混合内容をもつという。この場合、子要素の型についての制約が存在してもよいが、子要素の順序又は出現回数についての制約は存在しない。

```
[例] Mixed ::= '(S? '#PCDATA' (S? '| S? Name)* S?)'*
 | '(S? '#PCDATA' S?)'
```

ここで、Nameは子として出現してもよい要素の型を示す。

【妥当性制約】要素型の重複の禁止  
一つの混合内容宣言内に、同じ名前が複数回出現してはならない。

混合内容宣言の例を次に示す。

```
<!ELEMENT p (#PCDATA|a|ul|b|i|em)*>
<!ELEMENT p (#PCDATA | %font; | %phrase; | %special; | %form;)* >
<!ELEMENT b (#PCDATA)>
```

## 3.3. 属性リスト宣言

属性は、名前及び値の対を要素に関連付けるために用いる。属性指定は、開始タグ又は空要素タグ内でだけ可能とする。したがって、属性指定を認識するための生成規則は、12ページの§3.1 開始タグ、終了タグ及び空要素タグに示されている。属性リスト宣言は、次の目的で用いる。

- a) ある要素型に適用する属性の集合を規定する。
- b) 属性への型制約を設定する。
- c) 属性のデフォルト値を規定する。

属性リスト宣言は、ある要素型と関連付けられた各属性に対し、名前、データ型及び(存在すれば)デフォルト値を規定する。

```
[例] AttlistDecl ::= '<!ATTLIST' S Name AttDef* S? '>'
```

```
[例] AttDef ::= S Name S AttType S DefaultDecl
```

AttlistDecl規則に含まれるNameは、要素型の名前とする。ユーザのオプション指定によっては、宣言していない要素型に対して属性を宣言したならば、XMLプロセサは、警告を出してもよい。しかし、これはエラーとはしない。AttDef規則におけるNameは、属性の名前とする。

ある要素に対して、複数のAttlistDeclを与える場合、これらすべての内容はマージする。ある要素型の同じ属性に、複数の定義を与える場合には、最初の宣言を有効とし、他の宣言は無視する。相互運用性のためには、DTDの作成者は、ある要素型には高々一つの属性リスト宣言しか与えない、ある属性名には高々一つの属性定義しか与えない、及びすべての属性リスト宣言には少なくとも一つの属性定義を与える、という選択をしてもよい。相互運用性のためには、

XMLプロセサは、ユーザのオプション指定によっては、ある要素型に複数の属性リスト宣言を与えたり、ある属性に複数の属性定義を与えたりしたときに、警告を出してもよい。しかし、これは、エラーとはしない。

### 3.3.1. 属性の型

XMLの属性の型は、3種類とする。これらは、文字列型、トークン化型及び列挙型とする。文字列型は、値として任意のリテラル文字列をとる。トークン化型は、字句及び意味に関して、次に示す様々な制約をもつ。

Ⓔ AttType ::= StringType | TokenizedType | EnumeratedType

Ⓔ StringType ::= 'CDATA'

Ⓔ TokenizedType ::= 'ID'  
| 'IDREF'  
| 'IDREFS'  
| 'ENTITY'  
| 'ENTITIES'  
| 'NMTOKEN'  
| 'NMTOKENS'

#### 【妥当性制約】ID

ID型の値は、生成規則Nameにマッチしなければならない。一つのXML文書内では、一つの名前が、この型の値として複数回現れてはならない。つまり、IDの値は、要素を一意に特定しなければならない。

#### 【妥当性制約】1要素ごとに一つのID

要素型は、複数のID属性をもってはならない。

#### 【妥当性制約】ID属性のデフォルト

ID属性は、デフォルトとして、#IMPLIED又は#REQUIREDを宣言しなければならない。

#### 【妥当性制約】IDREF

IDREF型の値は、生成規則Nameにマッチしなければならない。IDREFS型の値は、Namesにマッチしなければならない。各々のNameは、XML文書内に存在する要素のID属性の値とマッチしなければならない。つまり、IDREFの値は、あるID属性の値とマッチしなければならない。

#### 【妥当性制約】実体名

ENTITY型の値は、Name生成規則にマッチしなければならない。ENTITIES型の値は、Namesにマッチしなければならない。各々のNameは、DTDで宣言する解析対象外実体とマッチしなければならない。

**【妥当性制約】名前トークン**

NMTOKEN型の値は、Nmtoken生成規則にマッチしなければならない。NMTOKENS型の値は、にマッチしなければならない。

列挙型の属性は、宣言した幾つかの値の一つを取ることができる。列挙型には、2種類ある。

- ```

[57] Enumerated- ::= NotationType | Enumeration
      Type
[58] NotationType ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
[59] Enumeration ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'

```

NOTATION属性は、その属性が付与されている要素を解釈するのに使用する記法を特定する。記法は、DTD内で宣言され、システム識別子及び公開識別子のどちらか、又は、両方に関連付けられる。

【妥当性制約】記法属性

この型の値は、宣言に含まれる幾つかの記法の名前の一つとマッチしなければならない。つまり、宣言に含まれる記法名は、すべて宣言されていないといけない。

【妥当性制約】列挙

この型の値は、宣言に含まれる幾つかのNmtokenトークンの一つとマッチしなければならない。

相互運用性のためには、同じNmtokenは、一つの要素型のいくつかの列挙型の属性として、複数回現れない方がよい。

3.3.2. 属性のデフォルト

属性宣言は、属性の指定が必須かどうかについての情報を与える。必須でない場合には、文書内で属性が指定されていないとき、XMLプロセサがどう処理しなければならないか又は処理するほうがいいのかの情報も与える。

- ```

[60] DefaultDecl ::= '#REQUIRED' | '#IMPLIED'
 | (('#FIXED' S)? AttValue)

```

属性宣言において、#REQUIREDはその属性が必須であること、#IMPLIEDはデフォルト値がないことを意味する。宣言が#REQUIREDでも#IMPLIEDでもないときには、AttValueの値が、デフォルト値となる。#FIXEDキーワードは、その属性の値がデフォルト値と常に同一でなければならないことを示す。デフォルト値を宣言している場合、この属性が省略されているのを見つけたなら、宣言したデフォルト値が属性値に指定しているとして、XMLプロセサは振舞うものとする。

**【妥当性制約】必須属性**

デフォルトの宣言が#REQUIREDキーワードの場合、属性リスト宣言で参照した要素型のすべての要素で、その属性を指定しなければならない。

【妥当性制約】属性デフォルトの正しさ  
宣言したデフォルト値は、宣言した属性型の字句制約を満たさなければならない。

【妥当性制約】固定の属性デフォルト  
属性が#FIXEDキーワードで宣言されたデフォルト値を持つ場合、その属性のインスタスはデフォルト値にマッチしなければならない。

属性リスト宣言の例を、次に示す。

```
<!ATTLIST termdef
 id ID #REQUIRED
 name CDATA #IMPLIED>
<!ATTLIST list
 type (bullets|ordered|glossary) "ordered">
<!ATTLIST form
 method CDATA #FIXED "POST">
```

### 3.3.3. 属性値の正規化

XMLプロセサは、属性値をアプリケーションに渡す前、または、妥当性を判定する前に、次のとおり正規化しなければならない。

[訳注(これは原文にはない)]以下の箇条書きは、“と”の間(もしくは‘と’の間)の文字の並びに対して繰り返し実行される条件分岐である。一度実行されるたびに、正規化された文字列が先頭から少しずつ構築されていく。

[訳注(これは原文にはない)] 2.11で述べたように、文字を読み込むルーチンがCR+LFやCRをLFに置き換えているものと考えると理解しやすい。

- 属性値の中の文字参照は、参照される文字で置き換えることによって処理する。
- 実体参照は、実体の置換テキストを再帰的に処理する。
- 属性値の中のスペース文字(#x20, #xD, #xA, #x9)は、#x20で置き換えることによって処理する。ただし、外部解析対象実体の一部、または、内部解析対象実体のリテラルリテラル実体値の中にCR(#xD)とLF(#xA)とが連続して現れる場合は、一つの#x20で置き換える。
- 他の文字は、正規化した値の中にそのまま置く。

さらに、属性の型がCDATAでない場合は、XMLプロセサは正規化された属性値に対して、次の処理をしなければならない。まず、先頭または末尾にあるスペース文字(#x20)をすべて取り除く。つぎに、連続するスペース文字(#x20)を一つのスペース文字(#x20)に置き換える。

妥当性を検証しないパーサは、宣言が見つからない属性は、すべて、CDATAを宣言しているとして扱うものとする。

## 3.4. 条件付きセクション

条件付きセクションとは、[文書型宣言の外部サブセット](#)の一部であって、制御キーワードの指定によって、DTDの論理構造に含めたり、除いたりする部分とする。

⌘ conditionalSect ::= includeSect | ignoreSect

- ❷ includeSect ::= '<![ S? 'INCLUDE' S? '[' extSubsetDecl ']]>'
- ❸ ignoreSect ::= '<![ S? 'IGNORE' S? '[' ignoreSectContents\* ']]>'
- ❹ ignoreSectCon- ::= Ignore ('<![ ignoreSectContents ']]>' Ignore)\*
 tents
- ❺ Ignore ::= Char\* - (Char\* ('<![ | ']]>') Char\*)

条件付きセクションは、DTDの内部サブセット及び外部サブセットと同様に、完全な宣言、コメント、処理命令又は入れ子になった条件付きセクションを、いくつか含んでよい。これらの間に、空白が現れてもよい。

条件付きセクションのキーワードがINCLUDEならば、条件付きセクションの内容はDTDの一部である。条件付きセクションのキーワードがIGNOREならば、条件付きセクションの内容は論理的にはDTDの一部ではない。構文解析を正しく行うためには、無視する条件付きセクション(IGNORE)に関しても、内容を読まなければならないことに注意すること。これは、入れ子になった条件付きセクションを見つけ、(無視する)最も外側の条件付きセクションを正しく検出することを目的とする。キーワードをINCLUDEとする小さな条件付きセクションが、キーワードをIGNOREとするより大きな条件付きセクションに含まれるならば、外側及び内側の条件付きセクションの両方とも無視する。

条件付きセクションのキーワードがパラメタ実体参照ならば、XMLプロセサは条件付きセクションの扱いを判断する前に、このパラメタ実体を展開しなければならない。

例を次に示す。

```

<!ENTITY % draft 'INCLUDE' >
<!ENTITY % final 'IGNORE' >

<![%draft;[
<!ELEMENT book (comments*, title, body, supplements?)>
]]>
<![%final;[
<!ELEMENT book (title, body, supplements?)>
]]>

```

## 4. 物理構造

XML文書は、一つ以上の記憶単位から構成する。この記憶単位を、実体という。実体は、内容をもち、文書実体(以降参照)及び外部DTDサブセットを除いて、名前で特定する。各XML文書は、文書実体と呼ぶ実体をもつ。XMLプロセサは、この文書実体から処理を開始する。文書実体が、文書のすべてを含んでもよい。

実体は、解析対象実体又は解析対象外実体とする。解析対象実体の内容は、解析対象実体の置換テキストと呼ぶ。このテキストは、文書の本体の一部として解釈する。

解析対象外実体は、内容がテキストでもそうでなくともよいリソースとする。テキストの場合、XMLでなくともよい。各解析対象外実体には、記法が関連付けられ、この記法は、名前で特定する。XMLプロセサが実体や記法の識別子をアプリケーションに渡すという要件以外は、XMLは解析対象外実体の内容を制限しない。





て、実体 amp, lt, gt, apos, quotを宣言することが望ましい。パラメタ実体の場合は、宣言は、参照に先行しなければならない。同様に、一般実体の場合は、属性リスト宣言のデフォルト値内での参照よりも先に、宣言が現れなければならない。

**【整形制約】解析対象実体**

実体参照は、**解析対象外実体**の名前を含んではならない。解析対象外実体は、ENTITY型又はENTITIES型として宣言した**属性値**としてだけ参照できる。

**【整形制約】再帰なし**

解析対象実体は、それ自体への参照を、直接にも間接にも含んではならない。

**【整形制約】DTDの中**

パラメタ実体参照は、**DTD**内にだけ、出現してよい。

文字参照及び実体参照の例を、次に示す。

```
Type <key>less-than</key> (<) to save options.
This document was prepared on &docdate; and
is classified &security-level;.
```

パラメタ実体参照の例を、次に示す。

```
<!-- declare the parameter entity "ISOLat2"... -->
<!ENTITY % ISOLat2
 SYSTEM "http://www.xml.com/iso/isolat2-xml.entities" >
<!-- ... now reference it. -->
%ISOLat2;
```

## 4.2. 実体宣言

実体は、次のとおりに宣言する。

- [70] EntityDecl ::= GEDecl | PEGecl
- [71] GEDecl ::= '<ENTITY' S Name S EntityDef S? '>'
- [72] PEGecl ::= '<ENTITY' S '%' S Name S PEGecl S? '>'
- [73] EntityDef ::= EntityValue | (ExternalID NDataDecl?)
- [74] PEGecl ::= EntityValue | ExternalID

**Name** は、**実体参照**において実体を特定する。解析対象外実体ならば、ENTITY型又はENTITIES型の属性値内で、実体を特定する。同一の実体が一回以上宣言されれば、最初の宣言を用いる。ユーザのオプション指定によっては、複数回宣言される実体に関し、XMLプロセサは、警告を出してもよい。

### 4.2.1. 内部実体

実体の定義が `EntityValue` のとき、これを内部実体という。これは、別個の物理的記憶単位をもたず、実体の内容は宣言内で与える。正しく置換テキストを生成するには、リテラル実体値内での実体参照及び文字参照の処理が必要となるかもしれないことに注意する。詳細は、27ページの § 4.5 内部実体置換テキストの構築を参照。

内部実体は、解析対象実体とする。

内部実体宣言の例を、次に示す。

```
<!ENTITY Pub-Status "This is a pre-release of the specification.">
```

### 4.2.2. 外部実体

内部実体でない実体は外部実体であって、次のとおりに宣言する。

```
[7] ExternalID ::= 'SYSTEM' S SystemLiteral
 | 'PUBLIC' S PubidLiteral S SystemLiteral
```

```
[7] NDataDecl ::= S 'NDATA' S Name
```

`NDataDecl` が存在すれば、この実体は、一般解析対象外実体とし、そうでなければ、解析対象実体とする。

【妥当性制約】記法が宣言されていること

`Name` は、宣言した記法の名前とマッチしなければならない。

`SystemLiteral` を、実体のシステム識別子と呼ぶ。システム識別子としてはURIを使い、その実体の内容を取り出すために用いてもよい。URIと共に使うことの多いシャープ記号(#)及びフラグメント識別子は、正式には、URI自体の一部ではない。フラグメント識別子が、システム識別子の部分として与えられている場合、XMLプロセサは、エラーを出してもよい。この標準情報(TR)の適用範囲外の情報(例えば、ある特定のDTDの特別なXML要素又は特定のアプリケーションの仕様によって定義された処理命令)によって上書きされない限り、相対的なURIは、その実体の位置、すなわち、その実体の宣言があるファイルに相対的とする。したがって、そのURIは、文書実体、外部DTDサブセットを含む実体、又は、いくつかの外部パラメタ実体に対して、相対的である。

XMLプロセサは、非ASCII文字がURIに含まれている場合、これを次のとおりに扱う。非ASCII文字は、UTF-8によって一つ以上のバイトで表現し、これらのバイトをURIの別扱い機構を用いて(すなわち、バイト値の16進数による表現をHHとしたとき、各バイトを%HHの形式に変換することによって)別扱いする。

[訳注(これは原文にはない)] この規定はHTML 4.0の附属書Bと RFC 2141"URN Syntax", R. Moats, May 1997にあるものである。

システム識別子以外に、外部実体は、公開識別子を含んでもよい。実体の内容を取り出すXMLプロセサは、この公開識別子を用いて、代替りのURIの生成を試みてもよい。XMLプロセサがこれに失敗した場合は、システムリテラルとして指定したURIを用いなければならない。マッチする前に、公開識別子内にある空白文字からなる文字列は、すべて単一のスペース文字(#x20)に正規化しなければならない。先頭及び末尾の空白文字はすべて削除しなければならない。

外部実体宣言の例を、次に示す。

```
<!ENTITY open-hatch
 SYSTEM "http://www.textuality.com/boilerplate/OpenHatch.xml">
<!ENTITY open-hatch
 PUBLIC "-//Textuality//TEXT Standard open-hatch boilerplate//EN"
 "http://www.textuality.com/boilerplate/OpenHatch.xml">
<!ENTITY hatch-pic
 SYSTEM "../grafix/OpenHatch.gif"
 NDATA gif >
```

## 4.3. 解析対象実体

### 4.3.1. テキスト宣言

外部解析対象実体は、テキスト宣言で始まってよい。

[77] TextDecl ::= '<?xml' VersionInfo? EncodingDecl S? '?>'

テキスト宣言は、そのままの形で現れなければならない。解析対象実体への参照を経由してはならない。外部解析対象実体において、テキスト宣言は、先頭以外のいかなる位置にも出現しない。

### 4.3.2. 整形式の解析対象実体

ラベルdocumentをもつ生成規則にマッチすれば、文書実体は整形式とする。ラベルextParsedEntをもつ生成規則にマッチすれば、外部の一般解析対象実体は、整形式とする。ラベルextPEをもつ生成規則にマッチすれば、外部パラメタ実体は整形式とする。

[78] extParsedEnt ::= TextDecl? content

[79] extPE ::= TextDecl? extSubsetDecl

置換テキストが、ラベルcontentをもつ生成規則にマッチすれば、内部の一般解析対象実体は、整形式とする。すべての内部のパラメタ実体は、定義から整形式になる。

実体はすべて整形式なので、XML文書の論理的及び物理的構造は、厳密に入れ子となる。開始タグ、終了タグ、空要素タグ、要素、コメント、処理命令、文字参照及び実体参照が、一つの実体で開始し、別の実体で終了することはない。

### 4.3.3. 実体における文字符号化

XML文書内の外部解析対象実体は、それぞれ別の文字符号化方式を用いてもよい。すべてのXMLプロセサは、UTF-8で符号化した実体、及びUTF-16で符号化した実体进行处理できなければならない。

[訳注(原文にはない)]原文ではeither?orであるが意識した。

UTF-16で符号化した実体は、ISO/IEC 10646の附属書E及びUnicodeの付録Bで規定するバイト順マーク(ZERO WIDTH NO-BREAK SPACE文字、#xFEFF)で始まらなければならない。これは、符号化方式の標識であって、XML文書のマーク付けの一部でも、文字データの一部でもない。XMLプロセサは、UTF-8で符号化した文書とUTF-16で符号化した文書との区別を行うために、この文字を認識可能でなければならない。

XMLプロセサは、UTF-8及びUTF-16で符号化した実体を読めなければならない、他の符号化方式が世界では用いられることも多く、それらの符号化方式を用いる実体をXMLプロセサは処理できることが望ましい。UTF-8又はUTF-16以外の符号化方式を用いて格納する解析対象実体は、符号化宣言を含む[テキスト宣言](#)で始めなければならない。

```
[R] EncodingDecl ::= S 'encoding' Eq ("" EncName "" | "" EncName "")
[R] EncName ::= [A-Za-z] ([A-Za-z0-9._] | '-')* /* ラテン文字だけを含む符号化方式名 */
```

文書実体では、符号化宣言は、XML宣言の一部とする。EncNameは、使用する符号化方式の名前とする。

符号化宣言では、値"UTF-8"、"UTF-16"、"ISO-10646-UCS-2"及び"ISO-10646-UCS-4"は、Unicode及びISO/IEC 10646の各種符号化方式のために用いる。値"ISO-8859-1"から"ISO-8859-9"までは、ISO 8859の対応するパートのために用いる。値"ISO-2022-JP"、"Shift\_JIS"及び"EUC-JP"は、JIS X-0208-1997の各種符号化方式のために用いる。XMLプロセサは、ここに挙げた以外の符号化方式を認識してもよい。Internet Assigned Numbers Authority [IANA]に、(charsetsとして)登録された文字符号化方式については、ここに挙げたもの以外についても、登録された名前で参照することが望ましい。これらの登録された名前は、大文字・小文字の区別をせずに定義されているので、これらに対する比較を試みるプロセサは、大文字・小文字の区別をしない方法をとることに注意する。

外部の伝送プロトコル（すなわち、HTTP、MIMEなど）で与えられる情報が存在しないとき、XMLプロセサに渡された実体が、符号化宣言を含むにもかかわらず、宣言で示したものの以外の方式で符号化されている場合、符号化宣言が外部実体の最初以外の位置に出現した場合、又はバイト順マークでも符号化宣言でも始まらない実体が、UTF-8以外の符号化方式を使用した場合は、[エラー](#)とする。ASCIIはUTF-8のサブセットなので、通常のASCIIの実体は厳密には符号化宣言を必要としないことに注意。

処理できない符号化方式を使用した実体をXMLプロセサが発見したときは、[致命的エラー](#)とする。

符号化宣言の例を次に示す。

```
<?xml encoding='UTF-8'?>
<?xml encoding='EUC-JP'?>
```

## 4.4. XMLプロセサによる実体及び参照の扱い

次の表に、文字参照、実体参照及び解析対象外実体の呼出しが現れる文脈、並びに、それぞれの場合におけるXMLプロセサに要求される振舞いを要約する。一番左の列のラベルは、参照が現れる文脈を示す。

### 内容における参照

要素の[開始タグ](#)及び[終了タグ](#)の間の任意の場所での参照。非終端記号contentに対応する。

### 属性値における参照

[開始タグ](#)の属性の値、又は[属性宣言](#)におけるデフォルト値のいずれかでの参照。非終端記号AttValueに対応する。

### 属性値として出現

参照ではなく、**Name**として出現。ENTITY型として宣言した属性の値として出現するか、又はENTITIES型として宣言した属性の値におけるスペースで区切るトークンの一つとして出現する。

### 実体値における参照

実体の宣言における、パラメタ実体又は内部実体の**リテラル実体値**の中での参照。非終端記号EntityValueに対応する。

### DTDにおける参照

DTDの内部サブセット又は外部サブセットでの参照。ただし、EntityValue又はAttValueの外側とする。

|          | 実体の型       |            |            |         | 文字    |
|----------|------------|------------|------------|---------|-------|
|          | パラメタ       | 内部一般       | 外部解析対象実体一般 | 解析対象外実体 |       |
| 内容での参照   | 認識しない      | 取込み        | 検証のために取込み  | 禁止      | 取込み   |
| 属性値での参照  | 認識しない      | リテラル内での取込み | 禁止         | 禁止      | 取込み   |
| 属性値として出現 | 認識しない      | 禁止         | 禁止         | 通知      | 認識しない |
| 実体値での参照  | リテラル内での取込み | 処理しない      | 処理しない      | 禁止      | 取込み   |
| DTDでの参照  | PEとして取込み   | 禁止         | 禁止         | 禁止      | 禁止    |

#### 4.4.1. “認識しない”

DTDの外では、%文字は、いかなる特別な意味ももたない。したがって、DTDの中ではパラメタ実体参照として認識するものであっても、contentの中ではマーク付けとしては認識しない。同様に、適切に宣言した属性の値の中に現れる場合を除き、解析対象外実体の名前は認識しない。

#### 4.4.2. “取込み”

実体参照を処理するには、その**置換テキスト**を取り出し、処理する。参照自体の代わりに、参照があった位置で、文書の一部として含まれるものとして取り込む。置換テキストは、**文字データ**及び(パラメタ実体を除く。)マーク付けのいずれを含んでもよく、これらは、通常の方法で認識されなければならない。ただし、マーク付けの区切り子を別扱いするために用いる実体(amp, lt, gt, apos, quot)の置換テキストは、常にデータとして扱う(文字列"AT&T;"は、"AT&T;"に展開され、残されたアンド記号は、実体参照の区切り子としては認識しない。)。文字参照は、番号で示した文字を参照自体の代わりに取り込む。

#### 4.4.3. “検証のために取込み”

文書の妥当性を**検証**するには、XMLプロセサは解析対象実体への参照を認識したとき、その置換テキストを**取り込ま**なければならない。実体が外部実体であって、XML文書の妥当性を検証

しないときは、実体の置換テキストを取り込んでもよいが、取り込むことを義務づけられてはいない。妥当性を検証しないパーサが置換テキストを取り込まない場合、実体を認識したが、読み込まなかったことをアプリケーションに通知しなければならない。

この取決めは、SGML及びXMLの実体の機構が提供する自動取込み機能が、文書作成時のモジュール化を主な目的として設計されており、その他のアプリケーション(特に、文書のブラウジング)には、必ずしも適切ではない、という認識による。例えば、ブラウザは外部解析対象実体への参照を見つけると、その実体が存在するという表示だけを行い、表示を要求されたときにだけ、内容を取り出すかもしれない。

#### 4.4.4. “禁止”

次は禁止されており、**致命的エラー**とする。

- a) **解析対象外実体**への参照の出現。
- b) DTDの**EntityValue**又は**AttValue**以外の部分における、文字参照又は一般実体への参照の出現。
- c) 属性値内の外部実体への参照の出現。

#### 4.4.5. リテラル内での取込み

**実体参照**が属性値の中で現れたとき、または、パラメタ実体への参照がリテラル実体値の中で現れたとき、**置換テキスト**は、参照自体の代わりに、参照があった位置に文書の一部としてあったものとして処理される。ただし、置換テキストの中の一重引用符又は二重引用符文字は、常に通常の文字データとして扱われ、リテラルを終了させることはない。例えば、次の文書例は整形形式である。

```
<!ENTITY YN "Yes" >
<!ENTITY WhatHeSaid "He said &YN;" >
```

[訳注(これは原文にはない)] 一行目は原規定ではパラメタ実体となっているが明らかな誤りなので修正した。

一方、次の例は整形形式ではない。

```
<!ENTITY EndAttr "27'" >
<element attribute='a-&EndAttr;'>
```

#### 4.4.6. “通知”

**解析対象外実体**の名前が、**ENTITY**型又は**ENTITIES**型の属性値においてトークンとして現れたとき、妥当性を検証するプロセサは、アプリケーションに対して、その実体及び関連する記法の**システム**識別子並びに(存在すれば)**公開**識別子を通知しなければならない。

#### 4.4.7. “処理しない”

一般実体参照が、実体宣言における**EntityValue**内に現れるとき、一般実体参照は処理されないので、そのまま残る。

#### 4.4.8. “PEとして取込み”

外部解析対象実体の場合と同様に、パラメタ実体は、妥当性を検証するときだけ取り込む必要がある。パラメタ実体参照をDTD内に認識して取り込むとき、その置換テキストは、その前後に一つのスペース文字(#x20)の付加によって引き伸ばされる。パラメタ実体の置換テキストがDTD内の文法的トークンを完全に含むようにすることを、この規程は意図している。

### 4.5. 内部実体置換テキストの構築

内部実体の取扱いの規定で、実体値を二つの形式に区別することは役に立つ。リテラル実体値は、実体宣言内に実際に存在する、引用符で囲まれた文字列とする。これは、非終端記号EntityValueとマッチする。置換テキストは、文字参照及びパラメタ実体参照の置換え後における、実体の内容とする。

内部実体宣言内で与えるリテラル実体値(EntityValue)は、文字参照、パラメタ実体参照及び一般実体参照を含んでもよい。これらの参照は、リテラル実体値内に完全に含まれていなければならない。展開する実際の置換テキスト(先に示したものは、参照するパラメタ実体の置換テキストを含み、リテラル実体値内での文字参照の代わりに参照した文字を含む。しかし、一般実体参照はそのまま残し、展開してはならない。例えば、次の宣言を与えたとする。

```
<!ENTITY % pub "Éditions Gallimard" >
<!ENTITY rights "All rights reserved" >
<!ENTITY book "La Peste: Albert Camus,
© 1947 %pub;. &rights;" >
```

実体の置換テキスト"book"は、次のとおりとなる。

```
La Peste: Albert Camus,
© 1947 Éditions Gallimard. &rights;
```

参照"&book;"が文書の内容又は属性値内に出現すれば、一般実体参照"&rights;"は展開される。

これらの単純な規則は、複雑な相互作用をもちうる。難しい例についての詳細は、附属書35ページの附属書 D 実体参照及び文字参照の展開を参照のこと。

### 4.6. 定義済み実体

不等号(小なり)、アンド記号及び他の区切り子を別扱いするには実体参照及び文字参照のどちらも使用できる。いくつかの一般実体 ( amp, lt, gt, apos, quot ) をこの目的のために使用する。番号による文字参照も、この目的のために使用できる。文字参照は、認識されると直ちに展開され、文字データとして扱われるので、番号による文字参照"&#60;"及び"&#38;"は、文字データ内に出現する<及び&を別扱いするために使用できる。

すべてのXMLプロセサは、宣言されているかどうかに関係なく、これらの実体を認識しなくてはならない。相互運用性のためには、妥当なXML文書は、これらの実体を使用する前に他の実体と同様に宣言する。実体を宣言する場合は、別扱いする1文字を置換テキストとして指定した内部実体、又は、その文字への文字参照を指定した内部実体として、次のとおりに宣言しなければならない。

```
<!ENTITY lt "&#60;">
<!ENTITY gt ">">
```

```
<!ENTITY amp "&#38;">
<!ENTITY apos "'">
<!ENTITY quot """>
```

lt及びampの宣言内の"<"及び"&"文字は、実体の置換テキストが整形形式となるように二重に別扱いされることに注意。

## 4.7. 記法宣言

記法は、[解析対象外実体](#)の形式、記法属性を持つ要素の形式、または、[処理命令](#)の対象とするアプリケーションを特定する名前とする。

記法宣言は、記法の名前及び外部識別子を提供する。この名前は、外部実体宣言、属性リスト宣言、及び属性指定に用いる。外部識別子は、与えられた記法のデータを処理できるソフトウェア（ヘルパアプリケーションなど）を、XMLプロセッサ又はクライアントアプリケーションが探すために利用できる。

```
Ⓝ NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S?
 '>'
```

```
Ⓝ PublicID ::= 'PUBLIC' S PubidLiteral
```

XMLプロセッサは、宣言されていて、属性値、属性定義又は実体宣言で参照されているすべての記法について、XMLプロセッサは、記法の名前及び外部識別子をアプリケーションに提供しなければならない。さらに、外部識別子を、[システム識別子](#)、ファイル名又はその他の情報に展開してもよく、これらを用いて、アプリケーションは、その記法のデータを処理するプロセッサを起動する。しかし、XMLプロセッサ又はアプリケーションが動作するシステムでは利用できない記法を、XML文書が宣言し参照しても、これは、エラーとはしない。

## 4.8. 文書実体

文書実体は、実体の成す木構造のルートであって、[XMLプロセッサ](#)が処理を開始する対象とする。この標準情報(TR)は、XMLプロセッサが、文書実体の存在する場所をどのように見つけるかは規定しない。他の実体と異なり、文書実体は名前をもたず、いかなる識別もなしにプロセッサへの入力ストリームに出現してもよい。

# 5. 適合性

## 5.1. 妥当性を検証するプロセッサ及び検証しないプロセッサ

適合XMLプロセッサは、妥当性を検証するもの及び妥当性を検証しないものの二つに分類される。

妥当性を検証するプロセッサも妥当性を検証しないプロセッサも、読み込んだ文書実体及び他のすべての解析対象実体において、この標準情報(TR)の整形形式制約への違反を報告しなければならない。

妥当性を検証するプロセッサは、[DTD](#)内の宣言によって示された制約への違反と、この標準情報(TR)が規定する妥当性制約への違反とを、すべて報告しなければならない。 これを実現するために、妥当性を検証するXMLプロセッサは、DTD全体と文書内で参照されているすべての外部解析対象実体とを読み込んで処理しなければならない。



妥当性を検証しないプロセサは、整形形式であることを確認するために、DTDの内部サブセット全体を含めた文書実体を調べることだけが義務づけられている。文書の妥当性を確認する必要はないが、読み込んでいないパラメタ実体への参照が最初に起きるまでに読み込んだDTDの内部サブセットとパラメタ実体とに現れるすべての宣言を処理しなければならない。すなわち、属性値を正規化し、内部実体の置換テキストを取込み、デフォルトの属性値を与えるために、これらの宣言にある情報を使用しなければならない。実体の宣言は上書きされる可能性があるため、妥当性を検証しないプロセサは、読み込んでいないパラメタ実体への参照より後に現れた実体宣言及び属性リスト宣言を処理してはならない。

## 5.2. XMLプロセサの使用

妥当性を検証するXMLプロセサの振舞いはほとんど予測可能である。すなわち、文書のすべての断片を読み込み、整形形式及び妥当性に対するすべての違反を報告しなければならない。妥当性を検証しないプロセサに必要とされることはそれより少ない。すなわち、文書実体以外の文書の断片を読み込む必要はない。したがって、XMLプロセサのユーザに対して重要な二つ効果をもつ。

- ある種の整形形式のエラー、特に、外部実体を読まなければ検出できないエラーは、妥当性を検証しないプロセサでは検出しなくてもよい。例えば、24ページの§ 4.4 XMLプロセサによる実体及び参照の扱いで禁止として説明されているいくつかの場合、並びに実体が宣言されていること、解析対象実体、及び再帰なしという見出しが付けられた制約が挙げられる。
- プロセサからアプリケーションに渡される情報は、プロセサがパラメタ実体及び外部実体を読み込むかどうかで違って来る。例えば、妥当性を検証しないプロセサは、属性値を正規化したり、内部実体の置換テキストを取込んだり、デフォルトの属性値を与えたりする必要はない。これらを行なうかどうかは、外部実体及びパラメタ実体内での宣言を既に読み込んでいるかどうかによる。

異なるXMLプロセサ間での相互運用性を最も高めるためには、妥当性を検証しないプロセサを使用するアプリケーションは、そのようなプロセサでは必要とされない振舞いに依存すべきではない。外部実体で宣言されている属性のデフォルトや内部実体を使用するような場合は、妥当性を検証するプロセサを使用する。

## 6. 表記法

XMLの形式的な文法は、簡単な拡張Backus-Naur Form(EBNF)表記法によって与える。文法の各規則は、次の形式で記号を定義する。

```
symbol ::= expression
```

記号は、正規表現で定義するときは大文字で始め、そうでなければ小文字で始める。リテラル文字列は引用符で囲む。

規則の右辺では、一つ以上の文字からなる文字列とマッチするために、次の式を使用する。

**#xN**

ここで、Nは16進の整数とする。ISO/IEC 10646の文字であって、標準形の(UCS-4)コード値を符号なし2進数として解釈したとき、指定した値と等しいものとマッチする。#xN形式の先頭にゼロがいくつか現れるかは意味をもたない。コード値における先頭のゼロの数は、文字の符号化によって決定されるのでXMLにとっては意味がない。

[a-zA-Z], [#xN-#xN]

指定した範囲の値(両端の値を含む。 )をもつ任意の文字とマッチする。

[^a-z], [^#xN-#xN]

指定した範囲外の値をもつ任意の文字とマッチする。

[^abc], [^#xN#xN#xN]

指定した文字以外の値をもつ任意の文字とマッチする。

"string"

二重引用符で囲むリテラル文字列とマッチするリテラル文字列とマッチする。

'string'

一重引用符で囲むリテラル文字列とマッチするリテラル文字列とマッチする。

これらの記号は、次の形式の組合せで使用する。ここで、A及びBは式とする。

(expression)

ここに示す組合せによる式(expression)を、一つのまとまりとして扱うために使う。

A?

Aのオプションな出現とマッチする(オプションのA)。

A B

Aの次にBが出現するものとマッチする。

A | B

A又はBのどちらかとマッチする。

A - B

AとマッチするがBとはマッチしない任意の文字列とマッチする。

A+

Aの1回以上の繰返しとマッチする。

A\*

Aの0回以上の繰返しとマッチする。

生成規則内で使用する他の記法を次に示す。

/\* ... \*/

コメント。

[ wfc: ... ]

整形形式制約。生成規則に関連した、整形形式の文書に関する制約を名前によって特定する。

[ vc: ... ]

妥当性制約。生成規則に関連した、妥当な文書に関する制約を名前によって特定する。

## 附属書 A. 参考文献

### A.1. 規定の参考文献

#### *IANA*

(Internet Assigned Numbers Authority) Official Names for Character Sets, ed. Keld Simonsen et al. <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>を参照。

#### *IETF RFC 1766*

IETF (Internet Engineering Task Force). RFC 1766: Tags for the Identification of Languages, ed. H. Alvestrand. 1995.

#### *ISO 639*

(International Organization for Standardization). ISO 639:1988 (E). Code for the representation of names of languages. [Geneva]: International Organization for Standardization, 1988.

#### *ISO 3166*

(International Organization for Standardization). ISO 3166-1:1997 (E). Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes [Geneva]: International Organization for Standardization, 1997.

#### *ISO/IEC 10646*

ISO (International Organization for Standardization). ISO/IEC 10646-1993 (E). Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane. [Geneva]: International Organization for Standardization, 1993 (plus amendments AM 1 through AM 7).

#### *Unicode*

The Unicode Consortium. The Unicode Standard, Version 2.0. Reading, Mass.: Addison-Wesley Developers Press, 1996.

### A.2. 他の参考文献

#### *Aho/Ullman*

Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. Reading: Addison-Wesley, 1986, rpt. corr. 1988.

#### *Berners-Lee et al.*

Berners-Lee, T., R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax and Semantics. 1997. (Work in progress; see updates to RFC1738.)

#### *Brüggemann-Klein*

Brüggemann-Klein, Anne. Regular Expressions into Finite Automata. Extended abstract in I. Simon, Hrsg., LATIN 1992, S. 97-98. Springer-Verlag, Berlin 1992. Full Version in Theoretical Computer Science 120: 197-213, 1993.

*Brüggemann-Klein and Wood*

Brüggemann-Klein, Anne, and Derick Wood. Deterministic Regular Languages. Universität Freiburg, Institut für Informatik, Bericht 38, Oktober 1991.

*Clark*

James Clark. Comparison of SGML and XML. <http://www.w3.org/TR/NOTE-sgml-xml-971215>を参照。

*IETF RFC1738*

IETF (Internet Engineering Task Force). RFC 1738: Uniform Resource Locators (URL), ed. T. Berners-Lee, L. Masinter, M. McCahill. 1994.

*IETF RFC1808*

IETF (Internet Engineering Task Force). RFC 1808: Relative Uniform Resource Locators, ed. R. Fielding. 1995.

*IETF RFC2141*

IETF (Internet Engineering Task Force). RFC 2141: URN Syntax, ed. R. Moats. 1997.

*ISO 8879*

ISO (International Organization for Standardization). ISO 8879-1986 (E). Information processing -- Text and Office Systems -- Standard Generalized Markup Language (SGML). First edition -- 1986-10-15. [Geneva]: International Organization for Standardization, 1986.

*ISO/IEC 10744*

ISO (International Organization for Standardization). ISO/IEC 10744-1992 (E). Information technology -- Hypermedia/Time-based Structuring Language (HyTime). [Geneva]: International Organization for Standardization, 1992. Extended Facilities Annex. [Geneva]: International Organization for Standardization, 1996.

## 附属書 B. 文字クラス

Unicode標準に定義するプロパティにしたがって、文字は基底文字(BaseChar)(これらは、発音符号を除くラテンアルファベットのアルファベット文字を含む。)、統合漢字(ideographic)及び結合文字(CombiningChar)(このクラスはほとんどの発音符号を含む。)にクラス分けする。これらのクラスを合わせて字(Letter)のクラスとする。10進数値(Digit)及びエクステンダ(Extender)のクラスもある。

Ⓕ Letter ::= BaseChar | Ideographic

Ⓖ BaseChar ::= [#x0041-#x005A] | [#x0061-#x007A] | [#x00C0-#x00D6] | [#x00D8-#x00F6] | [#x00F8-#x00FF] | [#x0100-#x0131] | [#x0134-#x013E] | [#x0141-#x0148] | [#x014A-#x017E] | [#x0180-#x01C3] | [#x01CD-#x01F0] | [#x01F4-#x01F5] | [#x01FA-#x0217] | [#x0250-#x02A8] | [#x02BB-#x02C1] | #x0386 | [#x0388-#x038A] | #x038C | [#x038E-#x03A1] | [#x03A3-#x03CE] | [#x03D0-#x03D6] | #x03DA | #x03DC | #x03DE | #x03E0 | [#x03E2-

#x03F3] | [#x0401-#x040C] | [#x040E-#x044F]  
| [#x0451-#x045C] | [#x045E-#x0481] | [#x0490-  
#x04C4] | [#x04C7-#x04C8] | [#x04CB-#x04CC]  
| [#x04D0-#x04EB] | [#x04EE-#x04F5] | [#x04F8-  
#x04F9] | [#x0531-#x0556] | #x0559 | [#x0561-  
#x0586] | [#x05D0-#x05EA] | [#x05F0-#x05F2]  
| [#x0621-#x063A] | [#x0641-#x064A] | [#x0671-  
#x06B7] | [#x06BA-#x06BE] | [#x06C0-#x06CE]  
| [#x06D0-#x06D3] | #x06D5 | [#x06E5-#x06E6]  
| [#x0905-#x0939] | #x093D | [#x0958-#x0961]  
| [#x0985-#x098C] | [#x098F-#x0990] | [#x0993-  
#x09A8] | [#x09AA-#x09B0] | #x09B2 | [#x09B6-  
#x09B9] | [#x09DC-#x09DD] | [#x09DF-#x09E1]  
| [#x09F0-#x09F1] | [#x0A05-#x0A0A] | [#x0A0F-  
#x0A10] | [#x0A13-#x0A28] | [#x0A2A-#x0A30]  
| [#x0A32-#x0A33] | [#x0A35-#x0A36] | [#x0A38-  
#x0A39] | [#x0A59-#x0A5C] | #x0A5E | [#x0A72-  
#x0A74] | [#x0A85-#x0A8B] | #x0A8D | [#x0A8F-  
#x0A91] | [#x0A93-#x0AA8] | [#x0AAA-#x0AB0]  
| [#x0AB2-#x0AB3] | [#x0AB5-#x0AB9] | #x0ABD  
| #x0AE0 | [#x0B05-#x0B0C] | [#x0B0F-#x0B10]  
| [#x0B13-#x0B28] | [#x0B2A-#x0B30] | [#x0B32-  
#x0B33] | [#x0B36-#x0B39] | #x0B3D | [#x0B5C-  
#x0B5D] | [#x0B5F-#x0B61] | [#x0B85-#x0B8A]  
| [#x0B8E-#x0B90] | [#x0B92-#x0B95] | [#x0B99-  
#x0B9A] | #x0B9C | [#x0B9E-#x0B9F] | [#x0BA3-  
#x0BA4] | [#x0BA8-#x0BAA] | [#x0BAE-#x0BB5]  
| [#x0BB7-#x0BB9] | [#x0C05-#x0C0C] | [#x0C0E-  
#x0C10] | [#x0C12-#x0C28] | [#x0C2A-#x0C33]  
| [#x0C35-#x0C39] | [#x0C60-#x0C61] | [#x0C85-  
#x0C8C] | [#x0C8E-#x0C90] | [#x0C92-#x0CA8]  
| [#x0CAA-#x0CB3] | [#x0CB5-#x0CB9] | #x0CDE  
| [#x0CE0-#x0CE1] | [#x0D05-#x0D0C] | [#x0D0E-  
#x0D10] | [#x0D12-#x0D28] | [#x0D2A-#x0D39]  
| [#x0D60-#x0D61] | [#x0E01-#x0E2E] | #x0E30  
| [#x0E32-#x0E33] | [#x0E40-#x0E45] | [#x0E81-  
#x0E82] | #x0E84 | [#x0E87-#x0E88] | #x0E8A  
| #x0E8D | [#x0E94-#x0E97] | [#x0E99-#x0E9F]  
| [#x0EA1-#x0EA3] | #x0EA5 | #x0EA7 | [#x0EAA-  
#x0EAB] | [#x0EAD-#x0EAE] | #x0EB0 | [#x0EB2-  
#x0EB3] | #x0EBD | [#x0EC0-#x0EC4] | [#x0F40-  
#x0F47] | [#x0F49-#x0F69] | [#x10A0-#x10C5]  
| [#x10D0-#x10F6] | #x1100 | [#x1102-#x1103]  
| [#x1105-#x1107] | #x1109 | [#x110B-#x110C]  
| [#x110E-#x1112] | #x113C | #x113E | #x1140  
| #x114C | #x114E | #x1150 | [#x1154-#x1155]  
| #x1159 | [#x115F-#x1161] | #x1163 | #x1165  
| #x1167 | #x1169 | [#x116D-#x116E] | [#x1172-  
#x1173] | #x1175 | #x119E | #x11A8 | #x11AB  
| [#x11AE-#x11AF] | [#x11B7-#x11B8] | #x11BA  
| [#x11BC-#x11C2] | #x11EB | #x11F0 | #x11F9  
| [#x1E00-#x1E9B] | [#x1EA0-#x1EF9] | [#x1F00-  
#x1F15] | [#x1F18-#x1F1D] | [#x1F20-#x1F45]  
| [#x1F48-#x1F4D] | [#x1F50-#x1F57] | #x1F59

| #x1F5B | #x1F5D | [#x1F5F-#x1F7D] | [#x1F80-  
#x1FB4] | [#x1FB6-#x1FBC] | #x1FBE | [#x1FC2-  
#x1FC4] | [#x1FC6-#x1FCC] | [#x1FD0-#x1FD3]  
| [#x1FD6-#x1FDB] | [#x1FE0-#x1FEC] | [#x1FF2-  
#x1FF4] | [#x1FF6-#x1FFC] | #x2126 | [#x212A-  
#x212B] | #x212E | [#x2180-#x2182] | [#x3041-  
#x3094] | [#x30A1-#x30FA] | [#x3105-#x312C]  
| [#xAC00-#xD7A3]

☞ Ideographic ::= [#x4E00-#x9FA5] | #x3007 | [#x3021-#x3029]

☞ CombiningChar ::= [#x0300-#x0345] | [#x0360-#x0361] | [#x0483-  
#x0486] | [#x0591-#x05A1] | [#x05A3-#x05B9]  
| [#x05BB-#x05BD] | #x05BF | [#x05C1-#x05C2]  
| #x05C4 | [#x064B-#x0652] | #x0670 | [#x06D6-  
#x06DC] | [#x06DD-#x06DF] | [#x06E0-#x06E4]  
| [#x06E7-#x06E8] | [#x06EA-#x06ED] | [#x0901-  
#x0903] | #x093C | [#x093E-#x094C] | #x094D  
| [#x0951-#x0954] | [#x0962-#x0963] | [#x0981-  
#x0983] | #x09BC | #x09BE | #x09BF | [#x09C0-  
#x09C4] | [#x09C7-#x09C8] | [#x09CB-#x09CD]  
| #x09D7 | [#x09E2-#x09E3] | #x0A02 | #x0A3C  
| #x0A3E | #x0A3F | [#x0A40-#x0A42] | [#x0A47-  
#x0A48] | [#x0A4B-#x0A4D] | [#x0A70-#x0A71]  
| [#x0A81-#x0A83] | #x0ABC | [#x0ABE-#x0AC5]  
| [#x0AC7-#x0AC9] | [#x0ACB-#x0ACD] | [#x0B01-  
#x0B03] | #x0B3C | [#x0B3E-#x0B43] | [#x0B47-  
#x0B48] | [#x0B4B-#x0B4D] | [#x0B56-#x0B57]  
| [#x0B82-#x0B83] | [#x0BBE-#x0BC2] | [#x0BC6-  
#x0BC8] | [#x0BCA-#x0BCD] | #x0BD7 | [#x0C01-  
#x0C03] | [#x0C3E-#x0C44] | [#x0C46-#x0C48]  
| [#x0C4A-#x0C4D] | [#x0C55-#x0C56] | [#x0C82-  
#x0C83] | [#x0CBE-#x0CC4] | [#x0CC6-#x0CC8]  
| [#x0CCA-#x0CCD] | [#x0CD5-#x0CD6] | [#x0D02-  
#x0D03] | [#x0D3E-#x0D43] | [#x0D46-#x0D48]  
| [#x0D4A-#x0D4D] | #x0D57 | #x0E31 | [#x0E34-  
#x0E3A] | [#x0E47-#x0E4E] | #x0EB1 | [#x0EB4-  
#x0EB9] | [#x0EBB-#x0EBC] | [#x0EC8-#x0ECD]  
| [#x0F18-#x0F19] | #x0F35 | #x0F37 | #x0F39  
| #x0F3E | #x0F3F | [#x0F71-#x0F84] | [#x0F86-  
#x0F8B] | [#x0F90-#x0F95] | #x0F97 | [#x0F99-  
#x0FAD] | [#x0FB1-#x0FB7] | #x0FB9 | [#x20D0-  
#x20DC] | #x20E1 | [#x302A-#x302F] | #x3099  
| #x309A

☞ Digit ::= [#x0030-#x0039] | [#x0660-#x0669] | [#x06F0-  
#x06F9] | [#x0966-#x096F] | [#x09E6-#x09EF]  
| [#x0A66-#x0A6F] | [#x0AE6-#x0AEF] | [#x0B66-  
#x0B6F] | [#x0BE7-#x0BEF] | [#x0C66-#x0C6F]  
| [#x0CE6-#x0CEF] | [#x0D66-#x0D6F] | [#x0E50-  
#x0E59] | [#x0ED0-#x0ED9] | [#x0F20-#x0F29]

☞ Extender ::= #x00B7 | #x02D0 | #x02D1 | #x0387 | #x0640  
| #x0E46 | #x0EC6 | #x3005 | [#x3031-#x3035]  
| [#x309D-#x309E] | [#x30FC-#x30FE]

ここで定義する文字クラスは、Unicode文字データベースから、次のとおりを得ることができる。

- a) 名前開始文字は、Li, Lu, Lo, Lt, NIカテゴリ内の一つでなければならない。
- b) 名前開始文字以外の名前文字は、Mc, Me, Mn, Lm, Ndカテゴリ内の一つでなければならない。
- c) 互換性領域にある文字(文字符号で#xF900より大きく#xFFFEより小さい文字)は、XMLにおける名前としては許されない。
- d) フォント分解が互換性分解をもつ文字(つまり、データベース内の5番目のフィールドに"compatibility formatting tag"があるもの。これは、5番目のフィールドが、"<"で始まることによって示される。)は許されない。
- e) 次の文字は、名前開始文字として扱う。これは、プロパティファイルが、これらの文字をアルファベットに類似すると見なすことによる。それらは [#x02BB-#x02C1], #x0559, #x06E5, #x06E6とする。
- f) 文字符号が[#x20DD-#x20E0]の文字は、(Unicodeの5.14に従って)除外する。
- g) 文字符号が#x00B7の文字は、プロパティリストにしたがって、エクステンダ(extender)に分類する。
- h) 文字#x0387は、これに相当する標準形が#x00B7なので、名前文字に追加する。
- i) 文字'!'及び'\_'は、名前開始文字として使ってよい。
- j) 文字'-'及び'.'は、名前文字として使ってよい。

## 附属書 C. XML及びSGML【非準】

XMLは、SGMLのサブセットとして設計されている。すなわち、すべての**妥当**なXML文書は、規格に適合するSGML文書にもなる。SGMLが文書に課す制限以外に、XMLがいかなる制限を課すかについての詳細は、[Clark]を参照のこと。

## 附属書 D. 実体参照及び文字参照の展開【非準】

この附属書は、24ページの§ 4.4 XMLプロセサによる**実体及び参照の扱い**で規定されている、実体参照及び文字参照を認識し展開する一連の流れを例によって示す。

DTDが、次の宣言を含む場合を考える。

```
<!ENTITY example "<p>An ampersand (&#38;) may be escaped
numerically (&#38;#38;) or with a general entity
(&amp;).</p>" >
```

XMLプロセサは、実体の宣言を構文解析した時点で文字参照を認識し、これを解決する。実体"example"の値として、次の文字列を保存する。

```
<p>An ampersand (&) may be escaped
numerically (&#38;) or with a general entity
(&amp;).</p>
```

文書内で"&example;"を参照すると、このテキストは再び構文解析される。このとき、要素"p"の開始タグ及び終了タグを認識し、三つの参照を認識し展開する。その結果、要素"p"は、次の内容(すべてデータであって、区切り子又はマーク付けは存在しない。)をもつ。

```
An ampersand (&) may be escaped
numerically (&) or with a general entity
(&).
```

規則及びその効果をより詳細に示すため、さらに複雑な例を示す。次の例で、行番号は参照の便宜のためだけに付ける。

```
1 <?xml version='1.0'?>
2 <!DOCTYPE test [
3 <!ELEMENT test (#PCDATA) >
4 <!ENTITY % xx '%zz;'>
5 <!ENTITY % zz '<!ENTITY tricky "error-prone" >' >
6 %xx;
7]>
8 <test>This sample shows a &tricky; method.</test>
```

これを処理すると、次のとおりとなる。

- a) 4行目で、37番の文字への参照を直ちに展開し、パラメタ実体"xx"を、シンボルテーブルに"%zz;"という値とともに保存する。置換テキストを再び走査することはないので、パラメタ実体"zz"への参照は認識しない。"zz"は、まだ宣言されていないので、走査されればエラーとなる。
- b) 5行目で、文字参照"&#60;"を直ちに展開し、パラメタ実体"zz"を"<!ENTITY tricky "error-prone" >"という置換テキストとともに保存する。これは、整形形式の実体宣言になる。
- c) 6行目で、"xx"への参照を認識し、"xx"の置換テキスト(すなわち,"%zz;")を構文解析する。"zz"への参照を続いて認識し、置換テキスト("<!ENTITY tricky "error-prone" >")を構文解析する。一般実体"tricky"は、この時点で宣言され、その置換テキストは"error-prone"になる。
- d) 8行目で、一般実体"tricky"への参照を認識し、展開する。要素"test"の完全な内容は、次の自己記述的な(非文法的な)文字列となる。つまり、This sample shows a error-prone method.

## 附属書 E. 決定的内容モデル【非準】

[互換性のため](#)、要素型宣言における内容モデルは、決定的でなければならない。

SGMLは、決定的内容モデル(SGMLでは、非あいまいと呼ぶ。)を要求する。SGMLシステムを用いて作成したXMLプロセサは、非決定的内容モデルをエラーとしてもよい。

例えば、内容モデル((b, c) | (b, d))は非決定的となる。これは、最初にbを与えたとき、モデル内のいずれのbとマッチするのか、その次の要素を先読みすることなしには、パーサは知ることができないことによる。この場合は、bの二つの出現は、一つにまとめることができ、



モデルは(b, (c | d))となる。こうすれば、明らかに最初のbは、内容モデル内の一つの名前とだけマッチする。パーサは先読みして、次にくるものを知る必要がない。cもdも受理される。

形式的に示せば次の通り。Aho, Sethi, and Ullman [Aho/Ullman]の3.9節のアルゴリズム3.5などの標準的なアルゴリズムを用いて、内容モデルから有限オートマトンを構成することができる。この種の多くのアルゴリズムでは、正規表現における各々の位置(つまり、正規表現の構文木における各々の末端ノード)に対して、follow setを構成する。ある位置に対するfollow setにおいて、複数の位置が同じ要素型名でラベル付けされていれば、その内容モデルはエラーとなり、エラーとして報告されることもある。

すべての非決定的内容モデルを等価な決定的内容モデルに変換することはできないが、多くの非決定的内容モデルを変換するアルゴリズムが存在する。Brüggemann-Klein 1991 [Brüggemann-Klein]を参照のこと。

## 附属書 F. 文字符号化方式の自動検出【非準】

XMLの符号化宣言は、各実体の内部ラベルとして機能し、どの文字符号化方式を使用するかを示す。しかし、XMLプロセサは内部ラベルを読む前にどの文字符号化方式を使われているかを知る必要があり、これが、内部ラベルが示そうとしていることに他ならない。一般的には、これは絶望的な状態となる。しかし、XMLにおいては、完全には絶望的ではない。これは、XMLが次の二つの点で一般的な場合に対する制限を加えていることによる。一つの制限は、どの実装も有限個の文字符号化方式だけをサポートするものと見なす。他の一つは、XMLの符号化宣言の位置及び内容を制限して、各実体で使用される文字符号化方式の自動検出を可能にする。また、多くの場合に、XMLのデータストリームに加え、他の情報が利用できる。ここでは、XMLの実体がプロセサに渡される時、(外部)情報を伴うかどうかによって、二つの場合に分ける。まず最初の場合を示す。

UTF-8形式又はUTF-16形式ではないXML実体は、最初の文字列を<?xml'とするXML符号化宣言で始めなければならないので、どの適合したプロセサも、入力にある2オクテット又は4オクテットを調べれば、次のどの場合があてはまるかを検出できる。このリストを読む際には、UCS-4の'<'が"#x0000003C"、'?'が"#x0000003F"、及びUTF-16のデータストリームの必要とするバイト順マークが"#xFEFF"ということを知っておくと役立つ。

- a) 00 00 00 3C: UCS-4, big-endian マシン (1234順)
- b) 3C 00 00 00: UCS-4, little-endian マシン (4321順)
- c) 00 00 3C 00: UCS-4, 普通ではないオクテット順 (2143)
- d) 00 3C 00 00: UCS-4, 普通ではないオクテット順 (3412)
- e) FE FF: UTF-16, big-endian
- f) FF FE: UTF-16, little-endian
- g) 00 3C 00 3F: UTF-16, big-endian, バイト順マークなし(したがって、厳密に言えば、エラー。)
- h) 3C 00 3F 00: UTF-16, little-endian, バイト順マークなし(したがって、厳密に言えば、エラー。)
- i) 3C 3F 78 6D: UTF-8, ISO 646, ASCII, ISO 8859の各パート, Shift-JIS, EUC, 並びに任意の他の7ビット, 8ビット又は混在幅の符号化方式であって、ASCII文字を通常の位置、幅及

び値とすることを保証するもの。これらのどれに対応するかを検出するためには、実際の符号化宣言を読み込まなければならない。しかし、これらすべての符号化方式は、ASCII文字に対して同じビットパターンを使用するので、符号化宣言自体は、正確に読み込むことができる。

j) 4C 6F A7 94: EBCDIC (又はその変種。どのコードページを使用するかを知るためには、符号化宣言全体を読み込まなければならない。)

k) その他: 符号化宣言なしのUTF-8。そうでないときには、データストリームが壊れているか、文書の断片であるか、何らかの形式に従って埋め込まれている。

この程度の自動判別でも、XMLの符号化宣言を読み込み、文字符号化方式の識別子を十分解析できる。識別子の解析は、類似する各々の符号化方式の一つ一つを区別するために必要になる(例えば、UTF-8及び8859を区別するため、8859の各パートを区別するため、使用している特定のEBCDICコードページを区別するため。)

符号化宣言の内容をASCII文字に限定しているため、どのシステムの符号化方式が使用されているかを検出すれば、プロセサは符号化宣言全体を正確に読み込むことができる。現実問題として、広く使用されている文字符号化方式は前述のシステムのいずれかにあてはまるので、オペレーティングシステム又は伝送プロトコルが与える外部情報を信頼できないときでも、内部ラベルで文字符号化方式をかなり正確に示すことがXML符号化宣言によって可能となる。

プロセサが文書の符号化方式を検出しさえすれば、それぞれの場合に対して別の入力ルーチン呼び出すか、又は入力する各文字に対し適切な変換関数を呼び出すことによって、適切に動作することができる。

自分自体にラベル付けをするいかなるシステムでも同様だが、ソフトウェアが、符号化宣言を更新せずに実体の文字集合又は符号化方式を変えれば、XMLの符号化宣言は機能しない。文字符号化ルーチンの実装者は、実体のラベル付けに使用する内部及び外部の情報の正確さの保証に注意すべきである。

[訳注(これは原文にはない)] 以下の記述はRFC 2376によって、すでに取って代わられているので、そちらを参照されたい。MIME型application/xmlの場合の扱いはRFC 2376では変更されている。

2番目の場合は、XMLの実体の他に、符号化方式についての情報が存在するときである。いくつかのファイルシステム及びネットワークプロトコルでは、その符号化方式についての情報が存在する。複数の情報が利用できる時、それらの相対的な優先度と、それらが矛盾したときの望ましい処理方法とは、XMLの配送に使用するより高水準のプロトコルの一部として規定するのがよい。例えば、内部ラベル及び外部ヘッダに存在するMIME形式のラベルの相対的な優先度についての規則は、MIME型text/xml及びapplication/xmlを定義するRFC文書の一部となるのが望ましい。しかし、相互運用性のため、次の規則を推薦する。

a) XMLの実体がファイルに存在すれば、バイト順マーク及び符号化宣言PIは、(存在すれば)文字符号化方式を決定するために使用する。他のすべてのヒューリスティック及び情報は、エラー回復のためだけに用いる。

b) XMLの実体をMIME型text/xmlで配送するときは、このMIME型のもつcharsetパラメタが文字符号化方式を決定する。他のすべてのヒューリスティック及び情報は、エラー回復のためだけに用いる。

c) XMLの実体をMIME型application/xmlで配送するときは、バイト順マーク及び符号化宣言PIを(存在すれば)文字符号化方式の決定のために使用する。他のすべてのヒューリスティック及び情報はエラー回復のためだけに用いる。

---

これらの規則は、プロトコルについての資料がないときにだけ用いる。特に、MIME型text/xml及びapplication/xmlが規定されれば、関連RFCの勧告が、これらの規則に取って代わる。

## 附属書 G. W3C XML作業グループ【非準】

この標準情報(TR)の原勧告は、W3C XML作業グループ(WG)が準備し、公開を承認した。WGがこの標準情報(TR)の原勧告を承認するということは、WGのすべての委員が承認投票を行ったということを必ずしも意味しない。XML WGの現在の委員及び以前の委員を次に示す。

Jon Bosak, Sun (Chair); James Clark (Technical Lead); Tim Bray, Textuality and Netscape (XML Co-editor); Jean Paoli, Microsoft (XML Co-editor); C. M. Sperberg-McQueen, U. of Ill. (XML Co-editor); Dan Connolly, W3C (W3C Liaison); Paula Angerstein, Texcel; Steve DeRose, INSO; Dave Hollander, HP; Eliot Kimber, ISOGEN; Eve Maler, ArborText; Tom Magliery, NCSA; Murray Maloney, Muzmo and Grif; 村田 真, 富士ゼロックス情報システム(株); Joel Nava, Adobe; Conleth O'Connell, Vignette; Peter Sharpe, SoftQuad; John Tigue, DataChannel

このページは意図的に空白のままにしています。