返回总目录

目　　录

第六部分　相关协议及标准

第二十一章　可扩展标记语言 1.0（第二版）规范

第二十二章　XML 术语参考

第二十三章　XML 技术动态

第二十一章　可扩展标记语言 1.0（第二版）规范

W3C 建议 （2000 年 10 月 6 日）

本版本

http://www.w3.org/TR/2000/REC-xml-20001006
http://www.w3.org/TR/2000/REC-xml-20001006.html
http://www.w3.org/TR/2000/REC-xml-20001006.xml
http://www.w3.org/TR/2000/REC-xml-20001006.pdf
http://www.w3.org/TR/2000/REC-xml-20001006-review.html

最新版本

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

上一版本

http://www.w3.org/TR/2000/WD-xml-2e-20000814
http://www.w3.org/TR/1998/REC-xml-19980210

编者

Tim Bray (Textuality and Netscape)　mailto: tbray@textuality.com
Jean Paoli (Microsoft)　mailto: jeanpa@microsoft.com
C. M. Sperberg-McQueen (University of Illinois at Chicago)　mailto: cmsmcq@uic.edu
Eve Maler (Sun Microsystems, Inc.)　mailto: elm@east.sun.com

摘要

本文档完整地描述了可扩展标记语言 XML，它是标准通用标记语言 SGML 的一个子集。其目的在于使得在 Web 上能以现有超文本标记语言 HTML 的使用方式提供、接收和处理通用的 SGML 成为可能。XML 的设计既考虑了实现的方便性，同时也顾及了与 SGML 和 HTML 的互操作性。

关于本文档

本文档已由 W3C 组织成员和其他相关各方审阅，并已被组织理事批准为 W3C 建议。这是一个稳定的文档，可以用作参考材料，也可以作为其他文档的正式参考文献。W3C 在建议制定过程中的作用是吸引对本规范的注意并促进它的广泛使用。这能增强 Web 的功能和互操作性。

本文档规定了一种用于 World Wide Web 的语法，此语法是通过取一个已存在并已广泛使用的文本处理国际标准(标准通用标记语言，经增补和更正的 ISO 8879:1986(E))的子集而创建的。它是 W3C XML 行动组(XML Activity)的工作成果，关于 XML 行动组的详细信息可以在 http://www.w3.org/XML 找到。英文版是唯一的正式版。在 http://www.w3.org/TR 可以找到现有

W3C 建议和其他技术文档的一个列表。

XML 1.0 第二版不是 XML 的一个新版本（1998 年 2 月 10 日首次发表）；它只是为了方便读者，并入了第一版勘误表中指出的错误和修改（在 http://www.w3.org/XML/xml-19980210-errata）。 本第二版的勘误表在 http://www.w3.org/XML/xml-V10-2e-errata。

本文档中的错误请报告给 xml-editor@w3.org，同时可以在此找到相关的存档。

注 C. M. Sperberg-McQueen 在第一版发表之后供职之处已有变化。他现在供职于 W3C，可以通过 cmsmcq@w3.org 和他联系。

21.1 绪　　论

可扩展标记语言，缩写为 XML，描述了一类称为 XML 文档的数据对象，同时也部分地描述了处理这些数据对象的计算机程序的行为。XML 是 SGML（标准通用标记语言[ISO 8879]）针对应用的一个子集，或者说是 SGML 的一种受限形式。根据定义，XML 文档是合乎规范的 SGML 文档。

XML 文档由称为实体的存储单元组成，实体包含解析（parsed）数据或未解析（unparsed）数据。解析数据由字符组成，其中一些字符组成字符数据，另一些字符组成标记。标记中包含了对文档存储格式(storage layout)和逻辑结构的描述。XML 提供了一种机制用于约束存储格式和逻辑结构。

称为 XML 处理器的软件模块用于读取 XML 文档，存取其中的内容和结构。XML 处理器被设想为是为另一个称为应用的模块作处理。本规范从 XML 处理器应如何读取 XML 数据以及应向应用提供哪些信息的这两个方面，描述了要求 XML 处理器作出的动作。

21.1.1 开发者和开发目标

XML 由 XML 工作组（原先的 SGML 编辑审查委员会）开发，此工作组由 World Wide Web Consortium（W3C）在 1996 年主持成立。工作组由 Sun Microsystems 的 Jon Bosak 负责，同样由 W3C 组织的 XML SIG（Special Interest Group）（原先的 SGML 工作组）积极参与了 XML 工作组的工作。XML 工作组的成员在附录中给出。工作组与 W3C 的联系人是 Dan Connolly。

XML 的设计目标如下：

- XML 应该可以直接在因特网（Internet）中使用。
- XML 应该支持大量不同的应用。
- XML 应该与 SGML 兼容。
- 处理 XML 文档的程序应该容易编写。
- XML 中的可选项应尽可能少，理想状况下应为零。
- XML 文档应该清晰明了，可读性强。
- XML 应易于设计。
- XML 的设计应该正式而且简洁。
- XML 文档应易于创建。
- XML 标记的简洁性较为次要。

本规范与其他相关的标准一起（Unicode 和 ISO/IEC 10646 定义了字符集，Internet RFC1766 定义了语言识别码，ISO 639 定义了语言名称代码，ISO 3166 定义了国家名称代码），提供了理解 XML 版本 1.0 和创建相应计算机处理程序所需的所有信息。

在完整保留所有文本和法律注意事项的前提下，本版本的 XML 规范可以自由分发。

21.1.2 术语

用于描述 XML 文档的术语在此规范的正文中定义。在这些定义中以及描述一个 XML 处理器的动作时，使用了下面的术语：

- 可以（may），允许合乎规范的文档和 XML 处理器按所描述的方式工作，但不要求必须如此。
- 必须（must），要求合乎规范的文档和 XML 处理器按所描述的方式工作；否则出现错误。
- 错误（error），对本规范中的规则的违反；其结果不确定。合乎规范的软件可以检测和报告错误，并可以从中恢复。
- 严重错误（fatal error），合乎规范的 XML 处理器必须检测到，并向应用报告的一类错误。在遇到严重错误之后，处理器可以继续处理数据以发现更多的错误并可以向应用报告这些错误。为了支持错误的更正，处理器可以向应用提供文档中未经处理的数据（字符数据和标记的混合体）。但是，一旦检测到一个严重错误，处理器必须停止正常的处理（也就是说，它必须停止以正常的方式向应用提供与文档逻辑结构有关的数据和信息）。
- 由用户选择（at user option），合乎规范的软件可以或者必须（取决于句子中的情态动词）按所描述的方式工作；如果它满足这个条件，它必须同时提供用户一种手段，使得用户能够启用和禁用所描述的工作方式。
- 有效性约束（validity constraint），适用于所有有效的 XML 文档的一种规则。违反有效性约束属于错误；由用户选择，进行验证的 XML 处理器必须报告这些错误。
- 格式约束（well-formedness constraint），适用于所有有效的 XML 文档的一种规则。违反格式约束属于严重错误。
- 匹配（match），（对于字符串和名字：）被比较的两个字符串或名字必须完全相同。在 ISO/IEC 10646 中有多种可能表示方式的字符（例如，既有预定义（precomposed）形式和基字符（base）+变音符形式的字符）只在两个字符串中的表示方式相同时才匹配。由用户选择，处理器可以将这些字符规范成某种规范形式。不进行字符的大小写转换。对于句法中的字符串和规则：如果一个字符串属于一个句法产生式产生的语言，则它匹配这个产生式；对于内容和内容模型：当一个元素符合“元素有效性”约束中的描述时，它匹配其声明。
- 兼容性考虑（for compatibility），仅用于保证与 SGML 兼容的 XML 特性。
- 互操作性考虑（for interoperability），是一个不具约束性的建议，目的是增加 XML 文档能被在 ISO 8879 的 WebSGML 改编附件之前已有的 SGML 处理器处理的可能性。

21.2 文　　档

如果一个数据对象满足本规范中格式良好的要求时，它是一个 XML 文档。一个规范的 XML 文档如果满足某些进一步的约束，它将更为有效。

每一个 XML 文档都有逻辑和物理结构。物理上而言，文档由称为实体的单元组成。一个实体可以引用（refer）其他实体，将它们包含在文档中。文档开始于“根（root）”或文档实体中。逻辑上而言，文档由声明、元素、注释、字符引用和处理指令组成，所有这些都在文档中用显式标记指明。

21.2.1 格式良好的 XML 文档

一个文本对象如果满足以下条件，它将是一个格式良好的 XML 文档：

（1）作为一个整体，它匹配文档（document）产生式。

（2）它满足本规范中定义的所有格式约束。

（3）此文档中直接或间接引用的每一个解析实体都是格式良好的。

文档

[1] document ::= prolog element Misc*

匹配 document 产生式意味着：

（1）它包含一个或多个元素。

（2）有且仅有一个称为根（root）或文档元素的元素，它不出现在其他任何元素的内容（content）中。对于其他所有元素，如果起始标签在另一个元素的内容中，则其结束标签也在同一元素的内容中。换一个更简单的说法，以起始标签和结束标签为界的各个元素，必须严格地嵌套。

这样做的结果是，对于每一个非根的元素 C，文档中另有一个元素 P，C 在 P 的内容中，而不在其他任何被 P 所包含的元素的内容中。P 被称为 C 的父元素（parent），而 C 被称为 P 的子元素（child）。

21.2.2 字符

一个解析实体包含文本（text），文本是一个字符（character）序列，可以表示标记或字符数据。一个字符是 ISO/IEC 10646[ISO/IEC 10646]中定义的文本最小单元。合法的字符包括制表符、回车、换行以及 Unicode 和 ISO/IEC 10646 中定义的合法的图形字符。不提倡使用[Unicode]6.8 节中定义的“兼容字符（compatibility characters）”。

字符范围

[2] Char ::= #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]

/* 除了替代块（surrogate block），FFFE 和 FFFF 以外的任意 Unicode 字符。*/

将字符代码编码成位模型的机制各个实体间可能会有所不同。所有的 XML 处理器必须接受 10646 中的 UTF-8 和 UTF-16 编码；用于指出所用编码或指定使用其他编码的机制在后面讨论。

21.2.3 通用句法成分

本节中定义了一些在句法中广泛使用的符号。

S（空白域）包括一个或多个空格字符（#x20）、回车、换行或制表符。

空白

[3] S ::= （#x20 | #x9 | #xD | #xA）+

为方便起见，字符被分为字母、数字和其他字符三类。字母可以是字母表中的字母，或是一个音节基字符（syllabic base character）后跟一个或多个组合字符，也可以是一个表意字符。在“B.字符的分类”中给出了每一类字符的特定定义。

名字（name）是以一个字母或某一标点符号开头的记号，后跟字母、数字、连字符、下划线、冒号或句号，这些符号统称为命名字符（name character）。以 xml 或其他任何以（（'X'|'x'）（'M'|'m'）（'L'|'l'））的字符串开头的名字，被保留用于本规范的此版本或后续版本的标准化。

注意 XML 名字中的冒号被保留用于名字空间（name space）实验。它的含义有待于日后标准化，那时那些将冒号用于实验目的的文档有可能需要更新（不保证 XML 采用的任何名字空间机制实际会采用冒号作为定界符）。实际上，这意味着除非用于名字空间实验，XML 文档编者不应该在 XML 名字中使用冒号，但 XML 处理器应该接受冒号作为一个命名字符。

Nmtoken（名字记号，name token）是任何命名字符的混合体。

名字和记号

[4] NameChar ::= Letter | Digit | '.' | '-' | '_' | ':' | CombiningChar | Extender

[5] Name ::= （Letter | '_' | ':'） （NameChar）*

[6] Names ::= Name （S Name）*

[7] Nmtoken ::= (NameChar)+

[8] Nmtokens ::= Nmtoken (S Nmtoken)*

常量数据是任何用引号括起的字符串，不包括用作定界符的引号。常量数据用于指明内部实体的内容（EntityValue），属性值（AttValue），以及外部标识符（SystemLiteral）。注意，对 SystemLiteral 的解析可以不扫描标记。

常量数据

[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] | [-'()+,./:=?;!*#@$_%]

注 虽然产生式 EntityValue 允许定义只包含单个 < 的实体（如，<!ENTITY mylt "<">），但是强烈建议避免这种用法，因为对此实体的任何引用都会引起一个格式正确性错误。

21.2.4 字符数据和标记

文本由字符数据和标记混合构成。标记包括起始标记、结束标记、空元素标记、实体引用、 字符引用、注释、CDATA 段定界符、文档类型声明、处理指令、XML 声明、文本声明、以及任何在文件实体顶层的空白（即，在文件元素之外且不在任何其他的标记中）。

其他所有非标记的文本组成文档的字符数据。

and 号（&）和左尖括号（<）只有作为标记定界符，或在注释、处理指令，或 CDATA 段中时才能以字面形式出现。它们在一个内部实体声明的字面实体数值中也是合法的。如果在其他地方需要用到这两个字符，它们必须用数值式字符引用来转义或分别用字符串&和<表示。右尖括号（>）可以用>表示，而当它在内容中的字符串“]]>”中出现，但此字符串不表示一个 CDATA 段的结束时，出于兼容性考虑，必须用“>”或一个字符引用转义得到。

在一个元素的内容中，字符数据可以是不包括任何标记的起始定界符的任意字符串。在一个 CDATA 段中，字符数据可以是不包括 CDATA 段结束定界符“]]>”的任意字符串。

为了允许在属性值中包含单引号和双引号，省略符或称单引号（'）可以被表示为“'”，而双引号（"）可以被表示为“"”。

字符数据

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

21.2.5 注释

注释可以在其他标记之外的文档中的任何位置出现。另外，它们可以在文档类型声明中语法允许的地方出现。它们不是文档字符数据的一部分，XML 处理器可以，但不必须，允许一个应用检索注释文本。出于兼容性考虑，字符串“--”（双连字符）不能在注释中出现。

注释

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

注释的一个例子：

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

注意，此文法不允许注释以 ---> 结尾。下面的例子不是格式正确的。

```
<!-- B+, B, or B--->
```

21.2.6 处理指令

处理指令（PI）允许文档中包含由应用来处理的指令。

处理指令

[16] PI ::= '<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'

[17] PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))

PI 不是文档字符数据的一部分，但必须传递给应用。PI 以用于指示传递给哪个应用的目标（PITarget）开头，目标名字“XML”，“xml”，等等，保留用于本规范的此版本或后续版本的标准化。XML 符号机制可以用于 PI 目标的形式化声明。参数实体在处理指令中不被识别。

21.2.7 CDATA 段

CDATA 段可以出现在字符数据可以出现的任何地方，它们用于转义包含会被识别为标记的字符串的文本块。CDATA 段以字符串“<![CDATA[”开始，以字符串“]]>”结束：

CDATA 段

[18] CDSect ::= CDStart CData CDEnd

[19] CDStart ::= '<![CDATA['

[20] CData ::= (Char* - (Char* ']]>' Char*))

[21] CDEnd ::= ']]>'

在一个 CDATA 段内，只有 CDEnd 字符串被识别为标记，因此左尖括号和“&”可以以它们的字面形式出现，不需要（也不能）被换码为“<”和“&”。CDATA 段不能嵌套。

一个 CDATA 段的例子，其中“<greeting>”和“</greeting>”被识别为字符数据，而不是标记：

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

21.2.8 序言和文档类型声明

XML 文档可以，也应该以一个 XML 声明开始，其中指明了所用 XML 的版本。 例如，以下是一个完整的 XML 文档，它是格式良好的，但不是有效的：

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

下面这个也同样：

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

版本号“1.0”应该用于表明对与本规范的此版本相一致，如果使用了值“1.0”但又与本规范的此版本不一致，那么这是文档的一个错误。XML 工作组打算赋予本规范的后续版本不同于“1.0”的数值，但这并不代表开发后续版本的承诺，也不代表如果有后续版本，会使用任何特殊的命名方案的承诺。因为不排除有后续版本的可能性，提供了本构造作为一旦需要时进行自动版本识别的手段。当处理器收到的文档标有它们不支持的版本时，可以给出一个错误。

XML 文档中标记的功能是描述文档的存储格式和逻辑结构，并将属性-值对和逻辑结构关联起来。XML 提供一种称为文档类型声明的机制，用于定义对逻辑结构的约束，支持预定义存储单元的使用。如果一个 XML 文档有相应的文档类型声明并且它遵循其中的约束，则称它是有效的（valid）。

文档类型声明必须位于文档第一个元素之前。

序言

[22] prolog ::= XMLDecl? Misc* (doctypedecl Misc*)?

[23] XMLDecl ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'

[24] VersionInfo ::= S 'version' Eq (' VersionNum ' | " VersionNum ")

[25] Eq ::= S? '=' S?

[26] VersionNum ::= ([a-zA-Z0-9_.:] | '-')+

[27] Misc ::= Comment | PI | S

XML 文档类型声明包含或指向标记声明，标记声明提供某一类文档的语法。这种语法被称为文档类型定义（document type definition，DTD）。文档类型定义可以指向一个外部子集（一种特殊类型的外部实体），或者可以在一个内部子集中直接包含标记声明，或者两者兼用。一个文档的文档类型定义由这两个子集合在一起组成。

标记声明可以是元素类型声明、属性表声明、实体声明或是符号声明。这些声明可以如下面规范性和有效性约束中所述，全部或部分地包含在参数实体中。

文档类型定义

[28] doctypedecl ::= '<!DOCTYPE' S Name (S ExternalID)? S? [VC: 根元素类型]
('[' (markupdecl | DeclSep)* ']' S?)? '>'
[WFC: 外部子集]
/* */

[28a] DeclSep ::= PEReference | S [WFC: 声明间的参数实体]
/* */

[29] markupdecl ::= elementdecl | AttlistDecl [VC: 严格的声明/参数实体嵌套]
| EntityDecl | NotationDecl | PI | Comment [WFC: 内部子集中的参数实体]

注意，要构建包含了一个既不指向外部子集也不包含内部子集的 doctypedecl 而格式正确的文件是可能的。

标记声明可以全部或部分地由参数实体的置换文本组成。本规范后面的各个非终结符（elementdecl，AttlistDecl，等等）产生式描述的是在所有的参数实体被包含（include）之后的声明。

除了在常量、处理指令、注释和被忽略的条件段的内容中出现的参数实体引用以外，DTD 中的其他任何地方（内部或外部子集以及外部参数实体）的参数实体引用都被识别。在实体值常量中的参数实体引用也被识别。内部子集中参数实体引用的使用限制如下所述。

有效性约束：根元素类型（Root Element Type）。

文档类型声明中的 Name 必须匹配根元素的类型。

有效性约束：严格的声明/参数实体嵌套。

参数实体的置换文本必须用标记声明严格嵌套。即，如果一个标记声明（上面的 markupdecl）的第一个或最后一个字符被包含于一个参数实体引用的置换文本中，两者必须都在此置换文本中。

格式约束：内部子集中的参数实体。

在内部 DTD 子集中，参数实体引用只能出现在标记声明出现的地方，而不能在标记声明内部出现（这个约束不适用于出现在外部参数实体内的引用，也不适用于外部子集）。

格式正确性约束: 外部子集

外部子集（如果有的话）必须匹配产生式 extSubset。

格式正确性约束: 声明间的参数实体

一个 DeclSep 内的参数实体引用的置换文本必须匹配产生式 extSubsetDecl。

同内部子集一样，外部子集和任何 DTD 中引用的外部参数实体，必须由一系列被非终结符 markupdecl 所允许的完整的标记声明组成，其中可以夹杂空白字符或参数实体引用。但是，外部子集和外部参数实体的部分内容可以通过使用条件段（conditional section）被有条件地忽

略，在内部子集中则不允许这么做。

外部子集

[30] extSubset ::= TextDecl? extSubsetDecl

[31] extSubsetDecl ::= (markupdecl | conditionalSect | PEReference | S)*

外部子集和外部参数实体与内部实体不同之处还在于：在外部子集和外部参数实体内，参数实体引用不仅可以出现在标记声明间，还可以出现在标记声明内。

下面是有文档类型声明的 XML 文档的例子：

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

系统标识符“hello.dtd”给出了文档 DTD 的 URI。

声明也可以如同下面这个例子一样直接（locally）给出：

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

如果同时使用外部和内部子集，内部子集被看成出现在外部子集之前，这意味着内部子集中的实体和属性表声明的优先级要比在外部子集中的高。

21.2.9 独立文档声明

当文件从 XML 处理器传递给应用时，标记声明可以影响它的内容，属性缺省值和实体声明是其中的例子。可以作为 XML 声明一个成分的独立文件声明，指明了是否存在着在文件实体外或在参数实体中的声明。外部标记声明被定义为出现在外部子集或参数实体（外部或内部，包括内部参数实体是因为并不强制不进行验证的处理器读取其中的标记声明）中的标记声明。

独立文档声明：

[32] SDDecl ::= S 'standalone' Eq (('"' ('yes' | 'no') '"') | ("'" ('yes' | 'no') "'")) [VC: 独立文档声明]

在一个独立文档声明中，值“yes”表示对于文档实体没有外部标记声明（不论是在 DTD 外部子集中，还是在由内部实体引用的外部参数实体中）会影响从 XML 处理器传递给应用的信息。值“no”表示有或可能有这样的外部标记声明。注意独立文档声明只是表示外部声明的存在，如果文档中存在对外部实体的引用，而这些实体已在内部声明时，不影响它的独立状态。

如果不存在外部标记声明，独立文档声明没有意义。如果存在外部标记声明，但没有独立文档声明，就假定取值“no”。

某些网络传输应用也许需要独立的文档，任何满足 standalone=“no”的 XML 文档可以通过一定的算法转换为独立文档。

有效性约束：独立文档声明。

独立文档声明必须取值为“no”，如果任何外部标记声明中包含：

- 有缺省值的属性声明，如果适用这些属性的元素出现在文档中而又没有给这些属性赋

值的话。

- （除了 amp，lt，gt，apos，quot 的）实体声明，而对这些实体的引用出现在文档中的话。
- 需要规范化的属性声明，这些出现在文档中的属性的值会因规范化而改变。
- 具有元素内容的元素类型声明，如果在这些类型的任一实例中直接出现空白域的话。

具有独立文档声明的 XML 声明的例子：

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

21.2.10 空白域处理

在编辑 XML 文档时，使用“"空白域"”（空格，制表符，空行，在本规范中用非终结符 S 表示）来分开标记以获得更好的可读性是很方便的。通常在文档的交付版本中不想包含这些空白域。另一方面，必须保留在交付版本中的有意义的空白域是很常见的，如在诗歌和源码中的空白域。

XML 处理器必须始终把不是标记的所有字符传递给应用。一个进行验证的 XML 处理器必须同时通知应用这些字符中的那一些组成了出现在元素内容中的空白域。

可以在元素中附加一个名为 xml:space 的特殊属性，以通知应用应该保留此元素中的空白域。在有效的文档中，此属性和其他属性一样，使用时必须声明。它必须被声明为枚举类型，只有“default”和“preserve”两个可能的值。例如：

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

“default”表示可以对此元素使用应用的缺省空白域处理模式，“preserve”表示应用应该保留所有的空白域。这适用于其所处元素的内容中的所有元素，除非被另一个 xml:space 属性的实例所覆盖。

任何文档的根元素被认为对应用的空白域处理方式不作要求，除非它给此属性赋了值或将此属性声明为带缺省值。

21.2.11 行尾处理

为编辑的方便起见，存储 XML 已析实体的计算机文档经常用行来组织。通常这些行用回车符（#xD）和换行符（#xA）的一些组合来分隔。

为了使应用的工作简单化，对于一个外部已析实体或内部已析实体的字面实体值中包含的任何双字符序列“"#xD#xA"”或单独的“#xD”，XML 处理器都应换成“#xA”传递给应用（这可以通过在进行解析前将所有行定界符规范成#xA 而方便地实现）。

21.2.12 语言标识

在进行文档处理时，标识出其内容所使用的自然或形式化语言经常是很有用的。可以在文档中插入一个名为 xml:lang 的特殊属性用于指出 XML 文档中任何元素的内容和属性所使用的语言。在有效的文档中，此属性和其他属性一样，使用时必须声明。此属性的值是[IETF RFC 1766]，“语言标识码”中定义的语言标识符。

注意，原有版本一中的产生式[33]到[38]已被删除。

举例如下：

```
<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 hei m Be m'n.</l>
  </sp>
```

xml:lang 所表示的语言选择适用于它所处元素的所有属性和内容，除非被此内容中的元素内的另一个 xml:lang 的实例所覆盖。

xml:lang 的一个简单声明可以采用如下形式：

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

但是如果合适的话，也可以给出特定的缺省值。在一本供英国学生使用的法文诗歌集中，评注和注解使用英语，xml:lang 属性可以这样声明：

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

21.3 逻 辑 结 构

每个 XML 文档包含一个或多个元素，它们的边界用起始标记和结束标记定界，或者，对于空元素，用一个空元素标记分隔。每一个元素有一个用名字标识的类型，有时称之为它的“通用标识符（generic identifier）”（GI），同时它可以有一个属性说明（attribute specification）集。每个属性说明有一个名字和一个值。

元素

```
[39]   element ::=   EmptyElemTag | STag content ETag
  [ WFC: 元素类型匹配 ] [   VC: 元素有效 ]
```

除了那些开头匹配（（'X'|'x'）（'M'|'m'）（'L'|'l'））的名字保留用于本规范的此版本和后继版本的标准化外，本规范不对元素类型和属性的语义、用法和名字（语法之外）作出限制。

格式约束：元素类型匹配。

元素结束标记中的 Name 必须和起始标记中的元素类型相匹配。

有效性约束：元素有效。

如果有一个与 elementdecl 相匹配的声明的 Name 与元素类型相匹配，且下述之一成立时，称此元素是有效的：

（1）此声明与 EMPTY 相匹配，同时此元素没有内容。

（2）此声明与 children 相匹配，同时子元素的序列属于内容模型中的正则表达式所产生的语言，在每对子元素间允许有空白域（匹配非终结符 S 的字符）。注意，仅包括空白的 CDATA 段不匹配非终结符 S，因此不能在这些位置出现。

（3）此声明与 Mixed 相匹配，同时内容由其类型匹配内容模型中的名字的字符数据和子元素组成。

（4）此声明与 ANY 相匹配，同时每个子元素的类型均已声明。

21.3.1 起始标记，结束标记和空元素标记

每一个非空 XML 元素以一个起始标记作为开始的标记。

1.起始标记

[40] STag ::= '<' Name （S Attribute）* S? '>' [WFC: 唯一的属性说明]

[41] Attribute ::= Name Eq AttValue [VC: 属性值类型]

[WFC: 无外部实体引用]

[WFC: 在属性值中没有<]

起始标签和结束标签中的 Name 给出了元素的类型。Name-AttValue 对被统称为元素的属性值进行说明；其中每一对中的 Name 被称为属性名；AttValue 的内容（在“或”定界符间的文本）被称为属性值。注意，在起始标签和空元素标签中各个属性值声明的次序没有意义。

格式约束：唯一的属性说明

一个属性名只能在同一个起始标记或空元素标记中出现一次。

有效性约束：属性值类型

属性必须被声明，其值必须是所声明的类型。

格式约束：无外部实体引用

属性值不能包含对外部实体直接或间接的实体引用。

格式约束：在属性值中没有<

在一个属性值中直接或间接引用的实体的置换文本（除了“<”）不能包含<。

起始标记的一个例子：

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

由一个起始标记开始的每一个元素必须用一个结束标记标记其结束，结束标记中的名字必须与起始标记中给出的元素类型相同：

2. 结束标记

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

结束标记的一个例子：

```
</termdef>
```

在起始标记和结束标记中的文本被称为元素的内容：

元素的内容

[43] content ::= CharData? ((element | Reference | CDSect | PI | Comment) CharData?)* /* */

如果一个元素为空，它必须表示为一个起始标记紧跟一个结束标记或空元素标记。一个空元素标记采用一种特殊的形式：

空元素标记

[44] EmptyElemTag ::= '<' Name （S Attribute）* S? '/>' [WFC: 唯一的属性说明]

不论元素是否用关键字 EMPTY 声明，空元素标记都可以用于任何没有内容的元素。出于互操作性考虑，空元素必须用于，且只能用于声明为 EMPTY 的元素。

空元素的例子：

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

21.3.2 元素类型声明

出于验证的目的，可以用元素类型和属性表声明限制 XML 文档中元素的结构。元素类型声明限制了元素的内容。

元素类型声明通常限制了子元素的类型。由用户选择，当声明提到的元素类型没有相应的声明时，XML 处理器可以给出警告，但这不是一个错误。

元素类型声明形式如下：

[45] elementdecl ::= '<!ELEMENT' S Name S contentspec S? '>' [VC: 唯一的元素类型声明]

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

其中 Name 给出了所声明的元素类型。

有效性约束：唯一的元素类型声明

元素类型只能声明一次。

元素类型声明的例子：

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

1.元素内容

当某一类型的元素只能包含用可选空白域（匹配非终结符 S）分隔的子元素（无字符数据）时，此元素类型具有元素内容。在这种情况下，有内容模型作为类型限制之一，内容模型是决定子元素类型和子元素出现顺序的一种简单语法。此语法用内容粒子（cp）构建，内容粒子由名字，内容粒子的选择表（choice list）或内容粒子的序列表（sequence list）组成：

元素内容的模型

[47] children ::= (choice | seq) ('?' | '*' | '+')?

[48] cp ::= (Name | choice | seq) ('?' | '*' | '+')?

[49] choice ::= '(' S? cp (S? '|' S? cp)* S? ')' [VC: 严格的组/参数实体嵌套]

[50] seq ::= '(' S? cp (S? ',' S? cp)* S? ')' [VC: 严格的组/参数实体嵌套]

其中每一个 Name 是可以作为子元素的元素的类型。选择表中出现的任意内容粒子在元素内容中允许出现的位置对应于选择表在语法中的位置。序列表中出现的所有内容粒子必须以相同的顺序出现在元素内容中。在名字或表之后的可选字符（optional character）决定了表中元素或内容粒子可以出现一次或多次（+），还是零次或多次（*），或是零次或一次（?）。没有这样一个操作符意味着元素或内容粒子必须恰好出现一次。这种句法和意义和本规范中的产生式中所使用的相同。

当且仅当一个元素的内容可以通过满足内容模型中的选择，序列和重复操作符得到，并且内容中的每一个元素与内容模型中的一种元素类型相匹配时，称此元素的内容与一个内容模

型相匹配。出于兼容性考虑， 如果文档的某个元素可以和内容模型中的一种元素类型多次匹配，这是一个错误。更详细的信息参见“E. 确定型内容模型”。

有效性约束: 严格的组/参数实体嵌套

参数实体的置换文本用括号括起的组严格嵌套。即，如果 choice，seq 或 Mixed 成分的开始或结束括号出现在某个参数实体的置换文本中，两者必须同在此置换文本中。出于互操作性考虑，如果一个参数实体引用出现在 choice，seq 或 Mixed 成分中时，它的置换文本不应为空，同时其置换文本的第一个和最后一个非空字符不应为一个连接符（|或,）。

元素内容模型的例子：

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

2.混合型内容

当某元素类型可以包含字符数据，其间可以随意穿插子元素时，称此元素类型具有混合型内容。在这种情况下，对子元素的类型可能有所限制，但对它们的次序和出现次数没有限制：

混合型内容声明

```
[51]  Mixed ::=  '(' S? '#PCDATA' (S? '|' S? Name)* S? ')*'
                 | '(' S? '#PCDATA' S? ')'                    [ VC: 严格的组/参数实体嵌套 ]
                                                              [ VC: 无重复类型 ]
```

其中 Name 给出了子元素的元素类型。关键字 #PCDATA 来自术语“已析字符数据（parsed character data）”。

有效性约束：无重复类型

同一名字在单个混合型内容声明中只能出现一次。

混合内容声明的例子：

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

21.3.3 属性表声明

属性用于联系名字-值对和元素。属性说明只能在起始标记和空元素标记中出现。属性表声明可以用于：

- 定义与一给定元素类型有关的属性集。
- 确定这些属性的类型限制。
- 提供属性的缺省值。

属性表声明详细说明了与给定元素类型相关联的每一个属性的名字，数据类型和缺省值（如果有的话）：

属性表声明

```
[52]  AttlistDecl ::=  '<!ATTLIST' S Name AttDef* S? '>'
[53]  AttDef ::=  S Name S AttType S DefaultDecl
```

AttlistDecl 规则中 Name 是元素的类型。由用户选择，当元素类型中的属性没有被声明时，

XML 处理器可以给出一个警告，但这不是一个错误。AttDef 规则中的 Name 是属性的名字。

当与某个给定元素类型相关的 AttlistDecl 超过一个时，这些声明中的内容被合并在一起。当给定元素类型的某个属性的定义超过一个时，绑定第一个定义，其余定义被忽略。出于互操作性考虑，DTD 的作者可以选择一个给定的元素类型至多有一个属性表声明，一个给定的属性名至多有一个属性定义，以及每个属性表声明至少有一个属性定义。出于互操作性考虑，当一个给定元素有超过一个的属性表声明或一个给定属性有超过一个的属性定义时，由用户选择，XML 处理器可以给出警告，但这不是一个错误。

1.属性类型

XML 属性有三种类型：字符串类型、一组记号化类型和枚举类型。字符串类型可以以任意字面字符串为值; 各个记号化类型有不同的词法和语义约束。文法中指出的有效性约束适用于属性值已按“属性表声明”中所述规范化了之后的情况。

属性类型

[54] AttType ::= StringType | TokenizedType | EnumeratedType

[55] StringType ::= 'CDATA'

[56] TokenizedType ::= 'ID' [VC: ID] [VC: 每种元素类型一个 ID] [VC: ID 属性的缺省值]

| 'IDREF' [VC: IDREF]

| 'IDREFS' [VC: IDREF]

| 'ENTITY' [VC: 实体名]

| 'ENTITIES' [VC: 实体名]

| 'NMTOKEN' [VC: 名字记号]

| 'NMTOKENS' [VC: 名字记号]

有效性约束：ID

ID 类型的值必须匹配 Name 产生式。作为此类型值的名字只能在 XML 文档中出现一次; 即，ID 类型的值必须能唯一标识元素。

有效性约束：每种属性类型一个 ID

每种属性类型只能有一个 ID 属性。

有效性约束：ID 属性的缺省值

ID 属性必须有一个声明为#IMPLIED 或#REQUIRED 的缺省值。

有效性约束：IDREF

IDREF 类型的值必须匹配 Name 产生式，IDREFS 类型的值必须匹配 Names 产生式；每一个 Name 必须匹配 XML 文档中某些元素 ID 属性的值；也就是说，IDREF 类型的值必须匹配某些 ID 属性的值。

有效性约束：实体名

ENTITY 类型的值必须匹配 Name 产生式，ENTITIES 类型的值必须匹配 Names 产生式；每一个 Name 必须匹配 DTD 中声明的未析实体的名字。

有效性约束：名字记号

NMTOKEN 类型的值必须匹配 Nmtoken 产生式; NMTOKENS 类型的值必须匹配 Nmtokens 产生式。

枚举类型的属性可以在声明中提供的取值表中取值。有两种枚举类型：

枚举属性类型

[57] EnumeratedType ::= NotationType | Enumeration

[58] NotationType ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'

[VC：符号属性]

[59] Enumeration ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')' [VC：枚举]

一个 NOTATION 类型的属性标识了一种用于解释与此属性相关的元素的符号，此符号用相关系统或公共标识符在 DTD 中声明。

有效性约束：符号属性

此类型的值必须与声明中所包含的符号名之一相匹配；声明中的所有符号名都必须声明。

有效性约束：每种属性类型一种记法

每种元素类型的 NOTATION 属性不能多于一个。

有效性约束：空元素没有记法

出于兼容性考虑，声明为 EMPTY 的元素不能声明类型为 NOTATION 的属性。

有效性约束：枚举

此类型的值必须与声明中所包含的 Nmtoken 记号之一相匹配。

出于互操作性考虑，同一 Nmtoken 只能在单个元素类型的枚举属性类型中出现一次。

2.属性缺省值

属性声明提供的信息指明了某属性是否必须出现，同时指明了在被声明的属性不是必须出现而文档中没有出现此属性的情况下，XML 处理器应如何处理。

属性缺省值

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

[VC：必须的属性]

[VC：合法的属性缺省值]

[WFC：在属性值中无<]

[VC：固定的属性缺省值]

在一个属性声明中，#REQUIRED 表示必须总是提供此属性，#IMPLIED 表示不提供缺省值。如果声明既不是#REQUIRED，也不是#IMPLIED，那么 AttValue 值包含了所声明的缺省值；关键字#FIXED 规定此属性必须总是有缺省值。如果声明了一个缺省值，当 XML 处理器遇到一个被省略的属性时，它将当成此属性以缺省值出现。

有效性约束：必须的属性

如果缺省值声明是关键字#REQUIRED，那么在所有此类型元素的属性表声明中必须有此属性。

有效性约束：合法的属性缺省值

被声明的属性缺省值必须满足被声明的属性类型的词法约束。

有效性约束：固定的属性缺省值

如果某属性的缺省值用关键字#FIXED 声明，此属性的所有实例必须匹配该缺省值。

属性表声明的例子：

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

3.属性-值对的规范化(Attribute-Value Normalization)

在将属性的值传给应用或检验其有效性之前，XML 处理器必须使用下面的算法（或使用其他能使传给应用的值与用此算法得到的值相同的方法）将其规范化：

（1）所有的行尾必须在输入时如 2.11 行尾处理中所述规范成 #xA，本算法的其余部分作用于以此方法规范化之后的文本。

（2）开始时规范化之后的值包含空字符串。

（3）对于未经规范化的属性值中的每个字符，实体引用或字符引用，从第一个开始，直到最后一个，做如下操作：

- 对于一个字符引用，将其所引用的字符加在规范化之后的值的末尾。
- 对于一个实体引用，对此实体的置换文本递归地使用本算法的第 3 步。
- 对于一个空白字符（#x20, #xD, #xA, #x9），在规范化之后的值的末尾加一个空格字符（#x20）。
- 对于其他字符，将其加在规范化之后的值的末尾。

如果属性值的类型不是 CDATA，那么 XML 处理器必须继续处理规范化之后的值，去掉其前导和尾随空格（#x20）字符，并将空格（#x20）字符序列替换成单个空格（#x20）字符。

注意，如果未经规范化的属性值中包含对空格字符（#x20）以外的空白字符的引用，那么规范化之后的值包含被引用的字符本身（#xD, #xA or #x9），而不是空格（#x20）。这与未经规范化的属性值中包含空白字符（不是引用）的情况不同，在那种情况下空白字符被置换成空格字符（#x20）。同时这也与未经规范化的属性值中所包含的实体引用的置换文本中包含空白字符的情况不同，在那种情况下，实体引用的置换文本被递归处理，空白字符被置换成空格字符（#x20）。

不进行验证的语法分析器应该将所有尚未读到声明部分的属性当成被声明为 CDATA。

以下是属性规范化的例子。有如下声明：

```
<!ENTITY d "&#xD;">
<!ENTITY a "&#xA;">
<!ENTITY da "&#xD;&#xA;">
```

下表中左边一列中的属性值说明在 a 声明为 NMTOKENS 的情况下规范化为中间一列的字符序列，在 a 声明为 CDATA 的情况下规范为右边一列中的字符序列。

属性值说明	a 声明为 NMTOKENS	a 声明为 CDATA
a=" xyz"	x y z	#x20 #x20 x y z

（续表）

属性值说明	a 声明为 NMTOKENS	a 声明为 CDATA
a="&d;&d;A&a;&a;B&da;"	A #x20 B	#x20 #x20 A #x20 #x20 B #x20 #x20
a="

A

B
" | #xD #xD A #xA
#xA B #xD #xA | #xD #xD A #xA #xA B #xD #xD |

注意，在 a 声明为 NMTOKENS 类型的情况下，最后一个例子不是有效的（但是是格式正确的）。

21.3.4 条件段

条件段是文档类型声明外部子集的一部分，取决于相应的关键字，它们或被包含在 DTD 逻辑结构之内，或被排除在 DTD 逻辑结构之外。

条件段

[61] conditionalSect ::= includeSect | ignoreSect

[62] includeSect ::= '<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>' /* */

[VC: 严格的条件段/参数实体嵌套]

[63] ignoreSect ::= '<![' S? 'IGNORE' S? '[' ignoreSectContents* ']]>' /* */

[VC: 严格的条件段/参数实体嵌套]

[64] ignoreSectContents ::= Ignore ('<![' ignoreSectContents ']]>' Ignore)*

[65] Ignore ::= Char* - (Char* ('<![' | ']]>') Char*)

有效性约束: 严格的条件段/参数实体嵌套

如果一个条件段的"<![", "["或 "]]>" 中的任意一个包含在一个参数实体中的置换文本中，它们必须全部在此同一置换文本中。

同内部或外部 DTD 子集一样，条件段可以包含一个或多个完整的声明，注释，处理指令，或嵌套的条件段，其间可以夹杂空白域。

如果条件段的关键字是 INCLUDE，那么条件段的内容是 DTD 的一部分，如果条件段的关键字是 IGNORE，那么条件段的内容逻辑上不是 DTD 的一部分。如果一个关键字为 INCLUDE 的条件段出现在更大的关键字为 IGNORE 的条件段中，内外两个条件段都被忽略。在对被忽略的条件段的内容进行语法分析时，从紧随关键字的 "[" 之后开始，除了条件段的开始 "<![" 和结尾 "]]>" 以外的所有字符都被忽略，直到找到相匹配的条件段结尾。在此过程中参数实体不被识别。

如果条件段的关键字是一个参数实体引用，处理器在决定是否包含或忽略此条件段前，必须先将该参数实体置换成其内容。

一个例子：

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

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

```
<!ELEMENT book (title, body, supplements?)>
]]>
```

21.4 物 理 结 构

一个 XML 文档可能包含一个或多个存储单元。它们被称为实体（entity）；它们都具有内容并且都用名字（name）进行标识（除了下面要提到的文档实体，和外部 DTD 子集之外）。每一个 XML 文档有一个称为文档实体的实体，它作为 XML 处理器处理的起点并可能包含了整个文档。

实体可以是已析的或未析的。已析实体（parsed entity）的内容被称为它的置换文本；此文本被看成是文档整体的一部分。

未析实体（unparsed entity）是一种资源，其内容可以是也可以不是文本，并且如果是文本的话，可以不是 XML。每一个未析实体有一个相关联的用名字标识的符号。除了要求 XML 处理器能向应用提供实体和符号的标识符之外，XML 对未析实体的内容不作任何限制。

已析实体以实体引用的方式使用名字来调用;未析实体用 ENTITY 或 ENTITIES 属性中给出的名字调用。

通用实体（general entity）是那些在文档内容中使用的实体。在本规范中，在不致引起混淆的情况下，普通实体有时用未修饰的术语 entity 来表示。参数实体是用于 DTD 内的已析实体。这两类实体用不同形式的引用，在不同的上下文中识别。另外，它们使用不同的名字空间；具有相同名字的参数实体和通用实体是两个截然不同的两个实体。

21.4.1 字符和实体引用(Character and Entity References)

一个字符引用 ISO/IEC 10646 字符集中的一个特定字符。例如一个不能用输入设备直接输入的字符。

字符引用

```
[66]  CharRef ::=  '&#' [0-9]+ ';'
                 | '&#x' [0-9a-fA-F]+ ';'                    [  WFC：合法字符 ]
```

格式约束：合法字符

用字符引用的字符必须匹配 Char 产生式。

如果字符引用以“&#x”开头的数字和字母（直到终结）提供了某字符在 ISO/IEC 10646 中代码的一个十六进制表示。如果它仅以“&#”开头的数字（直到终结）提供了某字符的代码的十进值表示。

实体引用（entity reference）引用一个命名实体的内容。对已析普通实体的引用使用"and"号（&）和分号（;）作为定界符。参数实体引用则使用百分号（%）和分号（;）作为定界符。

实体引用

```
[67]  Reference ::=  EntityRef | CharRef
[68]  EntityRef ::=  '&' Name ';'                          [  WFC：声明实体 ]
                                                           [  VC：声明实体 ]
                                                           [  WFC：已析实体 ]
```

[WFC：无递归]

[69] PEReference ::= '%' Name ';' [VC：声明实体]

[WFC：无递归]

[WFC：在 DTD 内]

格式正确性约束：声明实体

在一个没有任何 DTD 的文档，或一个只有不包含参数实体引用的内部 DTD 子集的文档，或一个“standalone='yes'”的文档内，在实体引用中给出的 Name 必须与实体声明中所给出的相匹配，但格式良好的文档不需要声明以下的这些实体：amp，lt，gt，apos 和 quot。参数实体的声明必须先于任何对它的引用。类似地，通用实体的声明必须先于任何在属性表声明中的缺省值中出现的对它的引用。注意对于在外部子集或外部参数实体中声明的实体，不进行验证的处理器不必要读取和处理它们的声明;对这些文档，仅当 standalone='yes'时，实体必须被声明的规则才是一个格式约束。

有效性约束：声明实体

在一个有外部子集或外部参数实体且“standalone='no'”的实体中，实体引用中给出的 Name 必须与实体声明中所给出的相匹配。出于互操作性考虑，有效的文档应该以 21.4.6 节中的简化形式声明实体 amp，lt，gt，apos 和 quot。参数实体的声明必须先于任何对它的引用。类似地，通用实体的声明必须先于任何在属性表声明中的缺省值中出现的对它的引用。

格式约束：已析实体

实体引用不能包含一个未析实体的名字。未析实体只能在声明为 ENTITY 或 ENTITIES 的属性值中引用。

格式约束：无递归

已析实体不能直接或间接地包含对自身的递归引用。

格式约束：在 DTD 内

参数实体引用只能在 DTD 中出现。

字符引用和实体引用的例子：

```
Type <key>less-than</key> (&#x3C;) to save options.
This document was prepared on &docdate; and
is classified &security-level;.
```

参数实体引用的例子：

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

21.4.2 实体声明（Entity Declaration）

实体以如下方式声明：

实体声明

[70] EntityDecl ::= GEDecl | PEDecl

[71] GEDecl ::= '<!ENTITY' S Name S EntityDef S? '>'

[72] PEDecl ::= '<!ENTITY' S '%' S Name S PEDef S? '>'

[73] EntityDef ::= EntityValue | (ExternalID NDataDecl?)

[74] PEDef ::= EntityValue | ExternalID

实体引用中的 Name 标识了该实体;对于未析实体，ENTITY 或 ENTITIES 属性的值标识了该实体。如果同一实体被声明了不止一次，绑定第一个声明。由用户选择，如果实体被多次声明，XML 处理器可以给出警告。

1.内部实体（Internal Entities）

如果实体定义是一个 EntityValue，被定义的实体被称为内部实体。内部实体没有单独的物理存储对象，实体的内容在声明中给出。注意字面实体值中一些实体和字符引用的处理可能要求产生正确的置换文本。

内部实体是已析实体。

内部实体声明的例子：

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

2.外部实体（External Entities）

如果实体不是内部的，那么它是一个外部实体，声明如下：

外部实体声明

[75] ExternalID ::= 'SYSTEM' S SystemLiteral

| 'PUBLIC' S PubidLiteral S SystemLiteral

[76] NDataDecl ::= S 'NDATA' S Name [VC：声明符号]

如果有 NDataDecl，那么这是一个通用未析实体；否则它是一个已析实体。

有效性约束：声明符号

Name 必须与符号的名字相匹配。

SystemLiteral 被称为该实体的系统标识符。这是一个 URI 引用（在 [IETF RFC 2396] 中定义，在 [IETF RFC 2732] 中更新），可以由此获得 XML 处理器的输入用于构建此实体的置换文本。] 片断标识符（以 # 开头）出现在系统标识符中是一个错误。如果一个片断标识符作为系统标识符的部分给出，XML 处理器可以给出一个错误。除非在本规范范围之外另外给出（如，一个特殊 DTD 中定义的专用 XML 元素类型，或一个特殊应用规范中定义的处理指令），相对 URI 指相对于实体声明所在资源的位置。因此，一个 URI 可能是相对于文件实体，或相对于包含外部 DTD 子集的实体，或相对于其他一些外部参数实体。

URI 引用需要对某些字符进行编码和转义。不允许出现的字符包括所有非 ASCII 字符，以及 [IETF RFC 2396] 前面列出的不被允许的字符，井号（#）、百分号（%）） 和 [IETF RFC 2732] 中允许的方括号除外。不被允许的字符必须用如下的方法转义：

（1）每个不被允许的字符首先被转换成一个或多个字节的 UTF-8 [IETF RFC 2279] 编码。

（2）任何对应于一个不被允许的字符的八位组用 URI 转义机制转义（即，将其转换成 %HH，其中 HH 是字节值的十六进制记法）。

（3）用得到的字符序列置换原来的字符。

除了系统标识符之外，外部标识符还可以包含公共标识符。试图查找实体内容的 XML 处

理器可以用公共标识符试着产生一个可选 URI。如果处理器无法做到这一点，它必须使用系统常量中定义的 URI。在试着匹配之前，公共标识符中所有空白字符串必须被规范为单个空格字符（#x20），同时必须去掉前导和尾随空白。

外部实体声明的例子：

```
<!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 >
```

21.4.3 已析实体（Parsed Entities）

1. 文本声明（Text Declaration）

每个外部已析实体可以以文本声明作为开始。

文本声明

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

文本声明必须以字面形式给出，而不能使用已析实体的引用。文本声明只能在外部已析实体的开头出现，不允许在其他任何地方出现。

2. 格式良好的已析实体（Well-Formed Parsed Entities）

如果文档实体匹配 document 产生式，那么它是格式良好的。如果外部通用已析实体匹配 extParsedEnt 产生式，那么它是格式良好的。如果外部参数实体匹配 extPE 产生式，那么它是格式良好的。

3. 格式良好的外部已析实体

如果文件实体匹配 document 产生式，那么它是格式正确的。如果外部普通已析实体匹配 extParsedEnt 产生式，那么它是格式正确的。如果外部参数实体匹配 extPE 产生式，那么它是格式正确的。根据定义，外部参数实体是格式正确的。

格式正确的外部已析实体

[78] extParsedEnt ::= TextDecl? content

[79] extPE ::= TextDecl? ExtSubsetDecl

如果内部普通已析实体的置换文本匹配 content 产生式，那么它是格式正确的。根据定义，所有内部的参数实体都是格式正确的。

实体符合格式正确性的一个结果是 XML 文件的逻辑和物理结构是严格嵌套的；起始标签、结束标签、空元素标签、元素、注释、处理指令、字符引用或实体引用都不能在一个实体中开始而在另一个实体中结束。

4. 实体中的字符编码（Character Encoding in Entities）

XML 文档中的每个外部已析实体都可以对其字符采用一种不同的编码方案。所有 XML 处理器必须能读编码为 UTF-8 或 UTF-16 的实体。

以 UTF-16 编码的实体必须以 ISO/IEC 10646 增补 E 和 Unicode 附录 B（零宽度不间断空格字符，#xFEFF）中所描述的字节次序标记（Byte Order Mark）开头。这是一个编码签名，即不是 XML 文档中标记的一部分，也不是 XML 文档字符数据的一部分。XML 处理器必须能用此字符区分 UTF-8 编码和 UTF-16 编码的文档。

虽然 XML 处理器只被要求能读取 UTF-8 和 UTF-16 编码的实体，普遍认为国际上还有其他编码方案。有时可能想让 XML 处理器读取以那些编码方案编码的实体。在没有外部字符编码信息（如 MIME 头）的情况时，以不同于 UTF-8 和 UTF-16 的编码方案存储的实体必须以包含编码声明的文本声明开头：

编码声明

[80] EncodingDecl ::= S 'encoding' Eq ('"' EncName '"' | "'" EncName "'")

[81] EncName ::= [A-Za-z] ([A-Za-z0-9._] | '-')* /* 编码方案的名字只包含拉丁字母 */

在文档实体中，编码声明是 XML 声明的一部分。EncName 是所用编码方案的名字。

在一个编码声明中，值"UTF-8"，"UTF-16"，"ISO-10646-UCS-2"和"ISO-10646-UCS-4"应该用于表示 Unicode 或 ISO/IEC 10646 中的各种不同编码和变换方案，值"ISO-8859-1"，"ISO-8859-2"，... "ISO-8859-9"应该用于表示 ISO 8859 的各个部分，而值"ISO-2022-JP"，"Shift_JIS"和"EUC-JP"应该用于表示 JIS X-0208-1997 的各种编码。XML 处理器可以识别其他编码方案; 建议对于在 Internet Assigned Numbers Authority [IANA]注册的字符编码方案(以字符集(charset)的方式)，除了以上所列的之外，引用时应使用其注册名。其他的编码应该使用带 "x-" 前缀的名称。欲与之匹配的 XML 处理器应该以大小写敏感的方式对字符编码的名称进行匹配。而且 XML 处理器处理字符编码的名称时，应该将在 IANA 注册的编码名称解释为在 IANA 注册的相应编码，不然就应该当成未知的编码（当然，不要求处理器支持所有在 IANA 注册的编码）。

在缺少外部传输协议（如 HTTP 或 MIME）所提供的信息时，以下情况均是错误：XML 处理器接收到的实体的编码方案与实体所含编码声明中指出的编码方案不同，编码声明不在外部实体的开头，既不以字节次序标记开头也不以编码声明开头的实体使用了不同于 UTF-8 的编码。注意因为 ASCII 是 UTF-8 的一个子集，以普通 ASCII 编码的实体不严格需要编码声明。

TextDecl 出现在外部实体开头以外的地方是一个严重错误。

当 XML 处理遇到的实体使用了它不能处理的编码时，是一个严重错误。如果一个 XML 实体被确认为使用了某种编码（由默认值、编码声明或高层协议确定），但是它包含了在此编码中非法的八位组序列的话，是一个严重错误。如果一个 XML 实体没有编码声明而它的内容不是合法的 UTF-8 或 UTF-16 编码的话，也是一个严重错误。

编码声明的例子：

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

21.4.4 XML 处理器对实体和引用的处理

下表汇总了字符引用、实体引用和对未析实体的调用可以出现的上下文，以及每种情况下 XML 处理器要求的动作。最左边一列的标签记指明了识别时的上下文：

内容中的引用 可以在元素的起始标记之后，结束标记之前的任何地方以引用形式出现，对应于非终结符 content。

属性值中的引用 可以在起始标记内的属性值中，或属性声明内的缺省值中以引用形式出现;对应于非终结符 AttValue。

作为属性值 可以以 Name 而不是以引用的形式出现，作为声明为 ENTITY 类型的属性的值，或可以作为声明为 ENTITIES 类型的属性值中的以空白分隔的记号之一。

实体值中的引用 可以在参数中或内部实体的实体声明内的字面实体值中以引用形式出现；对应于非终结符 EntityValue。

DTD 中的引用 可以在 DTD 的内部或外部子集中以引用形式出现，但需在 EntityValue 和 AttValue 之外。

	实体类型				字符
	参数	内部通用	外部已析通用	未析	
内容中的引用	不被识别	被包含	进行验证时被包含	被禁止	被包含
属性值中的引用	不被识别	作为常量被包含	被禁止	被禁止	被包含
作为属性值	不被识别	被禁止	被禁止	通知	不被识别
实体值中的引用	作为常量被包含	不处理	不处理	被禁止	被包含
DTD 中的引用	作为参数实体被包含	被禁止	被禁止	被禁止	被禁止

1. 不被识别（Not Recognized）

在 DTD 之外，百分号字符%没有特殊含义;因此在 DTD 中的参数实体引用在 content 中不被当成标记识别。类似地，除非未析实体的名字出现在已适当声明的属性的值中，否则它们不被识别。

2. 被包含（Included）

当一个实体的置换文本代替引用，被当成引用文档的一部分一样被查找和处理时，称此实体被包含。其置换文本可以包含字符数据和标记（不包括参数实体），其中标记必须以通常的方式识别，但用于转义标记定界符（实体 amp，lt，gt，apos 和 quot）的实体的置换文本总是被当成数据（字符串“AT&T;”展开为“AT&T;”尚存的“and”号&不被识别为实体引用的定界符）。当被表示的字符代替引用被处理时，称此字符引用被包含。

3. 进行验证时被包含（Included If Validating）

当 XML 处理器识别出一个对已析实体的引用，为了验证该文档，处理器必须包含此实体的置换文本。如果实体是外部的，而处理器不试图验证该 XML 文档，那么处理器可以，但不是必须，包含此实体的置换文本。如果一个不验证的解析器不包含此置换文本，它必须通知应用它识别出但没有读取此实体。

这条规范基于这样一个共识：由 SGML 和 XML 的实体机制提供的起初设计用于支持模块化创作的自动包含不一定适合于其他应用，尤其是文档浏览。例如，当浏览器遇到一个外部已析实体引用时，可能选择用可视方式表示其存在但只在被请求时才查找它进行显示。

4. 被禁止（Forbidden）

以下情况被禁止，并构成一个严重错误：

- 出现对未析实体的引用。
- 在 DTD 中出现任何字符或通用实体引用，除非它们出现在 EntityValue 或 AttValue 中。
- 属性值中出现对外部实体的引用。

5. 作为常量被包含(Included in Literal)

当实体引用出现在属性值中或参数实体引用出现在字面实体值中时，它们的置换文本代替引用被当成引用文档的一部分一样被查找和处理，但是置换文本中的单双引号总是被当成正常的数据字符而不会结束此常量。例如，下面的例子是格式良好的：

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

而这个例子不是格式良好的：

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

6. 通知（Notify）

当未析实体名字作为记号在声明为 ENTITY 或 ENTITIES 类型的属性的值中出现时，进行验证的处理器必须将此实体和它的相关符号的系统和公共（如果有的话）标识符通知给应用。

7. 不处理（Bypassed）

当实体声明内一个通用实体引用出现在 EntityValue 中时，它不被处理，保持不变。

8. 作为参数实体被包含(Included as PE)

和外部已析实体一样，参数实体只需在进行验证时被包含。当参数实体引用在 DTD 中被识别并被包含时，它的置换文本被前后各加上一个空格字符;其目的在于强制参数实体的置换文本在 DTD 中包含完整的的语法记号。

21.4.5 内部实体置换文本的构建（Construction of Internal Entity）

在讨论内部实体的处理时，区分两种形式的实体值是有帮助的。字面实体值（literal entity value）是实际出现在实体声明中用引号括起的字符串。对应于非终结符 EntityValue。置换文本（replacement text）是置换了字符引用和参数实体引用后的实体内容。

在内部实体声明（EntityValue）中给出的字面实体值可以包括字符引用、参数实体引用和通用实体引用。这些引用必须被整个包含于字面实体值中。如前述方式被包含的实际置换文本必须包含所有被引用的参数实体的置换文本，同时在字面实体值中必须包含所有代替字符引用的字符。但通用实体的引用必须保持不变，不被展开。例如，如果有以下的声明：

```
<!ENTITY % pub       "&#xc9;ditions Gallimard" >
```

```
<!ENTITY     rights "All rights reserved" >
<!ENTITY     book     "La Peste: Albert Camus, &#xA9; 1947 %pub;. &rights;" >
```

那么实体“book”的置换文本为：

```
La Peste: Albert Camus, ?nbsp;1947 ditions Gallimard. &rights;
```

一旦引用“&book;”出现在文档的内容或属性值中时，通用实体引用“&rights;”应该被展开。

这些简单的规则将可能会有复杂的相互作用。

21.4.6 预定义实体（Predefined Entities）

实体和字符引用都可以用于转义左尖括号，“and”号（&）和其他定界符。通用实体集合（amp，lt，gt，apos，quot）专门用于此目的。也可以使用数值字符引用；一旦被识别，它们立即被展开，同时它们必须被当成字符数据，因此数值字符引用“<”和“&”可以用于转义出现在字符数据中的<和&。

不管这些实体是否被声明，所有的 XML 处理器必须能识别它们。出于互操作性考虑，如其他实体一样，有效的 XML 文档应该在使用这些实体前先声明它们。如果声明的话，这些实体必须被声明为内部实体，其置换文本是被转义的单个字符或指向这个字符的字符引用。如下所示：

```
<!ENTITY lt       "&#38;#60;">
<!ENTITY gt       "&#62;">
<!ENTITY amp      "&#38;#38;">
<!ENTITY apos     "&#39;">
<!ENTITY quot     "&#34;">
```

21.4.7 符号声明（Notation Declarations）

符号用名字标识了未析实体的格式，具有符号属性的元素的格式以及处理指令所针对的应用的格式。

符号声明赋予符号一个名字用于实体,属性表声明和属性说明中，同时也给出了一个符号的外部标识符使得 XML 处理器或它的客户应用可以定位能以给定符号处理数据的助理应用。

符号声明

```
[82]   NotationDecl ::=   '<!NOTATION' S Name S (ExternalID |   PublicID) S? '>'
                                                                  [VC: 唯一的记法名字]
[83]   PublicID ::=   'PUBLIC' S PubidLiteral
```

XML 处理器必须向应用提供任何在属性值中，属性定义中或实体声明中定义或引用的符号的名字和外部标识符。它们还可以将外部标识符解析成系统标识符、文档名，或是应用调用相应处理器处理给定符号格式的数据的所需的其他信息（但如果 XML 处理器或应用所运行的系统中没有处理 XML 文档声明和引用的符号的相应应用的情况，不是一个错误）。

21.4.8 文档实体（Document Entity）

文档实体（document entity）是实体树的根和 XML 处理器的处理起点。本规范没有规定 XML 如何定位文档实体；与其他实体不同，文档实体没有名字，而且可以完全不带任何标识地出现在处理器的输入流中。

21.5 一致性（Conformance）

21.5.1 进行验证和不进行验证的处理器（Validating and Non-Validating Processors）

合乎规范的 XML 处理器可以分为两类：进行验证的和不进行验证的。

进行验证和不进行验证的处理器都必须报告在文档实体的内容中和任何其他它们读到的已析实体中对格式约束的违反。

进行验证的处理器必须报告违反 DTD 声明中所述约束的情况以及不满足本规范中给出的有效性约束的情况。要完成这一点，进行验证的 XML 处理器必须读取和处理整个 DTD 和所有在文档中引用的外部已析实体。

不进行验证的处理器只被要求检查文档实体和整个内部 DTD 子集的格式。虽然它们不被要求检查文档的有效性，但它们必须处理它们读取的所有内部 DTD 子集和参数实体中的声明，直到遇到第一个没有读取的参数实体的引用;也就是说，它们必须根据这些声明中的信息规范化属性值，包含内部实体的置换文本，并提供缺省属性值。它们在遇到第一个没有读取的参数实体的引用后，不应处理其后的实体声明或属性表声明，因为此实体中包含的声明可能覆盖前面的声明。

21.5.2 使用 XML 处理器

进行验证的处理器的行为是高度可预测的；它必须读取文档的所有部分，报告所有对格式和有效性的违反。对一个不进行验证的处理器的要求要低一点；它不需要读取文档实体以外的任何文档部分。这对 XML 的处理器的用户而言可能会有两个重要的影响：

- 某些格式错误，尤其是那些要求读取外部实体的，可能不会被不进行验证的处理器检测到。例如称为声明实体、已析实体和无递归的约束。
- 取决于处理器是否读取参数和外部实体，从处理器传给应用的信息可能会有所不同。例如，不进行验证的处理器可能不规范化属性值，不包含内部实体的置换文本，或不提供缺省属性值，这些动作要求先读取外部或参数实体中的声明。

为了使不同 XML 处理器间的互操作有最大的可靠性，使用不进行验证的处理器的应用不应依赖于不要求这些处理器具备的动作。那些要求使用如缺省值或在外部实体中声明内部实体等功能的应用应该使用进行验证的 XML 处理器。

21.6 符号（Notation）

本规范中 XML 的正式句法用一种简单的扩展巴科斯范式（Extended Backus-Naur Form，EBNF）给出。句法中的每一条规则定义了一个记号，形式如下：

symbol ::= expression

如果记号用正则表达式定义，则它以大写字母开头，否则以小写字母开头。字符串（literal strings）用引号括起。

在规则右边的表达式中，以下表达式用于匹配一个或多个字符的字符串：

#xN

N 是一个十六进制的整数，当 ISO/IEC 10646 中某个字符的规范（UCS-4）代码值作为无符号二进制数与 N 相等时，此表达式匹配这个字符。#xN 中的前导 0 没有意义，在相应的代码值中的前导 0 的个数则由所用字符编码方案决定，对 XML 没有意义。

[a-zA-Z]， [#xN-#xN]

与其值在指定范围内的任何字符相匹配（含界，inclusive）。

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

与其值为所枚举的值之一的 Char 相匹配。在一对方括号内枚举和范围可以混用。

[^a-z]， [^#xN-#xN]

与其值在指定范围之外的任何 Char 相匹配。

[^abc]， [^#xN#xN#xN]

与任何不在给定字符集内的 Char 相匹配。在一对方括号内被禁值的枚举和范围可以混用。

"string"

与双引号中字符串相匹配。

'string'

与单引号中字符串相匹配。

这些符号可以按下列方式组合，以匹配更复杂的模式，其中 A 和 B 表示简单表达式：

(expression)

expression 被当成一个单元，可以如本表描述进行组合。

A?

与零个或一个 A 相匹配，即 A 可选。

A B

与 A 后跟 B 的模式相匹配。

A | B

与 AB 之一相匹配，但不同时匹配。

A - B

与任何匹配 A 但不匹配 B 的字符串相匹配。

A+

与一个或多个 A 相匹配。

A*

与零个或多个 A 相匹配。

其他在产生式中使用的符号有：

/* ... */

注释

[wfc: ...]

格式约束；用名字标识一个对与某个产生式相关联的格式良好的文档的约束。

[vc: ...]

有效性约束；用名字标识一个对与某个产生式相关联的有效文档的约束。

附　录

A. 参考文献

A.1　标准参考文献

IANA

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

IETF RFC 1766

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

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

与其值为所枚举的值之一的 Char 相匹配。在一对方括号内枚举和范围可以混用。

[^a-z]， [^#xN-#xN]

与其值在指定范围之外的任何 Char 相匹配。

[^abc]， [^#xN#xN#xN]

与任何不在给定字符集内的 Char 相匹配。在一对方括号内被禁值的枚举和范围可以混用。

A.2　其他参考文献

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.)

Br gemann-Klein

Br gemann-Klein, Anne. Formal Models in Document Processing. Habilitationsschrift. Faculty of Mathematics at the University of Freiburg, 1993. (See ftp://ftp.informatik.uni-freiburg. de/documents/papers/ brueggem/habil.ps.)

Br gemann-Klein and Wood

Br gemann-Klein, Anne, and Derick Wood. Deterministic Regular Languages. University Freiburg, Institut ion Informatik, Bericht 38, Oktober 1991. Extended abstract in A. Finkel, M. Jantzen, Hrsg., STACS 1992, S. 173-184. Springer-Verlag, Berlin 1992. Lecture Notes in Computer Science 577. Full version titled One-Unambiguous Regular Languages in Information and Computation 140 (2): 229-253, February 1998.

Clark

James Clark. Comparison of SGML and XML. See http://www.w3.org/TR/NOTE-sgml-xml-

971215.

IANA-LANGCODES

(Internet Assigned Numbers Authority) Registry of Language Tags, ed. Keld Simonsen et al. (See http://www.isi.edu/in-notes/iana/assignments/languages/.)

IETF RFC2141

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

IETF RFC 2279

IETF (Internet Engineering Task Force). RFC 2279: UTF-8, a transformation format of ISO 10646, ed. F. Yergeau, 1998. (See http://www.ietf.org/rfc/rfc2279.txt.)

IETF RFC 2376

IETF (Internet Engineering Task Force). RFC 2376: XML Media Types. ed. E. Whitehead, M. Murata. 1998. (See http://www.ietf.org/rfc/rfc2376.txt.)

IETF RFC 2396

IETF (Internet Engineering Task Force). RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax. T. Berners-Lee, R. Fielding, L. Masinter. 1998. (See http://www.ietf.org/rfc/rfc 2396.txt.)

IETF RFC 2732

IETF (Internet Engineering Task Force). RFC 2732: Format for Literal IPv6 Addresses in URL's. R. Hinden, B. Carpenter, L. Masinter. 1999. (See http://www.ietf.org/rfc/rfc2732.txt.)

IETF RFC 2781

IETF (Internet Engineering Task Force). RFC 2781: UTF-16, an encoding of ISO 10646, ed. P. Hoffman, F. Yergeau. 2000. (See http://www.ietf.org/rfc/rfc2781.txt.)

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 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 Annexe. [Geneva]: International Organization for Standardization, 1996.

WEBSGML

ISO (International Organization for Standardization). ISO 8879:1986 TC2. Information technology -- Document Description and Processing Languages. [Geneva]: International Organization for Standardization, 1998. (See http://www.sgmlsource.com/8879rev/n0029.htm.)

XML Names

Tim Bray, Dave Hollander, and Andrew Layman, editors. Namespaces in XML. Textuality, Hewlett-Packard, and Microsoft. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/REC-xml-names/.)

B. 字符的分类（Character Classes）

根据 Unicode 标准中定义的特征，字符被分为基类字符（其中包括没有变音符的拉丁字母）、表意字符和组合字符（其中包括大多数的变音符）；这些类合起来组成了字母类。数字和扩展符（extender）也各自被分成类。

字符

[84] Letter ::= BaseChar | Ideographic

[85] 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]

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

[87] 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

[88] 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]

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

在此定义的字符类可以从 Unicode2.0 字符库中如下导出：

- 名字的起始字符必须属于 Ll, Lu, Lo, Lt, Nl 中的一类。
- 除了起始字符之外的命名字符必须属于 Mc，Me，Mn，Lm 或 Nd 中的一类。
- 兼容区（即代码大于#xF900，小于#xFFFE 的字符）中的字符不允许在 XML 名字中

出现。

- 不允许出现具有字体或兼容性分解的字符（即数据库中第 5 项有以"<"开始的"compatibility formatting tag"的字符 ）。
- 下列字符被当成名字起始字符而非名字字符，因为特性文件中将它们归于字母类：[#x02BB-#x02C1]，#x0559，#x06E5，#x06E6。
- 不允许出现字符#x20DD-#x20E0（与 Unicode2.0 的 5.14 节保持一致）。
- 字符#x00B7 被分为扩展符，因为特性文件中对它是这么标识的。
- 字符#x0387 被当成一个命名字符，因为#x00B7 是它的等价规范形式。
- 字符':'和'_'可以作为名字起始字符。
- 字符'-'和'.'可以作为命名字符。

C. XML 和 SGML（非正式）

XML 被设计为 SGML 的一个子集，表现在每一个有效的 XML 文档应该也是一个合乎规范的 SGML 文档。对 XML 在 SGML 之外对文档所加的限制的详细讨论参见[Clark]。

D. 实体和字符引用的展开（非正式）

本附录中举例说明了在 21.4.4 一节中规定的实体和字符引用的识别和展开的次序。

如果声明包含在 DTD 中。

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

那么 XML 处理器将在对实体声明进行解析时识别出字符引用，并在将下面的字符串存为实体“example”的值前解析这些字符引用：

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

文档中对“&example;”的引用会导致对文本的重新分析，此时元素“p”的起始和结束标记被识别，三个引用被识别和展开，其结果是一个包含下面内容（所有数据、无定界符或标记）的“p”元素：

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

一个更复杂的例子可以完整地说明这些规则和它们的作用。在下面的例子中，行号仅仅是为了方便说明。

```
1 <?xml version='1.0'?>
2 <!DOCTYPE test [
3 <!ELEMENT test (#PCDATA) >
4 <!ENTITY % xx '&#37;zz;'>
5 <!ENTITY % zz '&#60;!ENTITY tricky "error-prone" >' >
6 %xx;
7 ]>
8 <test>This sample shows a &tricky; method.</test>
```

这个例子会导致下列动作：

- 在第 4 行，对字符 37 的引用会被立即展开，参数实体“xx”以值“%zz;”存于记号表中。因为置换文本不被再次扫描，对参数实体“zz”的引用不会被识别（而且如果它被识别的话则是一个错误，因为“zz”还没有被声明）。
- 在第 5 行，字符引用“<”被立即展开，而参数实体“zz”以置换文本“<!ENTITY tricky "error-prone" >”被存储，此置换文本是一个格式良好的实体声明。
- 在第 6 行，对“xx”的引用被识别，“xx”的置换文本（即"%zz;"）被解析。对“zz”的引用随后被识别，它的置换文本（"<!ENTITY tricky "error-prone" >"）被解析。此时通用实体“tricky”被声明，它的置换文本是“error-prone”。
- 在第 8 行，对通用实体"tricky"的引用被识别，并被展开，因此"test"元素的全部内容为一个自我描述的(不合语法)字符串 This sample shows a error-prone method。此例表明了一种有错误倾向的方法。

E. 确定型内容模型(非正式)

元素类型声明中的内容模型要求是确定型的。这个要求是出于和 SGML 的兼容性考虑（SGML 称为"无歧义的"）；用 SGML 系统生成的 XML 处理器可能会把非确定型内容模型标为错误。例如，内容模型（（b, c）｜（b, d））是非确定型的，因为给定一个初始 b，解析器没有在向前看以知道 b 后是什么元素之前，无法知道匹配模型中的哪个 b。在这种情况下，两个对 b 的引用可以简化成单个的引用，使得模型成为（b,（c｜d））。此时初始的 b 只和内容模型中的一个名字明确匹配。解析器不需要向前看其后的内容。c 或 d 都能被接受。

更正式的说法：使用 Aho，Sethi 和 Ullman 所著[Aho/Ullman]3.9 节中的标准算法 3.5，可以从内容模型构造出一个有限状态自动机。在很多这样的算法中，对应正则表达式中的每一个位置（即正则表达式的语法树中的每个叶子节点），都构造一个随集（follow set）；如果任一位置的随集中不止一个后继位置被标为同一元素类型时，那么此内容模型出错，并且可以被报为错误。

存在将许多但不是所有非确定型内容模型自动规约为等价的确定型模型的算法；参见 Brgemann-Klein 1991 [Brgemann-Klein].

F. 字符编码的自动检测（非正式）

XML 编码声明在实体中以内部标签的方式工作，用于指出使用了何种字符编码。然而，在 XML 处理器能读取这个内部标签前，显然它必须知道当前使用的是何种字符编码——而这正是此内部标签要试图指出的。通常情况下，这是一种无法解决的情况。但在 XML 中并非如此，因为 XML 在两个方面对这种情形作出了限制：假定每一种实现只支持一个有限的字符编码集，并且，为了使得正常情况下自动检测每个实体中所用字符编码成为可能，限制了 XML 编码声明的位置和内容。同时，很多情况下除了 XML 数据流本身之外，另外还有可用的信息源。根据 XML 实体交给处理器时没有或有任何的附带（外部）信息，可以区分出两种情况。我们先考虑第一种情况。

F.1　无外部编码信息时的检测

因为每一个没有外部编码信息且非 UTF-8 或 UTF-16 格式的 XML 实体必须以 XML 编码声明开头，其开始的几个字符必须为'<?xml'，任何合乎规范的处理器可以在两到四个八位组的输入后，检测出适用于下列何种情况。在读这张表时，知道这些是有帮助的：在 UCS-4 中，'<'是"#x0000003C"，'?'是"#x0000003F"，UTF-16 数据流的字节次序标记要求为"#xFEFF"。记法 ## 用于表示任意的字节值，但两个连续的 ## 不能同时为 00。

有字节次序标记：

- 00 00 00 3C: UCS-4，big-endian 编码的计算机（1234 次序）
- 3C 00 00 00: UCS-4，little-endian 编码的计算机（4321 次序）
- 00 00 3C 00: UCS-4，异常的八位组次序（2143）
- 00 3C 00 00: UCS-4，异常的八位组次序（3412）
- FE FF: UTF-16，big-endian
- FF FE: UTF-16，little-endian

无字节次序标记：

- 00 00 00 3C
3C 00 00 00
00 00 3C 00
00 3C 00 00　　UCS-4 或其他 32 位码元的编码，同时 ASCII 字符的码值就是 ASCII 值，次序分别为 big-endian（1234），little-endian（4321）和两种异常的字节次序（2143 和 3412）。必须读取编码声明以确定使用的是 或其他被支持的 32 位编码。
- 00 3C 00 3F　　UTF-16BE，big-endian 的 ISO-10646-UCS-2 或其他 16 位码元的 big-endian 的编码，其中 ASCII 字符的码值就是 ASCII （必须读取编码声明以确定使用的是哪一种）。
- 3C 00 3F 00　　UTF-16LE，little-endian 的 ISO-10646-UCS-2 或其他 16 位码元的 little- endian 的编码，其中 ASCII 字符的码值就是 ASCII （必须读取编码声明以确定使用的是哪一种）。
- 3C 3F 78 6D　　UTF-8，ISO 646，ASCII，ISO 8859 的一些部分，Shift-JIS，EUC，及其他任何 7 位，8 位或混合宽度的能保证 ASCII 字符有它们正常的位置，宽度，取值的编码；具体其中哪一个适用需读取实际的编码声明来检测确定，但因为所有这些编码中 ASCII 字符的位模式相同，所以能够可靠地读取编码声明本身。
- 4C 6F A7 94　　EBCDIC（在某些变种中，完整的编码声明必须能用于确定使用了哪一个代码页）。
- 其他　　没有编码声明则为 UTF-8，否则不是数据流被标错了（没有所需的编码声明），被损坏了，是不完整的，就是被包含在某种外层数据中。

注：在上述不需要读取编码声明来确定所用编码的情况下，4.3.3 节仍然要求读取可能出现的编码声明，并检查其中的编码名称是否与实体实际的编码相一致。同时，现在不要求有编码声明的情况有可能会因为新的字符编码方案的发明而变得必须使用编码声明用于确定所用的编码。

这种层次的自动检测足以用于读取 XML 编码声明和解析字符编码标识符。字符编码标识符仍然是必须的，它用于区分编码方案集中的单个成员（例如从 8859 中区分出 UTF-8，8859 各个部分间的相互区分，以及区分所用的特定 EBCDIC 代码页，等等）。

因为编码声明的内容限于 ASCII 字符，一旦处理器检测到使用的是哪一个编码方案集，它能够可靠地读取整个编码声明。因为在实际中，所有广泛使用的字符编码都可以归于上述种类中，XML 编码声明保证了可靠的内嵌(in-band)字符编码标注，即使是在操作系统或传输协议级的外部信息源并不可靠的情况下。像 UTF-7 那样重用了 ASCII 编码值的字符编码方案有可能无法可靠地被检测。

一旦处理器检测到所用的字符编码，它就可以作出合适的动作，或是针对每种情况调用单独的输入例程，或是对每个输入的字符调用版本合适的转换函数。

和任何自标注（self-labeling）的系统一样，一旦任何软件改变了实体的字符集或其编码而没有相应修改编码声明的话，XML 的编码声明将无法工作。字符编码方案的实现者必须小心，以保证用于标注实体的内部和外部信息的正确性。

F.2 外部编码信息的优先级

第二种可能的情况是 XML 实体有附带的信息，如在一些文档系统和网络协议中。当具有多个信息源时，它们间的相对优先级和首选冲突处理方法必须在传输 XML 的高层协议中给出。例如，内部标签和外部文档头中的 MIME 类型标签的相对优先级应该是定义 text/xml 和 application/xml MIME 类型的 RFC 文档的一部分。然而出于互操作性考虑，建议使用下列规则：

- 如果 XML 实体是在一个文档中，用字节次序标记和编码声明 PI（如果有的话）来确定字符编码。所有其他信息源和推断都仅仅用于错误恢复。
- 如果 XML 实体传递时标为 text/xml MIME 类型，那么 MIME 类型的 charset 参数决定了字符编码方法;所有其他信息源和推断都仅仅用于错误恢复。
- 如果 XML 实体传递时标为 application/xml MIME 类型，那么用字节次序标记和编码声明 PI（如果有的话）来确定字符编码。所有其他信息源和推断都仅仅用于错误恢复。

这些规则只适用于缺少协议级文档时的情况；特别是，当相关 RFC 中定义了这些 text/xml 和 application/xml MIME 类型时，RFC 中的建议取代这些规则。

G. W3C XML 工作组（非正式）

本规范由 W3C XML 工作组（WG）完成并批准发表。工作组批准了本规范并不一定表示工作组的所有成员一致同意本规范。现有和以前的 XML 工作组成员包括：

- 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, SoftQuad, Grif SA, Muzmo and Veo Systems
- MURATA Makoto (FAMILY Given), Fuji Xerox Information Systems
- Joel Nava, Adobe
- Conleth O'Connell, Vignette
- Peter Sharpe, SoftQuad
- John Tigue, DataChannel

H. W3C XML 核心工作组（非正式）

本规范的第二版由 W3C XML 核心工作组完成。在此版本发表时此工作组的成员包括：

Paula Angerstein, Vignette
Daniel Austin, Ask Jeeves
Tim Boland
Allen Brown, Microsoft
Dan Connolly, W3C (Staff Contact)
John Cowan, Reuters Limited
John Evdemon, XMLSolutions Corporation
Paul Grosso, Arbortext (Co-Chair)
Arnaud Le Hors, IBM (Co-Chair)
Eve Maler, Sun Microsystems (Second Edition Editor)
Jonathan Marsh, Microsoft
MURATA Makoto (FAMILY Given), IBM
Mark Needleman, Data Research Associates
David Orchard, Jamcracker
Lew Shannon, NCR
Richard Tobin, University of Edinburgh
Daniel Veillard, W3C
Dan Vint, Lexica
Norman Walsh, Sun Microsystems
Franis Yergeau, Alis Technologies (Errata List Editor)
Kongyi Zhou, Oracle

I. 文档制作说明（非正式）

第二版用 XMLspec DTD 书写（见其文档）。其 HTML 版本用 xmlspec.xsl，diffspec.xsl 和 REC-xml-2e.xsl XSLT 样式表制成。其 PDF 版本使用 html2ps 和一个 distiller 程序制成。

第二十二章　XML 术语及词汇参考

XML 专业技术词汇表（草案）
XML Professional Tech-Glossary（Draft）
本词汇表提供给读者和 XML 专业技术人员。供大家参考：

A

application（应用）
attribute （属性）
API（Application Programming Interface 应用编程接口）
ADO（ ActiveX Data Objects ActiveX 数据对象）
ANSI （American National Standards Institute 美国国家标准研讨会）
ASP （Active Server Pages 活动服务器页面）

B

C

character（字段）
class（类）
combination（组合）
CAD（Computer Aided Design 计算机辅助设计）
CAM（Computer Aided Manufacturing 计算机辅助制造）
CGI（Common Gateway Interface 公共网关接口）
CDF（Channel Definition Format 频道定义格式）
COM （Component Object Model 结构对象模式）
CORBA（Common Object Request Broker Architecture 共同对象要求处理结构）
CSS（Cascading Style Sheets 层叠样式表）
CML （Chemical Markup Language 化学标识语言）

D

definition（定义）
declaration（声明）
delimiter（定界符）
DTD（Document Type Definition 文件类型定义）
DSSSL（Document Style Semantics and Specification Language 文档样式语义和规范语言）
DOM（Document Object Model 文档对象模式）

DDML（Document Definition Markup Language 文件定义标识语言）
DSO（Data Source Object 数据源对象）

E

encoding （编码）
entity（实体）
EDI（Electronic Data Interchange 电子数据交换）
ECMA（European Computer Manufacturers Association 欧洲计算机协会）
EEMA（European Electronic Messaging Associations 欧洲电子信息协会）

F

G

GCA（Graphic Communications Association of America 美国图形通信协会）

H

HTML（HyperText Markup Language 超文本标识语言）
HGML（Hyper Graphics Markup Language 超图像标识语言）
HTTP（HyperText Transfer Protocol 超文本传输协议）
HyTime（Hypermedia/Time-based Structuring Language - ISO/IEC 10744 超媒体/基于时间的结构语言）

I

identifier（标识符）
implicit（隐含）
instance（实例）
IEC（国际电工委员会）
IETF（Internet Engineering Task Force 工程任务组）
ISUG（International SGML Users' Group 国际 SGML 用户组）
ISO（International Standards Organization 国际标准组织）
IDL （Interface Definition Language 接口定义语言）

J

JVM（Java Virtual Machine Java 模拟机）

K

keyword（关键字）

L

M

map（参照）
model（模式）
MathML（Mathematical Markup Language 数学标识语言）
MCF（Meta Content Framework 元内容格式）
MTS（Microsoft Transaction Server 微软交易服务器）

N

Namespace（名字空间）

O

ODBC（Open Database Connectivity 开放数据库连接）
OSD（Open Software Description 开放软件描叙）
OTP（Open Trading Protocol 开放网络贸易协议）
OFX（Open Financial Exchange 开放式金融交易）
OPS （Open Profiling Standard 开放轮廓标准）
OQL（Object-orientated Query Language 面向对象查询语言）
OFE （Open Financial Exchange 开放金融交换）
OFX （Open Financial Exchange 开放金融交换）

P

parameter（参数）
P3P（Platform for Privacy Preferences 私有参数平台）
PDF（Portable Document Format 便携文件形式）
PNG（Portable Network Graphics 小型网络图形）
PGML（Precision Graphics Markup Language 精密图像标识语言）
Parser（解析器）

Q

quantity（量）

R

record（纪录）
RDF （Resource Description Framework 资源描叙框架）
RFC（Request For Comments 注释要求）
RMD （Required Markup Declaration 必需的标识语言）
RTF（Rich Text Format 丰富文本形式）

S

set（集合）
sequence（序列）
separator（分隔符）
space（空格）
subelement（字元素）
syntax（语法）
SGML （Standard Generalised Markup Language 通用标识语言标准）
SAX（Simple API for XML XML 的简单应用编程接口）
SP（SGML Parser SGML 解析器）
SQL（Structured Query Language 结构查询语言）
SMIL （Synchronized Multimedia Integration Language 同步多媒体综合语言）
STEP（Standard for the Exchange of Product Model Data 产品类型数据交换标准）
SVG （Scalable Vector Graphics 可升级矢量图形）
SDML （Signed Document Markup Language 有符号文件标识语言）
Schema（大纲）
Stylesheet（样式表）

T

Tag（标签）
text（文本）
token（助记符）

U

upper-case（大写）
UCS（Universal Character Set 通用字符集）
URL（Uniform Resource Locator 通用信息定位器）
URI （Universal Resource Identifier 通用源识别器）

V

virtual（虚拟）
VML （Vector Markup Language 矢量标识语言）
VRML （Virtual Reality Modeling Language 虚拟现实造型语言）
Valid（合法）

W

W3C（World Wide Web Consortium 万维网络联盟）
WFC （Windows Foundation Classes 窗口基础类）

WIDL （Web Interface Definition Language 网络接口定义语言）
WSP（Web Standards Project 网络标准项目）
WWW（World Wide Web 万维网络）
Well_formed（结构完整）

X

XML（The Extensible Markup Language 可扩展标识语言）
XLL（XML Linking Language 可扩展链接语言）
XSL（XML Style Language 可扩展类型语言）
XHTML （Extensible HyperText Markup Language 可扩展超文本标识语言）
XPath（XML Path Language XML 路径语言）
XPointer （XML Pointer Language XML 指针语言）
XFDL （Extensible Forms Description Language 可扩展创体描叙语言）

Y

Z

第二十三章　XML 技术动态

23.1　XML 1999 技术动态

3 月

3 月 21 日

Oracle 在 Oracle TechNet 上第一次发布了为 Java 服务的 Oracle XML 解析器的 bug 修补器（未注册）。

3 月 22 日

MDSAX 1.0 版本产品发布了。MDSAX 是一组程序员级的工具。它用来支持 Java SAX 解析器和解析过滤起的工作。

3 月 30 日

Tom Harding 应用可扩展协议起草了一份 Java 的应用实现。这个协议是一个纯 XML 的协议。它支持发送和接收 XML 文件时系统稳固的连接。

4 月

4 月 1 日

Java 中的 OpenXML（开放式 XML） XML parser（分析器）1.0.5 版本面世。

4 月 4 日

Jonathan Robie's 建立了一个讨论 XQL（XML 询问语言）的邮件列表。这个讨论组主要是回答关于语言定义的问题，怎样应用的问题，怎样有产品实现的问题。它致力于在将来的 XQL 发展中实现一致。

4 月 19 日

Richard Tobin 在 XML parser（分析器）RXP 中增加了 namespace（名字空间）的支持。

4 月 22 日

W3C 的发布了一个新的 XSL 草稿说明，其中包括对格式化对象的说明。

4 月 23 日

FourThought LLC 已发放第二公众阿尔发版的 4XSL（版本 0.6.1）,它支持大多数 XSL 变

换语言。它兼容 12 月 XSL 的工作草稿。它用 Python 书写并且通过了 Solaris 和 Linux 的测试。

4 月 24 日

Michael Kay 发布了 SAXON 的 4.2 版本。它加入了一个 XSL 的编译器。这个编译器以 XSL 样式表为输入，并以一个 Java 应用程序为输出。这个 Java 应用程序可以再不依赖于原样式表的情况下处理 XML 文件。

这个版本的发布支持了 1998 十一月发布的 XSL 工作草稿。

5 月

5 月 7 日

万维网联盟发布了第一份 XML Schemas（模式）的工作草稿。

5 月 25 日

IBM 公司的用于 C++的 XML 解析工具（XML parser for C++，XML4C）现已支持 Solaris 系统。IBM 的 XML4C 以 C++中一个简便的子集写成。XML4C 使得其应用能方便地读写 XML 的数据。一单独 C++共享库提供类以分析、生成、操纵,并且确认 XML 文档。

了解更多或下载该工具。

5 月 26 日

用改进的 LotusXSL 安排 XML 的格式。XSL 提供在浏览器或者在服务器端格式化和变换 XML 的机制。它允许开发者把抽象数据 XML 的语义学例子变换成简报语言,如 HTML 语言。

6 月

6 月 1 日

Sun 公司已发布了 Java Project X 的技术版本二。该发布版本为 SAX 1.0 增添了完全的支持,修改了多样混合的错误,并且改善了性能。

6 月 8 日

IBM 的 alphaWorks 部门发布了 easyXML Bean Suite，其中包括 XMLHolder、XMLElement、XMLAttribute 和 XMLFileGenerator JavaBeans。easyXML Bean Suite 允许 Java 的应用程序处理 XML。

6 月 17 日

Vertex Industries 公司宣布了支持 IBM MQ Series 的接口（Interface）。Vertex Industries 已计划推出下一代中间件工具的新部件（Component）。这个名为 Vixen 的模块为信息在 XML 文件和有效数据结构之间的传输和翻译提供了有效地手段。经过演变的 XML 支撑技术和它的跨平台的功能使得在不同环境中运作的应用程序之间可以交换数据。VIXEN 正在运用 NetWeave Middleware 和 IBM MQ-Series 作为平台间数据交换的渠道。

6 月 30 日

W3C XML 布局标准：万维网联盟已经批准了一项规范，该规范允许开发人员设计 XML 文档的布局，并迈出了允许开发人员通过 XML 文档使用样式表的第一步。样式表可以让使用者定义生成的文件，制定颜色、字体或字体的大小等。

7 月

7 月 14 日

Oracle 免费奉献编程语言接口。Oracle 不久前免费推出常用编程语言 Java、C、C++、PL/SQL 之间的扩展标记语言（XML）接口，为的是方便对 Oracle 老的应用软件中数据的使用。Oracle XML Parser for C 和 Oracle XML Parser for C++加入了 Oracle 已有的 Java 和 PL/SQL 语法分析程序（parser）。C 和 C＋＋构件支持 DOM（Document Object Model）和 SAX（Simple API for XML）接口。这些语法分析程序连同 Oracle Application Server 一起使用，可用来访问 Oracle 8 和 Oracle 8i 中的数据。Oracle XML Utilities 和 XSQL Servlet 产品可用来对 Oracle 数据库读出或写入 XML 数据。此外，Oracle 还推出了 Oracle XML Parserfor Java 的版本 2.0，包括 Extensible Stylesheet Language Transformation 处理器。Oracle 的新语法分析程序出台后，现有的商业应用软件无需改写就能使用 XML。Oracle XML Parser for Java 版本 2.0 不但能使 XML 文件之间可以互相转换，而且能变化 XML 至其他格式，如 HTML，或提取数据然后存入 Oracle 8i 数据库，一次完成。

7 月 14 日

Clarus 公司是世界领先的网络应用供货商。它联合工业界领先的合作伙伴一起宣称支持 XML 作为电子商务和应用的统一标准。

7 月 16 日

资金交易的革命。

FXMarketT 是一个革命性的网上在线外汇兑换交易系统。这个新的信息系统是一个透明的“市场交易者”。它提供私有交易和价格查询的服务。

每一宗交易,交易者都能通过屏幕时时查看自己和其他重要的交易。

Forex Digital Services,Inc（Forex 数字服务有限公司）使用的是 XML 技术。这将有利于建立一个全球跨平台的外汇兑换管理通信系统。XML 将成为 Forex 的国际标准。

7 月 20 日

OASIS（the Organization for the Advancement of Structured Information Standards 结构信息标准开发组织）已经同意加入一个关于网络标准和电子商务的讨论组。这个讨论组是由美国总统行政办公室下属的科学和技术政策办公室组织的。这个讨论组在 7 月 20 日讨论了变化中的技术,市场和环境将会怎样改变网络标准的发展,电子商务和知识的管理。

7 月 20 日

XMLS（XMLSolutions Corporation ）宣布已和 LTP（Lisle Technology Partners）结为战略

联盟。LTP 从技术产品方向和资源两方向给 XMLSolutions 以支持。

7 月 20 日

XML Parser for Java（基于 Java 的 XML 解析器）更新了。现在已有了 2.0.13 版本。新版本中包括了旧有的 DOM（Document Object Model 文档对象模式），而且被系列化了。XML 解析器是完全用 Java 编写的。它包括用于解析、产生、管理 XML 文件的类和方案。

请参见：http://www.alphaworks.ibm.com/tech/xml4j。

7 月 21 日

OAGI（Open Applications Group Inc. 开放式应用工作组）宣布了一个向导计划，用以把 OAGIS（Open Applications Group Integration Specification 开放式应用工作组整体说明）中的 XML 定义移植到 BizTalk 框架说明中来。包括了 Candle 公司、Compaq Computer 公司、HK Systems 有限公司、IBM 制造系统、Microsoft、NEC 公司、PeopleSoft 有限公司和 PricewaterhouseCoopers。在 OAGI 中的 XML 工作组打算更新现有的基于 W3C XML 1.0 版本说明的 OAGI XML 文件。OAGI 的目的是构造一个可兼容的 XML-schema 文件集合。并把它公布于 BizTalk 组织的资源库中。

7 月 23 日

IETF 与 W3C 支持数字签名。

IETF 和 W3C 在日前举行的第 45 届 IETF 会议上宣布，将联合推出支持 Web 上数字签名的策略。双方将使用 XML 实现数字签名，数字签名使用公用和专用密钥加密以识别用户。为这项技术制定 XML 标准是非常重要的，因为 XML 应用程序逐渐成为热门。许多公司在借贷票据和其他文件上使用 XML 格式，它们需要通过不同的应用程序读取申请者的签名。这项标准将制定一套 XML 标记，它把数字签名标记为加密的识别码，并且映射到相应的 Web 资源上，如证书供应商用于认证的 URL 上。XML 将与现在的 X.509 或是 PGP 证书加密技术配合使用。合作双方希望年底能够推出标准。但是 XML 数字签名也面临着一些问题，因为人们对由密码提供商这类的第三方来验证签名缺乏信任。政府是值得信任的第三方，但政府这么做的可能性很小。美国政府就不希望成为签名数据库的管理者。可能这么做的机构是银行或美国邮政总局，他们可将处理数字签名作为他们传统服务的延伸。

7 月 23 日

XMLWriter 和 XMLNorm 发布。

请详见：http://www.oasis-open.org/cover/megginsonJavaCl19990721.html

7 月 26 日

DTI（Digital Telecommunications, Inc. 数字通信有限公司）发布了语音服务的基于网络的基本框架（Architecture）。ESP（Extensible Service Policy 可扩展的服务策略）的基本框架使得第一次人们能利用网络技术在当今的网络系统或是下一代网络系统上实现动态实时语音通信。

7 月 26 日

SoftQuad 和 Object Design 两家公司在下一代 XML 网络印刷应用方面结成战略联盟。它们宣布把 SoftQuad 的 XMetaL XML 作业环境同 Object Design 的 eXcelon XML 电子商务信息服务器结合起来。

7 月 27 日

Commerce One 公司发布了 CBL（Common Business Library 公用商业资源库）2.0.版本。

CBL 是工业界第一个被 Microsoft、CommerceNet、UN/CEFACT 和 OASIS 认证的覆盖面广泛的 XML 文件资源库。CBL 的目的是加速网上电子商务的发展。有了 CBL 2.0 定义的文件框架，我们可以在网上准确无误地交换各种样式的文件。

7 月 28 日

ADVISORY/META 工作组电话会议内容:XML 的未来。

ADVISORY/META 工作组预言：到 2004 年，XML 市场将消失。因为它将成为一个重要的内嵌基本技术。它将很快融于平台之中。ADVISORY/META 工作组提醒各个公司在为纯 XML 技术开发投入巨额资金时要十分小心。

如果读者对 ADVISORY/META 工作组有兴趣,请参见:http:// www.metagroup.com。

9 月

9 月 13 日

Carta 公司在它的完整的中期电子商务方案中结合了 XML 和 BizTalk 的框架。

Carta 公司是加州的一家网络技术公司。它在它的 InStore eCommerce 产品中加入了 XML 标准。InStore 是一个完整的中期电子商务方案应用，它是建立在 MCIS（Microsoft Commercial Internet System 微软商务网络系统）之上的。Carta 在它的产品中运用了 BizTalk 的框架使得它的终端后台商务应用能严密结合，以实现真正的终端电子商务应用。

9 月 13 日

微软发表电子商务蓝图。

由于看好网络购物的增长潜力，微软公司宣布推出其电子商务架构 BizTalk 以及相关产品 Commerce Server、Small Business CommerceServices、BizTalk Server，而为了建立 MSN 入口网站上的购物目录，微软最近兼并了 CompareNet 公司。

依据 IDC 的预测，今年全球上网人口将达到 1.47 亿人，其它的调查资料也指出，1998 年圣诞节时网上交易量比 1997 年增加 230%，显示出电子商务市场的潜力无穷，而微软将通过 BizTalk 来整合各种平台，并由微软入口网站 MSN 来提供行销服务，使得各种规模的企业皆可利用网站拓展业务。

微软的电子商务是以 Windows NT Server、SQL Server 为基础，企业可运用 BizTalk Server、Commerce Server、Small BusinessCommerce Services 建设电子商务网站。微软希望能在今年内让全球上百万家的企业进入电子商务的领域。

据称，Commerce Server 可为中大型企业建置功能强大的网站，Small Business Commerce Services 则可使小型企业迅速建设网站，至于 BizTalk 则采用 XML 及业界标准，借此可整合各种平台、系统，未来微软的 Windows、Office、BackOffice 都将支持 BizTalk。

微软将使 MSN 成为买卖双方交易的公开市场，为此，微软已兼并 CompareNet 购物网站，同时微软也计划与 AltaVista 等单位共同推动微软护照（Microfost Passport），使消费者登录一次即可在网络上通行无阻。

9 月 14 日

S2 Systems 和 Sequoia Software 宣布它们将在电子商务上合作以推动医疗保健产业的发展。

S2 Systems 是一家全球领先的专门提供电子商务产品的公司。Sequoia Software 是一家提供交互入口方案的公司。它们的合作是为了建立基于 XML 的电子医疗保健方案。这一方案将为医疗保健交费用户、供货商、职员团体和医疗保健品消费者提供在线的信息传输服务。这种方案是基于 NT 系统的。

9 月 14 日

开发者们在网络标准上存在分歧。

八月份 W3C 向业界推荐了 Extensible Hypertext Markup Language（XHTML）作最终评价。但是从那以后，他们就没了开始的劲头。现在开发者们还没能统一意见，到底是用哪个标准主导网络浏览器。

XHTML 重写了现阶段普遍实用的 HTML。重写之后每个独立的工业都能根据自己特别的要求比较方便地设计网页。有了 XHTML 之后 HTML 可以用 XML 重写。一些 XML 开发者抱怨 XHTML 中使用 namespaces 过于松散。现在的分歧在于区分 namespaces 的网络地址，或者说 URL 的实际使用。

9 月 16 日

未来数年间微软计划将通过 Windows 2000 及相关应用软件，满足所有用户开发网页的需求。这标志着微软的大量产品计划重心偏于网页。

微软工作短期目标是强化 Windows 2000 的功能。根据微软有关其 Windows DNA（Distributed interNet Architecture）2000 软件的技术细节来看，微软希望凭借自己在业界的巨大的影响力推动开发者编写电子商务上应用的软件。很显然微软非常希望能够在迅速发展的电子商务市场上抢占重要的份额。

中期目标是更新 Visual Studio 工具包。

长期目标是推动 XML 的发展。微软给予了 XML 高度的评价：“资料、商务运作、应用软件及物件在网路上充份交互运作，XML 是支持它们主要的技术。”微软将让所有操作系统和全套 BackOffice 都支持 XML。

9 月 16 日

微软和 SAP 合作，旨在使移动通信设备和商业软件结合起来。

SAP 是德国软件界的一家很有影响的大公司。他们的联合是通过 BizTalk 框架把基于

Windows CE 及其它操作系统的移动通信设备和 SAP 公司的商业应用软件连接在一起。

9 月 16 日

IBM 正在筹划建立一个新的网站。这个网站的目标是为开发商们提供有益于开发软件的信息、资源和建议。这个新的网站叫做 DeveloperWorks。它涵盖了 Java 编成语言，XML 网络标准和开放资源的 Linux 操作系统。它还计划关注电子商务相关软件的结构和为设计非英语软件的 Unicode 的发展。

这个新的网站将在两个星期内推出，当然它在完全成熟之前，还是要经过几个月的 beta 测试。

9 月 20 日

BizTalk 应用框架得到了能源界主流企业的支持。美国石油研究会（The American Petroleum Institute）和核子能技术公司宣布支持应用于电子商务的 XML 框架。

美国石油研究会的财会，电子商务经理 Kendra Martin 说：“我们之所以选择了 BizTalk，是因为它有最好的资金支持，它对垂直结构市场所制定的标准也是最好的。有了 BizTalk，我相信我们能和更多的公司和标准制定者们合作，更快更好地从 XML 给电子商务带来的好处中获利。”

Martin 还列举了 BizTalk 可能给石油产业带来的好处：

(1) BizTalk 可以使新的，必须合并的产业实体更方便地合并。

(2) 为用户服务的发展速度会大幅度提高。

(3) 合并多商业单元会带来更多活力。

(4) 公司和消费者、合作商之间的联系更加丰富。在电子商务中建立更多新的商业联系将变得更加容易。

9 月 21 日

此新闻引自 ChinaByte。

IBM 揭开了一种新数字钱包技术的面纱，并表示，它正在和 MasterCard 合作开发使用这种技术的系统。按照所制定的计划，MasterCard 将提供业内第一种兼容电子商务标识语言（ECML）的电子钱包。

电子商务标识语言（ECML）是由 IBM，MasterCard、微软、Sun，以及其它几个企业在今年 6 月发布的，是一种开放式的标准，允许任何一位商家接受来自任何一种兼容 ECML 电子钱包所输入的信息。这种技术可以使钱包供应者迅速“发现”网上购物表格需要填写的区域，并以电脑速度自动传送正确的数据，整个过程只需 IBM 消费者钱包用户点击“自动填写”选项。使用 IBM 消费者钱包，网上购物者只需输入一次自己的信用卡信息，并把它安全地存储在一个图标（或者“钱包”）中以备将来网上购物之用。

目前，MasterCard 还在开发自有品牌的消费者钱包，以提供给自己的会员银行使用。IBM 表示，它和 MasterCard 之间达成的协议，还可以使 MasterCard 的会员银行通过一个特别联盟程序，获得自己的电子钱包。另外，MasterCard 还计划对现有的安全电子交易（SET）钱包进行升级。

9 月 21 日

开放的 eBook 印刷结构说明 1.0 版本（Open eBook Publication Structure Specification version 1.0）在微软提出这个建议的一年之后终于正式出台了。这将加速电子出版业的发展。

这个说明将对书籍格式加以描述，便于书籍方便地、数字地传输。它是基于 XML 说明的。这显然有益于用户在不同的界面下浏览文章。

如果读者对这个说明有兴趣，可以浏览 http://www.openebook.org/。

9 月 22 日

XMLSolutions 发布 XMLZip。

XMLSolutions 公司是 E2E 的领导企业,今天宣布最新的为 Windows NT 和 Linux 操作系统服务的 XMLZip 面世了。这是工业界第一个压缩 XML 文件的工具。这为处理大型的 XML 文件提供了方便。大型的 XML 文件占用了服务器和客户端过多的传输时间和存储空间。

XMLZip 允许用户决定压缩的程度。这使得在连续使用 DOM API 时不至于降低应用的水准。

举一个例子,一个 1.2Mb 的 XML 文件经过 XMLZip 处理后只剩 136kb 了!

9 月 22 日

OASIS（the Organization for the Advancement of Structured Information Standards 促进结构信息标准组织）今天宣布 WebMethods 将成为 Robin Cover XML（www.oasis-open.org/cover/）网站的最新赞助商。WebMethods 提供大量的文献和标准。它是工业界一个在线的资源库。各种有关结构信息标准的信息。包括 XML, SGML, HyTime 等等。比如说:教程、介绍、XML 喜爱工业界的应用实例、研讨会议的详细清单、XML 的软件清单、XML 相关的标准信息以及常见问题回答。

9 月 23 日

Harbinger 宣布支持微软的 BizTalk Framework 在电子商务的应用。

Harbinger 公司是一家跨国电子商务软件服务商。该公司宣布,它将与微软合作,支持 BizTalk Framework。同时,还宣布将与微软共同建立标准,把 XML schema 运用到在线目录管理系统中去。

它们合作主要在于四个方面：

(1) 将 BizTalk Framework 加入到 Harbinger 新产品中去。

(2) 和微软共同制作在线目录管理系统的标准。

(3) Harbinger 将使它的 harbinger.net（SM）电子商务接口可以交换 BizTalk 兼容的 XML 文件。这使得它的网络用户可以方便的接收到 XML 文件。

(4) Harbinger 打算通过支持 BizTalk Framework 来支持它的 TrustedLink（TM）。这使得 Harbinger 的 40,000 用户可以享受在线的 BizTalk 兼容的 XML 文件的翻译。

9 月 27 日

Asymetrix Ships ToolBook II Instructor 7.1 将提供对 ICC、DHTML、XML 和 Actions Editor

的支持。

Asymetrix Ships ToolBook II Instructor 7.1 是一个强力升级的编辑工具。Asymetrix Learning Systems 公司是一家在线学习软件的领头企业。它今天宣布,ToolBook II Instructor 7.1 版本 7.1 正式发布了。这是一个为专业开发商、编程者、设计指导者和培训员准备的在线编辑软件。

9 月 27 日

SciQuest 公司成功演示了在 Ariba Network 电子商务服务 cXML（Commerce XML）流水线上的在线买卖过程。

SciQuest 公司是一个领先的基于网络的交互式网上市场。它主要提供医药、生物、化学以及教育组织的试验产品。它今天宣布,成功演示了在 Ariba® Network（TM）服务器上成功的、安全的电子商务交换。

9 月 28 日

webMethods 将 cXML（Commerce XML）交换推向 Ariba Network。

WebMethods 公司是一家为商业界实现联合提供开放解决方案的领先企业。它在今天宣布它提供在 Ariba® Network（TM）平台上进行快速安全电子交换的解决方案。

9 月 28 日

X-Collaboration 软件公司和微软将推出一个关于 Phoenix 的中小型商业的研讨会。

X-Collaboration 软件公司（X-Collaboration Software Corporation XCSC）是一个网络合作服务的公司。它和微软今天宣布，它们将推出一个关于 Phoenix 附近的中小型商业的研讨会，主要是怎样利用 XML、网络合作以及 Office 2000 来增强知识和信息的管理。

研讨会将由微软代表主持,他将提出微软对通过在网络上整理、编辑、印刷文件来提高生产效率的观点。

研讨会将于 10 月 13 日早上 9 点在 Phoenix,Arizona 的微软分支机构中举行。

9 月 28 日

Bentley 的 ModelServer Integrator 宣布支持 XML and aecXML。

Bentley 合作公司今天宣布它的基于网络的 ModelServer® Integrator（TM）产品支持 XML and aecXML（architecture, engineering and construction XML）。ModelServer Integrator 从不同的数据库和软件应用中收集信息,然后及时迅速地把信息在整个组织的成员面前表现出来。ModelServer Integrator 对 XML 的支持使得用户们可以充分利用在电子商务数据交换中已有的各种支持工具、技术和标准。

10 月

10 月 8 日

SYS-CON 宣布正式发行 XML-Journal（XML 杂志）。

SYS-CON 是 Java Developer's Journal（Java 开发者杂志）的发行商。它为 XML 和网络开

发商们提供有价值的新闻，以及 XML 工业标准,跨平台开放 XML 建设的有用信息。XML-Journal 是一本全新的、也是第一本专门关注 XML 技术的双月刊杂志。读者可以从 http：www.XML-Journal.com获得更多信息。

10 月 20 日

XML Global 建立了一个开发 XSLT 网络通信标准的联合论坛。

XSLT（extensible stylesheet language transformations）是可扩展样式表转型语言的简称。XML Global 是一家 XML 提供商业方案的公司。它今天宣布为 XSLT（www.XSLT.com）提供一个新入口网址。这样能给网络开发商和编程者以更多更便捷的信息服务。

XSLT 正在快速成为 Internet 上 XML 转型处理的主干语言。

10 月 21 日

Excelergy 发布能源分化工业 XML 标准资源库。

Excelergy 公司是能源分化工业的技术领头羊。今天它正式宣布发行专业的能源分化工业 XML 标准资源库。Excelergy 致力于在激烈竞争氛围中，为客户和服务零售商提供开放的通信方法。

10 月 25 日

System 1 软件公司同意为商业器材金融业定制和推广特定的 BizTalk Schema。

System 1 软件有限公司是商业器材金融业中电子商务和投资者方案设计的领先企业。今天它宣布它已设计和推广了一个基于微软 BizTalk Framework 的商业器材金融业专用 schema。

除此之外，System 1 软件有限公司也加入到了微软 BizTalk 服务器 beta 版的设计编程中来了。这是为了使得 BizTalk 服务器和 schema 能完美地和 System 1 的产品兼容。

11 月

11 月 2 日

Commerce One 宣布工业界的第一个电子商务 XML 开发工具集 XDK。

XDK 1.0 版本将推动在工业界通过使用 XML Schema 文件广泛实现一体化的发展。XDK 是全世界第一个 XML schema 工具集。毫无疑问地它将促成工业界基于 XML 的电子商务发展。它是面向开发者的一套工具，读者可以从 Commerce One 的 MarketSite 免费下载得到（http：www.marketsite.net）。

11 月 4 日

OAG（Open Applications Group 开放应用组）向 BizTalk.org 提供电子商务 XML Schema。

OAG 是最大的 XML 印刷商。它今天宣布它向 BizTalk.org 提供了 122 个商务交换 XML Schema。

11 月 5 日

ConneXt 通过 XML 加快消费者信息系统的应用进程。

ConneXt 有限公司是西雅图的一家软件公司。它主要的发展方向是为器材市场提供先进的报价，客户端服务解决方案。ConneXt 宣布它的 ConsumerLinX 解决方案在各种应用编程界面上支持 XML。

11 月 12 日

e-conten 组织了一个主体是 XML 电子商务的贡献系列研讨会。

e-content 是 Interleaf 有限公司的一家子公司。它提供基于 XML 的电子商务解决方案。今天它宣布组织一个系列研讨会。这个系列研讨会主要讨论 XML 和无线网络技术对传统商务模型的冲击。系列研讨会的标题是“XML，电子商务，无线网络的力量”。

11 月 17 日

Netfish Technologies 已跻身全球电子商务解决方案提供商发展最快者行列。

Netfish Technologies 是一家基于 XML 的电子商务解决方案提供商。它已经被 Oracle 列为全球前 50 位的电子商务解决方案提供商。这项角逐是四月份开始的。共有 28 个国家的公司参加了评选。

11 月 17 日

e-content 推出第一个可升级 XML 动态网络服务器。

e-content 是 Interleaf 有限公司的一家子公司。它提供基于 XML 的电子商务解决方案。它今天宣布第一个可升级 XML 动态网络服务器 BladeRunner Web（TM）。这是为下一代网站服务的服务器。BladeRunner Web 提供的是对端对端 XML 内容的管理。这使得提供 XML 内容的一端，可以向几乎任何一种网络终端发布几乎任何一种形式的文件。

11 月 20 日

W3C 推出 XSLT 与 Xpath 建议。

W3C 推出 XSLT（XSL Transformations）与 XPath 建议。这是 W3C 推出正式标准之前的最后一步。它们主要的功能是对 XML 文件的转型与样式表现的控制。如果读者想更多地了解它们，请参看 XSLT，Xpath。

11 月 30 日

ACORD 和 IFX 论坛宣布在 XML 标准的发展上展开合作。

在 San Francisco 召开的 ACORD 半年度标准会议上，ACORD 宣布承认与 IFX（http://www.IFXForum.org）论坛的成员互相认可制度。它们都是致力于保险和金融业数据电子交换 XML 标准的组织。它们的合作集中在基于 IFX 标准和 ACORD 保险业 XML 推荐标准，以制定一个支持基于 XML 交易的框架。

12 月

12 月 1 日

RosettaNet 发展道路上的一个里程碑，发布了第一批 10 个 XML PIP 说明。

RosettaNet 是一个独立的联盟。它的目标是提供连锁电子商务标准。今天宣布发布第一批 10 个 XML PIP（XML Partner Interface Processes）说明。

12 月 1 日

来自全球的不同组织支持 ebXML 全球电子商务动议。

EbXML（电子商务 XML 动议）是 United Nation/CEFACT 和 OASIS 的一项联合动议。它已吸引了全球各种组织的支持。在 San Jose 召开的大会中，有来自全球不同组织的 120 名的代表。它们包括 ACORD、Accredited Standards Committee （ASC） X12、Commerce One、DataChannel、DISA、UN/EDIFACT、IBM、OAGOracleSun Microsystems 等。

12 月 2 日

XMLSolutions 在网上演示 EDI-to-XML 翻译。

XMLSolutions 宣布任何 EDI 系统的用户可以通过访问 http：www.xmls.com 把任何标准的 X12 或是 EDIFACT 文件翻译成 XML。

XMLSolutions 收到 EDI 文件之后将运用 XEDI 方法实现 EDI-to-XML 翻译（http：www.xedi.org）。然后通过电子邮件把翻译发还给用户。

12 月 6 日

微软今天宣布 BizTalk Framework 文件说明 1.0 版本正式出台。BizTalk Framework 文件说明为基于 XML 的解决方案提供的技术专用说明。它使得在线交易和网络信息交换变得轻松自如。

12 月 8 日

中国 XML 联盟举行第一次开放式宣讲会。宣讲会的主题有：《Application Integration Using XML》，《中国 XML 联盟发展研究报告》，《中国 XML 联盟发展计划》。宣讲会取得了圆满的成功。

12 月 8 日

W3C 与 WAP 论坛宣布正式联盟关系，共同制定下一代网络的说明。

W3C（万维网联盟）与 WAP（Wireless Application Protocol 无线应用协议 http://www.wapforum.org）论坛今天宣布共同制定下一代网络说明，来保障无线设备接入网络的成功。

12 月 13 日

Workforce eServices 的主要提供商 Icarian 宣布加入 HR-XML Consortium。

Icarian 公司是 Workforce eServices 的主要提供商。它今天宣布加入新成立的 HR-XML Consortium。HR-XML Consortium 是一个非盈利性联盟。它的主要方向是建立和推动和人力资源相关的 XML 标准的制定。这些标准可以为商业间电子商务交换和公司内部信息交换中的人力资源部分提供依据。

HR-XML 也有其 framework，它的功能是使得人力资源应用的内部和外部信息自由交换成

为可能。

读者可以在 http：//www.hr-xml.org/找到 HR-XML Consortium 的主页。

12 月 14 日

XMLSolutions 加入联合国和 OASIS 的行业，宣布支持 ebXML 全球动议。

XMLSolutions 今天宣布支持 ebXML 全球动议（Electronic Business XML Initiative）。联合国和 OASIS 已经提出了这项动议，用来统一标准化全球 XML 商业标准。EbXML 邀请各种组织加入它们的行业来意通知定义给统一的 XML 框架。已经有超过 50 家公司、联盟、全球化标准组织加入了这个行列。

12 月 15 日

OASIS 为 XML.org 的注册/资源库发布草稿说明。

OASIS 今天宣布 OASIS 注册/资源库技术说明草稿正式公开。新的说明将运用于 XML.org 的注册/资源库建设当中去。它将便利 XML 资源库跨网络的互操作性。

12 月 15 日

OASIS 把 XML-DEV 加入到 XML.org 中。

OASIS 揭示了它将把 XML-DEV 这个邮件组加入到它的业界接口 XML.org 中去。XML-DEV 邮件组始于 1997 年。它是一个开放的全力支持 XML 应用于发展的邮件组。它一直是寄身于英国的 Imperial College。在 XML 99 大会的闭幕式上，来自英国 Nottingham 大学的 Peter Murray 教授正式宣布 XML-DEV 将移居 XML.org，成为 OASIS 的资源。

12 月 21 日

OpenBox 实验室为开发商提供 XML 协议工具的网上接口。

Innovision 公司是 XML 协议的领头羊。它与今天宣布 OpenBox 实验室已经有了它的公共网络接口 http：//www.innovision.com。OpenBox 实验室专为企业开发商和工业标准组织提供先期的 XML 技术服务。

12 月 23 日

XMLFund 向 Digital Counterpart 投资，用以加速基于 XML 的电子商务解决方案的发展。

Digital Counterpart Inc（DCI）是网络 OSS（operations support system）软件解决方案的提供商。专为通信服务商提供解决方案。它已成功地获得了 XMLFund 的第一批投资。

DCI 的 eCounterpart（TM）解决方案应用尖端的 XML 技术为通信服务商提供强大的应用。如：基于网络的订单接入、跟踪和管理应用。又如为通信公司提供的自动双向网关功能。

23.2　XML 2000 技术动态

1 月

1 月 5 日

Flashline.com 组建资源库的后台是 eXcelon 应用平台。

eXcelon 应用平台是 Object Design 基于 XML 的产品。它主要是用来支持动态数据交换。Flashline.com 组件资源库是业界第一个开放的组件 XML 文档集。提供各种组件信息。读者可以在 http：//www.componentregistry.com/entrypage.jsp 找到它。

1 月 5 日

OASIS 为协调 XML 联盟设立新职位。

OASIS 今天任命 Scott McGrath 为首任会员服务总监（Member Services Manager）。这是一个全新的职位，将负责协调 130 多个非盈利性国际 XML 组织。

1 月 5 日

uBid.com 依靠微软基于 XML 的 E-Commerce Site 建构全球最大的商务网站。

uBid.com 宣布与微软结成战略联盟。

uBid.com 是一家全球领先的在线拍卖和电子商务站点。它今天宣布 uBid.com 采用微软 Windows DNA 平台上的 XML 技术来建构它的站点。这是全球最大的基于 XML 的在线商务站点。如今 uBid.com 几乎抢占了所有运用 XML 的交易。和同类网站相比，它是最好的。

为了提升整个站点的运作，uBid.com 最近重新设计了它的报价引擎。它现在运用了一系列微软的组件：MSMQ，MTS 和 COM。所有的组件都在微软的服务器环境中运作。

uBid.com 主页位置在 http://www.ubid.com/。

1 月 5 日

Harbinger 的 XML 注册用户达到创纪录。

Harbinger 是全球最大的网络电子商务软件、服务和解决方案的提供商之一。它今天宣布已有超过 40%的注册用户在 1999 年中选择了基于 XML 的技术。这将大大加速 B2B 在网络上的发展。

1 月 5 日

免费的电子商务词汇表 2.0 版本推出。

Profile Systems（http：//www.xml.org.cn:8188/News/www.profilesys.com）是一家电子商务解决方案提供商。它宣布包括有 1300 多个定义的 2.0 版本已经推出了。它有一个标准的 Windows 帮助界面。这个词汇表将对那些已经加入电子商务行列和即将加入的公司大有帮助。

读者可以在 http://www.profilesys.com/library/glossary.html 免费下载电子商务词汇表 2.0 版本。

1 月 6 日

Sterling Commerce 和 Edifecs Commerce 联合提供 XML 电子商务整合解决方案。

Sterling Commerce 今天宣布 Edifecs 的 SpecBuilder 正式面世。SpecBuilder 是 Sterling Commerce 的 GENTRAN 商业整合过程中的一项可扩展功能项。有了 GENTRAN 和 SpecBuilder，一家公司可以方便地实现以 XML、EDI 和其他专用商业格式存储的信息整合。现在加入一个新的电子商务伙伴开销减少了 25%。

1 月 10 日

XML 受青睐，OASIS 资助增长创历史新高点。

一批来自欧洲、北美洲和亚洲的组织加入到支持 OASIS，这个全球性的推动 XML 应用的非盈利性组织的行列中来了。新的组织包括：Aerospatiale、Bowstreet、Cohesia、eCredit.com、InformIT 、JetForm 、NII Enterprise Promotion Association 、ProNet Technology Partners 和 StreamServe。

1 月 11 日

Sequoia 宣布它的 XML Portal Server 兼容微软的 BizTalk 1.0 版本框架。

Sequoia 软件有了 BizTalk 框架的支持，就可以用 XML 消息来实现不同的信息系统的整合工作。Sequoia 软件公司是一家提供企业动态信息接口（enterprise information portals EIP）的领先企业。

1 月 12 日

InfoShark 提出的 CARD Schema 被 BizTalk.org 正式接受。

InfoShark 今天正式宣布它所提出的 CARD（Commerce Accelerated Relational Data 商务加速相关数据）Schema 已由微软 BizTalk 框架所接受。

1 月 12 日

EcomXML 和 RosettaNet 一起努力推进 B2B 的电子商务发展。

EcomXML 是一家 B2B 的电子商务解决方案的提供商。RosettaNet 是一个发展和应用 XML 电子商务标准的联盟（http：//www.rosettanet.org/general/index_general.html）。EcomXML 已经发布了它的 ecomTalk Server beta 版本。ecomTalk Server 是为用 XML 实现的供应链提供管理。

1 月 27 日

XHTML 开辟了 Web 的新领域。

万维网联盟（W3C）今天正式公布了 XHTML（可扩展超文本标记语言）规范。它是用于互联网的 XML 家族的首位成员。它更容易操作和使用，而计算机打开及处理此类文件也更快捷。XHTML 对 HTML 保持兼容性，这样网站维护人员就不用对原有的网页重新编写，而目前用户使用的大多数常用浏览器软件也可直接阅读 XHTML 文件而不用升级。

2 月

2 月 14 日

XMLFund 向 XML Global Technologies 投资。

XML Global 有两项主要技术产品。一是 GoXML（TM）（http：//www.goxml.com/ ），XML 内容搜索引擎。二是 DataXCHG，一个 XML/EDI 电子商务转型引擎。这两项产品都是网络 B2B 交易中所必不可少的。由此吸引了 XMLFund 不小的投资。

2 月 14 日

XMLSolutions 推出独一无二的双向 XML-EDI 交易解决方案。

XMLSolutions 推出独一无二的双向 XML-EDI 交易解决方案。XMLSolutions 是一家顶尖的提供基于 XML 的 EDI 解决方案和 XML Schema 管理产品的产品服务商。它今天正式宣布 XEDI Translator 1.0 面世了。XEDI Translator 使得从事 EDI 的公司可以自动地把文件转换为 XML。

2 月 15 日

Sun-Netscape 联盟 iPlanet（TM）电子商务解决方案支持 XML Connect（TM）。

OnDisplay 今天宣布一个 Sun-Netscape 的联盟 iPlanet（TM）电子商务解决方案支持 OnDisplay 的 XML Connect（TM）动议。XML Connect（TM）是一个免费的、可下载软件。致力于建立无需 XML 服务器的安全可靠的 XML 商业文件传输。

2 月 15 日

Information Architects 和 Red Hat 联合努力推动 Linux 上基于 XML 的内容动态集散解决方案的产生。

Information Architects（iA http://www.ia.com/）是一家专门提供基于 XML 的内容动态集散解决方案的公司。它今天正式宣布加入 Red Hat 的 Independent Software Vendor Partner（独立软件提供商合作伙伴）计划，共同推动 Linux 上的电子商务。

2 月 16 日

巴尔默亲自展示了微软最新研制的编程开发工具 Visual Basic 7.0。

星期二在离微软发布视窗 2000 会场不远的地方，微软首席执行官兼总裁巴尔默亲自展示了微软最新研制的编程开发工具 Visual Basic 7.0。这套软件预计今年下半年开始销售。

Visual Studio 7.0 增加了许多基于 XML 和 COM++的开发工具，支持拖放式 Web 服务，使分布式计算大大简化。这套软件的一个特性 ASP++ Web Forms 可使程序员开发出 Visual Basic 格式的表单供网。

2 月 17 日

Internet.com 和 mondus.com 签署一项电子商务协议。

Internet.com（http：//www.internet.com/）是一个电子商务因特网技术网络。mondus.com（http：//www.mondus.com/ ）是一个专为中小企业赢取配额的电子交易场所。两家公司的业

务都会因此大幅度提升。

2 月 17 日

Oracle 雄心勃勃，打算在操作系统竞争上再次胜出。

Oracle 今天宣布服务于 Windows 2000（TM）的 Oracle8i（TM） Release 2 面世了。Oracle 是 Windows NT 数据库市场中的主导力量。Oracle8i（TM） Release 2 的面世提升了数据库方面的表现，并且能与微软的最新操作系统很好的结合。

从最新的 Dataquest 市场调查报告中看，Oracle 在 Windows NT 上的数据库相关股份占了 47.3%。从 Oracle 的技术网络（OTN）上，下载了共大约 160,000 份 Windows NT 上的 Oracle8i 拷贝。Oracle8i 是现今唯一的操作系统上支持 Java 和 XML 的数据库。

读者可以从以下网址下载 Oracle8i（TM） Release 2：http：//technet.oracle.com/ 。

3 月

3 月 2 日

Locus Dialogue 宣布与 Speechwise Technologies 在电子商务方面结成战略盟友。

Locus Dialogue 是一家正在飞速成长的语音技术公司。Speechwise Technologies 是一家提供专为基于语音的电子商务解决方案的公司。他们将一起在 Locus Dialogue 的 SpeechPortal（TM）平台上建立运用 VoiceXML 的语音电子商务解决方案。

3 月 3 日

Westfield 集团选用 eXcelon 的 Dynamic B2B 产品来提高它的保险业接口品质。

EXcelon 是一家领先的专为动态 B2B 提供解决方案的公司。它今天宣布 Westfield 集团已经选用了它的 Dynamic 应用平台和 Javlin 产品（EJB Dynamic Data-Cache Server）应用于 Westfield 未来的整体架构当中。Westfield 集团是保险业界的一家知名企业。具体地说，Westfield 将使用 eXcelon 的 Dynamic 应用平台来组织和管理内部的信息流。这些信息是完全 XML 的。EXcelon 的产品支持快速和有效的存取 XML 和 XSL 数据。这很好地解决了企业内部信息交换的需要。

3 月 6 日

Extensibility, Inc 推出 XMLschema.com 3.0 版本。

XMLschema.com（TM）是一个由 Extensibility, Inc.提供的一项在线服务。它是用来简化合法的电子商务语法的，简单的说就是检验和提供 Schema。Extensibility, Inc.于 3 月 6 日正式宣布了新版本的出台。这使得任何应用 Schema 的应用程序都可以利用这项服务。这项服务是完全自动的，部分还是免费的。

3 月 15 日

微软 MSXML2000 年 3 月最新更新。

这些更新支持更加强大的对 XSLT/Xpath 的支持。关于它的新特点，请参阅：http：//msdn.microsoft.com/workshop/xml/general/msxmlprev.asp。

3 月 16 日

DbXML 是专为大量 XML 文件的集散设计的管理系统。同时 DbXML 还结合了一些非常漂亮的在现今网络环境下无缝的整合方法。DbXML 将大大简化下一代网络应用开发的过程。读者可以在 http：//www.dbxml.org/了解到更多信息。

3 月 18 日

ebXML 的传输路游包装规范（ebXML Transport, Routing and Packaging Specification）已正式上线了。它主要提供了：

(1) 路由信息的结构和头。
(2) 交换信息的模板过程。
(3) 对任何数字信息内容的的支持。
(4) 安全协议保护。
(5) 对可变声轨的支持。
(6) 对自动报错的支持。
(7) 安全可靠的消息传输。
(8) 定义了在服务中实现交互的信息项。
(9) 提供缺省的引导程序服务。

读者可以在 http：//www.ebxml.org/index.htm 处下载说明的全文。

4 月

4 月 6 日

朗讯科技公司这个星期宣布正式发布它的 Lucent Speech Server。这项产品使得用传统的语言服务方式就可以接触到网络。有了这套软件产品，服务商们就可以为商家或是消费者提供语音的方式接入网络信息。这一方向的盈利在未来几年得可望达到数亿之多。可能的相关服务包括新闻、天气、股票、证券、购物和娱乐等。

5 月

5 月 1 日

Soap.Weblogs.Com 上线了!

SOAP 自从 4 月 26 日推出 1.1 版本之后，就显得异常活跃。SOAP 是一个在分散、分布的环境之下互相传递信息的轻量级的协议。它是一个基于 XML 的协议，有以下三个部分组成：一是打包，用以定义消息内部结构和处理过程；二是编码信息，用以表述应用程序定义的数据类型；三是关于远端调用和相应的约定。SOAP 协议可以和其他很多协议结合起来。

Userland 的 CEO 今天宣布 Userland 为 SOAP 建立了一个新的网站，一个新的网络口号，那就是 soap.weblogs.com。UserLand 这次是和 Microsoft、DevelopMentor、IBM 以及 Lotus 一起合作，一同设计定制 SOAP 协议。它们相信这个协议能够带来网络应用的发展革命。

soap.weblogs.com 将及时地反馈 SOAP 的最新信息，使得它成为应用开发 SOAP 工程人员

的集散地。soap.weblogs.com 有健全的成员制度，还有许多 SOAP 相关问题的讨论组、留言板等。soap.weblogs.com 代表了 UserLand 在 XML-RPC 上的不懈努力。

5 月 4 日

GCA（Graphic Communications Association 图形交流协会）最近宣布它原来的 EDI 部门正式改名为 B2B 标准部门。

这个部门的主要目标也变为致力于制定和研究在不同商业组织与实体之间的电子信息交换格式和规范的制定。GCA 是一个全球享有盛名的组织联合体。大约有超过 300 家公司是它的正式成员。这些公司涵盖了几乎全部的印刷、出版和信息技术领域。为了适应日新月异的信息技术发展，GCA 深入地加入到 XML 技术的研究当中。

5 月 8 日

Bovone Stefano 宣布一个 REXP（Rendering Engine for XML/XSL to PDF ，XML/XSL 到 PDF 的翻译引擎）的诞生。REXP 是一个开放的工程，每一个人都欢迎献出有益的力量。简单的说，REXP 是 FO engine（Formatting Objects 样式对象引擎）的早期实现。XML 的 FO（format object）的争夺战已经打响。FOP（http://xml.apache.org/fop）是全球第一家由 XSL FO 驱动的打印形式，由 Java 应用读入的树型结构然后产生 PDF 格式。REXP 是建立在 FOP 0.9.2 版本之上的（详情请见 Apache）。REXP 现在是由 Genoa 大学的生物物理和电子工程系组织开发。

6 月

6 月 15 日

W3C/IETF 发布 Canonical XML1.0 版工作草案。

IETF/W3CXML 签名工作组与 6 月 13 日发布了新的 Canonical XML Version 1.0 工作草案。该工作草案是第二个建议草案，作为采用 W3C 的 XPath 数据模型来实现典范 XML 标准的另一途径，并进行了一些根本性的改变，将影响 XML 文档的规范性序列化。另一种实现规范形式的 XML 文档的方法是采用 XML 信息集合（XML Information Set）。

6 月 20 日

Sun、IBM 和 Oracle 帮助 Oasis 推出首期在线注册库。

在 Sun Microsystems、IBM、Oracle 等主要合作伙伴的帮助下，建立行业互操作的 XML 联盟 Oasis 通过 www.xml.org 发布了首期 XML 登记和注册库（XML registry and repository）。www.xml.org 网站的目标是成为存放 XML Schema 的仓库,帮助应用方案提供者建立未来电子商务的架构。Oasis 的执行主任 Laura Walker 说：“不是现在，甚至可能不是在两年之内，但是最终 Internet 将依赖 XML Schema 来进行数据交换。”在 e-business 交易——尤其是垂直市场的 B2B 交换中，使用相同的 Schema 来翻译 XML 标记对平滑的数据传输来说是不可或缺的。目前，开发者和 XML 标准实体仍在定义在不同应用中使用的 Schema。其中得到 Sun 和 IBM、Oracle 支持的 Oasis 和微软推出的 Biztalk 的重点都是在制定电子商务的通用框架和标准。Walker 称 xml.org 站点开放地接受行业递交的 XML Schema 草案。虽然目前该站点还没有 Schema，但是 Oasis 期望不久在其中有大量的 Schema。

6 月 21 日

W3C 于今天正式推出 P3P（Platform for Privacy Preferences Project）。

比如用户在一家商店购物。在进入之前，用户有简单的办法知道他的个人信息在整个购物期间，以及在购物之后会受到怎样的保护。比如说商店会用简洁醒目的标示告诉用户。这样用户在购物期间就非常了解自己个人信息的状态。再让我们来设想一下在网上购物的情况：用户可能是用信用卡交易。商务网站会怎样处理用户的个人信息？P3P 就是一个这方面的解决方案。在 P3P 中有两个关键的组件。一个是服务器端的。它是用来根据商务网站的特别要求生成 XML 形式的用户个人信息处理政策。这就像是贴在商店橱窗外的告示。另一个组件是放在客户端的。它的作用就是在用户进入网站购物之前把网站的个人信息处理政策提供给用户。用户可以至始至终地了解个人信息的状态。P3P 已经有了 1.0 的 Spec。

读者可在以下网址查询关于 P3P 的信息：http：//www.w3.org/P3P/。

6 月 21 日

IBM 和 Lotus 在 SOAP 的未来发展中角色凸现。

本周, IBM 和 Lotus 加入到 Microsoft、DevelopMentor 和 UserLand Software 的行列，成为 SOAP 修订规范 1.1 版（revised Simple Object Access Protocol specification v1.1）的作者之一。简单对象访问协议（Simple Object Access Protocol，简称 SOAP）定义了如何通过跨语言和平台的方式利用 XML 发送消息在 IBM's alphaWorks 中发布早期版本。

6 月 22 日

Microsoft 新的“NET” Internet 平台以核心 XML 技术为特征。

2000 年 6 月 22 日，Microsoft 公司展示了它的下一代软件和服务：Microsoft.NET（'DOT-NET'）平台。Microsoft 家族中的新成员.NET 产品和技术将取代以前的标准——下一代 Windows 服务（NGWS），其中包括用于开发者建立下一代 Internet 应用的软件和相关的新型智能设备。此外还披露了一些与 XML 有关的特征：.NET 是基于 Internet 的协议和标准，用于设备和服务之间进行交互，尤其是依赖于可扩展标记语言（XML）（Bill Gates 语）。.NET 设备软件将会是具有 XML 特征, 支持网络和.NET 服务之间的智能交互，这种技术是把用户引入采用.NET 用户经验技术的非 PC 设备的基础，如袖珍 PC、机顶盒、信元电话和游戏控制台。

6 月 22 日

W3C 更新同步多媒体综合语言（SMIL）Boston 规范的工作草案。

SMIL Boston 版本有以下两个设计目标:（1）定义一个简单的基于 XML 的语言使得人们可以编写交互性多媒体（2）允许在其他基于 XML 的语言中重用 SMIL 的语法，尤其是需要表现时序和同步的情况。SMIL Boston 定义为一套标记模块的集合,定义了特定区域的 SMIL 功能的语法。所有的模块都与文档对象模型（DOM）相联系。该版本是新一版的 SMIL 标准（被称为“SMIL Boston”）的第 4 个工作草案。该版本上一版本是 2000 年 2 月 25 日发布的。

6 月 26 日

信息结构发展组织推出 XML 计划。

OASIS（信息结构发展组织）在其主要成员 SUN、IBM、Oracle 的帮助下本周二推出 XML 计划的第一部分，发布了 XML 网站。方案提供商们建造电子商务的未来。这个组织相信，XML 计划不是一个短期行为，Internet 终将依赖于它而进行数据交换。

XML 是一种应用软件中数据交换的标准程序语言。对于 B2B 尤其重要。SUN 和一家文件内容管理系统开发商共同承担了设计这个站点的任务。IBM 则在站点测试阶段贡献了测试方案。Oracle 为站点的建设捐赠了 Oracle8i 数据库的使用许可。

6 月 27 日

BizTalk 框架说明 2.0 版本草案已经正式于 27 日推出。

这项框架针对 SOAP1.1 版本作了相对改变。它还包括了服务器与服务器之间安全传递消息的相应说明（reliable messaging 值得一看）。对 MIME 类型的支持也是它的特性之一。这使得在 XML 消息当中可以包含许多其他的非 XML 附件。这个问题在 ebXML 工作组当中也曾讨论过很多。相应的 BizTalk Server 2000 也将支持现在的 2.0 版本框架。读者可以在 http：//www.microsoft.com/biztalk/查看 2.0 版本说明的草案。

7 月

7 月 11 日

用 XML 实现 WAP 网页的工具问世。

一种使用 XML 将 HTML 描述的 Web 网页改写成可供上网手机或者便携式信息终端浏览的 WML 网页的软件面世了。该软件是由软件供应商 Ifasion Technologies（http://www.voice-net.co.jp/）推出的“EMPRESTO”。

7 月 30 日

Sun 的 StarOffice 中将开发基于 XML 的文件格式。

Sun 微系统公司最近在 O'Reilly 召开的开放代码会议上描述了一项计划，表明其 StarOffice 软件将走开放源代码的道路。Sun 将持续地推动 OpenOffice.org 源代码的开发并在该网站免费分发其认证的 StarOffice 最新版本。OpenOffice.Org 还将为文档规定 XML 的文件格式。StarOffice 是为所有主要平台开发的领先和全面的产品包，包括 Solaris、Windows、Linux 和 Macintosh 操作系统。

7 月 31 日

W3C 发布修改的 SMIL 动画规范。

W3C 今天发布了最新的 SMIL 动画标准（工作草案），这是为 XML 文档添加动态功能而制定的工作草案。除了定义适合于 XML 文档集成的基本 XML 动画元素之外，它还描述了一个整体动画框架。该框架建立在 SMIL 时序模型的基础之上，并进行了一些扩展。

8 月

8 月 1 日

Microsoft 发布 MSXML 解析器最新版本。

Microsoft 最近宣布了 MSXML7 月份 Beta 版。该版本是对 2000 年 5 月份的技术预览版（technology preview）的最新更新。7 月份的版本在原来的基础上增强了对符合 XSLT/XPath 标准的改进，并且 Microsoft Visual Basic 中实现了对 SAX2 （Simple API for XML）的支持，还紧密遵循 OASIS 的 Test Suit。该解析器的版本已经从技术预览版完全过渡到了 Beta 版，为到 2000 年秋季在网站正式发布进行产品通用性能的支持。XML 是 Microsoft.Net 平台的核心技术，而 MSXML3 为该平台建立了技术的基础，使得开发者迅速建立和传递基于 XML 的 Web 服务。使用 XML 进行沟通和数据交换的最大好处之一是互操作性，不过这必须建立在产品开发者和用户用统一的方式来处理 XML。这不必要求使用相同的软件或者编程语言，甚至也不需要在相同操作系统之上，但必须要求彼此的应用都遵循 W3C 制定的 XML 1.0 标准。

8 月 2 日

W3C 推出矢量图形技术说明。

SVG（Scalable Vector Graphics）的正式说明整整推迟了一年之久。SVG 是用来强化在各种不同环境不同场合之下的图形展现技术的。SVG 的目标是让计算机中的图形能够在不同的监视屏幕，比如是手机屏幕或是巨型的显示屏上，一致地展现。SVG 是通过 XML 来表示的。标准本来被计划在 1999 年 1 月份推出的。标准的延时推出是由于在去年夏天的时候所提出的草案带来了太多的讨论。

矢量图同一般意义下的通过点阵存取图像的方式不同。矢量描述的图形都是一些相对抽象的描述。在相对紧张的带宽要求之下矢量描述的图形能非常节约的传输。矢量描述图形的另外一个好处是可以在不同的目标屏幕上伸缩自如。SVG 的出现无疑对现在的终端浏览器的多元化大有裨益。

现在的图形还离不开点阵式的存储方法。SVG 标准的出台并不是宣布网络上点阵图的绝迹。

具体的 SVG 标准说明位于 http：//www.w3.org/TR/SVG/ 。

8 月 5 日

微软和 IBM 合作创建 XML 标准。

作为一对老冤家，微软和 IBM 一直摩擦不断，从最初的操作系统数据库到桌面软件市场，两家公司从未停止过争斗。但现在，两家公司将暂时把敌对情绪放置一边，合作建立一个普遍结构，以使一些新的服务成为现实，而 XML 语言（Extensible Markup Language）将会成为一致的选择。

两家公司都承认，通过共同努力建立一个普遍网络标准，能够增加两家公司的产品在市场上成功的机会。微软平台小组的副总裁 Paul Maritz 说：“当然，微软和 IBM 依然是竞争对手，但我们两家的技术人员形成了一些共同的技术观点，这将成为双方合作的基础。”IBM 的 XML 技术程序主任 Bob Sutor 也承认：“和微软合作是极为重要的，它是非常重要的市场

参与者，我们如果能够在更多的技术领域达成一致，这个领域就将更快地成长。”

分析家认为，两家公司的联盟是非常必要的婚姻，Giga 信息集团公司分析家 Mike Gilpin 说：“这就像在冷战时期美国联系中国对抗苏联一样，不是因为我们喜欢和中国政府联盟，而是一种权利自衡政策。”分析家认为，尽管两家公司一直存在冲突，但它们将为了共同的利益去创建 XML 标准。

8 月 8 日

Sun 发布 SVG 开发工具。

Sun 最近宣布了几项支持 W3C 可升级矢量图形（Scalable Vector Graphics，简称 SVG）规范的软件。SVG 刚刚于 8 月 2 日已经成为 W3C 候选建议（Candidate Recommendation），离正式的建议只有一步之遥。Sun 对 SVG 的支持包括二维图形的 SVG 生成器、SVG Slide 工具包和有关 SVG 及其与 Java 结合使用的文章和教程等。其中，二维图形的 SVG 生成器可以通过 Java 程序将图形导出为 SVG 的格式。而且用户还可以在支持 SVG 的图形编辑工具中导入 SVG 文件。SVG Slide 工具包能够将一个 XML 文件转换到 SVG 幻灯演示文件。相关的介绍性文章和教程包括：“Scalable Vector Graphics （SVG）: An Executive Summary”、“Tutorial on Java Server Pages technology and SVG”和“Writing a custom Graphics2D implementation to generate SVG images”。

10 月

10 月 6 日

W3C 推出 Extensible Markup Language (XML) 1.0 (Second Edition)。

XML 1.0 规范第二版不是 XML 的一个新版本（1998 年 2 月 10 日首次发表）；它只是为了方便读者，并入了第一版勘误表中指出的错误和修改。