



**XML CORE STANDARDS  
VERSION 1.0**

**MARCH 2, 2007**

---

**Information and Technology Management Branch**

**❖ IM / IT Standards & Guidelines ❖**

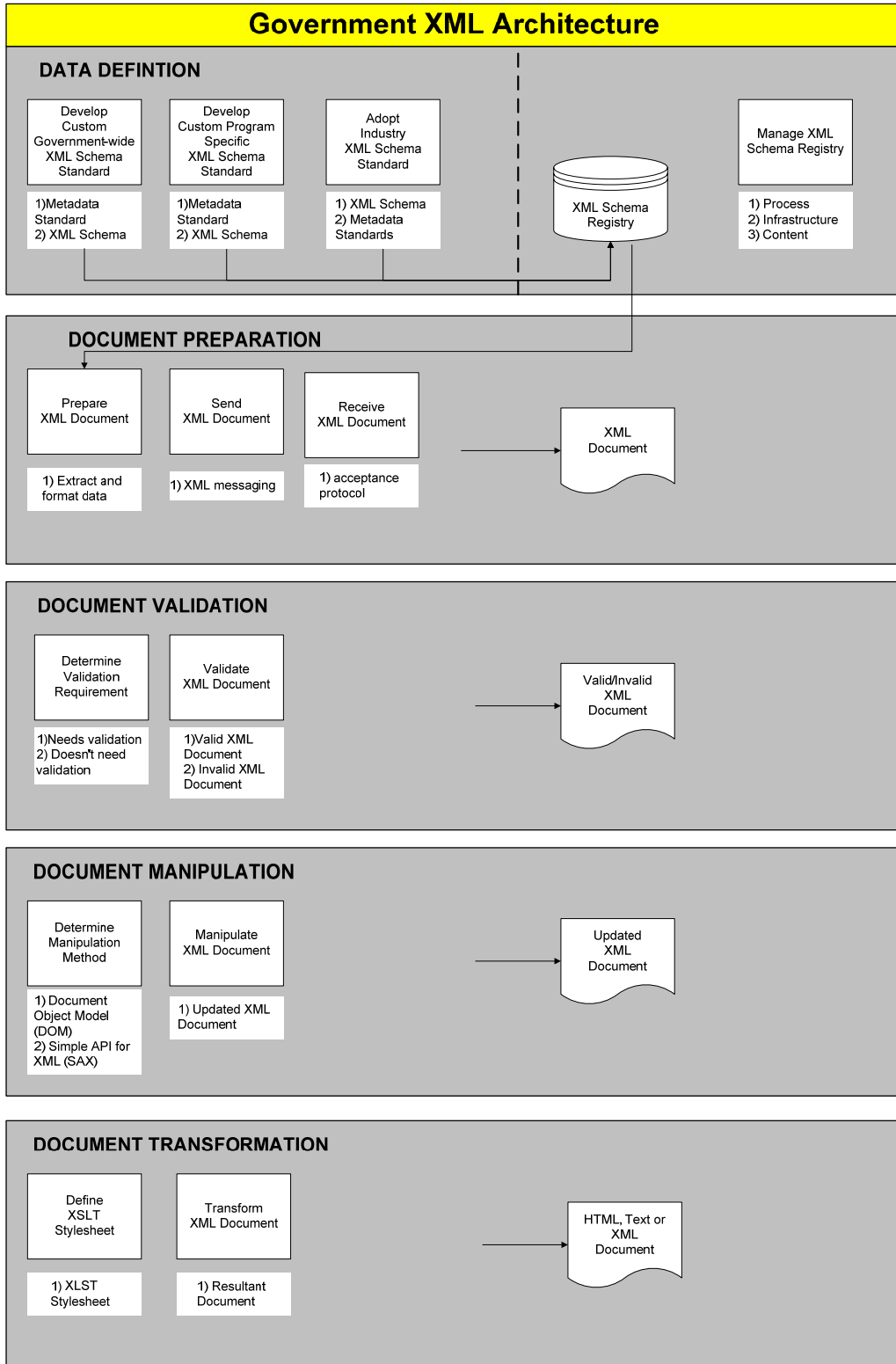
<b>1.</b>	<b>XML STANDARDS CONTEXT</b> .....	<b>3</b>
<b>2.</b>	<b>DOCUMENT PURPOSE</b> .....	<b>5</b>
<b>3.</b>	<b>XML VERSIONS</b> .....	<b>5</b>
<b>4.</b>	<b>XML DOCUMENT SYNTAX</b> .....	<b>6</b>
4.1	ELEMENT SYNTAX .....	6
4.2	ATTRIBUTE SYNTAX .....	7
4.3	COMMENT SYNTAX .....	7
4.4	PROCESSING INSTRUCTION SYNTAX .....	7
4.5	ENTITY REFERENCE SYNTAX .....	8
4.6	CDATA SECTION SYNTAX .....	8
4.7	DECLARATION SECTION SYNTAX .....	8
4.8	CHARACTER ENCODING SYNTAX .....	9

## 1. XML Standards context

XML Technologies span a wide variety of areas and standards. Ministry ADE process has identified the following areas of XML where standards could be built and evolved. Standards for each area are covered in separate documents.

- *XML Core Standards* : [“well-formed-ness” standards]. **This document.**
- *XML Validity Standards* : [DTDs and Schemas standards]
- *XML Namespaces* : [standards on resolution of tags with similar tag names]
- *XLink and XPointer* : [standards for creating hyperlinks in XML documents and standards for hyperlinks to point to more specific parts (fragments) in the XML document]
- *XML Catalogs* : [URL redirection standards]
- *XQuery* : [XML data querying standards]
- *XPath* : [standards on navigation through elements and attributes in an XML document]
- *XSL-FO (also known as XSL)* : [standards on formatting of XML data for output]
- *XSLT* : [standards on transforming XML documents]

The following diagram depicts the Government XML Architecture at a high level. This architecture is evolving and subject to changes :



## 2. Document Purpose

The purpose of this document is to provide a set of Core XML standards for XML documents to be “well-formed”. The document assumes that the reader has prior knowledge of XML and other related web technologies such as HTML, CSS etc. This document does not describe XML documents validity standards such as “DTD standards” and “XML Schema standards” etc. These are described in separate documents.

## 3. XML Versions

There are two official current versions of XML released by the World Wide Web Consortium (W3C) – *XML 1.0 fourth edition* and *XML 1.1 second edition* – both editions as of August 2006.

*XML 1.0* is widely implemented and still recommended by the Industry for general use and many parsers and tools are still XML 1.0 compliant. XML 1.0 only allows characters which are defined in Unicode 2.0, which includes most world scripts, but excludes scripts which entered in later versions of Unicode. XML 1.0 was not designed to fully accommodate newer versions of Unicode.

*XML 1.1* (unlike XML 1.0) is forward compatible with the Unicode Standard. The primary features in XML 1.1 that are not present in XML 1.0 are listed below :

- XML 1.1 is forward compatible with the Unicode Standard.
- XML 1.1 contains features that are intended to make XML easier to use for certain classes of users (mainframe programmers, mainly). For example, XML 1.1 has fixed the misalignment between the definitions of what marks the end of a line in XML and what Unicode defines this to be.
- XML 1.1 permits control characters (“#x1” through “#x1F”) in the documents through the use of character references most of which are forbidden in XML 1.0.
- XML 1.1 only disallows certain control characters but allows all other characters – regardless of whether the characters are defined or not in any particular (current or future) version of Unicode.

- XML 1.1 has inbuilt “*character normalization*” checking feature. Through this feature, XML 1.1 provides for XML 1.1 processors to verify whether a document is in a normal form or not; in the absence of this information, application programmers may need to perform normalization or make sure that their code does not rely on a particular form of text.
- XML 1.1 isn't fully backward compatible with XML 1.0.

**Of the two XML versions described above, Ministry ADE process supports and recommends usage of the *XML 1.1 second edition* which has been ratified as the official current version by W3C.**

**Older applications that have used XML 1.0 or have dependency on older version of Unicode version 2.0 and generate XML 1.0 compliant documents will continue to use XML 1.0 with assumptions that current XML 1.1 compliant tools and parsers would still support the older XML 1.0.**

## 4. XML Document Syntax

XML defines more strict syntax than HTML. A typical XML document has three sections : *Prolog*, *Body* and *Epilog*. Each section has its own syntax requirements. XML documents that comply with all the syntax rules are known as “*well formed*”. An XML application will always reject an XML document that is not well formed. “**Well formed**” only guarantees that the syntax is correct. The *Prolog section* is the first part of the XML document, before the root element. It contains optional header information, the XML declarations and comments and processing instructions. The *Body section* contains the root element and all its contents – the actual data in the XML document. The *Epilog section* must be placed after the end of the root element. It could contain comments and processing instructions.

The following paragraphs highlight the various XML syntax rules that must be adhered to while authoring/generating the XML documents so that the XML document is well-formed.

### 4.1 *Element syntax*

1. Element names must contain only letters, digits, period, hyphen, underscore and colon.
2. There can be only one root element and it must enclose the entire document body.
3. Elements must have a start tag and end tag (e.g <tag> value </tag>).
4. Elements must be correctly nested.
5. Content in XML document is case sensitive.
6. White spaces are irrelevant in XML markup like in HTML.

## 4.2 *Attribute syntax*

1. *Attribute names* must contain only letters, digits, period, hyphen, underscore and colon.
2. *Attribute values* must be enclosed within quotes (either double or single quotes, but not a hybrid of two).
3. If a double quote character is to be part of attribute value then that double quote character must be included as an escape character - “&quot;”. Example : location = “student&quot;s home” will be displayed as student’s home.
4. Only text is allowed in attributes. No tags are allowed in attributes.
5. An element cannot have more than one attribute with the same name.

## 4.3 *Comment syntax*

1. Comments in XML are very similar to that in HTML.
2. Comments can be put anywhere except inside tags.
3. Comments inside comments are not allowed.
4. The string “--” cannot be used anywhere inside the XML document except as a comment delimiter.

## 4.4 *Processing instruction syntax*

1. Processing instructions must be enclosed within the symbols “<?xml” and “?>”.
2. Two types of information are included as part of processing instructions : Name of the application targeted by this processing instruction and the information/data sent to the target application in the form of attributes.
3. The target application must be able to access and understand the processing instruction.

### Example of processing instruction standards

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet
  type="text/css"
  href="styleDoc.css" ?>
...
```

## 4.5 Entity reference syntax

1. Many characters can interfere with the XML parsing process, such as `<`, `>`, `%` etc. In order to avoid such interference, there is a kind of escaped character mapping (called entity mapping) supported by XML as shown in the following table which must be adhered to while authoring XML documents.

Character/entity	<	>	&	“	‘
Reference	<code>&amp;lt;</code>	<code>&amp;gt;</code>	<code>&amp;amp;</code>	<code>&amp;quote;</code>	<code>&amp;apos;</code>

2. Characters which may not be confusing to the parser but which may not be available on the keyboard should be referenced with the syntax “`&#characternumber`”. Example : A long dash character that is not available on the keyboard can be mentioned in an XML document as `&#8212;`

## 4.6 CDATA section syntax

1. Entity references described earlier will work in many cases but it becomes cumbersome when there are lots of special characters to be included in the XML document. In such cases, XML provides a section in the document called *CDATA section* that allows characters without any escape sequences. A CDATA section starts with "`<![CDATA[`" and ends with "`]]>`" as shown below :

```
<![CDATA[
Text containing any characters
...
]]>
```

2. Nested CDATA sections are not allowed. That is, a CDATA section cannot contain the string "`]]>`".

## 4.7 Declaration section syntax

1. The Declaration section is optional, but if present it must appear at the very beginning in the XML document.
2. *Version* attribute in the declaration section is mandatory.

3. Other attributes such as encoding, standalone (yes/no) etc are optional. See the example below :

```
<?xml version="1.0" encoding="UTF-8" standalone = "yes"?>
```

#### 4.8 Character encoding syntax

1. XML compliant applications must support at least one encoding such as UTF-8, UTF-16 and ISO-8859-1.
2. Ministry ADE process supports and recommends ISO-8859-1 encoding standard since this is available as a default encoding in the servers.
3. If declaration section is absent in the XML document, UTF-8 is treated as the default encoding by the XML processors. **For this reason, ADE process makes it mandatory for XML documents to contain declaration section with encoding as " ISO-8859-1" :**

```
<?xml version="1.1" encoding=" ISO-8859-1" ?>
```

4. Encoding attribute value is not case sensitive. See examples below :

```
<?xml version="1.1" encoding=" iso-8859-1" ?>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<?xml version="1.0" encoding="utf-16" ?>
```

**Ministry ADE process requires that all XML documents generated by tools or authored manually must be “*well formed*” and must be complying with all the syntax rules described above. XML applications will always reject an XML document that is not well formed.**

**Well-formed-ness does not guarantee anything about the usefulness or the *validity* of the data in the XML document. For most applications just “*well-formed-ness*” is not enough – the XML document must also be “*valid*” and obey additional rules. “*Validity*” of XML documents is checked either with *DTDs* or *XML Schema*, which are described in “XML validity standards” document in the ADE process.**