

XML as a Data Language

Introduction To XML



XML - eXtensible Markup Language



- originally intended for human authoring/reading
- textual
- mostly whitespace insensitive
- copies syntax principles from HTML

```
<elementName
        childName1="attribute value text"
        childName2="more attribute value text">
        value text <subElementName>sub element value</subElementName>
        more element value text
</elementName>
```

XML - eXtensible Markup Language



- WWW Consortium (w3c) standard
 - Versions 1.0 and 1.1 exist.
 - Almost all usage is version 1.0
- Designed to handle large and complex documents
 - namespaces to handle naming conflicts



XML - Love it/Hate it



- Reasons people love XML
 - Standard, robust, versioned, familiar (to those who have seen HTML)
 - Textual with standard way to specify charset encoding
 - Standard file preamble/first-line optional but recommended
 - <?xml version="1.0" encoding="utf-8"?>
 - Has a robust schema language
 - Precise specification of syntax and infoset (data model), excellent interoperability
- Reasons people hate XML
 - inefficient, verbose as a data language
 - ex: no real arrays, just repeating elements
 - two kinds of child nodes (element children, attribute children)
 - they can have the same name two different child namespaces
 - namespace mechanism seems too complex
 - whitespace problems
 - text character restrictions







```
<title kind="draft 1">iso lorem txt facto</title>
<title kind="draft
1">
   iso lorem
   txt facto
</title>
```

- Order of multiple attributes is not significant
- Whitespace (other than single spaces) is *normally* not significant
 - For attributes: gets collapsed to single spaces
 - XML tools will often wrap or unwrap lines
- XML loading converts all CRLF and isolated CR to LF

XML - Prefixes, Qualified Names (QName)



```
<ex:myEnclosingElement
  xmlns:ex="http://example.com"
  xmlns:pre="urn:foo.com/data1">
```

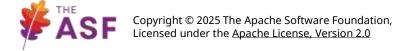
xmlns is an XML keyword

```
<pre:myTextItem
  ex:myAttr="a value">
  this text is the element content
</pre:myTextItem>
```

These URIs are just unique IDs.

They are never fetched.

</ex:myEnclosingElement>



XML - Default Namepace



xmlns with no prefix defines the default namespace

This element and enclosed elements with no prefix are in the default namespace.

http://example.com is a reserved URI for examples in XML.



XML - No Namepace



<enclosingElement
xmlns="">

<foo><bar>6.847</bar></foo>

• • •

</enclosingElement>

In XML data with no xmlns attributes the elements have *no namespace*

Or you can explicitly shut off default namespace

xmlns with no prefix AND no URL removes the default namespace

Contained elements with no prefix have no namespace

XML - Quoting, Character Entities



```
<elementName
  attributeName=<mark>'</mark>a "value" with quotes<mark>'</mark>>
  ... element content ...
</elementName>
<elementName
  attributeName=<mark>"</mark>a &quot;value&quot; with quotes<mark>"</mark>>
  ... element content ...
</elementName>

    " "

    ' '

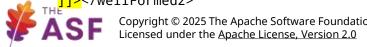
• & &
• > >
• < <
• 
 decimal numeric character entity (13 is Carriage Return aka CR)
• 
 hex numeric character entity (x0d is Carriage Return aka CR)
```

• XML 1.0 does not allow any ASCII control characters (00 to 1F) except TAB, LF, CR XML₀, converts CRLE to LF, and CR (alone) to LF.

XML - Element Quoting



```
<malformedXML>
 This content uses XML syntax literally
 in the element value to <emphasize>
 & generally mess with things.
</malformedXML>
<wellFormed1>
 This content uses XML syntax literally
 in the element value to <a href="https://emphasize&gt;">&lt;</a>emphasize</a>&gt;
 & generally mess with things.
 Note the whitespace is not guaranteed to be preserved.
</wellFormed1>
<wellFormed2><![CDATA[</pre>
 This content uses XML syntax literally
 in the element value to <emphasize>
 & generally mess with things.
 But now the space, tab, and lines are preserved. Even for pretty printed XML.
 (Except line endings CRLF/CR still become LF)
 Character entities cannot be used at all.
 1></wellFormed2>
```



XML data and CR (0x0D) line endings



XML does not round-trip CR <foo>mydata</foo>

6 characters for the character entity

- Read by XML _loader_ you get
 - an element with a string of length 8 as content
 - the string contains "mydata" plus a single CR character.
- Write it out with default writer you get in output: <foo>mydata
 CR
- A single CR character with code point 0x0D
- Read by XML again, and you get in memory:
 <foo>mydata

 foo>

A single <u>LF</u> character with code point 0x0A

- Because XML converts CRLF and isolated CR to LF!!!
 - XML wants data to be whitespace insensitive
 - But REAL data often is very particular about whitespace, control-chars, etc.
- To preserve CR
 - Special writer always writes out "
 ", but must be aware of quoting context.
 - ✓ Special reader/writer always converts CR to some other character that is preserved.
- Daffodil uses <mark>0xE00D</mark> in the *Unicode Public Use Area* (PUA)
 - See "Daffodil and the DFDL Infoset" at https://daffodil.apache.org/infoset/



XML data and the NUL character



- XML documents cannot contain NUL.
 - No way, No how.
 - Not even as �
- Daffodil uses OxE000 in the Unicode Public Use Area (PUA)
 - See "Daffodil and the DFDL Infoset" at https://daffodil.apache.org/infoset/

XML - Whitespace, Pretty Print and CDATA



- For XML data to be human accessible, must be pretty-printed (indented)
- Beware pretty-printing of XML.
 - It does not necessarily preserve string data.
- Consider this string element:

```
<callSign>BB823<callSign>
Pretty print at deep indent level might do this
BB823
</callSign>
<callSign><![CDATA[BB823]]><callSign>
```

- Most (All?) Pretty printers respect CDATA and will not corrupt this.
- Note: xml:space='preserve' does NOT fix this.

XML as a Data Language



- Requires some effort
- Map XML Illegal chars to Unicode Private Use Area (PUA) chars (Especially NUL)
- Map CR to Unicode PUA
- Pretty printing can cause trouble
 - CDATA bracketing needed around all xs:string elements
 - Protects significant whitespace from being harmed



XML - Element Content Types



Element with Simple Type Content

```
<courseNum>6.847</courseNum>
```

<courseDesc>Intro to Computer Science</courseDesc>

- Element with Simple Type Content with Attributes
 - Not used by DFDL v1.0 when value is non-empty

<courseNum creditHours="9">6.847</courseNum>

XML - Element Content Types



Element with Empty Content

```
<middleName/>
equivalent to

<middleName></middleName>
```

Element with Empty Content with Attribute

```
<middleName xsi:nil='true'/>
```

This is XML's very clumsy way of expressing 'null' or 'nilled' values.

XML - Element Content Types



- Element with Element-only Complex Type Content
 - Complex Type in XML means "may contain child elements"

```
<book>
  <title>Plants of the Amazon</title>
  <isbn>1-2345678-90123</isbn>
  </book>
```

- Element with Mixed Content (for real text+markup cases)
 - Mixture of text and elements
 - HTML-like
 - Not used by DFDL v1.0

```
<bookReview>
As entertaining as the tome <title>Plants of the Amazon</title>
is, I found it full of errors. You can find this book using its
<isbn>1-2345678-90123</isbn> at your favorite online bookstore.
</bookReview>
```