Unicode Constraints Creating Common Alerting Protocol (CAP) Alerts

Sean Donelan – sean@donelan.com

Suggested Unicode Constraints While Creating CAP Alerts

The Common Alerting Protocol v1.2 and IPAWS Profile v1.0 were written during the era of XML 1.0 and Unicode 3.0. Later XML and Unicode versions improved robustness and security. Alert origination system programmers mostly rely on XML and Unicode libraries doing the right thing, and sometimes they are unable to change the behavior of XML or Unicode libraries. Older software libraries may not properly handle problems corrected by later errata and versions. Newer software libraries may have backported some errata fixes or added new API options which need to be invoked.

Whether using new or old software libraries, CAP producers should be conservative encoding CAP Alerts with Unicode and XML.

CAP v1.2 producers should follow these Unicode character constraints writing CAP Alerts.

1. Only valid XML Unicode characters should be used, i.e., any Unicode character, excluding NUL, restricted characters, the surrogate blocks, FFFE and FFFF. The unrestricted characters allowed by both the XML 1.0 and the XML 1.1 standards:

   ```
   #x9 | #xA | #xD | [#x20-#x7E] | #x85 | [#xA0-
   #xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
   ```

   Alert origination system software should verify only Unicode characters are used during in the CAP alert authoring process. Software libraries have default character translations, which can conceal non-UTF8 character issues, making it difficult to diagnose problems later. This usually requires explicit quality control testing during software development because mistaken character sets issues frequently manifest as valid, but wrong, characters in the other character set. Problems commonly occur using copy/paste text between external applications, e.g., Microsoft Word. Problems with the trademark and euro currency symbol, ellipsis, single and double "smart quotes," en and em dash, and the OE ligature characters are often reported. For example, checking the application cut/paste text, before encoding in UTF-8, for byte values 0x80 through 0x9F can indicate potential WINDOWS-1252 character set transformation problems. Another indicator, some applications replace non-Unicode characters with a QUESTION MARK or INVERTED QUESTION MARK or the Unicode REPLACEMENT CHARACTER.

2. The following Unicode characters should be avoided in CAP XML (from the XML 1.1 standard, even if using XML 1.0):

   Document authors are encouraged to avoid "compatibility characters," as defined in Unicode. The characters defined in the following ranges are also discouraged. They are either control characters or permanently undefined Unicode characters:

   ```
   [#x1-#x8], [#xB-#xC], [#xE-#x1F], [#x7F-#x84], [#x86-
   #x9F], [#xFDD0-#xFDDF],
   [#x1FFFE-#x1FFFF], [#x2FFFE-#x2FFFF], [#x3FFFE-
   #x3FFFF],
   ```

```
[#x4FFFE-#x4FFFF], [#x5FFFE-#x5FFFF], [#x6FFFE-
#x6FFFF],
[#x7FFFE-#x7FFFF], [#x8FFFE-#x8FFFF], [#x9FFFE-
#x9FFFF],
[#xAFFFE-#xAFFFF], [#xBFFFE-#xBFFFF], [#xCFFFE-
#xCFFFF],
[#xDFFFE-#xDFFFF], [#xEFFFE-#xEFFFF], [#xFFFFE-
#xFFFFF],
[#x10FFFE-#x10FFFF].
```

3. Line breaks should be written as a single LINE FEED (#xA).
   XML parsers translate line breaks to a single #xA. But line-break handling changed between XML 1.0 and XML 1.1 to resolve some interoperability issues with different operating systems. Using only a LINE FEED avoids those interoperability issues.
4. Control codes #x1 through #x1F and #x7F through #x9F should be written as numeric character references, except for whitespace and line breaks.
   Most control codes should have been avoided by bullet points 1, 2 and 3 above. The remaining control codes, except whitespace and line breaks, should be encoded as numeric character references in both XML 1.0 and XML 1.1.
   See the XML 1.0 and 1.1 standards for the detailed rules when to escape characters, e.g., elements don't require line breaks and tabs to be escaped, but attributes do. It's best to avoid using the other control characters in XML.
5. Only the five XML predefined character entities (amp, lt, gt, apos, quot) should be written in XML output.
   CAP Alerts should not contain other XML named entities, external entities or parameter entities.
   Essentially all XML allowed Unicode characters should be written as normal text in UTF-8, except when escaping XML markup characters with the predefined character entities and escaping control codes with numeric character references. Other numeric character entities should be limited, because they make parsing and matching text strings more complicated. HTML entities and other markup should not be used.
6. CAP creators should fully normalize all XML parsed entities, as defined in XML 1.1 even if using XML 1.0, before the processes that save, digitally sign or post a CAP Alert.
   CAP creators should validate that their messages are both well-formed and fully normalized before saving, signing or posting. Normalized XML parsed entities avoid many Unicode interoperability and security issues with CAP consumers by creating predictable and consistent text strings. CAP processors, i.e., aggregators or gateways, should verify but not transform, XML input is well-formed and fully normalized. Caution: fully normalized is not the same as Canonical XML.