Infrastructure - Bug #1034

identifier in resolve response objectLocation/url not escaped

2010-10-29 22:37 - Dave Vieglais

Status:	Closed	**Start date:**	2010-11-07
Priority:	Normal	**Due date:**	
Assignee:	Rob Nahf	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Sprint-2011.01		
Milestone:	CCI-0.5	**Story Points:**	
Product Version:	*		

Description

The objectLocation/url values should contain valid URLs. Currently this is not the case as reserved characters don't appear to be properly escaped. For example:

http://cn-dev.dataone.org/cn/object/knb:testid:2010302142227709

should be written:

http://cn-dev.dataone.org/cn/object/knb%3Atestid%3A2010302142227709

Subtasks:

Task # 1104: Add decode function to balance encoding ones	**Closed**
Task # 1036: add url encoding to url string in cn/resolve filter	**Closed**
Task # 1037: develop integration tests for the urlencoded OLLs	**Closed**

History

#1 - 2010-10-30 01:59 - Matthew Jones

Although in principal I agree with the sentiment here, percent encoding on the server side is exceedingly difficult to do in practice, because according to RFC 3986 percent encoding should only occur on reserved characters when they occur in a part of the URI that defines them as having a reserved purpose. If the characters are being used for their reserved purpose, then they must not be escaped, and if they are being used in a part of the URI for which they do not have a reserved purpose, they should not be escaped. Knowing when to escape is difficult. For example, Dave's example above is wrong, in that the colons above need not be percent encoded because they are in the PATH component of the URI, which lists a colon as an allowable character from RFC 3986:

From RFC 3986, section 3.4 (http://tools.ietf.org/html/rfc3986#section-3.3):

pchar = unreserved / pct-encoded / sub-delims / ":" / "@"

Wikipedia summarizes the issue like so:

From http://en.wikipedia.org/wiki/Percent-encoding (emphasis mine):

'When a character from the reserved set (a "reserved character") has special meaning (a "reserved purpose") *in a certain context*, and a URI scheme says that it is necessary to use that character *for some other purpose*, then the character must be percent-encoded. ....'

'URIs that differ only by whether a reserved character is percent-encoded or appears literally are normally considered not equivalent (denoting the same resource) *unless it can be determined that the reserved characters in question have no reserved purpose*. This determination is dependent upon the rules established for reserved characters by individual URI schemes.'

The second paragraph above must be read carefully, but it basically says that whether or not a reserved character should be percent encoded depends on the URI scheme in use AND on where (in which component) the character appears in the URI. A simple example is for the http URI scheme, in which the "/" character is reserved within the path component of the URI but is not reserved within the query component, so it needs to be percent encoded within the path component but only if it is not used for its intended purpose as a path delimiter, and it should not be percent encoded in the query part of the URI. Here's the relevant section from RFC 3986, which defines the syntax of the query component:

From RFC 3986, section 3.4 (http://tools.ietf.org/html/rfc3986#section-3.4):

query = *(pchar / "/" / "?")

'The characters slash ("/") and question mark ("?") may represent data within the query component. Beware that some older, erroneous implementations may not handle such data correctly when it is used as the base URI for relative references (Section 5.1), apparently because they fail to distinguish query data from path data when looking for hierarchical separators. However, as query components are often used to carry identifying information in the form of "key=value" pairs and one frequently used value is a reference to another URI, it is sometimes better for usability to avoid percent-encoding those characters.'

Quite complex. But it gets worse...

For another example,imagine that we are trying to represent the 'get()' URL for an object who's identifier is a URN. The URN scheme itself is a specific case of a URI subject to percent encoding rules (but differing somewhat in how the rules work from the HTTP scheme). URNs have the following BNF notation:

::= "urn:" ":"
where is the Namespace Identifier, and is the Namespace Specific String.

Here there are two colons that have a reserved purpose, the first of which separates the 'urn' literal prefix from the NID, and the second of which separates the NID from the NSS. Under no circumstances should either of these two colons be escaped in a URN. In addition, colons are not necessarily reserved characters within the NSS, and so they should be used without percent encoding within the NSS, unless the particular NID defines them as reserved within the NSS. So, for just the URN part of this, doing proper percent encoding requires a thorough consideration of the rules of URIs, URNs, and particular NIDs to determine whether or whether not a particular character such as a colon should be percent encoded within the NSS. Then layer on top of that the rules that are needed to take that properly-percent-encoded URN and embed it in a GET URL with additional proper percent encoding (basically percent encoding on top of percent encoding).

Other URI schemes that we use for identifiers will have different rules, whereby at times it would be appropriate to escape a second colon, and at other times it would be inappropriate. Taken across all URI schemes, the problem is extremely challenging for just our identifiers alone.

In addition, in order to accomodate both URIs schemes like HTTP and LSID URNs and non-URI schemes like DOIs, DataONE doesn't even define its Identifiers as URIs -- they are defined as opaque unicode strings sans the non-printing characters (still somewhat ambiguously defined I think). Thus, these are all legitimate identifiers in DataONE, even though only the first two are actually URIs
1) http://a.com/b/c:6?d/e
2) urn:lsid:a.com:b:c/d:e
3) doi:10.6574/2013.a.b/c
4) foo.1.1

In URI (1), none of the "/" or ":" characters should be percent encoded (including the last slash between d and e in the query component). In URI (2), none of the colons should be percent encoded, and the "/" between c and d should not be encoded either. In (3) and (4), we should do no percent encoding because they are not even URIs. I can come up with many examples in which we should do percent encoding according to the rules, but my main point is to show the areas where a simplistic view of percent encoding rules is wrong for our identifiers, and that its even more complex when these percent-encoded identifiers are embedded in URLs. I posit that coming up with a universal percent encoding scheme for DataONE identifiers is intractable on the server side, and therefore we should treat these identifiers as they were originally intended: as opaque strings. If a particular D1 identifier instance happens to represent a URI, and if a client wants that URI to be properly referenced and used by various URI-aware tools, then it should be incumbent upon the client to submit the properly percent-encoded URI as the D1 identifier in the first place; all D1 would do is to properly store and return the identifier as it was submitted originally by the client.

Finally, to stir the pot even a little more.... In addition to the rules about percent encoding in URIs, there are additional rules about escaping in certain document formats such as XML. So, if a URI is embedded within an XML element, certain characters will need to be escaped to properly transport the data in the XML element container, and then unescaped when it is extracted. These rules overlap with but are different from the rules in URI percent encoding, but note that if an XML reserved character is found in a URI (notably '%', "&', '<', '>', and others), then it will need to be XML escaped when being serialized into XML such as SystemMetadata, and unescaped when being deserialized. Because % is itself a reserved character in XML and is used as an escape character in URI percent encoding, a properly URI-percent-encoded then XML-escaped character will look quite complex indeed. For example, transporting an escaped slash in a URI path component would look like this in an XML element: %2F

Looking forward to the discussion that this stirs up. ;) Let the wild rumpus begin...

#2 - 2010-10-30 02:29 - Dave Vieglais

As with any string forming the data of an XML element or attribute, it must be properly encoded to be serialized to XML and similarly decoded when deserialized. In this particular case, the XML fragment example is correct as "%" is not a reserved character in XML. The five characters that are, and so must be appropriately escaped in XML are:

quot "
amp &
apos '
lt <
gt >

Similarly, the identifier is being appended as a single path element at the end of the REST URLs. As such, the data value should be properly encoded so as to be unambiguous and not misinterpreted as indicating additional path or data beyond the single data value which is the identifier.

As such, when serializing an identifier to be embedded as we do as the final path element on a URL, the characters must be properly escaped. Similarly when the URL is deserialized, the data being transported, in our case the identifier, must be properly decoded.

I agree that the guidance for encoding URIs is a bit ambiguous, however in our situation it is quite clear since we are effectively dealing with a single path element that must be transmitted without further interpretation.

So basically, it seems there is some confusion about identifier representation. When putting an identifier with reserved XML characters in an XML document, those characters must be escaped to avoid misinterpretation. Similarly with putting an identifier in a URL as a path element. This also applies to other encoding schemes as well - JSON, SQL, etc.

That has nothing to do with the actual string characters or interpretation of an identifier, regardless of what scheme is using. It has everything to do with proper encoding of data for packaging in the language of choice.

#3 - 2010-10-30 03:43 - Dave Vieglais

Some further details on encoding path elements in URLs (rfc3986). The ABNF for path elements is:

path = path-abempty ; begins with "/" or is empty
/ path-absolute ; begins with "/" but not "//"
/ path-noscheme ; begins with a non-colon segment
/ path-rootless ; begins with a segment
/ path-empty ; zero characters

path-abempty = *("/" segment)
path-absolute = "/" [segment-nz *("/" segment)]
path-noscheme = segment-nz-nc *("/" segment)
path-rootless = segment-nz *("/" segment)
path-empty = 0

segment = *pchar*
*segment-nz = 1*pchar*
segment-nz-nc = 1(unreserved / pct-encoded / sub-delims / "@")
; non-zero-length segment without any colon ":"

pchar = unreserved / pct-encoded / sub-delims / ":" / "@"

unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"

pct-encoded = "%" HEXDIG HEXDIG

sub-delims = "!" / "$" / "&" / "'" / "(" / ")"
/ "*" / "+" / "," / ";" / "="

The case of interest is the "segment", which is a sequence of pchar, which in turn is defined as:

unreserved OR pct-encoded OR sub-delim OR ":" OR "@"

so indeed it is valid to have colons appear in path segments (provided it is not the first path segment), however it is not invalid to encode the colon (or "@").

This doesn't really clarify anything with respect to the ":" other than to indicate that apparently "%3A" and ":" are equivalent when appearing in a URL path segment.

#4 - 2010-11-04 22:13 - Rob Nahf

There seems to be a method in HttpServletResponseWrapper called encodeURL() (string in, encoded string out) that's the logical first place to start. Testing will likely consist of a series of potentially unsafe identifiers against cn/resolve. Not sure whether these would be best positioned as unit tests or as integration tests.

#5 - 2010-11-07 20:14 - Rob Nahf

- Target version changed from Sprint-2010.44 to Sprint-2010.45

I'm having trouble creating test data with some identifiers ('?'). Not sure if mn.create() has any validation functionality.

#6 - 2010-11-08 17:41 - Rob Nahf

- Status changed from New to In Progress

#7 - 2010-11-08 20:57 - Rob Nahf

- Position set to 23

#8 - 2010-11-09 23:35 - Rob Nahf

a stab at a concise encoding rule: "For get operations, identifiers must be represented by characters within the the pchar character set defined in RFC 3986 (unreserved, sub-delims,":",or "@"), or else the non-conforming characters will be replaced by percent encoded representations." Note, the general URI reserved characters "/","?","#" will be escaped by this rule, avoiding the situation where an identifier is interpreted as a different identifier with a query portion following. The UTF-8 representations of non-ascii characters will be percent encoded.

With regards to double-encoding situations, see RFC 3986 Section 2.4 "When to encode or decode" (http://tools.ietf.org/html/rfc3986#section-2.4), especially the last paragraph:

Because the percent ("%") character serves as the indicator for
percent-encoded octets, it must be percent-encoded as "%25" for that
octet to be used as data within a URI. Implementations must not
percent-encode or decode the same string more than once, as decoding
an already decoded string might lead to misinterpreting a percent
data octet as the beginning of a percent-encoding, or vice versa in
the case of percent-encoding an already percent-encoded string.

#9 - 2010-11-10 17:15 - Rob Nahf

implementation notes: The Java URI class offers functionality for encoding URLs, honoring different encoding rules depending on the part of the URI (scheme, authority, path, query, fragment). However it conforms to a different (probably outdated) standard: RFC 2396. there is marked difference between the RFC2396 and RFC 3986, in terms of allowable ASCII characters. (2396 not 3986 is "?" "/" "[" "]" while 3986 not 2396 is ":" ";" "," "$" "&" "+" "="). So will need to create own encoder.

#10 - 2010-11-11 16:20 - Dave Vieglais

Confirming recent discussion on this issue- URLs contain only US-ASCII characters. RFC 3986, section 2.3:

unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"

ALPHA = %41-%5A and %61-%7A (A-Z and a-z)
DIGIT = %30-%39 (0-9)

Unicode identifiers will need to be appropriately percent escaped. For example, the identifier:

□□□□□□□□□□□□□□□

Would be represented in the GET URL as:

http://some.mn.org/mn/object/%A9%u2014%u03C0%B0%u2018%u03C0%B0%u221A%u2013%AE%B0%u2030%A5%C8

Correction, the encoded URL should be:

http://some.mn.org/mn/object/%E0%B8%89%E0%B8%B1%E0%B8%99%E0%B8%81%E0%B8%B4%E0%B8%99%E0%B8%81%E0%B8%A3%E0%B8%B0%E0%B8%88%E0%B8%81%E0%B9%84%E0%B8%94%E0%B9%89

#11 - 2010-11-12 17:48 - Dave Vieglais

%E0%B8%89%E0%B8%B1%E0%B8%99%E0%B8%81%E0%B8%B4%E0%B8%99%
E0%B8%81%E0%B8%A3%E0%B8%B0%E0%B8%88%E0%B8%81%E0%B9%84%E0%B8%94%E0%B9%89
(line-break added)

#12 - 2010-11-12 22:57 - Rob Nahf

- Position set to 2

- Target version changed from Sprint-2010.45 to Sprint-2010.46

- Position deleted (34)

#13 - 2010-11-12 22:58 - Rob Nahf

- Position deleted (3)

- Position set to 2

#14 - 2010-11-15 16:32 - Dave Vieglais

Some additional comments and code samples at:

http://mule1.dataone.org/ArchitectureDocs/GUIDs.html#serializing

#15 - 2010-11-16 17:34 - Rob Nahf

- Position deleted (8)

- Position set to 3

#16 - 2010-11-22 17:33 - Dave Vieglais

- Position set to 31

- Position deleted (21)

- Target version changed from Sprint-2010.46 to Sprint-2010.47

#17 - 2010-11-30 17:55 - Rob Nahf

- Target version changed from Sprint-2010.47 to Sprint-2010.48

- Position set to 1

- Position deleted (29)

#18 - 2010-12-13 18:34 - Dave Vieglais

- Position deleted (35)

- Target version changed from Sprint-2010.48 to Sprint-2010.50

- Position set to 7

#19 - 2011-01-03 02:04 - Dave Vieglais

- Target version changed from Sprint-2010.50 to Sprint-2011.01

#20 - 2011-01-10 17:26 - Rob Nahf

- Status changed from In Progress to Closed

d1_integration tests for ascii character encoding pass. unicode tests are deferred.