

# URI fragments, and the future

Norman Gray, University of Glasgow, UK

IVOA TCG meeting, Pune

2011 October 16



University  
of Glasgow

- Started with an RFC comment on the VOEvent spec
- ...but I think it's a bigger issue
- ...and a practical one, rather than a theoretical nit-picking one.

I'm not trying to warn of an apocalypse, here, but there's a smart thing to do, here (and by implication...)

Punctu-ation, isn't  
ju'st!dec\$ora/tion

<http://www.ietf.org/rfc/rfc3986.txt>

<http://www.ivoa.net/Documents/REC/Identifiers/Identifiers-20070302.html#defs>

[ivo://authorityID/resourceKey#local\\_ID](ivo://authorityID/resourceKey#local_ID)

VOEvent identifiers are defined in VOEvent 2.0, §2.2

---

## rfc 3986 is unambiguous

RFC 3986, §3.5:

- “The fragment identifier component of a URI allows indirect identification of a secondary resource by reference to a primary resource and additional identifying information.”
- “Fragment identifier semantics are independent of the URI scheme and thus cannot be redefined by scheme specifications.”
- “the fragment identifier is not used in the scheme-specific processing of a URI”

# potential problem 1: apis

A URI-handler API could be constructed in such a way that the implementation couldn't get access to the fragment.

This *would not be a bug* in the API



The screenshot shows the Java API documentation for the class `URLStreamHandler`. The page includes a navigation bar with links for Overview, Package, Use Tree, Deprecated, Index, and Help. The class is defined as `public abstract class URLStreamHandler` extending `Object`. A descriptive paragraph explains that this class is the common superclass for all stream protocol handlers. Below the description, there are sections for Constructor Summary and Method Summary. The Method Summary table lists various methods such as `equals`, `getHostPort`, `getHostAddress`, `hashCode`, `hostToEqual`, `openConnection`, `openConnection`, `parseURL`, `sameFile`, and `toURL`.

Method	Description
<code>equals(URL u1, URL u2)</code>	Provides the default equals calculation.
<code>getHostPort()</code>	Returns the default port for a URL parsed by this handler.
<code>getHostAddress(URL u)</code>	Get the IP address of our host.
<code>hashCode(URL u)</code>	Provides the default hash calculation.
<code>hostToEqual(URL u1, URL u2)</code>	Compares the host components of two URLs.
<code>openConnection(URL u)</code>	Opens a connection to the object referenced by the url argument.
<code>openConnection(URL u, Proxy p)</code>	Same as <code>openConnection(URL)</code> , except that the connection will be made through the specified proxy. Protocol handlers that do not support proxying will ignore the proxy parameter and make a normal connection.
<code>parseURL(URL u, String spec, int start, int limit)</code>	Parses the string representation of a url into a url object.
<code>sameFile(URL u1, URL u2)</code>	Compare two urls to see whether they refer to the same file, i.e., having the same protocol, host, port, and path.
<code>toURL(URL u, String protocol, String host, int port, String file, String ref)</code>	

---

## potential problem 2: caches

- A proxy or cache must ignore the fragment
- RFC §6.1: “When URIs are compared to select (or avoid) a network action, such as retrieval of a representation, fragment components (if any) should be excluded from the comparison.”
- That is, you ask the proxy/cache for `ivo://auth/obj#frag`, you get `ivo://auth/obj`
- This also *is not a bug* in the cache

---

potential problem 3, uri++

URIs won't last forever



---

## potential problem 3, uri++

- URIs won't last forever

- At some point – a decade? half a century? – there will be a replacement.

- URIs are important: there will be a mapping + gateways + proxies

- ...which may not be optional

Those gateways cannot be guaranteed to be friendly to URI schemes which depend on behaviour which the URI specification declares must not happen.

---

## non-problem: uris as names

- TAPRegExt uses URIs as *names*: `ivo://ivoa.net/TAPRegExt#upload-http`

- Here, there's no suggestion that the `#upload-http` 'thing' is a differently-retrieved resource

- This goes *with* the grain of the URI definition

Future-proof IVOA Recommendations by not going against the grain of the underlying specifications.

If the resource will ever be *retrieved*, then the standard should explicitly note that the fragment processing is client-side.

Norman Gray, <http://nxg.me.uk>