

ONE URL STANDARD PLEASE

JANUARY 30, 2017 | DANIEL STENBERG | 19 COMMENTS

Following up on the problem with our current lack of a universal URL standard that I blogged about in May 2016: [My URL isn't your URL](#). I want a *single, unified* URL standard that we would all stand behind, support and adhere to.

What triggers me this time, is yet another issue. A friendly [curl](#) user sent me this URL:

```
http://user@example.com:80@daniel.haxx.se
```

... and pasting this URL into different tools and browsers show that there's not a wide agreement on how this should work. Is the URL legal in the first place and if so, which host should a client contact?

- **curl** treats the '@'-character as a separator between [userinfo](#) and host name so 'example.com' becomes the host name, the port number is 80 followed by rubbish that curl ignores. (wget2, the next-gen wget that's in development works identically)
- **wget** extracts the example.com host name but rejects the port number due to the rubbish after the zero.
- **Edge** and **Safari** say the URL is invalid and don't go anywhere
- **Firefox** and **Chrome** allow '@' as part of the userinfo, take the '80' as a password and the host name then becomes 'daniel.haxx.se'

The only somewhat modern "spec" for URLs is the [WHATWG URL specification](#). The other major, but now somewhat aged, URL spec is [RFC 3986](#), made by the IETF and published in 2005.

In 2015, [URL problem statement and directions](#) was published as an Internet-draft by Masinter and Ruby and it brings up most of the current URL spec problems. Some of them are also discussed in Ruby's [WHATWG URL vs IETF URI](#) post from 2014.

What I would like to see happen...



Which group? A group!

Friends I know in the WHATWG suggest that I should dig in there and help them improve their spec. That would be a good idea if fixing the WHATWG spec would be the ultimate goal. I don't think it is enough.

The WHATWG is highly browser focused and my interactions with members of that group that I have had in the past, have shown that there is little sympathy there for non-browsers who want to deal with URLs and there is even less sympathy or interest for URL schemes that the popular browsers don't even support or care about. URLs cover much more than HTTP(S).

I have the feeling that WHATWG people [would not like this work to be done within the IETF](#) and vice versa. Since I'd like buy-in from both camps, and any other camps that might have an interest in URLs, this would need to be handled somehow.

It would also be great to get other major URL "consumers" on board, like authors of popular URL parsing libraries, tools and components.

Such a URL group would of course have to agree on the goal and how to get there, but I'll still provide some additional things I want to see.

Update: I want to emphasize that I do not consider the WHATWG's job bad, wrong or lost. I think they've done a great job at unifying browsers' treatment of URLs. I don't mean to belittle that. I just know that this group is only a small subset of the people who probably should be involved in a unified URL standard.

A single fixed spec

I can't see any compelling reasons why a URL specification couldn't reach a stable state and get published as "the" URL standard. The "living standard" approach may be fine for certain things (and in particular browsers that update every six weeks), but URLs are supposed to be long-lived and inter-operate far into the future so they really really should not change. Therefore, I think the IETF documentation model could work well for this.

The WHATWG spec documents what browsers do, and browsers do what is documented. At least that's the theory I've been told, and it causes a spinning and never-ending loop that goes against my wish.

Document the format

The WHATWG specification is written in a pseudo code style, describing how a parser would "walk" over the string with a state machine and all. I know some people like that, I find it utterly annoying and really hard to figure out what's allowed or not. I *much* more prefer the regular RFC style of describing protocol syntax.

IDNA

Can we please just say that host names in URLs should be handled according to IDNA2008 ([RFC 5895](#))? WHATWG URL doesn't state any IDNA spec number at all.

Move out irrelevant sections

"Irrelevant" when it comes to documenting the URL format that is. The WHATWG details several things that are related to URL for browsers but are mostly irrelevant to other URL consumers or producers. Like section "5. application/x-www-form-urlencoded" and "6. API".

They would be better placed in a "URL considerations for browsers" companion document.

Working doesn't imply sensible

So browsers accept URLs written with thousands of forward slashes instead of two. That is not a good reason for the spec to say that a URL may legitimately contain a thousand slashes. I'm totally convinced there's no critical content anywhere using such formatted URLs and no soul will be sad if we'd restricted the number to a single-digit. So we should. And yeah, then browsers should reject URLs using more.

The slashes are only an example. The browsers have used a "liberal in what you accept" policy for a lot of things since forever, but we must resist to use that as a basis when nailing down a standard.

The odds of this happening soon?

I know there are individuals interested in seeing the URL situation getting worked on. We've seen articles and internet-drafts posted on the issue several times the last few years. Any year now I think we will see some movement for real trying to fix this. I hope I will manage to participate and contribute a little from my end.

◀ CHROME ◀ CURL AND LIBCURL ◀ EDGE ◀ FIREFOX ◀ IETF ◀ RFC ◀ SAFARI ◀ URL ◀ WGET ◀ WHATWG

19 THOUGHTS ON "ONE URL STANDARD PLEASE"

 **Stephan Sokolow**

JANUARY 30, 2017 AT 09:53

I'd be cautious about restrictions on use of slashes in URLs, given that I've seen some very weird URLs coming out of blind concatenation as a result of templating engines not wanting to provide proper URL handling functions.

(Especially Jekyll, as used by GitHub pages, where people tend to rely on their installation for convenience, so they can't just extend it themselves with their own URL-manipulating template tags/filters.)

 ★ **Daniel Stenberg**

JANUARY 30, 2017 AT 10:01

@Stephan: then I'd like you to explain to me and the world the value in having browsers and tools accept more than (say) 9 slashes in a row. Sure there might be some very rare producers doing such URLs, but they should be identified, get bugs filed against them and fixed (or shamed, if they don't fix them).

Just because there is stupid software in the world, it doesn't mean we need to adjust the spec to make their crap compliant.

The "liberal in what you accept" rule is not working.

 **Stephan Sokolow**

JANUARY 30, 2017 AT 11:15

I was more urging that you err on the "9 or less" side than the "two" side. I don't see a problem from that for URLs originating in templating engines.

However, that does also remind me of another place where issues could creep out of the woodwork... applications which generate URLs from filesystem paths. (eg. native applications which rely on URL parsing to succeed before they can recognize the "file" scheme and apply realpath())

I know I'm probably just being skittish, but it feels uncomfortable to imagine what data loss bugs might be exacerbated in native applications coded to assume that all un-normalized but percent-escaped POSIX paths are valid URL path components.

(eg. While Geeqie doesn't use URLs, it IS an example of an application where load/save cycling is not idempotent if a path in an image collection fails to load... ostensibly as a convenience feature for clearing out stale entries.)

 **Simon Pieters**


JANUARY 30, 2017 AT 11:48

> I'm totally convinced there's no critical content anywhere using such formatted URLs and no soul will be sad if we'd restricted the number to a single-digit.

We can gather data to get an idea about where that limit should be, if we are to add a limit.

I checked httparchive (495,618 pages; 16,797,474 response bodies), looking for http or https URLs in src or href attributes with 3 or more slashes. 720 have 3 slashes, 271 have 4 slashes, 0 have 5+ slashes.

Searching in <https://nerdydata.com/> ("We index approximately 160 million websites") for href="http:// in "Deep Web" gives 0 results, 4 slashes approx. 5,170 found, 3+ slashes approx. 92,120 found. Searching for href="https:// gives 0 results, 4 slashes approx. 1,175 found, 3+ slashes approx. 3,290 found.

 **Henri Sivonen**

JANUARY 30, 2017 AT 12:09

> my interactions with members of that group that I have had in the past, have shown that there is little sympathy there for non-browsers who want to deal with URLs

On the contrary, WHATWG specs are useful for non-browser software to know what to do in order to be able to interact with the Web as it is (for better or worse) authored by people who test mainly in browsers.

It would be more accurate to say that there's little sympathy for non-browsers that want to deal with URLs in a browser-incompatible way.

> interest for URL schemes that the popular browsers don't even support or care about

Do you have examples of URL schemes that curl supports but are a problem if implementing from the WHATWG spec?

> I can't see any compelling reasons why a URL specification couldn't reach a stable state and get published as "the" URL standard. The "living standard" approach may be fine for certain things (and in particular browsers that update every six weeks), but URLs are supposed to be long-lived and inter-operate far into the future so they really really should not change. Therefore, I think the IETF documentation model could work well for this.

What do you suggest is done if a spec bug is discovered after the spec has been declared “stable”?

> The WHATWG specification is written in a pseudo code style, describing how a parser would “walk” over the string with a state machine and all. I know some people like that, I find it utterly annoying and really hard to figure out what’s allowed or not. I much more prefer the regular RFC style of describing protocol syntax.

The RFC style doesn’t work well for specifying how to handle erroneous input interoperably, which is why it’s not used for the parsing spec.

As for figuring out what you’re supposed to output, the URL Syntax section seems isomorphic to the RFC style even though it doesn’t use the exact same formatting.

> So browsers accept URLs written with thousands of forward slashes instead of two. That is not a good reason for the spec to say that a URL may legitimately contain a thousand slashes.

And indeed the spec doesn’t say that! There are exactly two slashes in the relevant section at <https://url.spec.whatwg.org/#scheme-relative-special-url-string>

> I’m totally convinced there’s no critical content anywhere using such formatted URLs and no soul will be sad if we’d restricted the number to a single-digit. So we should. And yeah, then browsers should reject URLs using more.

Why does robustly handling excessive slashes make you sad? Again, the spec doesn’t say that an arbitrary number of slashes is legitimate. It just says how to handle that case if that’s what you encounter.

> The slashes are only an example. The browsers have used a “liberal in what you accept” policy for a lot of things since forever, but we must resist to use that as a basis when nailing down a standard.

The whole XHTML episode is a counter example. I think countering that history needs more explanation.

 ★ Daniel Stenberg

JANUARY 30, 2017 AT 13:31

> WHATWG specs are useful for non-browser software to know what to do in order to be able to interact with the Web

You then assume that the browsers decide what “the Web” is and how URLs work – I say the web and the Internet (as URLs go way outside of the web too) is bigger than that. If I use curl with a URL and then run wget with that same URL, I interacted with the web twice and didn’t use any browser. I would still like them both to understand the URL the same way. They do that by adhering to the same spec.

> Do you have examples of URL schemes that curl supports but are a problem if implementing from the WHATWG spec?

I keep adding support to my projects for URLs that browsers support but my projects didn’t. And that’s for things like many-slashes – which you even say the WHATWG spec doesn’t allow, which makes the situation even more crazy. If the syntax spec says there must be two, but all parsers handle thousands. What does “must be two” really mean then?

Some of the other minor discrepancies we have are problematic because we define our way of working with URLs as per RFC 3986 and we’ve done that since 2005, so breaking that compliance risk breaking things for users so we’re reluctant. That could be fixed, with communication and with code over time.

But I’m not discussing how to fix curl here. I’m discussing how to get us to a single unified URL standard and my discussion here is what I think could work to get the large community on board. Even the most hard-core IETF people. I don’t think we can land a single standard without a group that is more than WHATWG.

> What do you suggest is done if a spec bug is discovered after the spec has been declared “stable”?

Like we’ve always done for specs through the history of IETF and others: we write erratas and when the errors or the deviances are too many we make an updated spec. I don’t see how URLs are a unique “protocol” in that way. That way you could also more easily identify software and products that would still be supporting the old format.

> The whole XHTML episode is a counter example

First, XHTML came after users had already been taught how to do things and that browsers are tolerant, and secondly I don’t think URLs are at a “HTML-level” of forgiveness or that they need that.

Modern protocols are instead designed strictly to deny/reject/close hard if not adhering, just so that implementations instead are encouraged to follow the spec to the point, for the specs to be clearer and by doing that achieving inter-op faster and more solidly.

 Robert O’Callahan

JANUARY 30, 2017 AT 23:26

A lot of URLs are created by people who paste them into HTML and tested only according to “does it work in browsers?” If you want a spec to handle all those URLs as the author intended, then it needs to handle them as browsers do. That’s not an opinion about how things should be, it’s just a fact.

> Like we’ve always done for specs through the history of IETF and others: we write erratas and when the errors or the deviances are too many we make an updated spec.

The difference between that and a “living standard” is that in a living standard the errata are folded into the spec at all times so authors and developers don’t have to go chasing around to find the errata, or fail to do so and accidentally use an obsolete version of the spec.

> Modern protocols are instead designed strictly to deny/reject/close hard if not adhering, just so that implementations instead are encouraged to follow the spec

That works when you have a very limited set of producers and consumers of the protocol. It doesn’t work when content is being directly produced by humans (e.g. HTML and URLs) and you have more than one consumer implementation (or even just multiple versions of one implementation).

 ★ Daniel Stenberg

JANUARY 30, 2017 AT 23:36

I know how URLs are made. I know what a “living standard” means. Yes I’m convinced we can be strict even with many producers and consumers (and it has been done elsewhere).

My post here was written with all that knowledge. I still firmly think that may laid out steps would be good and could help push toward a single unified URL standard.

 Mathias

JANUARY 30, 2017 AT 13:45

*I can’t see any compelling reasons why a URL specification couldn’t reach a stable state and get published as *the* URL standard.*

That’s the ultimate goal of any standard. What makes you think the URL Standard is any different?

 ★ Daniel Stenberg

JANUARY 30, 2017 AT 13:53

@Mathias: that's exactly what I'm saying. I don't think URLs are different.

 Mathias

JANUARY 30, 2017 AT 15:37

When I say "URL Standard", I mean the existing WHATWG URL Standard. What makes you think the WHATWG doesn't share the above-mentioned goal?

I'm confused as to how [creating another standard](#) would somehow lead to a stable spec any quicker. That only seems possible if the spec ignores reality, which would make it not very useful as far as specs go.

 ★ Daniel Stenberg

JANUARY 30, 2017 AT 23:31

@Mathias: they have explicitly said it is not their goal, I think that's wrong, as I write in this post.

We need "another standard" if whatwg is not up to the game to make that unified single standard. I don't care who makes it, as long as it gets done. And as I already wrote, I don't think whatwg is the right venue for that work.

 Mathias

JANUARY 31, 2017 AT 17:55

they have explicitly said it is not their goal

<https://twitter.com/annevk/status/826012526789931008> literally confirms this as the ultimate goal. Where did anyone say otherwise?

 ★ Daniel Stenberg

JANUARY 31, 2017 AT 20:25

There's no shortage of people telling me it is a "living standard" (as you can see in whatwg issues) and [you can also see that](#) "the ultimate" goal of having a fixed version is so far away that even discussing setting a final dead-line or considering a cut-off data where we can say it is done for this version can't be done yet.

 Daurinator

JANUARY 30, 2017 AT 14:40

I like my URI parsing to be as strict as possible.

I've so found found that sticking to RFC3986 to be the most sensible option to take.

I'd like to know what issues there are with RFC3986, and get an actual list of things to address.

I think the main issues, as you mention, is i18n.

For domain name components, I think settling on IDNA2008 is sensible; though library support perhaps is barely there (libidn2 only gained proper support *this year*: it'll take a while to filter out).

For scheme, userinfo, path and query components, we probably want some form of UTF8. The IRI spec (RFC 3987) *might* be enough here?

I don't want to support multiple slashes, or the other weird things browsers have gained support for in pursuit of postel's maxim.

 ★ Daniel Stenberg

JANUARY 30, 2017 AT 15:01

@Daurinator: two things come to mind that we do in curl that aren't RFC3986 (apart from adding support for one to three slashes): we escape-encode incoming spaces and we percent-encode > 7bit inputs, to act more like browsers. (Primarily important for redirects.)

 Simon Pieters

JANUARY 30, 2017 AT 16:11

> Like we've always done for specs through the history of IETF and others: we write errata and when the errors or the deviances are too many we make an updated spec.

Then it's still not a "fixed spec", like you said you wanted? Errata is delaying publishing bug fixes until there are "too many"; meanwhile those bug fixes are often missed because people don't pay attention to errata. Living standard means the number of too many errata is > 0.

 ★ Daniel Stenberg

JANUARY 30, 2017 AT 16:18

@Simon: that's what I consider a fixed spec. Like how RFC2616 specified HTTP/1.1 until the RFC7230 series came out. A fixed spec that defined how it worked, until the update was out that then unified the world again. A "living" spec we can't even know if we're compliant with it at any point since it might've changed since last week. Who's compliant then and to what?

I think URLs should be able to live for an extended period of times. Much much longer than the browsers update cycles.

 Tobie Langel

JANUARY 31, 2017 AT 16:16

> I think URLs should be able to live for an extended period of times. Much much longer than the browsers update cycles.

That's easily achievable by making every change to the URL spec backwards compatible.

COMMENTS ARE CLOSED.