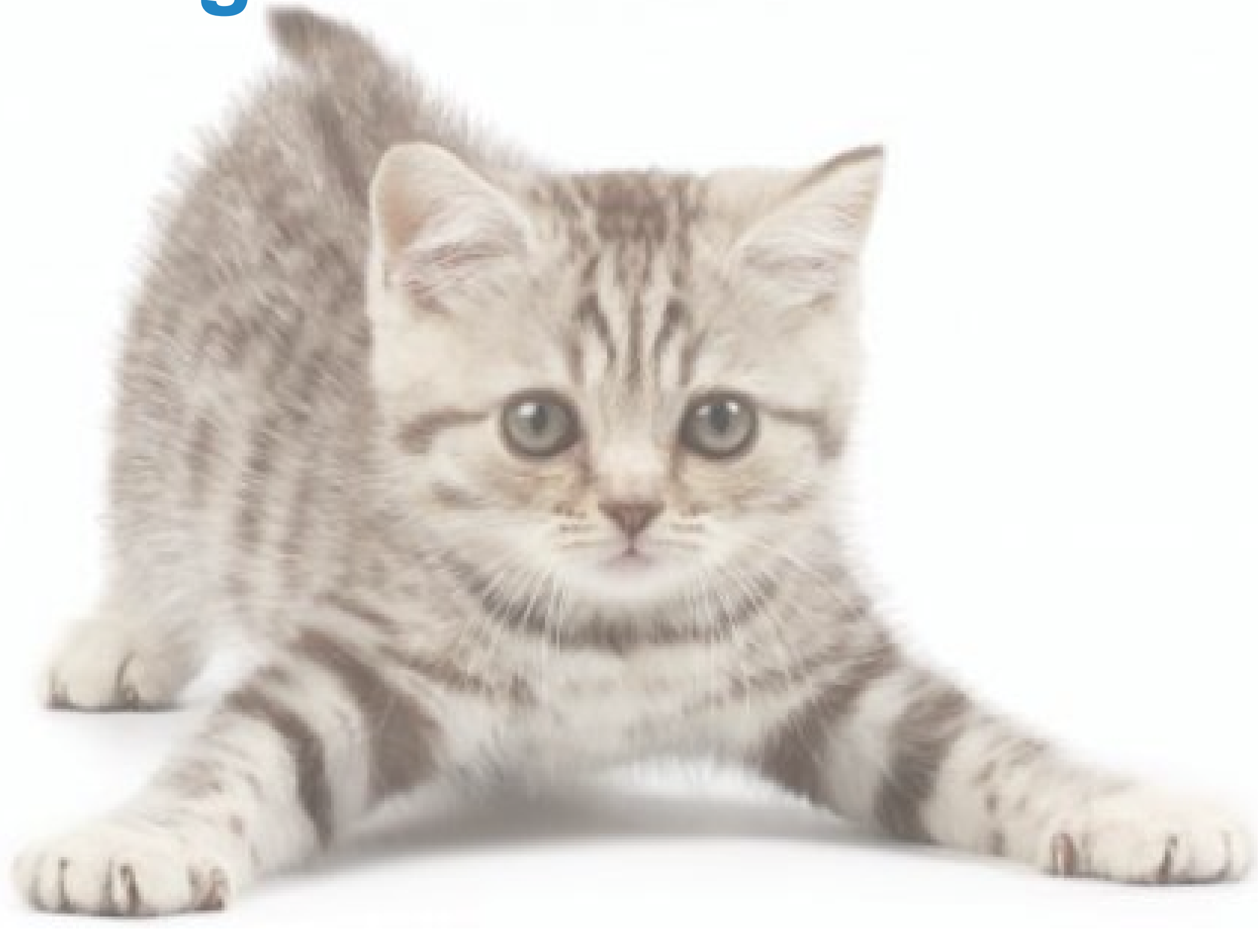
Meet purl: a "mostly" universal software package URL that purrs



source: https://www.severnedgevets.co.uk/pets/advice/advice-new-kitten-owners

Philippe Ombredanne

My mission: **make it easier to reuse FLOSS**

Enthusiast FLOSS developer
AboutCode, Linux kernel, a bit on strace, SPDX (and Eclipse, JBoss, and more)

CTO at nexB Inc. a software company helping software teams understand where their code comes from (and its licensing, vulnerability, quality, etc) with a combo of:

- FLOSS tools
- a commercial enterprise Dashboard

Why should you care?

- If you use more than one package environment and programming language
- You need to talk ABOUT packages across these boundaries
- Inventory all packages in your system or app
    - or just every packages (libraries.io)
    - or any package (grafeas)

The problem

I am telling you that I am using **"file"**, a fine package

- Pypi: https://pypi.python.org/pypi/**file ?**
- npmjs: https://www.npmjs.com/package/**file ?**
- Cargo: https://crates.io/crates/**file ?**
- Debian: https://packages.debian.org/stretch/**file ?**

source: http://www.iemoji.com/view/emoji/1853/smileys-people/thinking-face

The problem

We build and release software by massively consuming and producing software packages such as NPMs, RPMs, Rubygems, etc.

Each package manager, platform, type or ecosystem has its own conventions and protocols to identify, locate and provision software packages.

source: http://www.iemoji.com/view/emoji/1853/smileys-people/thinking-face

The origins

- needed something for ScanCode to point to packages in a uniform way
- Grafeas was defining some Resource URI of sorts that looks damned good (kudos to JFrog)
- Libraries.io was inventorying all the things
- Other package indexes seem all to use mostly similar approaches with subtle differences

When tools, APIs and databases process or store multiple package types, it is difficult to reference the same software package across tools in a uniform way.

The solution

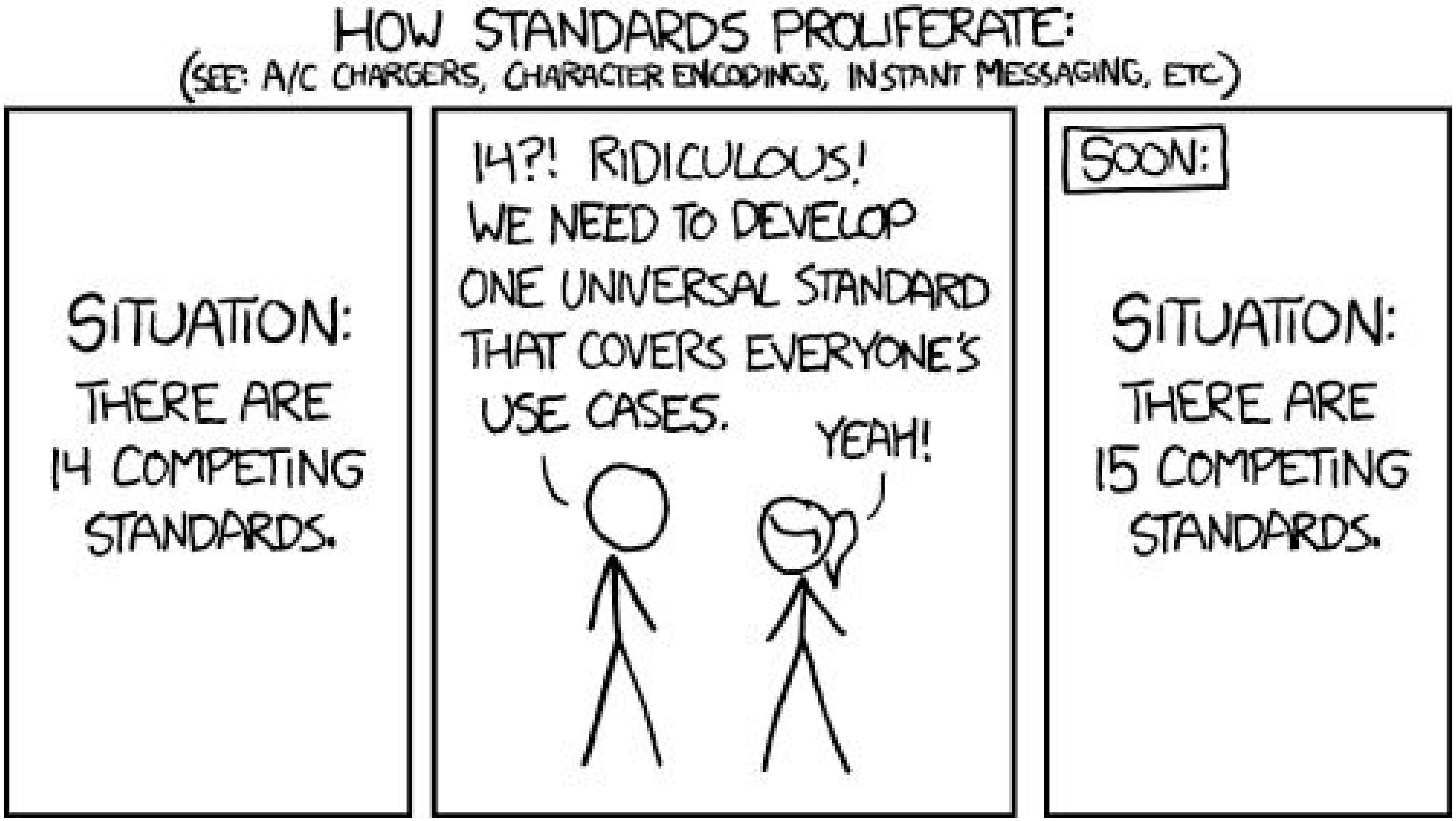
An expressive and simple package URL

To discuss about, identify & locate software packages reliably across:

- tools,
- DBs, indexes,
- APIs,
- and languages.

source: http://pluspng.com/png-25497.html

Avoiding the standards trap?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

source: https://xkcd.com/927/

The approach

1. A social experiment, starting an open conversation
2. Simple but nothing new, just enacting existing ways

A purl or package URL is an attempt to **standardize** existing approaches to reliably identify and locate software packages.

What is a purl?

Six data elements

type, namespace/name, version, qualifiers, subpath

A syntax for a URL string

```
bitbucket:birkenfeld/pygments-main@244fd47e07
deb:debian/curl@7.50.3-1?arch=i386&distro=jessie
```

Syntax

```
docker:gcr.io/customer/dockerimage@sha256:244fd47e07d1004f0aed9c

gem:jruby-launcher@1.1.2?platform=java

github:package-url/purl-spec@244fd47e07d1004f0aed9c

golang:google.golang.org/genproto#googleapis/api/annotations

maven:org.apache.xmlgraphics/batik-anim@1.9.1?repository_url=repo.spring.io

npm:foobar@12.3.1

nuget:EnterpriseLibrary.Common@6.0.1304

pypi:django@1.11.1

rpm:fedora/curl@7.50.3-1.fc25?arch=i386&distro=fedora-25
```

Six data elements

- **type**: the package "type" or package "protocol" such as maven, npm, nuget, gem, pypi, etc. Required.
- **namespace**: some name prefix such as a Maven groupid, a Docker image owner, a GitHub user or organization. Optional and type-specific.
- **name**: the name of the package. Required.
- **version**: the version of the package. Optional.
- **qualifiers**: extra qualifying data for a package such as an OS, architecture, a distro, etc. Optional and type-specific.
- **subpath**: extra subpath within a package, relative to the package root. Optional.

But, wait! this is a URI!?

- This is a locator alright hence a URL
- This is not a purists debate
- This has been reviewed by URL/URI "authorities"

One tidbit that needs ironing

- Single scheme vs. multiple schemes e.g:

```
pgk:pypi/django@2.0

vs.

pypi:django@2.0
```

- Implementation in multiple languages! HELP!

Credits and contributors

Alexios Zavras @ Intel
Anand Gaurav @ Nuget
Andrew Nesbitt @ libraries.io
Anne van Kesteren @ Whatwg and W3C
Brian Fox @Maven and Sonatype
Dan Rollo @ JFrog/Artifactory and Grafeas
Guillem Jover @ Debian
Jack Firth @ Racket and Google
Jannis Gebauer @ pyup
Jiri Popelka and Fridolín Pokorný @Red Hat fabric8 openshift analytics
Jonas Öberg @ FSFE
and more more missing

kasper3 @ Nuget
Liz Rice @ aquasecurity
Mark Nottingham @ http ;)
Nick Cross @ Red Hat
Rebecca Turner @ npm
Sam Boyer @Golang/dep
Sebastian Schuberth @ HERE Technologies and ORT
Stephen Milner and Jason Shepherd @ Red Hat Victims
Steve Springett @ OWASP
Sven Slootweg @joe2pi91
Todd Gamblin @ LLNL and spack
Vincent Batts @ Atomic and Red Hat
Wendy "R2wenD2" @Grafeas and Google
William Bartholomew @ Microsoft

Tools and "spec"

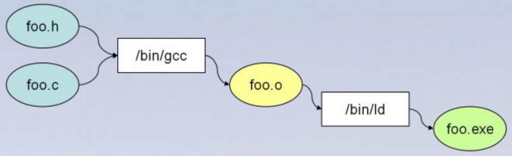
https://github.com/package-url/purl-spec

Go and Python implementations

https://github.com/package-url/packageurl-go
https://github.com/package-url/packageurl-python

(Java, .Net and JS on the way?)

Thank you!



Credits

Special thanks to all the people who made and released these awesome free resources:

- Presentation template by SlidesCarnival
- Photographs by Unsplash
- Icons from openclipart.org

And all the FLOSS software authors!