Improving Reliability with Advanced Content Negotiation

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Revisiting the contract between client and server

Advanced content negotiation

- Vendor-specific media types (input and output)

Versioning in a RESTful world

- Should RESTful APIs be versioned?

Revisiting the Contract Between Client and Server

URI
(resource identifier)

HTTP method

Payload

(represented by media types like application/json)

Revisiting the Contract Between Client and Server

"application/json" tells us something about the format of the data, but not about the type

“A REST API should spend almost all of its descriptive effort in defining the media type(s) used for representing resources and driving application state, or in defining extended relation names and/or hypertext-enabled mark-up for existing standard media types.”

Roy Fielding - https://bit.ly/2Kmsung

```
{ "value": [ {author}, { author} ],

  "links": [ …,
      {
        "href": "http://host/api/authors",
        "rel": “self",
        "method": “GET“},…]

}
```

~~Accept: application/json?~~

Accept: application/…

HATEOAS and Content Negotiation

We’re dealing with two different representations of the same resource

HATEOAS and Content Negotiation

Self-descriptive message sub constraint

- Each message must include enough info to process it

We’re returning the wrong representation

- We’re not as strict as we could be

Vendor-specific Media Types

application/vnd.marvin.hateoas+json

Vendor-specific Media Types

Top-level type

$\downarrow$

application/vnd.marvin.hateoas+json

Vendor-specific Media Types

Top-level type

↓

application/vnd.marvin.hateoas+json

↑

Vendor prefix

Vendor-specific Media Types

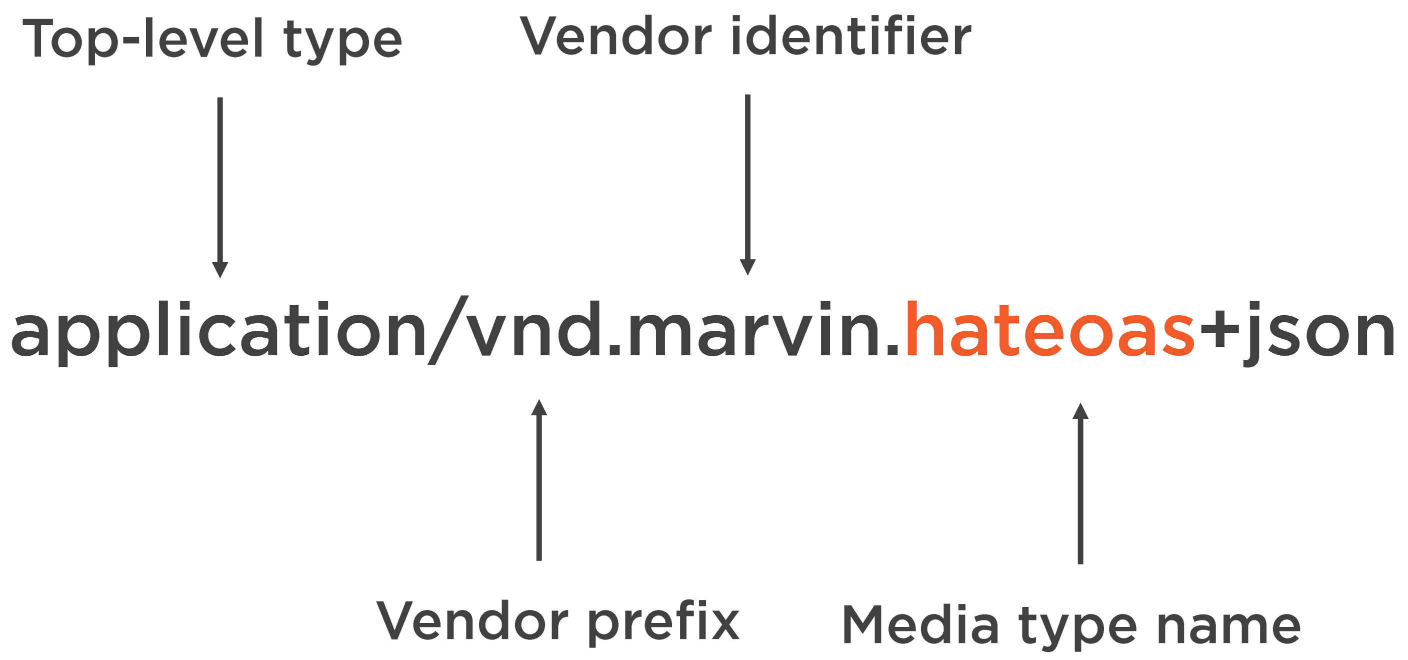
Top-level type

Vendor identifier

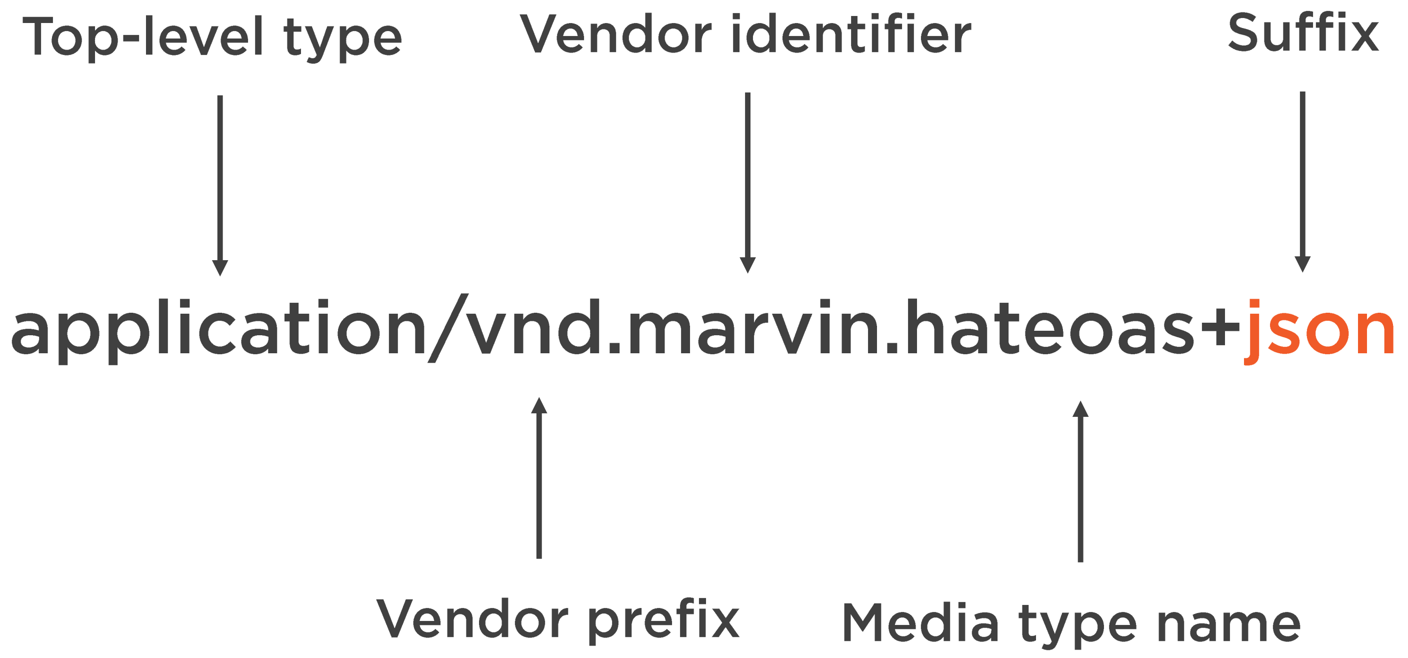
application/vnd.marvin.hateoas+json

Vendor prefix

Vendor-specific Media Types



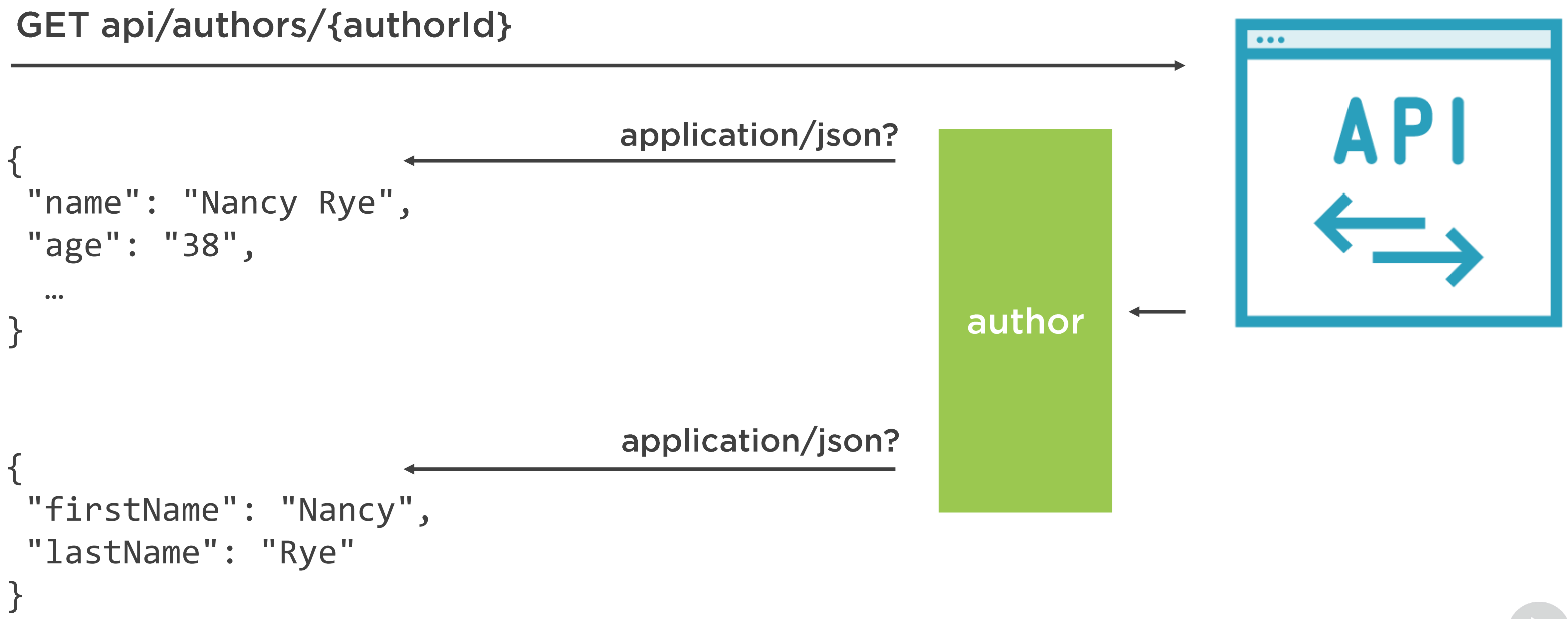
Vendor-specific Media Types



Demo

HATEOAS and content negotiation

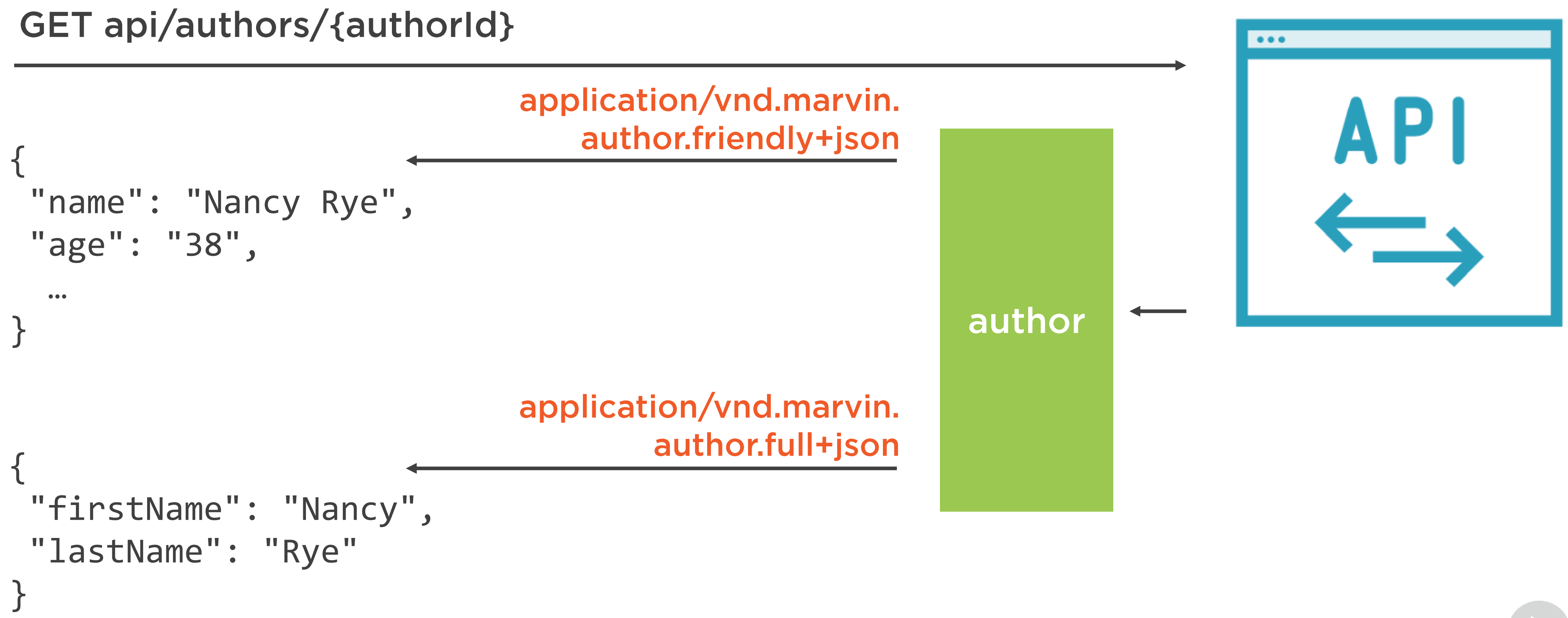
Tightening the Contract Between Client and Server with Vendor-Specific Media Types



```
{
 "name": "Nancy Rye",
 "age": "38",
  …
}
```

```
{
 "firstName": "Nancy",
 "lastName": "Rye"
}
```

Tightening the Contract Between Client and Server with Vendor-Specific Media Types



GET api/authors/{authorId}

application/vnd.marvin.
author.friendly+json

```
{
 "name": "Nancy Rye",
 "age": "38",
   …
}
```

application/vnd.marvin.
author.full+json

```
{
 "firstName": "Nancy",
 "lastName": "Rye"
}
```

Semantic Media Types

Semantic media types are media types that tell something about the semantics of the data – ie: what the data means

- Vendor-specific media types

Combining Semantic Media Types with HATEOAS

application/vnd.marvin.author.friendly+json

- Friendly representation without links

application/vnd.marvin.author.friendly +hateoas+json

- Friendly representation with links

application/vnd.marvin.author.full+json

- Full representation without links

application/vnd.marvin.author.full +hateoas+json

- Full representation with links

Combining Semantic Media Types with HATEOAS

There should be only one suffix per media type, and only officially registered suffixes should be used

Combining Semantic Media Types with HATEOAS

application/vnd.marvin.author.friendly+json

- Friendly representation without links

application/vnd.marvin.author.friendly .hateoas+json

- Friendly representation with links

application/vnd.marvin.author.full+json

- Full representation without links

application/vnd.marvin.author.full .hateoas+json

- Full representation with links

Combining Semantic Media Types with HATEOAS

Always provide a default representation that will be returned when no semantic information is passed through (eg: application/json)

Demo

Working with vendor-specific media types on output

Working with Vendor-specific Media Types on Input

When inputting data we can use vendor-specific media types as well through the Content-type header

Working with Vendor-specific Media Types on Input

appliation/json & application/vnd.marvin .authorforcreation+json.

- Representation without date of death

application/vnd.marvin. authorforcreationwithdateofdeath+json

- Representation with date of death

Demo

Working with vendor-specific media types on input

Versioning in a RESTful World

Functionality

Business rules

Resource representations

Versioning in a RESTful World

Through the URI

- api/v1/authors

Through query string parameters

- api/authors?api-version=v1

Through a custom header

- “api-version”=v1

“Don’t”

Roy Fielding (on versioning APIs, http://bit.ly/2hBPQXi)

Versioning in a RESTful World

Use HATEOAS to adapt to changes in functionality & business rules

Use CoI (Code on Demand) to adapt to changes in media types/resource representations

Evolvability

Versioning in a RESTful World

Version media types to handle change in representations

- application/vnd.marvin.author.friendly.**v1**+json
- application/vnd.marvin.author.friendly.**v2**+json

… or use friendly names

Summary

Use vendor-specific media types to differentiate between resources with and without HATEOAS links

Use semantic media types (implemented with vendor-specific media types) to attach meaning to representation requests

- Improves evolvability and reliability

Summary

Adapting to change

- HATEOAS for changes to functionality and business rules
- Versioned media types (until code on demand is feasible)

The REST architectural style was created with systems in mind that should live for years or decades, not months