 Ruben Verborgh

Your JSON is not my JSON

A case for more fine-grained content negotiation

Ruben Verborgh, Ghent University – iMinds – IDLab *30 November 2016*

¶ **Information resources can be expressed in different representations along many dimensions such as format, language, and time. Through content negotiation, HTTP clients and servers can agree on which representation is most appropriate for a given piece of data. For instance, interactive clients typically indicate they prefer HTML, whereas automated clients would ask for JSON or RDF. However, labels such as "JSON" and "RDF" are insufficient to negotiate between the rich variety of possibilities offered by today's languages and data models. This position paper argues that, despite widespread misuse, content negotiation remains the way forward. However, we need to extend it with more granular options in order to serve different current and future Web clients sustainably.**

Content on the Web—theory and practice

The way clients and servers exchange information on the Web is modeled by the **REST architectural style** [1]. It calls the main unit of information a *resource*, a conceptual relation between an identifier and a set of values. Those values are *representations*, concrete expressions of the resource. The Web's resource identifiers are URLs: using the HTTP protocol, clients can obtain a representation of a resource identified by a given URL. Through a mechanism called **content negotiation** [2], a client indicates its preference for certain kinds of representations, such that the server can respond accordingly.

An example of a resource is "the weather forecast for Amsterdam", which might have a URL such as `http://example.org/weather/amsterdam`. Using the HTTP headers `Accept` and `Accept-Language`, a client can indicate its preference for respectively the MIME types `application/ld+json` or `application/json`, and the languages English (`en-us`) or Dutch (`nl`). The server then tries to satisfy the client's preferences, identifying its eventual choice through the `Content-Type` and `Content-Language` headers.

In practice, however, content negotiation on the Web suffers from two issues. First, many websites avoid content negotiation by resorting to (only) representation-specific URLs. For example, the forecast in HTML might be available at `http://example.org/weather/amsterdam.html`, but a JSON version instead at `http://example.org/weather/amsterdam.json` or even `http://api.example.org/weather.php?city=AMS`. This absence of representation-agnostic resource URLs makes the client/server contract **unsustainable** [3]: clients use different identifiers for the same concepts solely because they read them in different formats, and new representations cannot be added in the future without introducing yet another set of identifiers. This unnecessarily results in the maintenance of **different interfaces** [4] for a single information source.

Second, servers insufficiently detail the properties of their representations. For example, a large part of APIs will indicate the content type of their responses with the MIME type `application/json`. While technically correct, this MIME type is almost always a severe *underspecification* of the actual format of the message, which adheres to much stricter rules than only the JSON syntax. Most JSON APIs will reply with a specific structure, using fields with a specific meaning, thereby conforming to an implicit schema

and interpretation. Clients relying on such knowledge act beyond the `application/json` parsing rules, thereby reducing sustainability through this **unwritten additional contract** [5].

Where MIME types fall short

Underspecification at the syntactical level

From the examples above, we conclude that MIME types remain at the syntactical level: they indicate what parser the client has to use. Such an indication is definitely necessary, especially when clients are compatible with multiple syntaxes. For example, data in the RDF model can be expressed in various syntaxes, such as RDF/XML, Turtle, TriG, and JSON-LD. These syntaxes are tied to certain levels of compatibility: RDF/XML and Turtle do not support RDF graphs, whereas TriG and JSON-LD do, so clients have reasons to prefer one over the other. These examples prove that scenarios in which servers support a handful of content types and syntaxes (as opposed to only HTML and JSON) can realistically occur.

However, even on the syntactical level, MIME types do not tell the whole story. Technically speaking, any JSON-LD document is also a JSON document, which in turn is also binary data. This means that for a JSON-LD document, the MIME types `application/ld+json`, `application/json`, and `application/octet-stream` are all *correct*, in the sense that they do not convey false assumptions to the client. However, crucially, only `application/ld+json` allows the client to **interpret its contents as RDF** [6] (in absence of other mechanisms such as the `Link` header). Hence, serving a JSON-LD document with a regular JSON MIME type is an underspecification similar to how the current generation of APIs advertises JSON but actually replies with specific JSON subsets.

Underspecification at the structural level

In the case of JSON-LD, the situation is even more complex, as there are two possible interpretation approaches. When interpreting a JSON-LD document as an RDF graph, no issues arise, since all possible serializations of an RDF graph as JSON-LD will be parsed into an equivalent graph. In contrast, when interpreting it as a regular JSON document, clients depend on a specific structure, as an RDF graph can be serialized into infinitely many different JSON structures. Through **JSON-LD framing** [7], the JSON-LD can be shaped into a certain JSON structure, but the MIME type will not contain information about this.

In general, all structurally flexible representation syntaxes (HTML, XML, JSON, CSV, …) parsed without a structurally independent interpretation (as possible with RDF-based syntaxes) suffer from this problem. In order to perform meaningful operations on a server's responses, clients typically make built-in structural assumptions that the MIME time strictly speaking does not allow for. The structural aspects of representations are thus underspecified as well.

Underspecification at the modeling level

Beyond the structural aspects of information are its modeling aspects. On the one hand, they concern the interpretation of specific structural entities in a certain way. For instance, many clients of specific JSON APIs will not only assume the structural presence and location of a certain field, but also that this field has a certain interpretation. If the generic `application/json` MIME type is used—as is most often the case—this interpretation is technically not allowed.

On the other hand, RDF-based syntaxes avoid the dependency on an implicit structure and interpretation, as the resulting graph uses IRIs, which have a universal meaning. However, it is unspecified what kind of IRIs the client can expect as entry points. For instance, when accessing a list of people, it is unclear what ontologies a client will encounter. They could be modeled using FOAF, Schema.org, and/or others. Therefore, when a client parses a Turtle representation and evaluates a query for the triple pattern `?p rdf:type foaf:Person`, a result count of zero matches could indicate either that the list does not contain any people, or that the server used another vocabulary. Again, the issue is underspecification of the representation's characteristics, this time on the level of modeling.

Possible (but inadequate) workarounds

An obvious approach to tackle the above problems seems the definition of more specific MIME types. For instance, a specific API could identify its own subset of JSON as `application/vnd.myapiresponse+json`. Such a solution has two blocking problems. First, there is no formal connection between this MIME type and JSON, as the `+json` suffix is merely a convention. Therefore, clients that might be syntactically compatible cannot know with certainty what parser to use; this for instance **applies to some JSON clients of JSON-LD** [6]. Second, and more importantly, pursuing such highly specific MIME types would result in combinatorial explosion. As an example, consider hypothetical subtypes of RDF syntaxes. We would need to define `text/vnd.mymodel.turtle`, but also `application/vnd.mymodel.trig` plus all other possible derivatives—and this is just a one-dimensional extension. In the case of JSON-LD, different structural representations would also need to be identified; even though all of these different MIME types would still be syntactically and structurally compatible with a regular JSON-LD parser. In other words, due to the many dimensions of possible variation, more specific MIME types, certainly for RDF, are a dead end.

Another option one could consider is offering multiple structural and modeling alternatives within a same representation. For instance, an RDF document could describe an entity using multiple vocabularies within the same representation. However, we cannot anticipate all possible vocabularies, and even providing only the most important ones has a negative impact on resource size. Furthermore, different vocabularies serve different *purposes*, some of which might conflict. For instance, Schema.org is purposely sloppy with is modeling constraints; as such, Schema.org markup is highly useful for purposes such as discovery, but inadequate for in-depth reasoning. Interleaving vocabularies might thus interfere with the outcomes of a reasoning process, or even introduce inconsistencies.

Toward more fine-grained content negotiation

Given that the above workarounds are inadequate, we direct our attention instead to fixing the current limitations of content negotiation. Fortunately, the HTTP content negotiation mechanism is extensible beyond MIME types. As an example, the **Memento protocol** [8] defines time-based negotiation for resources using the `Accept-Datetime` request header and the `Memento-Datetime` response header. A single URL thereby becomes the entry point to earlier representations of the same resource. Multiple dimensions of content negotiation can be combined together simply by using multiple headers.

Therefore, to negotiate beyond the syntactical level of MIME types, we need dimensions to capture the structural and modeling levels. First, the server needs to be able to *indicate* in its response what structure and/or model a given representation uses. This can happen through the HTTP **`Link` header** [9] and the **`profile` link relation type** [10]. A *profile* can be defined as "additional semantics that can be used to process a resource representation, such as constraints, conventions, extensions, or any other aspects that do not alter the basic media type semantics" [10], which covers our use cases of structures and models. Concretely, we would define IRIs for specific JSON interpretations, RDF vocabularies, and JSON-LD frames. The MIME type in the `Content-Type` header then identifies possible parsers, whereas profile links identify constraints within the syntax that allow additional interpretations. This way, profiles can be reused across syntaxes, which is especially relevant for RDF variants. We thereby avoid the combinatorial problems that more specific media types would bring. Moreover, *multiple* profile IRIs can be specified for a given representation, for instance, to combine expectations on the syntactical, structural, and modeling levels, or to describe content consisting of multiple facets.

Second, the client needs to be able to *request* specific profiles within the supported syntaxes indicated in the `Accept` field. This is possible by **explicitly indicating profiles** [10] as additional parameters of the MIME types listed in the `Accept` field. However, when multiple syntaxes are supported, this might lead to repetition of profiles, possibly complicating negotiation or introducing combinations again. Hence, an

`Accept-Profile` header might be considered, in analogy to other negotiation dimensions. As noted above, and in contrast with most other headers, profiles can be cumulative instead of mutually exclusive.

Sustainability: guaranteeing constants amid change

In the world of metadata, individual records frequently outlive the technologies used to describe them. Therefore, it is important to rely on sustainable publication mechanisms. In this context, "sustainable" means providing the right constants in an inevitably changing technological landscape. From the discussion above, it is apparent that formats and data models are part of the change; hence, no client/server contract should depend on them as main anchors.

Instead, content negotiation is an ideal mechanism, since its constants are almost exclusively tied to the information, and not to any specific technology (apart from DNS and HTTP). Relying on representation-independent URLs, late-binding them at runtime to concrete representations, is a backward- and forward-compatible way of dealing with the reality of representation heterogeneity. As such, we do not contribute a novel proposal, but rather re-argue the applicability of this existing mechanism. However, given that there exist more dimensions of heterogeneity nowadays, we advocate an active multi-dimension approach to negotiation, proving that the one-dimensional extensibility of MIME types is insufficient.

The limits of MIME types have already been exposed several times. Specifications such as JSON-LD suggest using the JSON MIME type when **needed for backward compatibility** [6]. Existing Web APIs lack sufficiently detailed MIME types, resulting in a **severe lack of interface reuse** [11] and a plethora of developer guidelines. Fortunately, positive examples emerge as well: **Activity Streams** [12] are served with the `application/ld+json` MIME type and a profile of `http://www.w3.org/ns/activitystreams`. Interestingly, this profile IRI is *actionable* information, as it resolves to a JSON-LD context. The context, however, is in turn served with the JSON-LD MIME type, without a more specific profile to indicate that a JSON-LD context is available. This begs the question whether standards should rely on MIME types at all, or rather focus on profiles in a syntax-agnostic way. If they do, profiles can evolve over time by recursively using profile-based content negotiation.

Particularly exciting is that *multiple* profiles can be combined in a *single* response, in contrast to the single-dimensional nature of MIME types. This allows multiple facets of metadata to **coexist in a single representation** [11], much like how webpages are typically composed out of different information boxes. Clients can then selectively extract those parts of information they want to interact with, as opposed to depending on purpose-specific information structures. However, in order for clients of different capability levels to interact with servers and with each other, it remains important they can obtain tailored representations of the same content using the same interchangeable identifiers. Consequently, content negotiation at various levels of granularity is a future-proof choice.

References

[1] Fielding, R.T. (2000), *Architectural Styles and the Design of Network-Based Software Architectures*, PhD thesis, University of California, available at: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[2] Fielding, R.T. and Reschke, J. (2014), *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, Request For Comments No. 7231, Internet Engineering Task Force, available at: https://tools.ietf.org/rfc/rfc7231.

[3] Verborgh, R., van Hooland, S., Cope, A.S., Chan, S., Mannens, E. and Van de Walle, R. (2015), "The Fallacy of the Multi-API Culture: Conceptual and Practical Benefits of Representational State Transfer (REST)", *Journal of Documentation*, Vol. 71 No. 2, pp. 233–252, available at: http://freeyourmetadata.org/publications/rest.pdf.

[4] Verborgh, R. (2013), "The lie of the API", November, available at: https://ruben.verborgh.org/blog/2013/11/29/the-lie-of-the-api/.

[5] Fielding, R.T. (2008), "REST APIs must be hypertext-driven", Untangled – Musings of Roy T. Fielding, October, available at: http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven.

[6] Sporny, M., Longley, D., Kellogg, G., Lanthaler, M. and Lindström, N. (2014), *JSON-LD 1.0*, Recommendation, World Wide Web Consortium, available at: http://www.w3.org/TR/json-ld/.

[7] Longley, D., Sporny, M., Kellogg, G. and Lanthaler, M. (2016), *JSON-LD Framing 1.1*, Draft Community Group Report, World Wide Web Consortium, available at: http://json-ld.org/spec/latest/json-ld-framing/.

[8] Van de Sompel, H., Nelson, M. and Sanderson, R. (2013), *HTTP Framework for Time-Based Access to Resource States – Memento*, Request For Comments No. 7089, Internet Engineering Task Force, available at: https://tools.ietf.org/rfc/rfc7089.

[9] Nottingham, M. (2010), *Web Linking*, Request For Comments No. 5988, Internet Engineering Task Force, available at: https://tools.ietf.org/rfc/rfc5988.

[10] Wilde, E. (2013), *The 'Profile' Link Relation Type*, Request For Comments No. 6906, Internet Engineering Task Force, available at: https://tools.ietf.org/rfc/rfc6906.

[11] Verborgh, R. and Dumontier, M. (n.d.). "A Web API ecosystem through feature-based reuse", available at: https://arxiv.org/abs/1609.07108.

[12] Snell, J.M. and Prodromou, E. (2016), *Activity Streams 2.0*, Candidate Recommendation, World Wide Web Consortium, available at: https://www.w3.org/TR/activitystreams-core/.

Cite this article in your publications

- Use the **BibTeX entry** to easily refer to this article.
- Alternatively, you can refer to this article as:
  Verborgh, R. (2016), "Your JSON is not my JSON – A case for more fine-grained content negotiation", in *Proceedings of the Workshop on Smart Descriptions & Smarter Vocabularies*.

Comment on this article