Access 4 Learning Community

Powered by: SIF

Simple. Secure. Scalable. Standard.

SIF Infrastructure Specification 3.3: Version Indication & Negotiation

www.A4L.org

Version 3.3, May 2019

Preface

This document, *SIF Infrastructure Specification 3.3: Version Indication & Negotiation*, is an addendum to the main volumes of the SIF Infrastructure Specification. It describes features that are experimental because they may represent areas that are rapidly evolving or rely on external specifications that are still in their draft state. Experimental features of SIF are not in scope for SIF compliance and certification processes. It is thus not compulsory for a product adopting SIF to incorporate these features in its implementation.

This document is our attempt to adopt two draft technical specifications by W3C and IETF as listed below. They are currently (May 2019) undergoing many changes.

- W3C's content negotiation by profile: https://w3c.github.io/dxwg/conneg-by-ap/
- IETF's negotiating profiles in HTTP: https://profilenegotiation.github.io/I-D-Accept--Schema/I-D-accept-schema

When experimental features of SIF gain enough attention and maturity, they will be considered for inclusion in the main part of SIF Infrastructure Specification.

SIF adopters and standard governance bodies are welcome to evaluate the experimental features and provide feedback to A4L for future improvements.

Disclaimer

The information, software, products, and services included in the SIF Implementation Specification may include inaccuracies or typographical errors. Changes are periodically added to the information herein. The SIF Association may make improvements and/or changes in this document at any time without notification. Information contained in this document should not be relied upon for personal, medical, legal, or financial decisions. Appropriate professionals should be consulted for advice tailored to specific situations.

THE SIF ASSOCIATION, ITS PARTICIPANT(S), AND THIRD PARTY CONTENT PROVIDERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY, RELIABILITY, TIMELINESS, AND ACCURACY OF THE INFORMATION, SOFTWARE, PRODUCTS, SERVICES, AND RELATED GRAPHICS CONTAINED IN THIS DOCUMENT FOR ANY PURPOSE. ALL SUCH INFORMATION, SOFTWARE, PRODUCTS, SERVICES, AND RELATED GRAPHICS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. THE SIF ASSOCIATION AND/OR ITS PARTICIPANT(S) HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS INFORMATION, SOFTWARE, PRODUCTS, SERVICES, AND

RELATED GRAPHICS, INCLUDING ALL IMPLIED WARRANTIES AND CONDITIONS OF: MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT.

IN NO EVENT SHALL THE SIF ASSOCIATION, ITS PARTICIPANT(S), OR THIRD PARTY CONTENT PROVIDERS BE LIABLE FOR ANY DIRECT, INDIRECT, PUNITIVE, INCIDENTAL, SPECIAL, CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF USE, DATA, OR PROFITS, ARISING OUT OF OR IN ANY WAY CONNECTED WITH THE USE OR PERFORMANCE OF THIS DOCUMENT, WITH THE DELAY OR INABILITY TO USE THE DOCUMENT, THE PROVISION OF OR FAILURE TO PROVIDE SERVICES, OR FOR ANY INFORMATION, SOFTWARE, PRODUCTS, SERVICES AND RELATED GRAPHICS OBTAINED THROUGH THIS DOCUMENT OR OTHERWISE ARISING OUT OF THE USE OF THIS DOCUMENT, WHETHER BASED ON CONTRACT, TORT, STRICT LIABILITY, OR OTHERWISE, EVEN IF THE SIF ASSOCIATION, ITS PARTICIPANT(S), OR THIRD PARTY CONTENT PROVIDERS HAVE BEEN ADVISED OF THE POSSIBILITY OF DAMAGES. IF YOU ARE DISSATISFIED WITH ANY PORTION OF THIS DOCUMENT OR WITH ANY OF THESE TERMS OF USE, YOUR SOLE AND EXCLUSIVE REMEDY IS TO DISCONTINUE USING THIS DOCUMENT.

Permission and Copyright

Document Conventions

Terms and Abbreviations

Term / Abbreviation	Description
A4L	Access 4 Learning.
HTTP	Hypertext Transfer Protocol.
JSON	JavaScript Object Notation. http://json.org/

Term / Abbreviation	Description
REST	Representational state transfer, an architectural style of properties and constraints applied to components, connectors and data elements in architecture. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
RESTful Web services	Web services that follow REST. For simplicity in this document, RESTful Web services is restricted to those that are HTTP-based.
SIF	Systems Interoperability Framework.
SIF Broker	A broker that mediate the interactions between SIF consumers and SIF providers.
SIF (Service) consumer	Any application that makes requests of, subscribes to, and receives Events from, one or more SIF providers.
SIF (Service) provider	An application that provides access to Objects of a selected type (e.g. for students) in accordance with its service interface and publishing of related Events.
SSL	Secure Socket Layer.
TBD	To be determined or defined.
XML	Extensible Markup Language. http://www.w3.org/XML/

Notations

Text in YELLOW emphasizes the area for attention in the body of the document.

`Text in BLUE` highlights characters appearing in HTTP transfer.

1. Context

Schema definitions abstract the structure, representation and rules of electronic data in an agreed format such that different parties can create, understand, exchange, store and trust the data conforming to their definitions. Typically, schemas continually evolve to align with the changing nature of a business domain.

Everything works well when different parties share the same schema definition. As new versions become available, however, these parties will not necessarily keep up in their pace of upgrade with the latest version of the schema so that they will become outofalignment with one another over time.

SIF consumers and providers face challenges because of schema evolution. An assumption in the past has been that a SIF environment will only support one version of a schema at a time; this assumption is challenged and impractical in a dynamic enterprise context. When a SIF consumer sends or receives an object to a SIF provider via REST, there are no adequate mechanisms currently in place for the provider to determine the exact version of the schema appropriate to the object to validate, extract and process the object and vice versa. The best guess currently is by interpreting the optional SIF namespace parameter declared in the object. The namespace itself, http://www.sifassociation.org/datamodel/au/3.4 for instance, is however deliberately coarse-grained, being associated with a minor version rather than a specific version. It is associated with schema versions 3.4, 3.4.1, 3.4.2 and so forth. Without knowing the actual version to use, the provider won't be able to fully and unambiguously process the object.

More generally, even though XML schemas are carefully designed for changes, maintaining both backwards and forward compatibility for the schemas is a major challenge[1]. One way to deal with schema differences - both type and version - between two parties is to let them negotiate their schemas as they exchange data, assuming they are flexible enough to deal with multi-versioning (i.e. supporting more than one version) and type variants so that at runtime the parties can establish a common schema which they both will accept.

[1] *See "XML Schema Versioning Use Cases" in Section 6 for further discussions.*

2. Problem Statement

There is a need for SIF consumers and SIF providers to declare precise schemas for objects sent over-the-wire and to negotiate which schema(s) to use at runtime. The goal is to decouple the rates of the two parties adopting new schemas as they become available such that they can be slightly out-of-alignment yet still able to exchange SIF objects with less friction and better interoperability.

For simplicity, any SIF broker - mediating between SIF consumers and providers - is seen as transparent in the negotiation process and hence will not be considered here. Nevertheless, it does not preclude a SIF broker from playing a future role in participating and supporting schema negotiations.

3. Method

Extensions to SIF infrastructure are proposed here to facilitate schema negotiations between SIF consumers and providers, covering both schema types and versions. They:

- borrow the concept of transparent content negotiation from RFC2295 in which HTTP user agents and HTTP servers handle negotiation of media type (PDF vs. HTML), language, charset, features (e.g. screen size) in HTTP content.
- extend IETF's work of HTTP profile negotiation (draft as on 9 Oct 2018) in which HTTP user agents and HTTP servers indicate and negotiate the profile used to represent a specific resource.
- adapt W3C's work on content negotiation by profile (draft as on 19 Mar 2019) prescribing how HTTP user agents negotiate content with HTTP servers according to profiles.

The approach is to represent a schema as a profile in HTTP such that schemas can be negotiated as profiles over HTTP. W3C defines a profile as "a named set of constraints on one or more identified base specifications, including the identification of any implementing subclasses of datatypes, semantic interpretations, vocabularies, options and parameters of those base specifications necessary to accomplish a particular function".

3.1 Schema Identification

A SIF schema's identity needs to be concise and precise so that data artefacts can be produced from the right schema and processed unambiguously.

A schema can be plainly treated as an online resource like an XML namespace. A common naming convention for uniquely identifying such a resource utilizes URNs from RFC8141:

```
urn:{NID}:{NSS}
```

NID and NSS are the namespace identifier (case insensitive) and the namespace specific string (case sensitive) respectively. With regards to SIF, NID is "sif" whereas NSS is:

	NSS Syntax[2]	Examples

Data Model	`data/`{domain}/{version}{schematype}
	• {domain} is the domain for the data model, i.e.
	  ○ "`us`" for the USA data model
	  ○ "`uk`" for the UK data model
	  ○ "`au`" for the Australian data model
	  ○ "`nz`" for the New Zealand data model
	  ○ "`gcore`" for the gCore global data model (TBD)
	• {version} is the data model's version identifier
	• {schematype} is the payload's type. If unspecified (see notes below). {schematype} is interpreted as SIF's default which is XML schema.
		`data/au/3.4.4+pesc`
		`data/au/3.4.4`

Infrastructure	`inf/global`/{version}{schematype} • {version} is the infrastructure's version identifier {schematype} is the payload's type. If unspecified (see notes below). {schematype} is interpreted as SIF's default which is XML schema.	`inf/global/3.3+goessner` `inf/global/3.3`

Notes

1. "`:`" is not to be used in NSS to specify a structure or hierarchy to avoid confusion with the role of "`:`" in separating "`urn`" from NID, and NID from NSS.
2. Valid {schematype}s and their applicability in different domains and media types are listed below. They do not provide detailed versioning information for the schema types to reduce cluttering and complexity. For instance, JSON Schema has version draft7, draft6, draft5, etc. but its corresponding {schematype} is simply the plain "`+ietf`".

[2] *The format for NSS is pending the final definitions for profiles in https://w3c.github.io/dxwg/ucr/#ProfilesRequirements.*

schema type	Description	Applicable Media type	inf/global	data/gcore	data/us	data/uk	data/au	data/nz
[unspecified]	SIF default which is XML schema.	*/xml	Supported	Supported	Supported	Supported	Supported	Supported
`+pesc`	For JSON instances transformed from XML instances using PESC's convention.	*/json	Supported	Supported	Supported	Supported	Supported	Supported
`+goessner`	For JSON instances transformed from XML instances using Goessner's convention.	*/json	Supported	Supported	Supported		Supported	Supported

3. The following lines convey equivalent {schematype}s as per URN specifications RFC8141, even though this document prefers the first line:

```
urn:sif:data/au/3.4.4
urn:SIF:data/au/3.4.4
urn:Sif:data/au/3.4.4
```

RFC4229 defines two HTTP headers for the versioning of an HTTP content: `Content-Version` and `Derived-From`. Both are unsuitable for the versioning of the underlying schema to which the HTTP content conforms since they refer to the version of the HTTP content rather than the version of the schema to which the HTTP content conforms.

3.2 Schema Declarations

SIF consumers and providers can declare the schema(s) of the message body they send or receive from other parties using the following HTTP headers[3]:

[3] *RFC6648 recommends that a domain name should be incorporated into the prefix of related HTTP headers to avoid name clashes. Having said that, these headers would ideally begin with something like "`SIF-`". However, since existing SIF*

- `Content-Profile` (section 3.2.1)
- `Accept-Profile` (section 3.2.2)
- `Link` (section 3.2.3)
- Warning: 214 - "Transformation Applied" (section 3.2.4)

3.2.1 Content-Profile

`Content-Profile`[4] is a new HTTP header proposed by the HTTP profile negotiation specification. This specification adapts `Content-Profile` to declare the schema ID of the SIF object in an HTTP message body. The following table summarizes where this header can appear in SIF:

HTTP Method	HTTP Request	HTTP Response	Conveyed
GET		Optional	HTTP header
POST	Optional	Optional	HTTP header
PUT	Optional	Optional	HTTP header
HEAD		Mandatory	HTTP header

More precisely:

- A SIF consumer adds `Content-Profile` in an HTTP `POST` or `PUT` request to declare the schema ID for the message body it sends to a SIF provider.
- Reciprocally, a SIF provider adds `Content-Profile` in an HTTP `GET`, `POST`, or `PUT` response to declare the schema ID for the message body it returns to a SIF consumer.
- A SIF provider adds in an HTTP `HEAD` response to list the schema ID for the message body that it can return to a SIF consumer as if it is a normal HTTP `GET` response. There is actually no message body in an HTTP `HEAD` response (RFC7231). See section 3.2.4 for applying this header.

specific headers do not follow this recommendation, the HTTP headers proposed here will not include "`SIF-`". Future versions of the SIF infrastructure should revisit and reconsider this recommendation.

[4] The first draft of IETF's HTTP profile negotiation specification defined "`Profile`" instead of "`Content-Profile`". W3C's content negotiation by profile specification uses "`Content-Profile`".

When present, `Content-Profile` will declare one and only one schema ID as specified in section 3.1. `Content-Profile` must always be declared with `Content-Type` for backwards compatibility in case the receiving party does not support schema negotiation or the schema identified by `Content-Profile` does not always map to an exact media type. `Content-Profile`'s schema type indicator must also be compatible with `Content-Type` (e.g. "`+goessner`" for "`application/json`", unspecified for "`application/xml`" or "`text/xml`"[5]).

The following example HTTP request/response declares that the message body conforms to a particular SIF data model schema:

```
Content-Profile: urn:sif:data/au/3.4.4
Content-Type: application/xml
…
<StudentPersonal …>
…
</StudentPersonal>
```

For an infrastructure message body in JSON format, the headers are like:

```
Content-Profile: urn:sif:inf/global/3.3+goessner
Content-Type: application/json
…
```

If a SIF consumer does not support `Content-Profile` but a SIF provider does, the provider should return the content conforming to its default schema and declare the schema in both `Content-Profile` and `Content-Type` along with the content. On the other hand, if a SIF consumer sends a `Content-Profile` but a SIF provider does not understand its referenced schema in order to process the message body, the provider will return an error with `406` as the HTTP status code. As per the HTTP profile negotiation specification, the SIF provider will also return `Accept-Profile` (section 3.2.2) in the response to indicate its supported schema(s). A SIF consumer can avoid receiving `406` by first using the HTTP `HEAD` method to discover available schemas (section 3.2.4).

3.2.1.1 HTTP Status Codes

The HTTP status codes returned by a SIF provider for a request containing a problematic `Content-Profile` are described below.

HTTP Code	HTTP Message	Meaning
400	Bad Request	Media/schema type mismatch between `Content-Type` and `Content-Profile`.

[5] *Use of "text/xml" is discouraged because of encoding issues.*

406	Not Acceptable	Unknown or unsupported schema ID in Content-Profile.

3.2.2 Accept-Profile

Accept-Profile is another new HTTP header proposed by the HTTP profile negotiation specification. The following table shows the conditions under which this header can appear in the HTTP protocol:

HTTP Method	HTTP Request	HTTP Response	Conveyed
GET	Optional	Mandatory when status code=406	HTTP header
POST	Optional	Mandatory when status code=406	HTTP header
PUT	Optional	Mandatory when status code=406	HTTP header

It is used in these situations:

- A SIF consumer adds Accept-Profile in an HTTP GET, POST or PUT request to declare the schema ID(s) it accepts and prefers for the message body returned by a SIF provider.
- A SIF provider adds Accept-Profile in an HTTP response with status code 406 (Not Acceptable) to declare the schema ID(s) of a message body it accepts and/or prefers, after the provider received an HTTP GET, POST or PUT request with an unknown/unsupported schema ID in Content-Profile (section 3.2.1) that it could not understand and process.

A message body returned by a SIF provider may be potentially be data model related, infrastructure related, or both. Hence, data model and infrastructure schema IDs may both appear in Accept-Profile. The example below declares that a SIF consumer expects that the message body to be returned by a SIF provider will be either the 3.4.4 version of the data model, or the 3.3 version of the infrastructure.

```
Accept-Profile: urn:sif:data/au/3.4.4, urn:sif:inf/global/3.3
```

If a SIF consumer sends Accept-Profile to a SIF provider who does not understand it, the provider will ignore Accept-Profile and return the content as usual.

3.2.2.1 HTTP Status Codes

The HTTP status codes returned by a SIF provider for a request involving a problematic `Accept-Profile` are described below.

HTTP code	HTTP message	Meaning
400	Bad Request	Media/schema type mismatch between `Accept` and `Accept-Profile`.
406	Not Acceptable	Unknown or unsupported schema ID(s) in `Accept-Profile`.

3.2.2.2 Schema Preference

If a SIF party supports multiple schemas and wishes to express its preference for each of the schemas, it can do so using the *quality value* parameter from RFC7231. This parameter assigns a relative preference weight to each value in the associated context, with the default value being 1.0. For instance, the following `Accept-Profile` header declares the data model's version preference chain of 3.4.4 in JSON over 3.4.4 in XML over 3.4.3 in XML:

```
Accept-Profile: urn:sif:data/au/3.4.3; q=0.8
Accept-Profile: urn:sif:data/au/3.4.4; q=0.9
Accept-Profile: urn:sif:data/au/3.4.4+pesc
Accept-Profile: urn:sif:inf/global/3.3
```

An equivalent representation as per HTTP standard RFC7230 is:

```
Accept-Profile: urn:sif:data/au/3.4.3; q=0.8, urn:sif:data/au/3.4.4; q=0.9,
urn:sif:data/au/3.4.4+pesc, urn:sif:inf/global/3.3
```

Here is how a SIF provider expresses its schema type preference for JSON over XML:

```
Accept-Profile: urn:sif:data/au/3.4.4; q=0.9
Accept-Profile: urn:sif:data/au/3.4.4+pesc
Accept-Profile: urn:sif:inf/global/3.3; q=0.9
Accept-Profile: urn:sif:inf/global/3.3+goessner
```

3.2.2.3 Schema Precedence

When a schema ID (or its equivalence) is listed more than once, the instance that appears first takes precedence. In the following example, the data model schema urn:sif:data/au/3.4.4 is shown three times and the first entry takes precedence. The effective quality value for this schema is 0.8:


```
Accept-Profile: urn:sif:data/au/3.4.4; q=0.8
Accept-Profile: urn:sif:data/au/3.4.4+xml; q=0.9
Accept-Profile: urn:SIF:data/au/3.4.4; q=0.6
```

3.2.2.4 Schema Default

A SIF party may wish to indicate its default schema when receiving a payload from another party. Since IETF and W3C's profile negotiation approaches do not provide such an indicator, the schema with quality value q=1.0 (or undeclared) is regarded as the default for the same type (data model vs. infrastructure). The ID of the default schema indicated in the example below is `gcore`:

```
Accept-Profile: urn:sif:data/au/3.4.4+pesc; q=0.9
Accept-Profile: urn:sif:data/gcore/1.0.0+goessner; q=1.0
```

Note that the data model and infrastructure default can both be declared like so:

```
Accept-Profile: urn:sif:data/au/3.4.4+pesc; q=0.9
Accept-Profile: urn:sif:data/gcore/1.0.0+goessner
Accept-Profile: urn:sif:inf/global/3.3+goessner
```

If more than one schema of the same type having quality value of 1.0 appears in `Accept-Profile`, the receiving party will choose the first one as the default and ignore the rest.

3.2.2.5 Using Accept-Profile with Accept

When the SIF consumer declares the schema ID(s) it accepts through `Accept-Profile` in an HTTP request, it must also declare `Accept` for backwards compatibility in case the receiving provider does not understand `Accept-Profile`. In terms of SIF schema identification, `Accept-Profile` holds more information than `Accept`.

In the following example, a SIF consumer declares that its most preferred data model is `openapi3` for JSON. A less preferred data model is `gcore` with XML support. The quality value parameters (RFC7231) can be applied to `Accept` as desired.

```
Accept-Profile: urn:sif:data/gcore/1.0.0; q=0.8
Accept-Profile: urn:sif:data/nz/3.0.1+openapi2; q=0.9
Accept-Profile: urn:sif:data/nz/3.0.1+openapi3
Accept-Profile: urn:sif:inf/global/3.3; q=0.8
Accept-Profile: urn:sif:inf/global/3.3+goessner
Accept: application/json, application/xml; q=0.9
```

When a provider is required to return `Accept-Profile` in an HTTP response, it can derive `Accept` by:

- listing all possible schema IDs in `Accept-Profile`

- making a best guess at what it prefers to receive from a SIF consumer, if it is not able to negotiate schema

3.2.3 Link

`Link` is standard HTTP header from RFC8288 to specify relationships in web content. W3C's content negotiation specification adopts `Link` to describe variations of the same web content available for negotiation. In its specification, a link is an identifier to a web content conforming to a profile and each variation of the web content (i.e. for a particular profile) has a unique link. In the same way, this specification makes use of the `Link` header to identify SIF schemas in HTTP:

- A SIF provider adds `Link` in an HTTP `GET` response to declare all the schema ID(s) of the message it can produce in a response.
- A SIF provider adds `Link` in an HTTP `LINK` response to declare all the schema ID(s) of the message it accepts in a request and produces in a response.

Here is a summary:

HTTP Method	HTTP Request	HTTP Response	Conveyed
GET		Optional	HTTP header
HEAD		Mandatory	HTTP header

In SIF, a link is an object service supporting a particular schema ID. Each link is specified with respect to the *context* object service which is found at the request URL plus `Content-Type` and `Content-Profile` (section 3.2.1) in an HTTP response. The syntax for expressing a link is:

```
Link: <{target URL}>; rel="{relationship}"; type="{media type}"; profile="{schema ID}"
```

where

- {target URL} is the URL for a SIF object service and can be absolute or relative. The party receiving a relative URL must resolve it to an absolute URL.
- {relationship} is either `self`, designating that the link is the context object service itself, or `alternate`, designating it as an alternative to the context object service.
- {schema ID} is one of those specified in section 3.1

A link or more is added to the response to relate object services to the context. In the following example, there are two links to the context object service, one providing

SchoolInfo XML objects conforming to the schema `urn:sif:data/au/3.4.4`. and the other to the JSON schema using the PESC convention:

```
Content-Type: application/xml
Content-Profile: urn:sif: data/au/3.4.4
Link: <../SchoolInfos>; rel="self"; type="application/xml"; profile="urn:sif:data/au/3.4.4"
Link: <../SchoolInfos>; rel=" alternate"; type="application/json";
profile="urn:sif:data/au/3.4.4+pesc"
```

As per RFC7230, multi-line `Link` headers can be collapsed into one line:

```
Content-Type: application/xml
Content-Profile: urn:sif: data/au/3.4.4
Link: <../SchoolInfos>; rel="self"; type="application/xml"; profile="urn:sif:data/au/3.4.4",
<../SchoolInfos>; rel="alternate"; type="application/json"; profile="urn:sif:data/au/3.4.4+pesc"
```

An advantage of using the `Link` header is that it can clearly state all the correct combinations of the SIF schemas and media types in its expressions. This is not possible to achieve using two separate headers `Accept` and `Accept-Profile`. A drawback of `Link` is there are two classes only: `self` and `alternate`. The order of preference for the profiles cannot be precisely declared like those using the quality value parameter in `Accept-Profile`.

3.2.4 Warning: 214 - "Transformation Applied"

Most of the time a SIF provider will honor the schema preference specified by a SIF consumer as an atomic operation on response. In certain situations, however, the SIF provider will instead choose to transfer the message body it generates, from a schema that it supports natively, to the scheme that best matches the preference requested by a SIF consumer. In this case, the SIF provider should include the following header from RFC7234 in its response:

```
Warning: 214 - "Transformation Applied"
```

HTTP Method	HTTP Request	HTTP Response	Conveyed
GET		Optional	HTTP header

When a transformation takes place, attributes from the original object may be dropped or attributes with “mocked” values may need to be added to the object such that the transformed object conforms to the desired schema.

3.3 SIF Schema Discovery

A SIF consumer can occasionally interrogate the schemas that a SIF provider supports by sending an HTTP `HEAD` request[6] that includes the `Accept` header to its object service of interest. The consumer should not specify `Accept-Profile` so that the provider will return the full range of schemas that it handles. It is also because when `Accept-Profile` is present, the SIF provider will attempt to return a schema ID that best matches `Accept` and `Accept-Profile` in `Content-Profile`.

```
HEAD /connectorPath/{ObjectType}s HTTP/1.1
Accept: application/xml
```

Note that the request cannot be normally specified at the object level like the example below because the SIF provider could return a 4xx status code (e.g. 404 Not Found) such that the consumer will not be able to retrieve the list of profiles. An exception is Functional Services will be illustrated in section 4.1:

```
HEAD /connectorPath/StudentPersonals/116b473f-8eb6-4542-8d8c-4535a9f7e634 HTTP/1.1
Accept: application/xml
```

Since the SIF broker may delegate each object type to a separate SIF provider, the interrogation results cannot be sourced from one provider only. The following headers are illegal; the broker or the provider in a direct environment must return 405 (Method Not Allowed):

```
HEAD * HTTP/1.1
```

```
HEAD /connectorPath HTTP/1.1
```

When a SIF provider supports schema negotiation and receives an HTTP `HEAD` request that it can process successfully, it must declare an HTTP `Link` header (section 3.2.3) containing information about its default (`ref="self"`) and alternative schemas (`ref="alternate"`) for an object service. These schemas are applicable to both the kinds of message bodies it can produce in a response and the kinds of message bodies it can accept in a request. For a data object service, `rel="self"` will always point to a data model schema and the infrastructure schema will appear in one of the `rel="alternate"` links:

```
Content-Type: application/xml
Content-Profile: urn:sif:data/au/3.4.4
Link: <../StaffPersonals>; rel="self";  type="application/xml";  profile="urn:sif:data/au/3.4.4",
  <../StaffPersonals>; rel="alternate"; type="application/xml";  profile="urn:sif:data/au/3.4.3",
  <../StaffPersonals>; rel="alternate"; type="application/xml";  profile="urn:sif:inf/global/3.3",
  <../StaffPersonals>; rel="alternate"; type="application/json"; profile="urn:sif:data/au/3.4.4+pesc",
  <../StaffPersonals>; rel="alternate"; type="application/json"; profile="urn:sif:inf/global/3.3+goessner"
```

[6] By definition, an HTTP `HEAD` response is like an HTTP `GET` one but without a message body. W3C's content negotiation using profile considers that the profiles returned in an HTTP `HEAD` response also apply to HTTP `POST`, HTTP `PUT`, and HTTP `DELETE` methods.

For an infrastructure object service, `rel="self"` will point to the infrastructure schema:

```
Content-Type: application/xml
Content-Profile: urn:sif:inf/global/3.3
Link: <../MyFunctServices>; rel="self";  type="application/xml";  profile="urn:sif:inf/global/3.3",
  <../MyFunctServices>; rel="alternate"; type="application/json"; profile="urn:sif:inf/global/3.3+goessner"
```

Apart from the schemas responding to the media type specified in the request (i.e. XML), the provider also announces the JSON-aware schemas in the above example. This information is useful for advanced SIF consumers.

The HTTP `HEAD` method can reduce the chattiness between SIF consumer and provider. For instance, once a SIF consumer knows the schemas (both types and versions) that a provider supports from the `Link header`, it can narrow the range of schemas declared in subsequent requests sent to the provider. Unlike the HTTP `OPTIONS` method, responses for the HTTP `HEAD` methods are cacheable. SIF consumers might utilize several HTTP caching features for these responses to improve performance. However, they should re-interrogate the SIF provider occasionally, in case the SIF provider changes its range of supported schemas dynamically.

3.4 Assumptions

SIF providers and consumers will store objects received from one another, along with the schema specified for the objects. This information will be useful when the objects are passed on to other parties.

SIF providers and consumers will always use the highest possible (i.e. most recent) version of the schema as the most strongly preferred one and this should be reflected in their expressed preference in `Accept-Profile`.

For simplicity, multiple objects in a message body must conform to the same schema.

4. Results

This section highlights how the extensions proposed in section 0 can be used to facilitate schema negotiation to a certain extent. More specifically, the objective is to support SIF consumers and SIF providers in negotiating and agreeing on a schema through extensions, by matching the preferences and capabilities of the SIF consumers and the availability of data on the SIF providers.

The approach proposed here, using schema negotiation, should not be regarded as the only way of handling multi-versioning in SIF.

4.1 Discovering Supported Schemas

There are several uses of the HTTP HEAD method in discovering the schemas supported by a SIF provider.

A SIF consumer detects that a SIF provider handles data models 3.4.4 and 3.4.3 with 3.4.4 being the default.

HTTP Request	HEAD /requestConnectorPath/MyObjects HTTP/1.1 Accept: application/xml …
HTTP Response	HTTP/1.1 200 OK Content-Type: application/xml Content-Profile: urn:sif:data/au/3.4.4 Link: <../MyObjects>;rel="self";type="application/xml";profile="urn:sif:data/au/3.4.4", <../MyObjects>;rel="alternate";type="application/xml";profile="urn:sif:data/au/3.4.3", <../MyObjects>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3" Content-Length: 0 …

The SIF consumer is also able to discover the schemas for a functional service that , which is 3.3.

HTTP Request	HEAD /serviceConnectorPath/MyFunctionalServices HTTP/1.1 Accept: application/xml …
HTTP Response	HTTP/1.1 200 OK Content-Type: application/xml Content-Profile: urn:sif:inf/global/3.3 Link: <../MyObjects>;rel="self";type="application/xml";profile="urn:sif:inf/global/3.3" Content-Length: 0 …

Here is a way for a SIF consumer to retrieve the ID of the schema for the phase of a functional service that the provider understands and provides. Note that the SIF RefId of the service must appear in the request URL:

HTTP Request	`HEAD /serviceConnectorPath/MyFunctionalServices/9d02787e-6f6f-466f-8fcc-f7cbb47a28d1/phaseOne HTTP/1.1`
	`Accept: application/json`
	…

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/json`
	`Content-Profile: urn:sif:inf/global/3.3+goessner`
	`Link: <../MyObjects>; rel="self"; type="application/json"; profile="urn:sif:inf/global/3.3+goessner"`
	`Content-Length: 0`
	…

As another example, a SIF consumer detects that a SIF provider does not support HTTP `HEAD` at all, implying that the SIF provider does not handle schema negotiation.

HTTP Request	`HEAD /requestConnectorPath/MyObjectTypes HTTP/1.1`
	`Accept: application/json`
	…

HTTP Response	`HTTP/1.1 405 Method Not Allowed`
	`Content-Length: 0`
	…

If the SIF provider does support HTTP `HEAD` in general, but not schema negotiation, it will not present `Link` or `Accept-Profile` header in the response.

HTTP Request	`HEAD /requestConnectorPath/MyObjectTypes HTTP/1.1`
	`Accept: application/json`
	…

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/json`
	…

Assume that a provider supports both XML and JSON schemas. When a XML based consumer asks for its supported schemas; the provider shows a full list in the `Link` header:

HTTP Request	`HEAD /requestConnectorPath/MyObjects HTTP/1.1`
	`Accept: application/xml`
	…

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/xml`
	`Content-Profile: urn:sif:data/au/3.4.4`
	`Link: <../MyObjects>;rel="self";type="application/xml";profile="urn:sif:data/au/3.4.4",`
	`<../MyObjects>;rel="alternate";type="application/json";profile="urn:sif:data/au/3.4.4+pesc",`
	`<../MyObjects>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3",`
	`<../MyObjects>;rel="alternate";type="application/json";profile="urn:sif:inf/global/3.3+goessner"`
	`Content-Length: 0`
	…

For a JSON based consumer's query, the provider shows a slightly different list. The returned `Content-Type` and the `ref="self"` link now point to the JSON schema:

HTTP Request	`HEAD /requestConnectorPath/MyObjects HTTP/1.1`
	`Accept: application/json`
	…

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/json`
	`Content-Profile: urn:sif:data/au/3.4.4+pesc`
	`Link:`
	`<../MyObjects>;rel="self";type="application/json";profile="urn:sif:data/au/3.4.4+pesc",`
	`<../MyObjects>;rel="alternate";type="application/xml";profile="urn:sif:data/au/3.4.4",`
	`<../MyObjects>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3",`
	`<../MyObjects>;rel="alternate";type="application/json";profile="urn:sif:inf/global/3.3+goessner"`
	`Content-Length: 0`
	…

A SIF consumer suspects that a provider supports JSON schemas but is not sure about which ones. It sends an HTTP HEAD request to the provider to find that out.

HTTP Request	`HEAD /requestConnectorPath/MyObjects HTTP/1.1`
	`Accept: application/json`
	…

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/json`
	`Content-Profile: urn:sif:data/nz/3.0.1+ietf`
	`Link:`
	`<../MyObjects>;rel="self";type="application/json";profile="urn:sif:data/nz/3.0.1+ietf",`
	`<../MyObjects>;rel="alternate";type="application/json";profile="urn:sif:data/nz/3.0.1+pesc",`
	`<../MyObjects>;rel="alternate";type="application/json";profile="urn:sif:data/nz/3.0.1+openapi3",`
	`<../MyObjects>;rel="alternate";type="application/xml";profile="urn:sif:data/nz/3.0.1",`
	`<../MyObjects>;rel="alternate";type="application/json";profile="urn:sif:inf/global/3.3+goessner"`
	`Content-Length: 0`
	…

4.2 Get Single and Multiple Objects

There are many variants of the HTTP `GET` method that can utilize schema negotiation. If a SIF consumer does not handle schema negotiation but a SIF provider does, the SIF provider must support backwards compatibility by returning `Content-Type` and conform to W3C's content negotiation by profile specification by returning a `Link` header. It should also return `Content-Profile` as recommended by IETF's HTTP profile negotiation specification.

HTTP Request	`GET /requestConnectorPath/ObjectTypes HTTP/1.1`
	`Accept: application/xml`
	…

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/xml`
	`Content-Profile: urn:sif:data/au/3.4.4`
	`Link:`
	`<../ObjectTypes>;rel="self";type="application/xml";profile="urn:sif:data/au/3.4.4",`
	`<../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3"`
	…

Note that the infrastructure schema is present in the `Link` header for completeness.

If a SIF consumer supports schema negotiation, it will declare at least one data model and one infrastructure schema that it can handle using `Accept-Profile`. The inclusion of an infrastructure schema is because the returned message body may need to describe an error in which case the

infrastructure schema will be used to produce the error message body. In the following example, a SIF provider simply uses the same schema as a SIF consumer and hence the response is straight forward.

HTTP Request	`GET /requestConnectorPath/ObjectTypes HTTP/1.1`
	`Accept: application/xml`
	`Accept-Profile: urn:sif:data/au/3.4.4, urn:sif:inf/global/3.3`
	`…`

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/xml`
	`Content-Profile: urn:sif:data/au/3.4.4`
	`Link:`
	`<../ObjectTypes>;rel="self";type="application/xml";profile="urn:sif:data/au/3.4.4",`
	`<../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3"`
	`…`

If the SIF consumer queries a data object service and specifies an infrastructure schema for the message body to be returned using `Accept-Profile` like the following example, the SIF provider will return an error 400 (Bad Request):

```
GET /requestConnectorPath/ObjectTypes HTTP/1.1
Accept: application/xml
Accept-Profile: urn:sif:inf/global/3.3"
…
```

The following SIF consumer can handle multiple data model versions and types. It prefers version 3.4.4 over 3.4.3, and the JSON over XML format. The SIF provider has its data conforming to 3.4.4 schema and honors the preference expressed by the consumer:

HTTP Request	`GET /requestConnectorPath/ObjectTypes HTTP/1.1`
	`Accept: application/xml; q=0.9, application/json`
	`Accept-Profile: urn:sif:data/au/3.4.3; q=0.9, urn:sif:data/au/3.4.4+pesc,`
	`urn:sif:inf/global/3.3; q=0.9, urn:sif:inf/global/3.3+goessner`
	`…`

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/json`
	`Content-Profile: urn:sif:data/au/3.4.4+pesc`
	`Link:`
	`<../ObjectTypes>;rel="self";type="application/json";profile="urn:sif:data/au/3.4.4",`
	`<../ObjectTypes>;rel="alternate";type="application/json";profile="urn:sif:inf/global/3.3+goessner"`
	`…`

The following SIF consumer supports schema negotiation and prefers data model 3.4.3 over 3.4.2. The SIF provider stores its data in 3.4.2 and wishes to honor the preference by *upgrading* its returning data to 3.4.3 through an upward transformation. Note the `Warning` header being returned to the SIF consumer.

HTTP Request	`GET /requestConnectorPath/ObjectTypes HTTP/1.1`
	`Accept: application/xml`
	`Accept-Profile: urn:sif:data/au/3.4.3, urn:sif:data/au/3.4.2; q=0.9,`
	`urn:sif:inf/global/3.3`
	`…`

HTTP Response	`HTTP/1.1 200 OK`
	`Content-Type: application/xml`
	`Content-Profile: urn:sif:data/au/3.4.3`
	`Warning: 214 - "Transformation Applied"`

	Link: <../ObjectTypes>;rel="self";type="application/xml";profile="urn:sif:data/au/3.4.3", <../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:data/au/3.4.2" , <../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3+goessner" …

The following SIF consumer supports schema negotiation. The SIF provider keeps its data in 3.4.2 and honors the preference by "downgrading" its returning data to 3.4.1 through a downward transformation. Note also the `Warning` header.

HTTP Request	GET /requestConnectorPath/ObjectTypes HTTP/1.1 Accept: application/xml Accept-Profile: urn:sif:data/au/3.4.1, urn:sif:inf/global/3.3 …
HTTP Response	HTTP/1.1 200 OK Content-Type: application/xml Content-Profile: urn:sif:data/au/3.4.1 Warning: 214 - "Transformation Applied" Link: <../ObjectTypes>;rel="self";type="application/xml";profile="urn:sif:data/au/3.4.1", <../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:data/au/3.4.2" , <../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3+goessner" …

A SIF consumer supports data model 3.4.3 whereas a SIF provider cannot, resulting in an error condition 406. The SIF provider also returns its supported schemas in the `Link` header:

HTTP Request	GET /requestConnectorPath/ObjectTypes HTTP/1.1 Accept: application/xml Accept-Profile: urn:sif:data/au/3.4.3, urn:sif:inf/global/3.3 …
HTTP Response	HTTP/1.1 406 Not Acceptable Link: <../ObjectTypes>;rel="alternate";type="application/json";profile="urn:sif:data/au/3.4.4+pesc", <../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:data/au/3.4.4" , <../ObjectTypes>;rel="alternate";type="application/json";profile="urn:sif:inf/global/3.3+goessner" <../ObjectTypes>;rel="alternate";type="application/xml";profile="urn:sif:inf/global/3.3" Content-Length: 0 …

4.3 Create Single Objects

Creating a single object is straightforward since the exact schema is present in `Content-Profile` and the only schema expected for a successful creation should be the same schema!

HTTP Request	POST /requestConnectorPath/ObjectTypes/ObjectType HTTP/1.1 Content-Type: application/xml Content-Profile: urn:sif:data/us/3.5 Accept: application/xml Accept-Profile: urn:sif:data/us/3.5, urn:sif:inf/global/3.3 …
HTTP Response	HTTP/1.1 201 Created Content-Type: application/xml Content-Profile: urn:sif:data/us/3.5 …

4.4 Update Single Objects

Updating a single object is simple. `Content-Profile` is always the schema for the object's data model and `Accept-Profile` refers to that of the infrastructure schema. If there are no errors, the message body of the response is empty:

HTTP Request	`PUT /requestConnectorPath/ObjectTypes/uid HTTP/1.1`
	`Content-Type: application/xml`
	`Content-Profile: urn:sif:data/au/3.4.4`
	`Accept: application/xml`
	`Accept-Profile: urn:sif:inf/global/3.3`
	`…`

HTTP Response	`HTTP/1.1 204 OK` `…`

Otherwise, it will contain an error object produced with the infrastructure schema, for instance:

HTTP Response	`HTTP/1.1 404 Not Found`
	`Content-Type: application/xml`
	`Content-Profile: urn:sif:inf/global/3.3`
	`…`

4.5 Delete Single Objects

An HTTP `DELETE` request does not have a message body. So the only new HTTP header is `Accept-Profile` (i.e. `Content-Profile` is absent):

HTTP Request	`PUT /requestConnectorPath/ObjectTypes/uid HTTP/1.1`
	`Accept: application/json`
	`Accept-Profile: urn:sif:inf/global/3.3+goessner`
	`…`

HTTP Response	`HTTP/1.1 204 OK` `…`

Otherwise, it will contain an error object produced with the infrastructure schema, for instance:

HTTP Response	`HTTP/1.1 404 Not Found`
	`Content-Type: application/json`
	`Content-Profile: urn:sif:inf/global/3.3+goessner`
	`…`

4.6 Create/Update Multiple Objects

In an HTTP `POST` or `PUT` requests for creating or updating existing objects, the message body consists of the objects to be actioned. The response is always either `createResponse`,

`updateResponse` or a generic error object. So, the `Content-Profile` and `Accept-Profile` values are straightforward. This consumer supports both XML and JSON equally; the provider can return one of the two responses at will:

HTTP Request	POST (or PUT) /requestConnectorPath/ObjectTypes HTTP/1.1 Content-Type: application/xml Content-Profile: urn:sif:data/au/3.4.4 Accept: application/xml, application/json Accept-Profile: urn:sif:inf/global/3.3, urn:sif:inf/global/3.3+goessner …
HTTP Response 1	HTTP/1.1 200 OK Content-Type: application/xml Content-Profile: urn:sif:inf/global/3.3 …
HTTP Response 2	HTTP/1.1 200 OK Content-Type: application/json Content-Profile: urn:sif:inf/global/3.3+goessner …

4.7 Delete Multiple Objects

To request a SIF provider to delete multiple objects, a SIF consumer submits a `deleteRequest` object for which a `deleteResponse` object will be returned. Both are sourced from the infrastructure schema which is hence referred by both `Content-Profile` and `Accept-Profile`.

HTTP Request	PUT /requestConnectorPath/ObjectTypes HTTP/1.1 methodOverride: DELETE Content-Type: application/xml Content-Profile: urn:sif:inf/global/3.3 Accept: application/xml Accept-Profile: urn:sif:inf/global/3.3 …
HTTP Response	HTTP/1.1 200 OK Content-Type: application/xml Content-Profile: urn:sif:inf/global/3.3 …

5. Caveats

5.1 SIF Standards

A SIF consumer is issued a dedicated SIF environment in which the services available to the consumer are listed. Among other things, a SIF consumer can discover the preset coarse-grained schemas of the infrastructure and the data model from the environment. It might be possible to incorporate the fine-grained schema IDs in the SIF consumer expects in the environment. One drawback of this approach is that the SIF environment will need to be updateable and maintained by a SIF consumer at runtime which might not be possible for security seasons particularly in a SIF brokered platform.

A SIF Functional Service provides a means of collaborating phased execution of a stateful process (i.e. a job) between parties (consumers and providers). Since each phase can involve data exchange, the data model and infrastructure schemas used in all the job phases are not guaranteed to be the same. It will thus be desirable for the connector that mounts the Functional Service to also support schema negotiation. A SIF consumer involved in a Functional Service will for example be able to retrieve the infrastructure schema for a particular phase with HTTP `HEAD` as proposed in this document. For simplicity, however, the number of schemas used in a Functional Service should be kept to minimal.

SIF events allow a SIF provider to communicate change notifications to SIF objects to multiple consumers. The modification to SIF events to support schema negotiation is out of scope. Nonetheless, it is logical to annotate each SIF event with a schema ID to enrich the information about its content.

5.2 HTTP Standards

In HTTP, certain HTTP responses can be cached to improve performance (RFC7234). In light of the work proposed here, the HTTP header `Cache-Control` must be used with caution when schema negotiation is also required. For instance, a SIF consumer can obtain a cached list of the schema IDs supported by a SIF provider by way of specifying `Cache-Control` appropriately in an HTTP `HEAD` request (where should the schema IDs be cached, how long will they be kept in the cache, etc.).

As per RFC7231, a SIF party should ignore HTTP headers that it does not understand. This allows SIF schema-aware consumers and providers to stay compatible to those that are not schema-aware and vice versa.

There are other features in the HTTP standards that support HTTTP content negotiation in some ways. One such example is redirection (RFC7231) which is limited to HTTP GET and `HEAD` queries. A server can present HTTP status code `300` (multiple choices, RFC7231) with a list of available resources and locations from which a user agent (e.g. a SIF consumer) can choose the most suitable. This approach cannot accommodate all the use cases identified in this proposal (which are `POST`, `PUT`, `DELETE` operations). Another way to mimic schema negotiation is to declare parameters in an HTTP request's query string. The parameters will be restricted by the limitations for query strings, e.g. maximum length, encoding, HTTP methods.

IETF has proposed new media types for JSON documents, namely `application/schema+json` and `application/schema-instance+json`. These are most likely applicable to HTTP headers `Accept` and `Content-Type`. They are not immediately useful for schema negotiation since the schema information cannot be declared using these media types alone.

6. References

Berners-Lee Fielding & Masinter (2005), “Uniform Resource Identifier (URI): Generic Syntax”, Network Working Group, https://tools.ietf.org/html/rfc3986

Fielding & Reschke (2014), “Hypertext Transfer Protocol -- HTTP/1.1: Message Syntax and Routing”, Internet Engineering Task Force (IETF), https://tools.ietf.org/html/rfc7230

Fielding & Reschke (2014), “Hypertext Transfer Protocol -- HTTP/1.1: Semantics and Content”, Internet Engineering Task Force (IETF), https://tools.ietf.org/html/rfc7231

Fielding, Gettys, Mogul, Frystyk, Masinter, Leach & Berners-Lee (1999), “Hypertext Transfer Protocol -- HTTP/1.1”, Network Working Group, https://tools.ietf.org/html/rfc2616

Fielding, Nottingham & Reschke (2014), “Hypertext Transfer Protocol -- HTTP/1.1: Caching”, Internet Engineering Task Force (IETF), https://tools.ietf.org/html/rfc7234

Goessner (2006), “Converting Between XML and JSON”, O'Reilly Media, Inc., https://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html

Holtman & Mutz (1998), “Transparent Content Negotiation in HTTP”, Network Working Group, https://tools.ietf.org/html/rfc2295

Mozilla (2018), “Content Negotiation”, MDN web docs, https://developer.mozilla.org/en-US/docs/Web/HTTP/Content_negotiation

Nottingham & Mogul (2005), “HTTP Header Field Registrations”, Network Working Group, https://tools.ietf.org/html/rfc4229

Nottingham (2017), “Web Linking”, Internet Engineering Task Force, https://tools.ietf.org/html/rfc8288

Postsecondary Electronic Standards Council (2018), “PESC Compliant JSON - Version 1.0.0”, for the latest, see http://www.pesc.org/standards-development-1.html

Saint-Andre, Crocker & Nottingham (2012), ‘Deprecating the "X-" Prefix and Similar Constructs in Application Protocols’, Internet Engineering Task Force (IETF), https://tools.ietf.org/html/rfc6648

SmartBear Software (2019), “OpenAPI Specification Version 3.0.2”, https://swagger.io/specification/

SmartBear Software (2019), “OpenAPI Specification Version 2.0”, https://swagger.io/specification/v2/

Stain-Andre & Klensin (2017), “Uniform Resource Names (URNs)”, Internet Engineering Task Force (IETF), https://tools.ietf.org/html/rfc8141

Svensson & Verborgh (2018), “Negotiating Profiles in HTTP”, Internet Engineering Task Force (IETF), https://profilenegotiation.github.io/I-D-Accept--Schema/I-D-Accept-Profile

Svensson (2016), “An http Header for Metadata Schema Negotiation”, Deutsche National Bibliothek, https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_14

W3C Dataset Exchange Working Group (2019), “Content Negotiation by Profile”, www.w3.org/TR/dx-prof-conneg/

W3C XML Schema Working Group (2006), “XML Schema Versioning Use Cases”, http://www.w3.org/XML/2005/xsd-versioning-use-cases/

Wright & Andrews (2018), "JSON Schema: A Media Type for Describing JSON Documents", Internet Engineering Task Force (IETF), https://tools.ietf.org/html/draft-handrews-json-schema-01

