Simple Triples Discovery Mechanism

16 March 2018

> *Editorial note* — This is an **first public draft** of a standard defining a simple, general-purpose discovery mechanism. This document is not endorsed by the FHISO membership, and may be updated, replaced or obsoleted by other documents at any time.
>
> The public tsc-public@fhiso.org mailing list is the preferred place for comments, discussion and other feedback on this draft.
>
> | | |
> |---|---|
> | Latest public version: | `https://fhiso.org/TR/triples-discovery` |
> | This version: | `https://fhiso.org/TR/triples-discovery-20180316` |

FHISO's **Simple Triples Discovery Mechanism** (or **Triples Discovery**) provides a way for internet-connected applications to attempt to gain information on any unfamiliar *terms* they may encounter, allow these *terms* to be better processed. Unknown *terms* can appear in data as a result of third-party extensions being used, when data conforming to a new standard is read by an older application, or if data conforming to other standards is present.

In *Triples Discovery*, an application makes an HTTP request to the *term name* IRI with an `Accept` header requesting a response in the N-Triples format. The details of these HTTP requests and their responses are given in §2. The N-Triples format is described in §3; it is extremely simple to parse, and is supported in various libraries by virtue of being a small subset of the more popular Turtle serialisation format.

1 Conventions used

Where this standard gives a specific technical meaning to a word or phrase, that word or phrase is formatted in bold text in its initial definition, and in italics when used elsewhere. The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **NOT RECOMMENDED**, **MAY** and **OPTIONAL** in this standard are to be interpreted as described in [RFC 2119].

An application is **conformant** with this standard if and only if it obeys all the requirements and prohibitions contained in this document, as indicated by use of the words MUST, MUST NOT, REQUIRED, SHALL and SHALL NOT, and the relevant parts of its normative references. Standards referencing this standard MUST NOT loosen any of the requirements and prohibitions made by this standard, nor place additional requirements or prohibitions on the constructs defined herein.

> *Note* — Derived standards are not allowed to add or remove requirements or prohibitions on the facilities defined herein so as to preserve interoperability between applications. Data

> generated by one *conformant* application must always be acceptable to another *conformant* application, regardless of what additional standards each may conform to.

If a *conformant* application encounters data that does not conform to this standard, it MAY issue a warning or error message, and MAY terminate processing of the document or data fragment.

This standard depends on FHISO's **Basic Concepts for Genealogical Standards** standard. To be *conformant* with this standard, an application MUST also be *conformant* with [Basic Concepts]. Concepts defined in that standard are used here without further definition.

> *Note* — In particular, precise definitions of *character*, *string*, *whitespace*, *language-tagged string*, *term*, *term name*, *discovery*, *namespace*, *namespace name*, *prefix notation*, *prefix*, *class*, *class name*, *type*, *property*, *property name*, *property value*, *property term*, *range*, *required property*, *datatype*, *datatype name*, *subtype*, *abstract datatype* and *language-tagged datatype* are given in [Basic Concepts].

Indented text in grey or coloured boxes does not form a normative part of this standard, and is labelled as either an example or a note.

> *Editorial note* — Editorial notes, such as this, are used to record outstanding issues, or points where there is not yet consensus; they will be resolved and removed for the final standard. Examples and notes will be retained in the standard.

In some of the examples in this standard, long lines are broken across multiple lines to improve readability. Where this has occurred, the continuation lines are prefixed with a "↪" to mark the continuation. To get the actual text, this character needs to be removed, and the continuation line appended to the previous line with a single space *character* (U+0020) separating the previous line's content from the continuation line's.

The grammar given here uses the form of EBNF notation defined in §6 of [XML], except that no significance is attached to the capitalisation of grammar symbols. *Conforming* applications MUST NOT generate data not conforming to the syntax given here, but non-conforming syntax MAY be accepted and processed by a *conforming* application in an implementation-defined manner.

This standard uses *prefix notation* when discussing specific *terms*. The following *prefix* bindings are assumed in this standard:

`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`
`xsd`	`http://www.w3.org/2001/XMLSchema#`
`types`	`https://terms.fhiso.org/types/`
`cev`	`https://terms.fhiso.org/sources/`

> *Note* — The particular *prefix* assigned above have no relevance outside this standard document as *prefix notation* is not used in the formal data model defined by this standard. This

> notation is simply a notational convenience to make the standard easier to read. The `cev` prefix is only used in examples.

2 HTTP requests and responses

Discovery is defined in §4.1 of [Basic Concepts] as being when an HTTP request to the *term name* IRI, made with an appropriate `Accept` header, results in a particular machine-readable format. This section defines how those HTTP requests and responses are made in *Triples Discovery*.

When an application opts to carry out *discovery* using this *Triples Discovery* mechanism on a *term* whose *term name* IRI has an `http` or `https` scheme, it SHALL make an HTTP `GET` request to the URI that results from the conversion of the *term name* IRI to a URI per §4.1 of [RFC 3987].

> *Note* — This standard does not specify how *Triples Discovery* works with *terms* using other IRI schemes, and the use of other schemes is NOT RECOMMENDED by §4 of [Basic Concepts].

The IRI to which the initial `GET` request is made is called the **discovery IRI**. It is the *term name* IRI with any fragment component removed.

> *Example* — The *term name* `https://example.com/events#Birth` contains a fragment component, therefore its *discovery IRI* is `https://example.com/events`.

The `GET` request SHOULD have an `Accept` header that is well-formed according to §5.3.2 of [RFC 7231], and which references the N-Triples media type, "`application/n-triples`". The request's `Accept` header MAY alternatively or additionally reference the media types of one of the alternative RDF formats described in §3.3 of this standard, but *conformant* servers need not support those formats.

If the *discovery IRI* is not a known *term name* and is not an IRI used for another purpose, it is RECOMMENDED that servers issue a 404 "Not Found" response. Applications MUST NOT consider a 404 or other HTTP error response to mean the *term* is invalid.

> *Note* — As it is only RECOMMENDED and not REQUIRED that parties defining new *terms* make information available online at the *term name* IRI, a 404 response can also mean the provider has chosen not to follow this recommendation. This might occur when a *term* is no longer supported by the organisation which originally defined it, but is still in use.

After any initial redirections, a *conformant* server SHOULD use the algorithm in §5.3 of [RFC 7231] to consider each of those media types listed in the `Accept` header which the server supports, including any documentation formats or other *discovery* formats outside the scope of this standard, to select the media type of the resource that will be served. If the server supports none of the listed media types, it SHOULD send a 406 "Not Acceptable" response; otherwise, if the selected media type is the N-Triples media type or a supported alternative type from §3.3, the server SHOULD continue with *Triples Discovery* as outlined here. If the `Accept` header was precisely "`application/n-triples`" then a *conformant* server MUST continue with *Triples Discovery*.

> *Note* — It is only RECOMMENDED and not REQUIRED that parties defining new *terms* arrange for HTTP content negotiation to be performed properly as described above and in [RFC 7231]. The reason for this is that some popular web servers do not make necessary configuration straightforward, and much of the published advice on the subject is to use basic pattern matching on `Accept` headers rather than proper content negotiation. An prominent example of such advice is [SWBP Vocab Pub], published as best practice by the World Wide Web Consortium. Recipes 3 and 4 from this can result in certain complex `Accept` headers being parsed contrary to [RFC 7231]; nevertheless, this standard allows server administrators to follow [SWBP Vocab Pub] while remaining *conformant* with this standard.

> *Example* — An application might send a `GET` request with the following `Accept` header:
>
> ```
> Accept: application/x-discovery; q=0.9, application/n-triples
> ```
>
> This is well-formed according to §5.3.2 of [RFC 7231]. The `q=0.9` in the `Accept` header is a quality value attached to the preceding media type. It indicates that the hypothetical `x-discovery` format is less preferred than N-Triples which by default has a quality value of 1.0. Placing a less preferred format before the preferred format is unorthodox but not prohibited.
>
> A server that supports both the `x-discovery` format and N-Triples SHOULD provide an N-Triples description of the *term* per this standard, but because the `Accept` header was not exactly "`application/n-triples`", this is not REQUIRED. If the server uses some form of pattern matching on the `Accept` header and concludes that `x-discovery` must be preferred as it is listed first, this behaviour, while incorrect, is still *conformant* with this standard.

Except when the *discovery IRI* is a *namespace name* as defined in §4.2 of [Basic Concepts], a *conformant* server SHALL issue an HTTP redirect response with a `Location` header containing the URL of a resource containing a description of the *discovery term* in the selected format including the *required triples* given in §4. This redirect SHOULD use a 303 "See Other" redirect, and MUST NOT be a permanent redirect such as a 301 "Moved Permanently".

> *Note* — A redirect is REQUIRED when the *discovery IRI* is a *term name* to avoid confusing the *term name* with the document containing its definition, which is found at the post-redirect URL. Neither this standard nor [Basic Concepts] currently defines *properties* for use with documents, but future FHISO standards might, and servers conforming to this standard MAY include in their response RDF *triples* outside the scope of this standard, such as [Dublin Core] metadata about the document. Without this requirement, such metadata would be indistinguishable from *properties* about the *term* subject to *discovery*.

> *Example* — Suppose an application wants to perform *discovery* on a hypothetical `https://example.com/events/Baptism` *term*. An application wanting to maximise the likelihood of a response from any *conformant* server might make the following request:

```
GET /events/Baptism HTTP/1.1
Host: example.com
Accept: application/n-triples
```

This standard does not specifically require support for HTTP/1.1, but it is currently the most widely used version of HTTP and servers are strongly encouraged to support it. If the server does, as the `Accept` header is exactly "`application/n-triples`", a *conformant* server MUST conclude this is a request for an N-Triples representation of the *term*. And as the *discovery IRI* is the *term name*, it MUST respond with a redirect:

```
HTTP/1.1 303 See Other
Location: https://example.com/events/Baptism.n3
Vary: Accept
```

In this case the redirect is to the original IRI but with `.n3` appended, however the actual choice of IRI is up to the party defining the *term* and running the `example.com` web server. When a server's response is dependent on the contents of an `Accept` header, §7.1.4 of [RFC 7231] says that this SHOULD be recorded in a `Vary` header, as it is in this example. In practice other headers are likely to be present too, probably including a `Date` header containing the current date and time, and a `Server` header identifying the web server software; these have been omitted for brevity.

The application would normally then make a second HTTP request to follow the redirect:

```
GET /events/Baptism.n3 HTTP/1.1
Host: example.com
Accept: application/n-triples
```

This request uses the same `Accept` header as the first, as HTTP redirects contain no information about the MIME type of the destination resource, so at this point the application does not know whether the server has done HTTP content negotiation.

The server's response to this request should be an N-Triples file containing information about the `Baptism` *term*.

```
HTTP/1.1 200 OK
Content-Type: application/n-triples

<https://example.com/events/Baptism>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <https://example.com/types/Event> .
```

The meaning of the N-Triples in the request body is described in §3.

Conformant servers MAY issue additional redirects during the *Triples Discovery* process, including 301 "Moved Permanently" redirects. Applications MUST NOT infer anything from the use of redirects: in particular, if one *term name* IRI permanently redirects to another *term name* IRI, applications MUST NOT assume the *terms* are synonymous.

Server support for *Triples Discovery* on *namespace names* is OPTIONAL, and a *conformant* server which opts not to support it *must not* generate a response in N-Triples or an alternative media type from §3.3, and SHOULD instead send a 406 "Not Acceptable" or 404 "Not Found" response, depending whether or not documentation of the *namespace* is available in an alternative format. If *Triples Discovery* of *namespaces* is supported and if the *discovery IRI* is a *namespace name*, a redirect is OPTIONAL, but response MUST contain the *required triples* given in §4.

Conformant servers MAY support any version of HTTP and any additional HTTP features.

> *Example* — Conditional HTTP requests per [RFC 7232] are an example of a feature that the operators of *conformant* servers MAY opt to support. Applications MAY repeat *discovery* on certain *terms* after some time has elapsed, and include an `If-Modified-Since` header in the request. A server that has opted to support conditional requests would respond with a 304 "Not Modified" if the results of *discovery* have not changed. If the results have changed, or if the server cannot determine whether they have changed, or if the server does not support conditional requests of this form, it would produce a 200 "OK" response containing triples describing the *term*.

3 N-Triples syntax

N-Triples is a line-based format. Each non-empty line contains a **triple**, which is a sequence of three elements separated by *whitespace*, and ending with a "`.`" (U+002E). In the simplest case, each of the three elements is a *term* written as an absolute IRI enclosed in "<" and ">" (U+003C and U+003E). These three elements are known as the **subject**, **predicate** and **object** of the *triple*, and are used in this *discovery* mechanism to record the *properties* of a *term*.

The *subject* of the *triple* SHALL be the *subject* of the *property*: that is, the *term* being described. The *predicate* SHALL be the *property name* of the *property*, and the *object* SHALL be its *property value*.

> *Example* — The following is one *triple* in the N-Triples format, which is a *type triple*:
>
> ```
> <https://example.com/types/Date>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
>   → <http://www.w3.org/2000/01/rdf-schema#Datatype> .
> ```
>
> To be valid N-Triples, the triple MUST be on a single line, and there SHOULD be exactly one space *character* (U+0020) separating each pair of IRIs. The "`.`" (U+002E) at the end of the line is a REQUIRED part of the N-Triples syntax to mark the end of a *triple*.
>
> In this example, the *subject* is a hypothetical `Date` *term*, the *predicate* is `rdf:type` and the *object* is `rdfs:Datatype`. This *triple* is therefore describing the `Date` *term* and saying that the value of its `rdf:type` *property* is `rdfs:Datatype`: i.e. that this `Date` *term* is a *datatype*.

If the *predicate* is `rdf:type` the *triple* is known as a **type triple**.

> *Note* — *Type triples* are used in *Triples Discovery* as a minimal way of noting the existence of a *term*.

The details of this syntax are defined in the [N-Triples] standard, and a *triple* must match the `triple` grammar production given in §7 of [N-Triples].

Conformant servers MUST produce N-Triples in the canonical form of N-Triples defined in §4 of [N-Triples]. A *conformant* application MUST support the canonical form of N-Triples, and MAY support more of the full syntax of N-Triples. If a *conformant* application encounters an N-Triples file using unsupported features, it MAY discard the whole file or MAY discard the *triple* containing the feature.

> *Note* — Canonical N-Triples is a form of N-Triples which does not allow arbitrary *whitespace*, comments or certain escape constructs. This results in a further simplification to the parsing of N-Triples by removing alternative ways of serialisation the same triple. It is relatively rare to encounter N-Triples data which is not in its canonical form.

> *Note* — For convenience, the `triple` production and some related productions are reproduced here from §7 of [N-Triples]:
>
> ```
> triple     ::=  subject predicate object '.'
> subject    ::=  IRIREF | BLANK_NODE_LABEL
> predicate  ::=  IRIREF
> object     ::=  IRIREF | BLANK_NODE_LABEL | literal
>
> IRIREF     ::=  '<' ([^#x00-#x20<>"{}|^`\]* | UCHAR)* '>'
> UCHAR      ::=  '\u' HEX HEX HEX HEX
>               | '\U' HEX HEX HEX HEX HEX HEX HEX HEX
> HEX        ::=  [0-9] | [A-F] | [a-f]
> ```
>
> The UCHAR escape sequences are not permitted in Canonical N-Triples, and therefore support for them is OPTIONAL in this standard. Unicode characters are written directly, without the need for any escaping, in Canonical N-Triples.
>
> Because this standard permits *conformant* applications to parse non-conformant data, and because UCHAR support is OPTIONAL, an application MAY simply parse any sequence of non-*whitespace characters* between "<" and ">" as an IRI without validating it against the `IRIREF` production.
>
> *Whitespace* handling is underspecified in the full N-Triples syntax. This is the subject of erratum 24 in [RDF Errata] and is likely to be addressed in a future version of [N-Triples], most probably by allowing more liberal use of *whitespace*. In Canonical N-Triples, a single space *character* (U+0020) is required after each of the three elements of the *triple*, thus:
>
> ```
> triple  ::=  subject #x20 predicate #x20 object #x20 '.'
> ```
>
> *Triples* are separated by `[#xD#xA]+`, which allows blank lines, and permits all the major styles of line endings.

3.1 Literals

Instead of being a *term*, the *object* of a triple MAY alternatively be a **literal**, which has a *string* value instead of an IRI, and is serialised in double quotes (U+0022). Backslashes (U+005C) MUST be used to escape any double quotes, backslashes, line feeds (U+000A) or carriage returns (U+000D) that appear literally in the *string*. If the *predicate* of the *triple* is a *property term* whose *range* is a *datatype*, then the *object* of the *triple* SHALL be a *literal* rather than a *term*; otherwise the *object* of the *triple* SHALL be a *term*.

> *Example —* If *discovery* is performed on a *datatype name*, the resulting N-Triples SHOULD include its *type* and *pattern*, as specified using the `rdf:type` and `types:pattern` *properties terms*. For a hypothetical date type, this might be as follows:
>
> ```
> <https://example.com/types/Date>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
>   → <http://www.w3.org/2000/01/rdf-schema#Datatype> .
> <https://example.com/types/Date>
>   → <https://terms.fhiso.org/types/pattern>
>   → "[0-9]{4}-[0-9]{2}-[0-9]{2}" .
> ```
>
> The *range* of the `rdf:type` *property term* is `rdfs:Datatype` which is a *class*, and therefore the *property value* is serialised in N-Triples as a *term*; however the *range* of the `types:pattern` *property term* is `types:Pattern` which is a *datatype*, and therefore its *property value* is serialised as a *literal*.

In N-Triples, a *literal* MAY optionally be followed by either a *language tag* or a *datatype name*, but not both. If either of these is present, it is placed after the quoted *string* in the serialisation: the *language tag* if present is preceded by an "@" (U+0040), and a *datatype name* is enclosed in "<" and ">" (U+003C and U+003E) and preceded by "^^" (U+005E twice).

> *Example —*
>
> ```
> <https://example.com/types/YearMonth>
>   → <https://terms.fhiso.org/types/pattern>
>   → "[0-9]{4}-[0-9]{2}"^^<https://terms.fhiso.org/types/Pattern> .
> <https://example.com/types/YearMonth>
>   → <http://www.w3.org/2000/01/rdf-schema#label>
>   → "Jahr und Monat"@de .
> ```
>
> The *object* of the first of these *triples* is a *literal* tagged with a *datatype name*, `types:Pattern`. The *object* of the second *triple* is a *literal* with a *language tag*, de representing German.

A *literal* followed by a *language tag* is used to serialise a *property value* which is a *language-tagged string*. If the *predicate* of the *triple* is a *property term* whose *range* is a *language-tagged datatype*, then the *object* of the *triple* SHALL be a *literal* with a *language tag*.

> *Note* — N-Triples has no mechanism for stating a default *language tag*, so the *language tag* MUST NOT be omitted when serialising a *language-tagged string*.

The use of *literals* with *datatype names* is NOT RECOMMENDED when N-Triples is used in *Triples Discovery*. *Conformant* applications MUST be able to parse *literals* containing them, but MAY ignore any *datatype names* encountered and parse the *literal* as if it were absent.

> *Note* — Applications are REQUIRED be able to parse *literals* containing *datatype names* so that they can parse N-Triples data that was not generated specifically for the purpose of FHISO's *Triples Discovery* mechanism.

> *Editorial note* — Because *Triples Discovery* is only intended as a *discovery* mechanism for obtaining the definition of a *term*, it only need accommodate *properties* that might reasonably be defined on *terms*. At present this does not include *properties* with polymorphic *datatypes*, which is when a *datatype name* might be needed. For this reason their use is NOT RECOMMENDED. If this mechanism is generalised in the future and used for genealogical data, rather than just metadata on *terms*, it will be necessary to support *language tags* and *datatype names* properly.

Conformant servers MUST NOT produce *triples* whose *object* is a *literal* with a *datatype name* unless the *datatype* is either the *range* of the *predicate* of the *triple* or is a *subtype* of the *range* of the *predicate*. *Conformant* servers also MUST NOT produce *triples* whose *object* is a *literal* with a *datatype name* naming a *abstract datatype*. Applications MAY discard any *triple* not conforming to these requirement.

> *Example* — The previous example included the following triple:
>
> ```
> <https://example.com/types/YearMonth>
>   → <https://terms.fhiso.org/types/pattern>
>   → "[0-9]{4}-[0-9]{2}"^^<https://terms.fhiso.org/types/Pattern> .
> ```
>
> A *conformant* server MAY generate this, even though the use of the *datatype name* is NOT RECOMMENDED. This is because the *range* of the `types:pattern` *property term* is defined to be `types:Pattern`, which is the *datatype name* used, and `types:Pattern` is not an *abstract datatype*.

> *Note* — *Literals* match the `literal` grammar production in §7 of [N-Triples] which is reproduced here for convenience:
>
> ```
> literal               ::=  STRING_LITERAL_QUOTE
>                              ('^^' IRIREF | LANGTAG)?
> STRING_LITERAL_QUOTE  ::=  '"' ([^#x22#x5C#xA#xD]
>                                 | ECHAR | UCHAR)* '"'
> ECHAR                 ::=  '\' [tbnrf"'\]
> LANGTAG               ::=  '@' [a-zA-Z]+ ('-' [a-zA-Z0-9]+)*
> ```

The ECHAR production provides a means of escaping *characters* to appear in a *literal*:

tab	U+0009	`\t`
backspace	U+0008	`\b`
line feed	U+000A	`\n`
carriage return	U+000D	`\r`
form feed	U+000C	`\f`
double quote	U+0022	`\"`
single quote	U+0027	`\'`
backslash	U+005C	`\\`

Canonical N-Triples only allows the use of `\"`, `\\`, `\r` and `\n`. Support for the other escapes, like the UCHAR escape mechanism, is OPTIONAL.

3.2 Blank nodes

N-Triples also allows the *subject* or *object* of a *triple* to be a **blank node**, which have serialisations in N-Triples beginning with "_:" (U+005F, U+003A). This *Triples Discovery* mechanism makes no use of *blank nodes* and *conformant* applications SHOULD ignore any triples containing them.

Example — The following *triple* has a blank node as its *object* and SHOULD be ignored.

```
<https://example.com/types/YearMonth>
  → <http://www.w3.org/2000/01/rdf-schema#isDefinedBy> _:1 .
```

Note — A future FHISO standard might use *blank nodes* to represent more complicated *property values* that cannot conveniently be represented by a *term* or a *literal*. For this reason, this standard does not prohibit *conformant* servers from generating *triples* using *blank nodes*.

Editorial note — FHISO recognises that *Triples Discovery* will need to be updated to include support for list-valued properties. RDF supports lists in the form of the `rdf:List` *class*, and while introducing `rdf:List` is appealing and potentially solves various other problems, it causes implementation difficulties in *Triples Discovery*. This is because, for simplicity's sake, we use N-Triples as the serialisation format, and, uniquely among the common RDF serialisation syntaxes, N-Triples lacks any clean syntax for representing an `rdf:List`. Instead, the N-Triples representation would involve a Lisp-like representation of the `rdf:List` with a series of blank nodes:

```
types:Date owl:unionOf _:1 .
_:1 rdf:type rdf:List .
_:1 rdf:first types:AbstractDate .
_:1 rdf:rest _:2 .
```

```
_:2 rdf:type rdf:List .
_:2 rdf:first rdf:langString .
_:2 rdf:rest rdf:nil .
```

To support this, support for *blank nodes* and the `rdf:first`, `rdf:rest` and `rdf:nil` constructs would have to become REQUIRED. This adds some significant complexity to the implementation, as the implementation would need to recognise and ignore malformed uses of these constructs.

For comparison, the same data expressed in Turtle (which has native support for lists) would be:

```
types:Data owl:unionOf ( types:AbstractDate rdf:langString ) .
```

Clearly this syntax is much cleaner; it also avoids the need for applications to support *blank nodes* and the `rdf:first`, `rdf:rest` and `rdf:nil` constructs. Turtle (and RDF/XML and JSON-LD, the other main formats) are a somewhat harder format to parse than N-Triples. There are plenty of libraries that will parse these formats, but this standard was designed to make parsing easy so that a library wasn't needed. Also, the interface to many parsing libraries is a stream of *triples* which have lists expanded into their `rdf:first`, `rdf:rest` and `rdf:nil` equivalents.

Making syntactic support for *blank nodes* OPTIONAL (whether using the `_:1` syntax or the `[]` syntax) would make writing a parser a bit simpler. Unfortunately, other features cannot be made OPTIONAL so readily, as dropping them would make it hard to parse information on existing vocabularies, or to use standard tools to create the discovery files served by *conformant servers*.

This issue needs resolving before list-valued *properties* can be added to [Basic Concepts].

3.3 Other formats

> *Note* — This section explains how alternative RDF formats may be used instead of N-Triples. Support for any other format is OPTIONAL.

N-Triples is not the only serialisation format for representing RDF data, nor is it the most commonly used.

> *Note* — Originally the only serialisation format for RDF was [RDF/XML], and despite its excessive verbosity, this format remains popular for compatibility with older datasets and applications. A more modern format is [Turtle] which is defined to be easily written and read by a human; it is a superset of N-Triples, though not as trivial to parse. Many RDF frameworks support these as well as N-Triples, and can convert between formats.

A *conformant* server MAY make information about *terms* available in other RDF formats too. If a server does so, it SHOULD provide the same information in these other RDF formats as it provides is N-Triples, and MUST ensure that the information made available in any other supported RDF formats

includes the *required triples*, as given in §4. A *conformant* server MUST NOT only make information available in an RDF format other than N-Triples.

A *conformant* application MAY request data in other RDF formats, and MAY request these formats in preference to or before N-Triples, but must also support N-Triples. It SHALL parse data in other RDF formats as if by converting them to N-Triples and parsing that per this standard.

> *Note* — In practice, such an application would likely use an RDF framework to parse all the supported RDF formats, N-Triples included, and then process the parsed triples that the framework provides.

4 Required triples

When *discovery* is carried out on an IRI (the *discovery IRI*) which is a known *term name* (including the *namespace name*, as defined in §4.2 of [Basic Concepts], if discovery on *namespaces* is supported), a *conformant* server SHALL ensure that the response includes certain **required triples**. The response MAY contain other *triples* in addition.

If the *discovery IRI* is a *namespace name* and if *discovery* of *namespace names* is supported, the set of *required triples* SHALL be the set of *type triples* for every *term* whose *namespace name* is the *discovery IRI*.

> *Note* — According to §2, *discovery* on a *namespace name* is OPTIONAL. However, if discovery is supported the *required triples* for such discovery is the full set of *terms* in the *namespace*. Thus a *conformant* server MUST respond to a *namespace name discovery IRI* either with an error code or with a response including all of the *terms* in the *namespace*.

> *Example* — [Basic Concepts] defines nine *terms* whose *term names* begin with the following IRI:
>
> ```
> https://terms.fhiso.org/types/
> ```
>
> This is the *namespace* of these nine terms, therefore *discovery* on that IRI either MUST yield an error (if *discovery* on *namespace names* is not supported) or MUST include the following triples:
>
> ```
> <https://terms.fhiso.org/types/constituentDatatype>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
> <https://terms.fhiso.org/types/nonTrivialSupertype>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
> ```

```
<https://terms.fhiso.org/types/Union>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/2000/01/rdf-schema#Class> .
<https://terms.fhiso.org/types/isAbstract>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<https://terms.fhiso.org/types/requiredProperty>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<https://terms.fhiso.org/types/pattern>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<https://terms.fhiso.org/types/nonTrivialSupertypeCount>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<https://terms.fhiso.org/types/constituentDatatypeCount>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<https://terms.fhiso.org/types/Pattern>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/2000/01/rdf-schema#Datatype> .
```

If in the future FHISO defines further *terms* in that *namespace*, the server MUST include them too.

Otherwise, the *required triples* SHALL include *triples* for every *required property* of the *class* which is the *type* of the *discovery IRI*. These *triples* all have the *discovery IRI* as their *subject*.

Example — [Basic Concepts] defines a `types:pattern` *term* whose *type* is `rdf:Property`. According to §5.2 of [Basic Concepts], `rdf:Property` has two *required properties*: `rdf:type` and `rdfs:range`. The set of *required triples* therefore includes *triples* whose *subjects* are `types:pattern` and whose *predicate* is each of these *required properties*:

```
<https://terms.fhiso.org/types/pattern>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<https://terms.fhiso.org/types/pattern>
  → <http://www.w3.org/1999/02/22-rdf-syntax-ns#range>
  → <https://terms.fhiso.org/types/Pattern> .
```

If the *discovery IRI* is a *class name* then the *required triples* SHOULD additionally include *type triples* for all known *terms* whose *type* is the *class* being discovered. At the least, this MUST include *type triples* for any *terms* with this *type* that were defined or referenced in the standard which defines the *class*, or which have the same *namespace* as the *class*.

> *Example —* [CEV Concepts] defines a *class* `cev:SourceDerivation`; that standard also defines a *term*, `cev:derivedFrom` whose *type* is that *class*. Therefore the *required triples* of `cev:SourceDerivation` include the two *required properties* for an `rdfs:Class` as well as the *type triple* for `cev:derivedFrom`:
>
> ```
> <https://terms.fhiso.org/sources/SourceDerivation>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
>   → <http://www.w3.org/2000/01/rdf-schema#Class> .
> <https://terms.fhiso.org/sources/SourceDerivation>
>   → <https://terms.fhiso.org/types/requiredProperty>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> .
> <https://terms.fhiso.org/sources/derivedFrom>
>   → <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
>   → <https://terms.fhiso.org/sources/SourceDerivation> .
> ```

5 References

5.1 Normative references

[Basic Concepts]
FHISO (Family History Information Standards Organisation). *Basic Concepts for Genealogical Standards*. First public draft. (See https://fhiso.org/TR/basic-concepts.)

[N-Triples]
W3C (World Wide Web Consortium). *RDF 1.1 N-Triples*. David Becket, 2014. W3C Recommendation. (See https://www.w3.org/TR/n-triples/.)

[RFC 3987]
IETF (Internet Engineering Task Force). *RFC 3987: Internationalized Resource Identifiers (IRIs)*. Martin Duerst and Michel Suignard, eds., 2005. (See https://tools.ietf.org/html/rfc3987.)

[RFC 7230]
IETF (Internet Engineering Task Force). *RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Roy Fielding and Julian Reschke, eds., 2014. (See https://tools.ietf.org/html/rfc7230.)

[RFC 7232]
IETF (Internet Engineering Task Force). *RFC 7232: Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*. Roy Fielding and Julian Reschke, eds., 2014. (See https://tools.ietf.org/html/rfc7232.)

5.2 Other references

[CEV Concepts]

FHISO (Family History Information Standards Organisation). *Citation Elements: General Concepts". Third public draft. See https://fhiso.org/TR/cev-concepts.

[Dublin Core]

Dublin Core Metadata Initiative. *Dublin Core metadata element set*. Dublin Core recommendation, version 1.1, 1999. See http://dublincore.org/documents/dcmi-terms/.

[RDF Errata]

W3C (World Wide Web Consortium). *RDF1.1 Errata.* (See https://www.w3.org/2001/sw/wiki/RDF1.1_Errata.)

[RDF/XML]

W3C (World Wide Web Consortium). *RDF 1.1 XML Syntax.* Fabien Gandon and Guus Schreiber, eds., 2014. W3C Recommendation. (See https://www.w3.org/TR/rdf-syntax-grammar/.)

[RFC 7231]

IETF (Internet Engineering Task Force). *RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* Roy Fielding and Julian Reschke, eds., 2014. (See https://tools.ietf.org/html/rfc7231.)

[SWBP Vocab Pub]

W3C (World Wide Web Consortium). *Best Practice Recipes for Publishing RDF Vocabularies*. Diego Berrueta and Jon Phipps, eds., 2008. W3C Working Group Note. (See https://www.w3.org/TR/swbp-vocab-pub/.)

[Turtle]

W3C (World Wide Web Consortium). *RDF 1.1 Turtle.* Eric Prud'hommeaux and Gavin Carothers, eds., 2014. W3C Recommendation. (See https://www.w3.org/TR/turtle/.)
