Citation Elements: General Concepts

11 September 2017

> *Editorial note* — This is a **second public draft** of the core part of FHISO's proposed suite of standards on Citation Elements. This document is not endorsed by the FHISO membership, and may be updated, replaced or obsoleted by other documents at any time.
>
> In particular, some examples in this draft use *citation elements* that are not even included in the draft Citation Elements: Vocabulary. These elements are very likely to be changed as the vocabulary progresses.
>
> 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/cev-rdfa-bindings` |
> | This version: | `https://fhiso.org/TR/cev-rdfa-bindings-20170911` |
> | Previous version: | `https://fhiso.org/TR/cev-rdfa-bindings-20170626` |

FHISO's suite of **Citation Elements** standards provides an extensible framework and vocabulary for encoding all the data about a genealogical *source* that might reasonably be included in a *formatted citation* to that *source*.

This document defines the general concepts used in FHISO's suite of Citation Elements standards, and the basic framework and data model underpinning them. Other standards in the suite are as follows:

— **Citation Elements: Vocabulary**. This standard defines a collection of *citation elements* allowing the representation of information normally found in *formatted citations* to diverse types of source.
— **Citation Elements: Bindings for RDFa**. This standard defines a means by which *citation elements* may be identified and tagged using RDFa attributes within HTML and XML *formatted citations*, allowing a computer to extract them in a systematic manner.
— **Citation Elements: Bindings for GEDCOM X**. This standard defines extensions to the GEDCOM X data model and its JSON and XML serialisations to allow *citation elements* to be represented in GEDCOM X.
— **Citation Elements: Bindings for ELF**. This standard defines how *citation elements* should be represented in FHISO's Extensible Legacy Format (ELF), a format based on and compatible with GEDCOM 5.5, but with the addition of a new extensibility mechanism.

> *Editorial note* — Not all of these documents are yet at the stage of having a first public draft.

1 Introduction

1.1 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.

Indented text in grey or coloured boxes, such as preceding paragraph, 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.

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.

1.2 Basic concepts

A **source** is any resource from which information is obtained during the genealogical research process. *Sources* come in many forms, including manuscripts, artefacts, books, films, people, recordings and websites. A full mechanism for describing *sources* is beyond the scope of this standard.

A **source derivation** is a directional link between two *sources*, indicating that the first *source* was derived from, cites or otherwise references the second *source*. The first *source* is referred to as the **derived source**, and the second the **base source**.

> *Note* — The term "derivation" is used very broadly in this standard, and includes relationships that might not normally be considered derivative. A *source derivation* exists between a digitisation, translation, transcription or index and the original document. A *source derivation* exists between a published genealogy and each *source* it cites. A *source derivation* also exists between a paper and a second paper which it is rebutting or commenting on.

A **citation** is an abstract reference to a specific *source* from which information has been used in some context. It SHOULD include sufficient detail that a third-party could readily locate the information themselves, assuming the *source* remains accessible.

A **formatted citation** is a *citation* that has been rendered into human-readable form, typically as a sentence or short paragraph that might be used as a footnote, endnote, tablenote or bibliography entry. There is no single standard on the correct form of *formatted citations*; many different style guides exist, each giving their own rules on how to construct a *formatted citation*.

> *Example* — A *formatted citation* produced for use in a footnote on the first use of the *source*, and conforming to [Chicago] might read:
>
> > [1] Christian Settipani, *Les ancêtres de Charlemagne*, 2nd ed. (Oxford: Prosopographia et Genealogica, 2015), 129–31.
>
> The [1] at the start of the citation is the hypothetical footnote number.

> *Note* — Footnotes and other reference notes sometimes contain information besides *citations*. This may include commentary on the accessibility, accuracy, authenticity or provenance of a *source*. As this information is not part of a *citation*, it is beyond the scope of this standard.

A **layered citation** is a *citation* that includes information about several *sources* between which *source derivation links* exist. The information in a *layered citation* about a specific *source*, whether the consulted *source* or one of *sources* from which it was derived, is known as a **citation layer**. A *citation* with just a single *citation layer* is called a **single-layer citation**.

The *citation layer* containing the information about the specific *source* which was consulted is known as the **head citation layer**. For a *single-layer citation*, its sole *citation layer* is necessarily the *head citation layer*.

> *Example* — A *citation* to a census return that was consulted on microfilm might contain information about the microfilm and as well as information about the census return, as in the following *formatted citation* from [Evidence Explained]:
>
> > 1810 U.S. census, York County, Maine, town of York, p. 435 (penned), line 9, Jabez Young; NARA microfilm publication M252, roll 12.

> In this example, the information before the semicolon pertains to the census return, while the information after it pertains to the microfilm. The microfilm and the census return are different *sources*, and a *source derivation* exists between them as the microfilm is derived from the census return. The information in the *citation* about microfilm forms the *head citation layer*, while the information about the census return forms a separate *citation layer*. As the *citation* contains two *citation layers*, it is an example of a *layered citation*.
>
> In this example, the *head citation layer* is not presented first in the *formatted citation*. Whether the *head citation layer* is presented first is a matter of style and emphasis, and it is common not to present the *head citation layer* first when it is a photographic or digital reproduction, as in this case.

> *Note* — *Layered citations* are often used to provide a partial statement of provenance, documenting how documents derived from one another. Many treatments of provenance also include information that is not included in citations, and hence not covered by this specification, such as a custody of ownership or characterization of the completeness of sources cited.

A **citation element** is a logically self-contained piece of information in a *citation layer* that might reasonably be included in a *formatted citation*. As this standard does not aim to provide facilities for the exhaustive description of *sources*, information about *sources* that is not normally included in *formatted citations* is not considered to be a *citation element*. *Citation elements* are represented in a sufficiently structured and language-independent way that applications can parse and reformat it in different styles and languages as needed.

> *Example* — The date that a *source* like a newspaper article was published is an example of a *citation element*. An American researcher might write the date as “Oct 8th, 2000”, while the same date might be written “zo. 8 okt. 2000” by a Dutch researcher. The *citation element* should use neither of these as its representation of the date and adopt a language-neutral format, such as one based on [ISO 8601].

The accompanying Citation Elements: Vocabulary standard defines many *citation elements*, covering the information normally found in *formatted citations* to a wide range of common *sources*. Applications MAY define their own *citation elements* or use those defined by a third-party standard; such *citation elements* are known as **extension citation elements**.

Conforming applications MUST NOT discard *citation elements*, except on the instruction of the user or as explicitly permitted in this standard. This applies to unrecognised *extension citation elements* too, though an application MAY opt not to display any such *citation elements*.

> *Editorial note* — Note that the definition of *citation element* limits it to information that might reasonably appear in a citation; thus, most items of metadata (such as who created the citation and when, or a globally-unique identifier for the citation or its layers) are not properly considered *citation elements* themselves.

> It is anticipated that metadata will be addressed in a future FHISO standard. Initial brainstorming on metadata implementation suggests that this document may be edited slightly to support metadata, perhaps by adding an optional identifier or context pointer to each element. The exact nature of such an edit, or if it will even be necessary, will depend on future development of that metadata standard.

A **citation element set** is a collection of *citation elements* that completely encode the information about a *source* that is present in a particular *citation layer*.

> *Example —* The example *formatted citation* to *Les ancêtres de Charlemagne* is represented by a *citation element set* containing the following seven *citation elements*:
>
> — The author: “`Settipani, Christian`”.
> — The title: “`Les ancêtres de Charlemagne`”.
> — The edition: “2”.
> — The place of publication: “`Oxford`”.
> — The publisher: “`Prosopographia et Genealogica`”.
> — The year of publication: “`2015`”.
> — The page range: “`129-131`”.
>
> The footnote number is not a *citation element* as it does not pertain to the *source*. The author and page range are not expressed here in quite the same form as the *formatted citation*, but an application can readily parse them to convert them to the required format because their format is defined by this standard.

When provided with the *citation element set* for each *citation layer* in the *citation*, knowledge of which is the *head citation layer*, information about the *source derivations* between *sources* referred to in each *citation layer*, and any necessary internal state, an application ought to be able to produce algorithmically a *formatted citation* in a reasonable approximation to any mainstream citation style. If higher quality *formatted citations* are desirable, applications SHOULD allow users to manually edit them to fine-tune their presentation, and SHOULD store the result for reuse. *Formatted citations* need not include all the information from a *citation element set* if the style dictates that certain information is omitted in the relevant context.

> *Note —* Producing *formatted citations* of a professional quality following a particular style guide is a difficult art about which books have been written. This standard does not require applications to produce *formatted citations*, and throughout this suite of standards, there is no expectation that an application choosing to do so should be able to do better than a “reasonable approximation” when generating *formatted citations* automatically. That is why this standard recommends that users be allowed to fine-tune them by hand if high quality *formatted citations* are required.

Citation element sets SHOULD NOT include *citation elements* for information that is not normally included in a *formatted citation*. They are not intended to provide a general mechanism for storing arbitrary information about *sources*.

> *Example — Formatted citations* do not normally include details such as the email addresses, phone numbers or academic affiliations of authors, so they should not be included in the *citation element set*. A more general mechanism for describing *sources* may well include such elements, but they are beyond the scope of this standard.

1.3 Characters and strings

Characters are specified by reference to their *code point* number in [ISO 10646], without regard to any particular character encoding. In this standard, *characters* may be identified in this standard by their hexadecimal code point prefixed with "U+".

> *Note* — The character encoding is a property of the serialisation, and not defined in this standard. Non-Unicode encodings are not precluded, so long as it is defined how characters in that encoding corresponds to Unicode characters.

Characters MUST match the `Char` production from [XML].

```
Char   ::=   [#1-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
```

> *Note* — This includes all *code points* except the null character, surrogates (which are reserved for encodings such as UTF-16 and not characters in their own right), and the invalid characters U+FFFE and U+FFFF.

A **string** is a sequence of zero or more *characters*.

> *Note* — The definition of a *string* is identical to the definition of the `string` datatype defined in [XSD Pt2], used in many XML and Semantic Web technologies.

Applications MAY convert any *string* into Unicode Normalization Form C, as defined in any version of Unicode Standard Annex #15 [UAX 15].

> *Note* — Normalization Form C and Normalization Form D allow easier searching, sorting and comparison of *strings* by picking a canonical representation of accented characters. The conversion between Normalization Forms C and D is lossless and therefore reversible, but the initial conversion to either form is not reversible. This allows a *conformant* application to normalise *strings* internally and not retain the unnormalised form; however, an application doing so MUST ensure the *string* is in Normalization Form C upon export, this being the more usual form for use in documents.

Characters matching the `RestrictedChar` production from [XML] SHOULD NOT appear in *strings*, and applications MAY process such characters in an implementation-defined manner or reject *strings* containing them.

```
RestrictedChar   ::=   [#x1-#x8] | [#xB-#xC] | [#xE-#x1F]
                         | [#x7F-#x84] | [#x86-#x9F]
```

> *Note* — This includes all C0 and C1 control characters except tab (U+0009), line feed (U+000A), carriage return (U+000D) and next line (U+0085).

> *Example* — As *conformant* applications can process C1 control characters in an implementation-defined manner, they can opt to handle Windows-1252 quotation marks in data masquerading as Unicode. Applications MUST NOT treat non-ASCII characters as ANSEL, the character set properly used in GEDCOM, as ANSEL's non-ASCII characters do not correspond to `RestrictedChars`.

Whitespace is defined as a sequence of one or more space *characters*, carriage returns, line feeds, or tabs. It matches the production `S` from [XML].

```
S  ::=  (#x20 | #x9 | #xD | #xA)+
```

Whitespace normalisation is the process of discarding any leading or trailing *whitespace*, and replacing other *whitespace* with a single space (U+0020) *character*.

> *Note* — The definition of *whitespace normalisation* is identical to that in [XML].

In the event of a difference between the definitions of the `Char`, `RestrictedChar` and `S` productions given here and those in [XML], the definitions in the latest edition of XML 1.1 specification are definitive.

1.4 Terms

> *Editorial note* — The concept of a *term* is new in this draft of the standard, introduced to refactor wording common to both *citation element names* and the new concept of a *datatype* into a single place.

A **term** consists of a unique, machine-readable identifier, known as the **term name**, paired with a clearly-defined meaning for the concept or idea that it represents. This standard uses *terms* as *datatypes* and *citation element names*, as defined in §2 and §3 of this standard respectively. *Term names* SHALL take the form of an IRI matching the `IRI` production in §2.2 of [RFC 3987].

> *Note* — IRIs have been chosen in preference to URIs because it is recognised that certain culture-specific genealogical concepts may not have English names, and in such cases the human-legibility of IRIs is advantageous. URIs are a subset of IRIs, and all the *terms* defined in this suite of standard are also URIs.

Term names are compared using the "simple string comparison" algorithm given in §5.3.1 of [RFC 3987]. If a *term name* does not compare equal to an IRI known to the application, the application MUST NOT make any assumptions about the *term*, its meaning or intended use, based on the form of the IRI or any similarity to other IRIs.

> *Note* — This comparison is a simple character-by-character comparison, with no normalisation carried out on the IRIs prior to comparison. It is also how XML namespace names are compared in [XML Names].

> *Example* — The following IRIs are all distinct for the purpose of the "simple string comparison" algorithm given in §5.3.1 of [RFC 3987], , even though an HTTP request to them would fetch the same resource.
>
> ```
> https://éléments.example.com/nationalité
> HTTPS://ÉLÉMENTS.EXAMPLE.COM/nationalit%C3%A9
> https://xn--lments-9uab.example.com/nationalit%c3%a9
> ```

An IRI MUST NOT be used as a *term name* unless it can be converted to a URI using the algorithm specified in §3.1 of [RFC 3987], and back to a IRI again using the algorithm specified in §3.2 of [RFC 3987], to yield the original IRI.

> *Note* — This requirement ensures that *term names* can be used in a context where a URI is required, and that the original IRI can be regenerated, for example for comparison with a list of known IRIs. The vast majority of IRIs, including those in non-Latin scripts, have this property. The effect of this requirement is to prohibit the use of IRIs that are already partly converted to a URI, for example through the use of unnecessary percent or punycode encoding.

> *Example* — Of the three IRIs given in the previous example on how to compare IRIs, only the first may be used as a *term name*. The second and third are prohibited as a result of the unnecessary percent-encoding, and the third is additionally prohibited as a result of unnecessary punycode-encoding.

The *terms* defined in this standard all have *term names* that begin `https://terms.fhiso.org/`. Subject to the requirements herein, third parties may also define additional *terms* for use as *datatypes* or *citation elements*. It is RECOMMENDED that any such *terms* use the `http` or preferably `https` IRI scheme defined in §2.7.1 and §2.7.2 of [RFC 7230] respectively, and an authority component consisting of just a domain name or subdomain under the control of the party defining the *extension citation elements*.

> *Note* — An `http` or `https` IRI scheme is RECOMMENDED because the IRI is used to fetch a resource during *discovery*, and it is desirable that applications implementing *discovery* should only need to support a minimal number of transport protocols. URN schemes like the `uuid` scheme of [RFC 4122] are NOT RECOMMENDED as they do not have transport protocols that can be used during *discovery*.
>
> The preference for a `https` IRI is because of security considerations during *discovery*. A man-in-the-middle attack during *discovery* could insert malicious content into the response, which, if undetected, could cause an application to process user data incorrectly, potentially

> discarding parts of it or otherwise compromising its integrity. It is harder to stage a man-in-the-middle attack over TLS, especially if public key pinning is used per [RFC 7469].

1.4.1 Prefix notation

Term names in the Citation Elements standard are sometimes referred to in **prefix notation**. This is a system whereby **prefixes** are assigned to IRIs that occur frequently as the leading portion of a *term name*. Then, instead of writing the *term name* in full, the leading portion of the *term name* is replaced by its *prefix* followed by a colon (U+003A) separator.

> *Example —* The *term name* `https://terms.fhiso.org/sources/title` is used in several of the examples in this standard. Instead of writing this in full, if the `cev` *prefix* is bound to the IRI `https://terms.fhiso.org/sources/`, then this IRI can be written in *prefix form* as `cev:title`.

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#`

> *Note —* The particular *prefixes* 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. Nevertheless, some serialisation formats, including the [CEV RDFa] bindings, do make use of *prefix notation* to shorten the serialised form of data.

1.4.2 IRI resolution

It is RECOMMENDED that an HTTP `GET` request to a *term name* IRI with an `http` or `https` scheme (once converted to a URI per §4.1 of [RFC 3987]), SHOULD result in a 303 “See Other” redirect to a document containing a human-readable definition of the *term* if the request was made without an `Accept` header or with an `Accept` header matching the format of the human-readable definition. It is further RECOMMENDED that this format should be HTML, and that documentation in alternative formats MAY be made available via HTTP content negotiation when the request includes a suitable `Accept` header, per §5.3.2 of [RFC 7231].

> *Note* — A 303 redirect is considered best practice for [Linked Data], so as to avoid confusing the *term name* IRI with the document containing its definition, which is found at the post-redirect URL. The *terms* defined in this suite of standards are not specifically designed for use in Linked Data, but the same considerations apply.

Parties defining *terms* MAY arrange for their *term name* to support **discovery**. This when an HTTP `GET` request to a *term name* IRI with an `http` or `https` scheme, made with an appropriate `Accept` header, yields 303 redirect to a machine-readable definition of the *term*.

> *Editorial note* — FHISO does not currently define a *discovery* mechanism, but anticipate doing so in a future standard. If such a standard is included in the initial suite of Citation Elements standards, it is likely to be RECOMMENDED that parties defining *terms* SHOULD arrange for them to support *discovery*, while application support for it would be OPTIONAL.

2 Datatypes

> *Editorial note* — The concept of a *datatypes* is new in this draft of the standard.

A **datatype** is a *term* which serves as a formal description of the values that are permissible in a particular context. Being a *term*, a *datatype* is identified by a *term name* which is an IRI. The *term name* of a *datatype* is also referred to as a *datatype name*.

A *datatype* has a **lexical space** which is the set of *strings* which are interpreted as valid values of the *datatype*. The definition of a *datatype* SHALL state how each string in its *lexical space* maps to a logical value, and state the semantics associated with of those values.

> *Note* — This definition of a *datatype* is sufficiently aligned with XML Schema's notion of a simple type, as defined in [XSD Pt2], that XML Schema's simple types can be used as *datatypes* in this standard. Best practice on how to get an IRI for use as the *term name* of XML Schema types can be found in [SWBP XSD DT]. Similarly, this standard's definition of a *datatype* is very similar to the definition of a datatype in [RDF Concepts], and RDF datatypes can be used as *datatypes* in this standard.

> *Example* — XML Schema defines an integer type in §3.4.13 of [XSD Pt2] which is well suited for use in this standard. XML Schema does not give its types IRIs, but it does give them `id`s, and following the best practice advice given in §2.3 of [SWBP XSD DT] gives it the following IRI:
>
> ```
> http://www.w3.org/2001/XMLSchema#integer
> ```
>
> This same type is also recommended for use in RDF by §5.1 of [RDF Concepts] which explicitly gives it the IRI above.

> The *lexical space* of this *datatype* is the space of all *strings* consisting of a finite-length sequence of one or more decimal digits (U+0030 to U+0039, inclusive), optionally preceded by a `+` or `-` sign (U+002B or U+002D, respectively). Thus the *string* "`137`" is within the *lexical space* of this *datatype*, but "`20.000`" and "`四十二`" are not, despite being normal ways of representing integers in certain cultures.

> *Editorial note* — The examples in this section use various XML Schema types. As the [CEV Vocabulary] takes shape, these should be replaced with types that are actually used in our vocabulary, which may or may not be XML Schema types.

The mapping from lexical representations to logical values need not be one-to-one. If a *datatype* has multiple lexical representations of the same logical value, a *conformant* application MUST treat these representations equivalently and MAY change a *string* of that *datatype* to be a different but equivalent lexical representation.

> *Note* — This allows applications to store such *strings* internally using as an entity (such as a database field or a variable) of some appropriate type without retaining the original lexical representation.

> *Example* — The XML Schema `integer` *datatype* used in the previous example is one where the mapping from lexical representation to value is many-to-one rather than one-to-one. This is due to *lexical space* including strings with a leading `+` sign as well as superfluous leading 0s, and means that "`00137`", "`+137`" and "`137`" all represent the same underlying value: the number one hundred and thirty-seven. Because *conformant* applications MAY convert strings between equivalent lexical representations, they MAY store them in a database in an integer field and regenerate *strings* in a canonical representation.

Strings outside the *lexical space* of a *datatype* MUST NOT be used where a *string* of that *datatype* is required. If an application encounters any such *strings*, it MAY remove them from the dataset or MAY convert them to a valid value in an implementation-defined manner. Any such conversion that is applied automatically by an application MUST either be locale-neutral or respect any locale given in the dataset.

> *Example* — XML Schema defines a `date` type in §3.3.9 of [XSD Pt2] which has a *lexical space* based on [ISO 8601] dates. If, in a dataset that is somehow identified as being written in German, an application encountering the *string* "`8 Okt 2000`" in a context where an XML Schema `date` is expected, it MAY convert this to "`2000-10-08`". However an application encountering the *string* "`8/10/2000`" MUST NOT conclude this represents 8 October or 10 August unless the document includes a locale that uniquely determines the date format. In this case, information that the document is in English is not sufficient as different English-speaking countries have different conventions for formatting dates.

2.1 Language-tagged datatypes

A **language-tagged datatype** is a *datatype* whose value consists of both a *string* from the *lexical space* of the *datatype* and a **language tag** to identify the language, and where appropriate the script and regional variant, in which that particular *string* is written. The *language tag* SHALL match the `Language-Tag` production from [RFC 5646].

> *Note* — The *language tag* is not itself part of the *lexical space* of the *datatype*, and is not embedded in the *string*, but is stored alongside it.

Language-tagged datatypes SHOULD be used whenever a *datatype* is needed to represent textual data that is in a particular language or script and which cannot automatically be translated or transliterated as required, and SHOULD NOT be used otherwise.

> *Example* — In a context where a year *Anno Domini* is required, a *language-tagged datatype* SHOULD NOT be used, and the *lexical space* of the *datatype* should encompass *strings* like, say, "2015". Even though an application designed for Arabic researchers might need to render this year as "٢٠١٥" using Eastern Arabic numerals, this conversion can be done entirely in the application's user interface, so a *language-tagged datatype* is not required and SHOULD NOT be used.

> *Example* — The [CEV Vocabulary] defines a *datatype* for representing the names of authors and other people, which has the following *term name*:
>
> `https://terms.fhiso.org/sources/AgentName`
>
> A person's name is rarely translated in usual sense, but may be transliterated. For example, the name of Andalusian historian صاعد الأندلسي might be transliterated "Ṣāʿid al-Andalusī" in the Latin script. Because machine transliteration is far from perfect, a *language-tagged datatype* SHOULD be used to allow an application to store both names. In this case, they would be tagged `ar` and `ar-Latn` respectively, meaning the Arabic language in its default script and in the Latin script.
>
> An author's names may also be respelled to conform to the spelling and grammar rules of the reader's language. An Englishman named Richard may be rendered "Rikardo" in Esperanto: the change of the "c" to a "k" being to conform to Esperanto orthography, while the final "o" marks it as a noun. The respelling would be tagged `eo`, the language code for Esperanto.

A *datatype* that is not a *language-tagged datatype* is called a **non-language-tagged datatype**.

2.2 Datatype patterns

A party defining a *datatype* SHALL specify a **pattern** for that *datatype*. This is a regular expression which provides a constraint on the *lexical space* of the *datatype*. Matching the *pattern* might not be sufficient to validate a *string* as being in the *lexical space* of the *datatype*, but a *string* that fails to match the *pattern* is guaranteed not to be in the *lexical space*.

> *Note* — Patterns are included in this standard to provide a way for an application to find out about the *lexical space* of a unfamiliar *datatype* through *discovery*. They are used during the *datatype correction* process defined in §4.4.

> *Editorial note* — We need to specify a particular dialect of regular expression. One option is the form defined in §21.2 of [ECMAScript] which has the advantage of being supported in most programming languages. It currently has relatively poor Unicode support (e.g. it lacks `\p`), though it seems likely this will improve in the next version of ECMAScript. Another option is to use the form defined in Appendix G of [XSD Pt2] which is much less widely supported, but has the advantage of being the standard form for defining *datatypes* in XML and RDF.

> *Editorial note* — We also need to specify exactly what *matching* a *pattern* means. In particular we want the complete *string* to match the *pattern*, so that "`Sept 2017`" does not match the *pattern* `[0-9]{4}`, despite the lack of `^...$` around the *pattern*.

> *Example* — The XML Schema `date` type mentioned in the previous example has the following *pattern* (here split onto two lines for readability — the second line is an optional timezone which the XML Schema `data` type allows).
>
> ```
> -?([1-9][0-9]{3,}|0[0-9]{3})-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])
> (Z|(\+|-)((0[0-9]|1[0-3]):[0-5][0-9]|14:00))?
> ```
>
> This *pattern* matches *strings* like "`1999-02-31`". Despite matching the *pattern*, this *string* is not part of the *lexical space* of this `date` type as 31 February is not a valid date.

A *datatype* with a *pattern* other than `.*` is known as a **structured datatype**, while one with a *pattern* of `.*` is known as an **unstructured datatype**. It is expected that most *datatypes* in common use, other than the `rdf:langString` *datatype* defined in §2.4.1 will be *structured datatypes*.

> *Note* — *Patterns* may be defined for *language-tagged datatypes* just for other *datatypes*. This means the classification of *datatypes* as *language-tagged* or *non-language-tagged* is orthogonal to their classification as *structured* or *unstructured*. Because *patterns* only constrain the *lexical space* of the *datatype*, they cannot be used to constrain the *language tag* in the value of a *language-tagged datatype*.

> *Example —* The `AgentName` datatype used to represent the names of authors and other people is a microformat which is constrained by a *pattern* meaning it is a *structured datatype*, but it is also a *language-tagged datatype* as names can be translated and transliterated.

2.3 Subtypes

A *datatype* MAY be defined as a **subtype** of another *datatype* which is referred to as its **supertype**. This is used to provide a more specific version of a more general *datatype*. The *lexical space* of the *subtype* SHALL be a subset of the *lexical space* of the *supertype*, and if an application is unfamiliar with the *subtype* it MAY process it as if it were the *supertype*. The *subtype* MUST be defined in such a way that at most this results in some loss of meaning but does not introduce any false implications about the dataset.

> *Note* — This does not require a *subtype* to define a *pattern* if the *supertype* does. Because the *lexical space* of the *subtype* MUST be a subset of that of the *supertype*, the *pattern* of the *supertype* may be used if the *subtype* does not define one. This might be done if additional restrictions made on *lexical space* of the *subtype* cannot readily be expressed using a regular expression.

> *Note* — It is only the *lexical space* of the *subtype* that is required to be a subset of the *lexical space* of the *supertype*. The set of *strings* that match the *pattern* of the *subtype* might not necessarily be a subset of that of the *supertype*. This is because the *pattern* is permitted to match *strings* outside the *lexical space*, as in the example of the date "`1999-02-31`".

A *datatype* MAY be defined to be a **abstract datatype**. An *abstract datatype* is one that MUST only be used as a *supertype* of other types. A *string* MUST NOT be declared to have a *datatype* which is an *abstract datatype*. *Abstract datatypes* MAY specify a *pattern* and SHALL have a *lexical space*.

> *Note* — The *lexical space* of an *abstract datatype* and any *pattern* defined on it serve to restrict the *lexical space* of all its *subtypes*. If no such restriction is desired, the *lexical space* may be defined as the space of all *strings*.

Subtypes may be defined of *language-tagged datatypes* as well as of other *datatypes*. If the *supertype* is a *language-tagged datatype* then the *subtype* MUST also be; and if the *supertype* is not a *language-tagged datatype* then the *subtype* MUST NOT be.

> *Note* — The concept of a *subtype* in this standard corresponds to XML Schema's concept of derivation of a simple type by restriction per §3.16 of [XSD Pt1]. XML Schema does not have concept compatible with this standard's notion of an *abstract datatype*, as in XML Schema only complex types can be abstract. If it is desirable to describe a FHISO *abstract datatype* in XML Schema, it should be defined as a normal simple type, with the information that it is abstract conveyed by another means.

2.4 Built-in datatypes

This standard gives special treatment to three *datatypes* defined in third-party standards.

2.4.1 The `rdf:langString` datatype

Any *language-tagged datatype* that is not defined to be a *subtype* of some other *datatype* SHALL implicitly be considered to be a *subtype* of the `rdf:langString` datatype defined in §2.5 of [RDFS]. This *datatype* is an *unstructured language-tagged datatype* and has the following properties:

Name	`http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`
Pattern	`.*`
Supertype	*none*
Abstract	no

> *Note* — Although this type is formally defined in the RDF Schema specification, this standard requires no knowledge of RDF; an implementer may safely use this *datatype* using just the information given in this section, and without reading [RDFS].

No constraints are placed on the *lexical space* of this *datatype*; the only restriction placed on the use or semantics of this *datatype* is that it SHOULD contain text in a human-readable form.

> *Note* — This type is the ultimate *supertype* of all *language-tagged datatypes*. This standard does not specify a comparable *datatype* which acts as the ultimate *supertype* of all *non-language-tagged datatypes*, nor of all *datatypes*.

> *Editorial note* — Possibly one or more of the `rdfs:Resource`, `rdfs:Literal`, `xsd:anyType`, `xsd:anySimpleType` and `xsd:anyAtomicType` would serve, but this needs careful consideration of the differences between *datatypes* in XML Schema, RDF and this standard. At present there is no compelling need for either of these additional *supertypes*.

2.4.2 The `xsd:string` datatype

This standard makes limited use of the `xsd:string` *datatype* defined in §3.3.1 of [XSD Pt2]. This is an *unstructured non-language-tagged datatype* which has the following properties:

Name	`http://www.w3.org/2001/XMLSchema#string`
Pattern	`.*`
Supertype	*none*
Abstract	no

It is a general-purpose *datatype* whose lexical space is the space of all *strings*; however it is not a *language-tagged datatype* and therefore it SHOULD NOT be used to contain text in a human-readable natural language.

> *Note* — This type is not the ultimate *supertype* of all *non-language-tagged* datatypes. This is because many other XML Schema *datatypes*, including `xsd:date` and `xsd:integer` are not defined as *subtypes* of `xsd:string` in XML Schema.

Use of this *datatype* is generally NOT RECOMMENDED: data that is in a human-readable form SHOULD use a *language-tagged datatype*, while data that is not human-readable SHOULD use a *structured datatype*.

If an application encounters a *string* with the `xsd:string` *datatype*, it MAY change the *datatype* to `rdf:langString` and assign the *string* a *language tag* of und, meaning an undetermined language.

> *Note* — The `xsd:string` *datatype* is included in this standard in order to align this data model more closely with the RDF data model, and in particular the [CEV RFDa] bindings which use this *datatype* as the default when no *language tag* is present. The above rule allowing conversion to `rdf:langString` means that applications MAY ignore the `xsd:string` *datatype*.

2.4.3 The `rdfs:Resource` datatype

This standard also makes use of the `rdfs:Resource` type defined in §2.1 of [RDFS] as the class of everything that can be expressed in RDF. In these Citation Elements standards, its use is more specific, and it is used as a *datatype* to represent resources identified by IRIs. In this context a resource might be a document or file that can be retrieved from that IRI, but it also includes an physical and abstract concept that are merely identified by an IRI.

> *Example* — The `rdfs:Resource` *datatype* is used to represent the website from which an online *source* can be retrieved.

> *Note* — The fact that FHISO is using `rdfs:Resource` in a more specific manner than RDF does not introduce a incompatibility between RDF and this FHISO standard. This is because, in RDF terminology, `rdfs:Resource` is not a datatype but something more general. All literals in RDF have a datatype, but IRIs are a distinct class of entity which do not have an RDF datatype. Instead the thing they represent has a type and `rdfs:Resource` is the most general possible type. A further complication is that all RDF datatypes are also subclasses of `rdfs:Resource`, but as `rdfs:Resource` is not itself an RDF datatype, it cannot appear in contexts where an RDF datatype is expected. For the purpose of FHISO's Citation Elements standards, the `rdfs:Resource` *datatype* is not a *supertype* of any other *datatype*.

The *lexical space* of thus *datatype* is the space of valid IRIs matching the `IRI` production in §2.2 of [RFC 3987]. It is a *non-language-tagged datatype* with the following properties:

Name	`http://www.w3.org/2000/01/rdf-schema#Resource`
Pattern	`[a-z][a-z0-9+.-]+:[^ ]+`
Supertype	*none*
Abstract	no

> *Editorial note* — This *pattern* almost certainly needs revising.

Applications MUST NOT define *subtypes* of `rdfs:Resource`.

> *Note* — This restriction is likely to be removed in a future version of this standard. If and when *subtypes* of `rdfs:Resource` are permitted, they will be almost certainly be used to describe the type of resource being referenced, rather than the type of IRI used to reference it. Therefore it is very unlikely that *subtypes* of `rdfs:Resource` will be permitted to define a *pattern* or further constrain the *lexical space* of the *datatype*.

> *Editorial note* — A future draft of this standard may clarify its relationship with the `xsd:anyURI` *datatype*. In RDF, the two are entirely unrelated as `xsd:anyURI` is used in RDF as the datatype of a literal, whereas `rdfs:Resource` is used as the type of the resource referenced by an IRI. But there may be a use case to make the two interchangeable, much as `xsd:string` is with an `rdf:langString` tagged with the *language tag* und.

2.5 Unions of datatypes

A **union of datatypes** is an unordered list of one or more different datatypes.

> *Note* — As defined in this standard, a *union of datatypes* is not itself a *datatype* as it lacks a *term name* to identify it, may not have a *pattern*, and cannot be used as a *subtype* or *supertype*. This is just a matter of nomenclature, and a future version of this standard might broaden the definition of a *datatype* to allow a *union* to be a *datatype*.

A *union of datatypes* MAY contain *language-tagged datatypes*, *non-language-tagged datatypes*, or a mixture of both.

The **lexical space** of a *union of datatypes* is the union of the *lexical space* of each of its constituent *datatypes*.

> *Note* — There is no requirement that the *lexical spaces* of each constituent *datatype* be disjoint.

> *Example* — *Unions of datatypes* are used as the *range* of *citation elements*, as defined in §4.2. In several cases a *union* of the following two *datatypes* is used:

> `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`
> `https://terms.fhiso.org/dates/AbstractDate`
>
> The former is an *unstructured datatype*, while the latter is an *abstract datatype* which serves as the *supertype* for various *structured datatypes* for dates. The inclusion of an *abstract datatype* provides a point of extensibility.

3 Citations elements

In the data model defined by this standard, a *citation element* consists of two parts, both of which are REQUIRED:

— a name, called the *citation element name*; and
— a value, called the *citation element value*.

A *citation element set* is defined to be an ordered list of *citation elements*; *conformant* applications MAY reorder the list subject to the following constraints:

— The relative order of *citation elements* must be preserved when they have the same *ultimate super-element* (as defined in §4.1 of this standard).
— When a *citation element set* contains a *citation element* with the *citation element name* `https://terms.fhiso.org/sources/localisedElement`, the previous element in *citation element set* with a different *citation element name* is referred to as its **localisation base**. The *localisation base* of any `localisedElement` *citation element* must not change if a *citation element set* is reordered.

> *Note* — The latter requirement can be avoided by processing `localisedElements` per §4.3.1 of this standard, and then removing them from the *citation element set*.

> *Note* — Subject to these constraints, this standard allows *citation element sets* to be reordered because some serialisation languages such as JSON and RDF do not guarantee to preserve the order of elements in certain important serialisation mechanisms: for example, object members in JSON and triples in RDF other than when RDF containers are used.

> *Editorial note* — The special `localisedElement` *citation element term* and the notion of its *localisation base* were called `translatedElement` and its *translation base* in earlier drafts of this standard.

3.1 Citation element names

The **citation element name** identifies the nature of the information contained in a particular *citation element*. It SHALL be a *term* that has been defined to be used as a *citation element name* in the manner required by §4 of this standard; a *term* defined for this purpose is called a *citation element term*.

> *Note* — This nomenclature draws a distinction between a *citation element name* and a *citation element term*. The former is part of a *citation element* and therefore part of the data describing a *source*, while the latter is an item of vocabulary used in the description. The *citation element name* is a *citation element term*.

> *Example* — The [CEV Vocabulary] defines a *citation element term* for the title of a *source*. Its *term name* is:
>
> `https://terms.fhiso.org/sources/title`
>
> A dataset might contain many *citation elements* with this as their *citation element name*.

3.2 Citation element values

The **citation element value** is the content of the *citation element* which SHALL be a *localisation set*. A **localisation set** is an ordered list of *strings*, which applications SHOULD *whitespace-normalise*. Each *string* in a *localisation set* SHOULD contain the same information, but translated, transliterated or otherwise localised.

Each *string* in a *localisation set* SHALL be tagged with a *datatype*, and SHALL additionally be tagged with a *language tag* if and only if the specified *datatype* is a *language-tagged datatype*. The *language tag* SHALL match the `Language-Tag` production from [RFC 5646], and SHOULD contain a script subtag per §2.2.3 of [RFC 5646] when transliteration has occurred.

> *Note* — Most often a *localisation sets* will contain only a single *string*, either because localisation is not relevant to that particular *citation element*, as might be the case with a straightforward page number, or because the creator of the *localisation set* only provided the particular version the user was expected to require. If more than one *string* is present, usually they will all have the same *datatype* and differ only in their *language tags*. Nevertheless, the mechanism allows for *strings* of different *datatypes* and there are rare situations where this functionality is needed.

> *Editorial note* — In the first public draft of this standard, instead of *localisation sets* there were *translation sets*, which were lists of *strings* each tagged with a *language tag*; there was no explicit notion of a *datatype*; and *citation element values* were either a *translation set* or a single *string*. In the new terminology this said *localisation sets* had to be homogenous, i.e. they had to have a single *datatype*.

> *Example —* The `title` *citation element* defined in the [CEV Vocabulary] would normally contain *strings* tagged with the `rdf:langString` *datatype*. An example `title` *citation element* might contain a *localisation set* with three `rdf:langString` *strings* in the following order:
>
> — the original title "`Η Γενεαλογία των Κομνηνών`" with *language tag* `el`, the language code for Greek in [ISO 639-1];
> — a transliteration, perhaps supplied algorithmically, with the value "`Hē Genealogia tōn Komnēnōn`" and *language tag* `el-Latn`, `Latn` being the code for the Latin script in [ISO 15924]; and
> — a French translation, "`La généalogie des Comnènes`", tagged with the language code `fr`.

3.2.1 Serialisation considerations

Although the *language tags* is REQUIRED for *language-tagged datatypes*, it need not be explicit in the serialisation. A serialisation format MAY provide a mechanism for stating the document's default *language tag*, and MAY provide a global default which SHOULD be a language-neutral choice such as und, defined in [ISO 639-2] to mean an undetermined language. In the absence of an explicit or implicit *language tag*, applications MUST NOT apply their own default, and MUST treat the *string* as if it had the *language tag* und.

> *Example —* The [CEV RDFa] standard provides a means for *citation elements* to be extracted from HTML, and uses HTML's `lang` attribute to provide a default *language tag* for the document or a part of the document. Thus, if the document begins `<html lang="pt_BR">`, it is not necessary to tag each *string* separately for them to be understood to be in Brazilian Portuguese. HTML does not define a default *language tag* that applies in the absence of a `lang` tag, and applications MUST NOT apply one.

If *localisation sets* are being serialised in XML, it is RECOMMENDED that the special `xml:lang` attribute defined in §2.12 of [XML] is used to encode the *language tag*.

Similarly, a *datatype* is REQUIRED, but it need not be explicit in the serialisation. A serialisation format MAY specify a *format default datatype* that applies when none is given explicitly. Ordinarily, if a *format default datatype* is specified, it SHOULD be the `rdf:langString` *datatype* defined in §2.4.1 of this standard.

> *Note* — This is called the *format default datatype* to avoid confusion with the *default datatype* defined per *citation element term* in §4.4 The *format default datatype* SHOULD be a *language-tagged datatype* to ensure that any *language tag* that is in the scope is retained in the data model, and as the most general *language-tagged datatype*, `rdf:langString` is RECOMMENDED. The *datatype correction* mechanism defined in §4.4 of this standard allow a *conformant* application to correct the *datatype* that have incorrectly defaulted to `rdf:langString`. In practice it is anticipated that many applications will apply *datatype correction* during import, and therefore the *format default datatype* becomes a fallback

> that applies if the *citation element term* does not define its own *default datatype*, or if this is unknown.

> *Example* — The [CEV RDFa] standard makes `rdf:langString` the *format default datatype* in most circumstances. Thus the *citation element* extracted from the following HTML fragment is interpreted as an `rdf:langString` *string*, even though it is not explicitly tagged as such:
>
> ```
> The Complete Peerage
> ```

3.2.2 Reordering, deduplicating and merging

Where possible, the first *string* in the *localisation set* SHOULD be the untranslated, and ideally untransliterated form of the *citation element value*. If it is known that the only available values are translations, the first *string* in the *localisation set* SHOULD be an empty string tagged with the *language tag* und, and the translations listed afterwards. An empty *string* in a *localisation set* means that its value is unknown, rather than that this particular translation is literally an empty string.

Conformant applications MAY reorder the *localisation set*, but MUST leave the first *string* first, so that applications wishing to use the original, untranslated, untransliterated form can do so.

> *Note* — A standard MAY define a serialisation format that does not preserve the order of a *localisation set*, but MUST take alternative steps to record the original version. For example, the language map in [JSON-LD] is very similar to a *localisation set* containing only `rdf:langString` *strings*, except that JSON's object notion, as given in §4 of [RFC 7159], does not preserve order. One possible solution is to append some private use subtag (per §2.2.7 of [RFC 5646]) to the first *language tag*.

In a *localisation set* which contains more than one *string* with the same *datatype* and *language tag*, or more than one *string* with the same *datatype* if it is a *non-language-tagged datatype*, any *string* other than the first non-empty *string* with that *datatype* and, if relevant, *language tag* is known as a **duplicate string**.

If an application encounters a *localisation set* with *duplicate strings*, it SHOULD ignore the value of any *duplicate strings* and MAY *deduplicate* the *localisation set*; where possible it SHOULD NOT *deduplicate* a *localisation set* that has been reordered from its serialised form.

> *Editorial note* — During feedback on the first public draft, concerns were expressed over whether *duplicate strings* might a necessary to express certain concepts; if so, they mustn't be ignored or *deduplicated*. Examples of where they might be needed are pseudonyms, places with multiple names, multiple page numbering systems, and dates with multiple prose forms. This requires further consideration.

To **deduplicate** a *localisation set*, the application first notes the *datatype* and, if present, the *language tag* of the first *string* in the *localisation set*. Next, all *duplicate strings* are deleted from the *localisation set*. Finally, if a *string* with the noted *datatype* and *language tag* remains after *deduplication*, the appli-

cation SHALL reorder the *localisation set* to ensure it is the first *string* in the *deduplicated localisation set*; if there is not, the application shall insert any empty *string* with that *datatype* and *language tag* as the first *string* in the *localisation set*.

If an application needs to **merge** two or more *localisation sets*, the contents of each *localisation sets* SHALL be combined in the order specified by this standard, and the application SHOULD *deduplicate* the resultant *localisation set*.

> *Note* — *Merging* of *localisation sets* only occurs as the result of the *deduplication* of *citation element sets* per §4.3. It specifies the *localisation sets* are merged in the order they appear in the *citation element set*.

If a *citation element* has a *citation element name* which is an empty *localisation set*, that *citation element* SHOULD be discarded.

> *Note* — This can occur as the result of removing *invalid strings* from a previously non-empty *localisation set*, as explained in §4.2.

4 Defining citation element terms

A **citation element term** is a *term* which has been defined specifically for use as a *citation element name* in the following manner. The party defining the *citation element term* SHALL provide a description of the intended purpose of the *citation element term* which SHOULD be made freely available to all interested parties, preferably by an HTTP request as described in §1.4.2 of this standard. In addition, the definition SHALL state:

- its *term name* (an IRI);
- whether it is a *sub-element* of some other *citation element term*, and if so which one, as defined in §4.1;
- its *range*: the *datatype* of its value space, as defined in §4.2;
- its *cardinality*: that is, whether it is *single-valued* or *multi-valued*, as defined §4.3; and
- an OPTIONAL *default datatype*, as defined in §4.4.

> *Editorial note* — Earlier drafts of this standard required the definition additionally to state whether the *citation element* was *translatable*. Now that all *citation element values* are *localisation sets* this concept is redundant, though whether the *range* is a *language-tagged datatype* serves a similar purpose.

4.1 Sub-elements

A *citation element term* MAY be defined as a **sub-element** of another *citation element term* which is referred to as its **super-element**. This is used to provide a refinement of a general *citation element term*. If an application is unfamiliar with the *sub-element* it MAY process it as if it were the *super-element*, with its *citation element value* unchanged. The *sub-element* must be defined in such a way that this only results in some loss of meaning, and does not imply anything false about the cited *source*.

> *Example —* The [CEV Vocabulary] defines a *citation element term* with the name
>
> `https://terms.fhiso.org/sources/creatorName`
>
> which contains name of a person, organisation or other entity who created or contributed to the creation of the *source*. Several *sub-elements* of it are defined, including
>
> `https://terms.fhiso.org/sources/interviewerName`
>
> which contains the name of an interviewer when the *source* is an interview. An interviewer can certainly be considered to have contributed to the creation of the interview.
>
> The [CEV Vocabulary] also defines a *citation element* with the name
>
> `https://terms.fhiso.org/sources/recipientName`
>
> which contains the party to whom a *source* such as a letter is addressed. In many respects it is similar to the *sub-elements* of `creatorName`, but because a recipient of a letter cannot be said to have contributed to the creation of the letter, and might not even be aware of its existence if it were not delivered, the `recipientName` element cannot be defined as a *sub-element* of `creatorName`.

The *range* of a *sub-element* SHALL be the same as that of its *super-element*.

> *Editorial note —* The *range* of a *sub-element* could be allowed to be a *subtype* of the *super-element's range*, as defined in §2.3 of this standard. At the moment there is no clear use case for this.

Any *sub-element* of a *single-valued super-element* MUST be *single-valued*.

The **super-element list** of a *citation element term* is an ordered list of IRIs defined inductively as follows. If the *citation element term* is not a *sub-element*, then its *super-element list* contains just that *citation element term*. Otherwise, its *super-element list* is the *super-element list* of its *super-element* to which its own *citation element term* is appended.

The **ultimate super-element** of a *citation element term* is defined as the first IRI in its *super-element list*.

> *Note* — This definition is equivalent to following the (possibly empty) chain of *super-elements* until it reaches something that is not a *sub-element*. It is used in specifying how applications are permitted to reorder *citation element sets*.

The **ultimate single-valued super-element** of a *single-valued citation element term* is defined as the first IRI in its *super-element list* that is a *single-valued citation element term*.

> *Note* — This definition is equivalent to following the (possibly empty) chain of *super-elements*, stopping at the last *single-valued* element in the chain. It is used in specifying the constraints on *sub-elements* that are *single-valued*.

The **most-refined common super-element** of a collection of *citation element terms* is defined as the last IRI that appears in the *super-element list* of every *citation element term* in the collection. It is only defined for *citation element terms* that share an *ultimate super-element*.

> *Note* — This definition is equivalent to following the chains of *super-elements* for each *citation element terms*, stopping at the first element that appears in each chain. It is used in specifying how to *merge citation elements*.

4.2 Range

The **range** of a *citation element term* is a *union of datatypes*, which describes what *citation element values* are valid in a *citation element* with this *citation element name*.

> *Example* — The [CEV Vocabulary] defines a *datatype* for representing the names of authors and other people, which has the following *term name*:
>
> `https://terms.fhiso.org/sources/AgentName`
>
> A *union of datatypes* consisting of just this one *datatype* is used as the *range* of several *citation element terms* defined in the [CEV Vocabulary] including:
>
> `https://terms.fhiso.org/sources/editorName`

Citation elements terms with non-textual *citation element values* such as numbers or dates SHOULD have *ranges* that include one or more *non-language-tagged datatype*.

> *Example* — The [CEV Vocabulary] defines an *abstract datatype* called `AbstractDate` which is used as the *supertype* of all *structured datatypes* for dates; it has the following *term name*:
>
> `https://terms.fhiso.org/dates/AbstractDate`
>
> Several *citation element terms* have a *range* consisting of a *union* of `AbstractDate` and `rdf:langString`. One such *citation element term* is:
>
> `https://terms.fhiso.org/sources/publicationDate`

> Because this *citation element* typically has non-textual *values*, frequently just a year, its *range* SHOULD include a *non-language-tagged datatype*: in this case, `AbstractData`. The inclusion of `rdf:langString` is to allow dates that cannot readily be represented in any of the available structured formats. An example might be a termly university publication dated “Michaelmas term, 1997”.

> *Editorial note —* The previous example may need revising once FHISO’s handling of date types has been finalised. In particular, the IRI of `AbstractDate` has not been discussed yet.

A *datatype* is said to be **compatible** with the *range* if either it is one of the *datatypes* listed in the *range*, or it is a *subtype* of a *datatype* that is *compatible* with the *range*.

> *Note* — This recursive definition of *compatibility* means that the *datatype* may be an indirect *subtype* (e.g. a *subtype* of a *subtype*) of one of the *datatypes* in the *range*.

A *string* in a *localisation set* which is used as a *citation element value* is said to be **invalid** if, after *datatype correction* has occurred per §4.4 of this standard, either the *string* is tagged with a *datatype* that is not *compatible* with the *range* of the *citation element term* used as the *citation element name*, or the *string* is outside the *lexical space* of that *datatype*. *Conformant* application SHOULD take steps to avoid creating *localisation sets* containing *invalid strings*.

> *Note* — An application might inadvertently create *invalid strings* if it does not know the *range* of a *citation element term* or does not properly understand the *lexical space* of some of the *datatypes* within that *range*. Applications MAY use the *pattern* of the *datatype* to identify some *strings* outside the *lexical space* of the *datatype* as a *string* that fails to match the *pattern* is guaranteed not to be in the *lexical space*; applications MAY also use deeper knowledge of the *lexical space* to identify more *invalid strings*.

Applications MAY use one or more *discovery* mechanism to obtain the information needed to determine which *strings* are *invalid*.

> *Note* — In order to determine whether a *datatype* is *compatible* with the *range*, the application will need to know or have access to the definition of the *datatype* and any *supertypes* to determine whether it is a *subtype* of a *datatype* listed in the range, as well as having access to definition of the *citation element term* to determine the *range*.

If the *range* of the *citation element term* includes one of the following *datatypes*, applications SHOULD change the *datatype* of the *invalid string* to that *datatype*:

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#langString
http://www.w3.org/2001/XMLSchema#string
```

If the *range* does not include either of these *datatypes*, applications MAY discard any *strings* that are found to be *invalid*. It is RECOMMENDED that this SHOULD be done prior to *deduplicating* a *localisation*

set, and it MAY be done at other times. A *conformant* application MUST NOT discard any *string* unless it is known to be *invalid* or as otherwise permitted by this standard.

Exceptionally, a *conformant* application MAY also discard any *string* which it has credible reason to believe contains malware or illegal content, or any *string* that is so long that the application cannot reasonably handle it.

> *Example* — An application might opt to discard all *strings* that appear to be Windows executables.

4.3 Cardinality

The **cardinality** of a *citation element term* records how many semantically distinct values it can have. A **multi-valued** *citation element term* is one that can logically have multiple values in a single *citation layer*. It SHOULD be reserved for situations where the values genuinely contains different information, and not used to accommodate transliterations, translations, or variant forms of something that is logically a single value. *Citation elements terms* that are not *multi-valued* are **single-valued**.

> *Example* — The `https://terms.fhiso.org/sources/title` *citation element term* is defined to be *single-valued*, as *citations* do not refer to the same *sources* by multiple titles (though they may translate or transliterate the title), so a *citation element set* MUST NOT contain more than one *citation element* with this *citation element name*; but it MAY contain several `https://terms.fhiso.org/sources/authorName` *citation elements*, as that is defined to be *multi-valued* to accommodate *sources* with several authors.

In a *citation element set* which contains more than one *citation element* whose *citation element names* have the same *ultimate single-valued super-element*, any *citation element* other than the first *citation element* with that *ultimate single-valued super-element* is known as a **duplicate citation element**.

> *Note* — *Citation element terms* that are declared as *multi-valued* do not have an *ultimate single-valued super-element* and are therefore never *duplicate citation elements*.

Citation element sets SHOULD NOT contain *duplicate citation elements*, and an application SHOULD take steps to avoid creating *duplicate citation elements*.

> *Note* — An application might inadvertently create *duplicate citation elements* if it does not know the *super-element* or *cardinality* of some of *citation element terms*.

When *duplication citation elements* are present, they MAY be *deduplicated*. To **deduplicate** a *citation element set*, the application SHOULD replace all the *citation elements* with a common *ultimate single-valued super-element* with a single replacement *citation element* with the following properties:

— a *citation element name* which SHALL be the *most-refined common super-element* of the *citation element terms* being replaced; and

— a *citation element value* which SHALL be a *localisation set* created by *merging* the *localisations sets* of each *citation element value* being replaced in the order they appear in the *citation element set*.

> *Example* — Consider the following *citation element set*, written in a hypothetical JSON format:
>
> ```
> [ "title": [ "fr": "Les ancêtres des Charlemagne",
>              "en": "The Ancestors of Charlemagne" ],
>   "title": [ "fr": "Les Ancêtres des Charlemagne",
>              "de": "Die Vorfahren von Karl dem Großen" ] ]
> ```
>
> Assuming the `title` *citation element term* is *single-valued*, an application MAY *deduplicate* the *citation element set* by *merging* the two *localisation sets* in order to get the following:
>
> ```
> [ "title": [ "fr": "Les ancêtres des Charlemagne",
>              "en": "The Ancestors of Charlemagne",
>              "fr": "Les Ancêtres des Charlemagne",
>              "de": "Die Vorfahren von Karl dem Großen" ] ]
> ```
>
> After *merging* the *localisation sets*, §3.2.2 says the application SHOULD *deduplicate* the resultant *localisation set*. This removes the second French title to give the following:
>
> ```
> [ "title": [ "fr": "Les ancêtres des Charlemagne",
>              "en": "The Ancestors of Charlemagne",
>              "de": "Die Vorfahren von Karl dem Großen" ] ]
> ```
>
> These rules mean that *single-valued citation elements* with the same *ultimate single-valued super-element* (in this example, with the same *citation element name*) are assumed to be given in order of preference for the purpose of *deduplicating* the *merged localisation set*, with the most preferred value first.

> *Note* — There is no requirement for an application to check for *duplicate citation elements* and *deduplicate* them; however it might be advisable for an application to do so when importing third-party data, or if it has recently learnt of new *extension citation elements* which are *single-valued*.

> *Editorial note* — This standard needs to define how to merge *citation element sets*. The following text is a start towards that.
>
> > If an application needs to **merge** two or more *citation element sets*, the contents of each *citation element set* shall be combined in order. The application SHALL identify any sets of *duplicate citation elements* in the combined *citation element set* and *deduplicate* them according to the rules above. An application MAY use one or more *discovery* mechanism to attempt to obtain machine-

> readable definitions of any *extension citation element* used in the *citation element set* before identifying *duplicate citation elements*.
>
> However the merger of *multi-valued* elements requires thought too. Even though the data model doesn't require deduplication, it is still necessary to prevent duplication of, say, authors.

4.3.1 List-flattening formats

Conformant applications MUST ensure that in *citation elements* whose *citation element names* are *multi-valued*, the *localisation set* in each *citation element value* remains separate.

> *Example* — The `authorName` *citation element term* is defined to be *multi-valued* because a source may have multiple authors, and each of them may have names that have been transliterated into different scripts. Suppose a researcher wants to cite the Anglo-Japanese Treaty document of 1902 which was (at least nominally) authored by the Marquess of Lansdowne and Count Hayashi Tadasu whose name is written in kanji as 林 董.
>
> The following hypothetical JSON serialisation is not allowed as it flattens *localisation sets* so it is no longer possible to determine how many authors there are, and which names are translations of which others.
>
> ```
> [ { "name": "https://terms.fhiso.org/terms/title",
>     "lang": "en",      "value": "The Anglo-Japanese Treaty" },
>   { "name": "https://terms.fhiso.org/terms/authorName",
>     "lang": "en",      "value": "Lord Lansdowne" },
>   { "name": "https://terms.fhiso.org/terms/authorName",
>     "lang": "jp",      "value": "林 董" },
>   { "name": "https://terms.fhiso.org/terms/authorName",
>     "lang": "jp-Latn", "value": "Hayashi Tadasu" } ]
> ```
>
> In this example, the *datatype* of each string has been omitted on the assumption that it defaults to `rdf:langString` and is corrected via the mechanism specified in §4.4 of this standard.
>
> This is an example of a *list-flattening format* that does not conform to this specification; a *list-flattening format* that does conform to this specification is found in the next example.

A serialisation format that does not keep the *localisation sets* of each *citation element value* separate is called a **list-flattening format**, and this standard provides a facility to allow such formats to comply with this standard by introducing a special *citation element term* with the following properties:

Name	`https://terms.fhiso.org/sources/localisedElement`
Range	*unspecified*
Cardinality	multi-valued
Super-element	*none*
Default datatype	*none*

> *Note* — This `localisedElement` *citation element term* has no *range* specified. No other *citation element terms* defined in accordance with this standard may have an unspecified *range*.

In a *list-flattening format*, an application MUST consider every value to be a separate *citation element value*, and therefore to be a *localisation set* with one element.

> *Note* — More often than not this assumption is expected to be valid, as more often than not *citation element sets* are expected not to include translated or transliterated elements.

When a *localisation set* with two or more *strings* needs to be serialised in a *list-flattening format*, the first *string* MUST be serialised according to the normal rules of the format, and subsequent *strings* MUST be serialised as if they were separate *citation element*, but with the `localisedElement` *citation element term* in place of the actual *citation element name*. This special *citation element* indicates that its value is not a distinct *citation element* and SHOULD instead be appended to the *localisation set* of its *localisation base* (i.e. the last preceding *citation element* which is not a `localisedElement`), and the `localisedElement` removed from the *citation element set*.

> *Example* — The hypothetical JSON serialisation in the last example can be fixed by using a `localisedElement` to serialise the transliterated version of Hayashi's name:
>
> ```
> [ { "name": "https://terms.fhiso.org/terms/title",
>     "lang": "en",      "value": "The Anglo-Japanese Treaty" },
>   { "name": "https://terms.fhiso.org/terms/authorName",
>     "lang": "en",      "value": "Lord Lansdowne" },
>   { "name": "https://terms.fhiso.org/terms/authorName",
>     "lang": "jp",      "value": "林 董" },
>   { "name": "https://terms.fhiso.org/terms/localisedElement",
>     "lang": "jp-Latn", "value": "Hayashi Tadasu" } ]
> ```
>
> The two `authorName` element are assumed to be separate *citation elements* and therefore to refer to different authors. The use of `localisedElement` signifies that this is not a different author. It immediately follows an `authorName` *citation element* with the value 林 董, and its value ("Hayashi Tadasu", tagged as `jp-Latn`) should be appended to that *localisation set*.

> *Note* — This standard does not say when the processing of `localisedElements` occurs. Ideally an application SHOULD do it during the process of reading a *list-flattening format*,

> but MAY do it later or not at all. If the application subsequently serialise the data in a non-*list-flattening format*, the `localisedElements` MAY still be present. Therefore applications reading non-*list-flattening format* SHOULD cope with the possibility of `localisedElements` being present.

If the *localisation set* in the *localisation base* already contains a string with the same *datatype* and *language tag*, an application MUST NOT overwrite or duplicate a *language tag*; the `localisedElement` SHOULD be ignored and MAY be removed from the the *citation element set*.

The use of *list-flattening formats* is NOT RECOMMENDED except where there is a good technical reason. The use of `localisedElements` other than in *list-flattening formats* is NOT RECOMMENDED.

4.4 Default datatypes

> *Editorial note —* The concept of a *default datatype* is new in this draft of the standard.

A *citation element term* MAY have a **default datatype** defined. When a *default datatype* is defined, it is used to provide an OPTIONAL **datatype correction** mechanism for correcting the *datatype* of a *string* in the *localisation set* of a *citation element value* in certain situations. The *default datatype* MUST be a *datatype* that is *compatible* with the *range* of the *citation element term*.

Datatype correction SHALL NOT be carried out unless the *datatype* of the *string* prior to *datatype correction* is one of the following *datatypes*, and not just a *subtype* of one of them:

```
http://www.w3.org/1999/02/22-rdf-syntax-ns#langString
http://www.w3.org/2001/XMLSchema#string
http://www.w3.org/2000/01/rdf-schema#Resource
```

> *Note —* It is anticipated that a large majority of times when *data correction* applies, the original datatype will be `rdf:langString`. Support for `xsd:string` and `rdfs:Resource` is only included in this *datatype correction* mechanism to accommodate certain corner cases in RDF processing that could arise in the [CEV RDFa] bindings.

Datatype correction SHALL only be applied to a *string* if it appears in a *citation element* whose *citation element name* is a *citation element term* that has a *default datatype*, and if that *default datatype* is a *datatype* whose *pattern* is known to the application, and if the *string* matches that *pattern*.

At any time when an application encounters a *string* which is eligible for *datatype correction* according to the above criteria, it MAY replace its *datatype* with the *default datatype*. It is RECOMMENDED that applications apply *datatype correction* during or shortly after the import of data in any serialisation format that defines a *format default datatype* of `rdf:langString`.

> *Note —* This standard does not limit when *datatype correction* occurs, and it MAY be desirable to apply it at times other than as *recommended above*. If an application exports an unknown *citation element* in a format that does not have a *format default datatype*, this

may result in explicit *datatypes* that still need *datatype correction*. Ideally, therefore, applications SHOULD cope with the possibility that *datatype correction* might be needed on any data being imported. Likewise, when an application gains access to the definitions of additional *citation element terms* or *datatypes*, this might allow it to identify further places where *datatype correction* is required. However, the only situation when *datatype correction* is REQUIRED by this standard is immediately prior to the removal of *invalid strings*, which process is itself OPTIONAL.

Example — The hypothetical JSON format used in several earlier examples included the following *citation element*:

```
[ { "name": "https://terms.fhiso.org/terms/authorName",
    "lang": "jp", "value": "林 董" } ]
```

This hypothetical format is supposed to default datatypes to `rdf:langString`, as RECOMMENDED by this standard.

The `authorName` *citation element* is defined in the [CEV Vocabulary] to have the following *default datatype*:

```
https://terms.fhiso.org/sources/AgentName
```

This *datatype* in turn defines the following *pattern*:

```
([^!#$%&@{|}]+@)?[^!#$%&@{|}]+(\|[^!#$%&@{|}]*(\|[^!#$%&@{|}]+)?)?
```

The *string* "林　董" matches this *pattern* — specifically it matches the second `[^!#$%&@{|}]+` part of the *pattern* — and therefore the *datatype correction* will change the *datatype* to this `AgentName` *datatype*.

Editorial note — The *pattern* quoted above for `AgentName` will almost certainly need changing as the `AgentName` *datatype* is properly specified.

Example — The `publicationDate` *citation element term* defined in the [CEV Vocabulary] has a *range* which is the *union* of the `AbstractDate` and `rdf:langString` *datatypes*; its *default datatype* is `GregorianDate`, a *subtype* of `AbstractDate` with the following *pattern*:

```
-?[0-9]{4,}(-(0[1-9]|1[0-2])(-(0[1-9]|[12][0-9]|3[01]))?)?
```

A *citation element set* might contain a `publicationDate` *citation element* whose *localisation set* contains the following two *strings*, both tagged with the *language tag* `en` and *datatype* `rdf:langString` (presumably implicitly as the result of no *datatype* being given in the serialisation):

```
Michaelmas term, 1997
1997-10
```

> The former *string* is not remotely close to matching the *pattern* for the `GregorianDate` *datatype*, so it is unaffected by *datatype correction*; however the latter *string* does match the *pattern* and so *datatype correction* MAY change its *datatype* to `GregorianDate`.
>
> This is an example of where a *localisation set* might usefully contain both *language-tagged datatypes* and *non-language-tagged datatypes*. The former gives the date in the correct form for inclusion in a *formatted citation*, while the latter allows an application to parse the date, for example to highlight contemporary sources to a user.

> *Editorial note* — FHISO's handling of dates is still very much unspecified, and in the present draft the preceding example should not be considered to be anything more than a hypothetical example containing situations in which *datatype correction* variously succeeds and fails. In particular, no decision has been taken on whether there even will be a `GregorianDate` *datatype*, let alone whether it is actually the *default datatype* of the `publicationDate` *citation element term*. If such a *datatype* is specified, it is unlikely to have precisely the *pattern* given above. Nevertheless, it is safe to assume that this *citation element term* will have a *default datatype* that is some *structured datatype* for dates.

Matching the *pattern* of a *datatype* does not guarantee the *string* necessarily belongs to the *lexical space* of that *datatype*, so it is possible that *data correction* might turn a valid *unstructured string* into an *invalid string*. An application SHOULD NOT perform *data correction* when it knows the result would be an *invalid string*.

> *Note* — The mechanism for handling *invalid strings* in §4.2 means that any *invalid string* that is inadvertently created as a result of this will be converted back to an `rdf:langString` or `xsd:string` rather than being discarded.

Applications SHOULD try to ensure that no *strings* are entered which match the *pattern* of the *default datatype* but are outside its *lexical space*. One strategy for ensuring this is to suggest an alteration to the *string* that would prevent it from matching the *pattern*; however applications MUST NOT make such an alternation other than at the instruction of the user.

> *Example* — The *string* "`1999-02-31`" matches the *pattern* for a `GregorgianDate` but is nonetheless outside the *lexical space* of that *datatype* as there was no such date. A *conformant* application might warn the user that this is not a valid Gregorian date; if the user confirms they really did mean to enter an *unstructured string* that looks like an invalid Gregorian date, the application MAY alter the string to make it not match the *pattern*. One way this could be done would be appending "`(sic)`" to the string; another option is to append an invisible Unicode character such U+2060 (word joiner).

If *datatype correction* would result in replacing a *non-language-tagged datatype* with a *language-tagged datatype*, then the application MUST tag the *string* with the *language tag* und.

> *Note* — This case only applies if the *string* was previously tagged with the `xsd:string` or `rdfs:Resource` *datatypes*, which this standard discourages when the data is indeed language-tagged.

5 Layered citations

In the data model defined in this standard, a *citation layer* is represented by a *citation element set* containing the information in the *citation layer*.

A *citation* is represented with the following three parts:

— an ordered list of one or more *citation layers*, each represented as a *citation element set*;
— a marker to identify one *citation layer* as the *head citation layer*; and
— an unordered set of *layer derivation links* encoding the *source derivations* between *sources* represented by the *citation layers*.

> *Note* — This standard does not specify the precise nature of the marker that identifies the *head citation layer*. Implementation strategies include attaching a boolean flag to precisely one of the *citation layers*, storing a pointer to the data structure in memory that represented the *citation layer*, or if the *citation layers* are stored in a relational database, the value of the primary key might be used.

In the common case of a *singe-layer citation*, the set of *layer derivation links* will be empty, and the sole *citation layer* present must be the *head citation layer*. This means that a *single-layer citation* can be represented using just a *citation element set*.

Applications SHOULD NOT reorder the list of *citation layers*, other than at the request of the user. The order of the *citation layers* is an indication of the preferred order for displaying the *citation layers*, and SHOULD begin with the one considered most important. This is not necessarily the *head citation layer*. Applications MAY ignore this order when displaying or formatting *citation layers*.

> *Note* — This is not an absolute prohibition on reordering, and *conformant* applications MAY if necessary use a technology that does not preserve the order of the *citation layers*.

5.1 Layer derivation links

When the *sources* represented by two *citation layers* are linked by a *source derivation*, a **layer derivation link** is used to encode this. It has three parts, all of which are REQUIRED:

— the *derived reference*, which is a reference to the *citation layer* representing the *derived source*;
— the *base reference*, which is a reference to the *citation layer* representing the *base source*; and
— the *source derivation type*, which is an IRI used to describe the nature of the *source derivation*.

The two references to *citation layers* in the *layer derivation link* SHALL refer to *citation layers* present in the current *citation*.

> *Note* — This standard does not specify the precise form of these references, and different implementations may implement it differently. A database-backed implementation might choose to assign a identifier to each *citation layer* using an auto-increment field, and make the references a copy of that identifier. Other implementations might implement the reference using a pointer to the data structures in memory that represents the *citation layer*. Serialisation formats will define their own representations of these references.

> *Editorial note* — Earlier drafts of this standard used a *layer identifier* to represent references, but left their form unspecified and allowed implementations to use alternative implementation techniques such as pointers. The new wording is not strictly a change, but makes it clearer that a formal *layer identifier* is not required. If a persistent identifier is subsequently required for *citation layers*, it is most likely to be added as a piece of metadata in a future metadata standard.

> *Note* — The data model allows multiple *layer derivation links* between the same pair of *citation layers*. This might be used when the relationship between the *sources* cannot be represented adequately by a single *source derivation type*.

The **source derivation type** SHALL be either an IRI defined in accordance with a future FHISO standard on source derivation types, or the following IRI which represents the most general case of derivation supported in this data model:

`https://terms.fhiso.org/sources/derivedFrom`

> *Editorial note* — Should we reuse the `prov:wasDerivedFrom` or `prov:wasInfluencedBy` properties from [PROV-O] instead of inventing our own `derivedFrom` *term*?

Applications MAY discard any IRI that it knows does not conform to the above requirement.

> *Editorial note* — FHISO intends to produce a Source Derivation Vocabulary standard giving a standard vocabulary of source derivation terms, for things like transcription, abstraction, translation, indexing, referencing, analysing, commenting on and rebutting. These will be sub-types of the `derivedFrom` *source derivation type*. The Source Derivation Vocabulary standard will also provide a mechanism for third parties to provider their own **extension source derivation types**, and provide a means of determining whether a given IRI is a *source derivation type*. If this document is ready for standardisation at the same time as this document, the previous paragraph will be updated to reference it.

5.1.1 Requirements for layer derivation links

> *Note* — The representation of a *citation* in this data model is equivalent to a directed graph whose vertex set is the set of *citation layers*, and whose edge set is the set of *layer derivation links*. Each edge is labelled with its *source derivation type*, while one vertex is labelled as the *head citation layer*. This graph is called the **citation layer graph**.

A *citation layer* is **directly derived** from another *citation layer* if there exists a *layer derivation link* whose *derived reference* is to the former *citation layer* and whose *base reference* is to the latter *citation layer*. The **direct base citation layer set** of a *citation layer* is the set of *citation layers* from which the first *citation layer* is *directly derived*.

The **complete base citation layer set** of a *citation layer* is defined recursively as follows. The *citation layer* itself is part of its *complete base citation layer set*. It also contains every *citation layer* in the *complete base citation layer set* of every *citation layer* in its *direct base citation layer set*.

> *Note* — This definition is simply makes the *complete base citation layer set* the transitive closure of the *direct base citation layer set*. It contains the *citation layer* itself together with every *citation layer* from which it is derived, directly or indirectly.

The *complete base citation layer set* of the *head citation layer* SHALL contain every *citation layer* in the *citation*. If an application encounters a *citation* for which this is not the case, it MAY discard any *citation layers* that are not in the *complete base citation layer set* of the *head citation layer*.

> *Note* — This requirement says that the *head citation layer* must be derived, directly or indirectly, from every other *citation layer* in the *citation*. There MUST NOT be additional *citation layers* that are unconnected to the *head citation layer*, or which are only derived from it. In graph theory terms, this is equivalent to saying the *citation layer graph* MUST be connected, and that every *citation layer* must be reachable from the *head citation layer*. This standard does not prohibit there being additional *layer derivation links* besides those needed to ensure these conditions, and in particular does not require that the graph be acyclic.

6 References

6.1 Normative references

[ISO 10646]

ISO (International Organization for Standardization). *ISO/IEC 10646:2014. Information technology — Universal Coded Character Set (UCS).* 2014.

[ISO 15924]

ISO (International Organization for Standardization). *ISO 15924:2004. Codes for the representation of names of scripts.* 2004.

[ISO 639-1]

ISO (International Organization for Standardization). *ISO 639-1:2002. Codes for the representation of names of languages — Part 1: Alpha-2 code.* 2002.

[ISO 639-2]

ISO (International Organization for Standardization). *ISO 639-2:1998. Codes for the representation of names of languages — Part 2: Alpha-3 code.* 1998. (See http://www.loc.gov/standards/iso639-2/.)

[RDFS]

W3C (World Wide Web Consortium). *RDF Schema 1.1*. W3C Recommendation, 2014. (See https://www.w3.org/TR/rdf-schema.)

[RFC 2119]

IETF (Internet Engineering Task Force). *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.* Scott Bradner, 1997. (See https://tools.ietf.org/html/rfc2119.)

[RFC 3987]

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

[RFC 5646]

IETF (Internet Engineering Task Force). *RFC 5646: Tags for Identifying Languages.* Addison Phillips and Mark Davis, eds., 2009. (See https://tools.ietf.org/html/rfc5646.)

[RFC 7230]

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

[RFC 7231]

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

[UAX 15]

The Unicode Consortium. “Unicode Standard Annex 15: Unicode Normalization Forms” in *The*

Unicode Standard, Version 8.0.0. Mark Davis and Ken Whistler, eds., 2015. (See http://unicode.org/reports/tr15/.)

[XML]

W3C (World Wide Web Consortium). *Extensible Markup Language (XML) 1.1*, 2nd edition. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan eds., 2006. W3C Recommendation. (See https://www.w3.org/TR/xml11/.)

6.2 Other references

[CEV RDFa]

FHISO (Family History Information Standards Organisation). *Citation Elements: Bindings for RDFa*. First public draft. (See https://fhiso.org/TR/cev-rdfa-bindings.)

[CEV Vocabulary]

FHISO (Family History Information Standards Organisation). *Citation Elements: Vocabulary*. Exploratory draft.

[Chicago]

The Chicago Manual of Style, 16th ed. Chicago: University of Chicago Press, 2010.

[ECMAScript]

Ecma International. *ECMAScript® 2017 Language Specification* (ECMA-262), 8th ed. 2017. (See https://www.ecma-international.org/ecma-262/8.0/.)

[Evidence Explained]

Elizabeth Shown Mills. *Evidence Explained*, 2nd ed. Baltimore: Genealogical Publishing Company, 2009.

[ISO 8601]

ISO (Internation Organization for Standardization). *ISO 8601:2004. Data elements and interchange formats — Information interchange — Representation of dates and times*. 2004.

[JSON-LD]

W3C (World Wide Web Consortium). *JSON-LD 1.0 — A JSON-based Serialization for Linked Data.* Manu Sporny, Gregg Kellogg and Markus Lanthaler, eds., 2014. W3C Recommendation. (See https://www.w3.org/TR/json-ld/.)

[Linked Data]

Heath, Tom and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*, 1st edition. Morgan & Claypool, 2011. (See http://linkeddatabook.com/editions/1.0/.)

[PROV-O]

W3C (World Wide Web Consortium). *PROV-O: The PROV Ontology.* Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik and Jun Zhao, eds., 2013. W3C Recommendation. (See https://www.w3.org/TR/prov-o/.)

[RDF Concepts]

W3C (World Wide Web Consortium). *RDF 1.1 Concepts and Abstract Syntax.* Richard Cyganiak,

David Wood and Markus Lanthaler, eds., 2014. W3C Recommendation. (See https://www.w3.org/TR/rdf11-concepts/.)

[RFC 4122]

IETF (Internet Engineering Task Force). *A Universally Unique IDentifier (UUID) URN Namespace*. P. Leach, M. Mealling and R. Salz, ed., 2005. (See https://tools.ietf.org/html/rfc4122.)

[RFC 7159]

IETF (Internet Engineering Task Force). *The JavaScript Object Notation (JSON) Data Interchange Format*. Tim Bray, ed., 2014. (See https://tools.ietf.org/html/rfc7159.)

[RFC 7469]

IETF (Internet Engineering Task Force). *Public Key Pinning Extension for HTTP*. C. Evans, C. Palmer and R. Sleevi, ed., 2015. (See https://tools.ietf.org/html/rfc7469.)

[SWBP XSD DT]

W3C (World Wide Web Consortium). *XML Schema Datatypes in RDF and OWL*. Jeremy J. Carroll and Jeff Z. Pan, 2006. W3C Working Group. (See https://www.w3.org/TR/swbp-xsch-datatypes/.)

[XML Names]

W3 (World Wide Web Consortium). *Namespaces in XML 1.1*, 2nd edition. Tim Bray, Dave Hollander, Andrew Layman and Richard Tobin, ed., 2006. W3C Recommendation. (See https://www.w3.org/TR/xml-names11/.)

[XSD Pt1]

W3 (World Wide Web Consortium). *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. Shudi Gao (高殊镝), C. M. Sperberg-McQueen and Henry S. Thompson, ed., 2012. W3C Recommendation. (See https://www.w3.org/TR/xmlschema11-1/.)

[XSD Pt2]

W3 (World Wide Web Consortium). *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. David Peterson, Shudi Gao (高殊镝), Ashok Malhotra, C. M. Sperberg-McQueen and Henry S. Thompson, ed., 2012. W3C Recommendation. (See https://www.w3.org/TR/xmlschema11-2/.)
