Citation Style Language Documentation

Release 1.0.1-dev

Rintze M. Zelle

Sep 04, 2018

Contents

CHAPTER 1

Primer — An Introduction to CSL

by Rintze M. Zelle, PhD

Table of Contents

1.1 Preface

This primer is an introduction to the Citation Style Language (CSL), an open XML-based language to describe the formatting of citations and bibliographies. For a more technical and in-depth description of CSL, see the CSL Specification.

1.2 What is CSL?

If you ever wrote a research paper, you probably included references to other works. Referencing is important in scholarly communication, as it provides attribution, and links together published research. However, manually formatting references can become very time-consuming, especially when you're dealing with multiple journals that each have their own citation style.

Luckily, reference management software can help out. Programs like Zotero, Mendeley, and Papers not only help you manage your research library, but can also automatically generate citations and bibliographies. But to format references in the desired style, these programs need descriptions of each citation style in a language the computer can understand. As you might have guessed, the Citation Style Language (CSL) is such a language.

1.3 Citation Formats

There are hundreds, if not thousands of different citation styles in use around the world. Fortunately, most citation styles fall within a few basic categories. CSL distinguishes between the following types:

1.3.1 In-text Styles

Citation styles can be divided in two main categories. The first category consists of **in-text** styles, where a *citation* in the sentence directly points to one or multiple entries in the *bibliography*. CSL subdivides this category in **author-date**, **author**, **numeric** and **label** styles.

Each citation points to one or more *bibliographic entries*. In CSL, each individual pointer is called a *cite*. For example, the citation "(Doe et al. 2002, Smith 1997)" contains two cites: one to a 2002 publication by Doe et al., and one to a 1997 publication by Smith. In the context of CSL, bibliographic entries are sometimes also called *references*.

"author-date" & "author" Styles

Cites of **author-date** styles show author names and the date of publication, e.g. "(Van der Klei et al. 1991; Zwart et al. 1983)", whereas cites of **author** styles only show names, e.g. "(Gidijala et al.)". Bibliographic entries are typically sorted alphabetically by author.

Note that many style guides use the confusing term "Harvard" to refer to **author-date** formats, even though most of these styles have no connection to Harvard University. There is also no such thing as a single official "Harvard" style.

Example Bibliography

Gidijala L, Bovenberg RA, Klaassen P, van der Klei IJ, Veenhuis M, et al. (2008) Production of functionally active *Penicillium chrysogenum* isopenicillin N synthase in the yeast *Hansenula polymorpha*. BMC Biotechnol 8: 29.

van der Klei IJ, Harder W, Veenhuis M (1991) Methanol metabolism in a peroxisome-deficient mutant of *Hansenula polymorpha*: a physiological study. Arch Microbiol 156: 15-23.

Zwart KB, Veenhuis M, Harder W (1983) Significance of yeast peroxisomes in the metabolism of choline and ethanolamine. Antonie van Leeuwenhoek 49: 369-385.

"numeric" Styles

Cites of **numeric** styles consist of numbers, e.g. "[1, 2]" and "[3]". Bibliographic entries are typically sorted either alphabetically by author, or put in the order by which they have been first cited.

Example Bibliography

1. Gidijala L, Bovenberg RA, Klaassen P, van der Klei IJ, Veenhuis M, et al. (2008) Production of functionally active *Penicillium chrysogenum* isopenicillin N synthase in the yeast *Hansenula polymorpha*. BMC Biotechnol 8: 29.
2. Zwart KB, Veenhuis M, Harder W (1983) Significance of yeast peroxisomes in the metabolism of choline and ethanolamine. Antonie van Leeuwenhoek 49: 369-385.
3. van der Klei IJ, Harder W, Veenhuis M (1991) Methanol metabolism in a peroxisome-deficient mutant of *Hansenula polymorpha*: a physiological study. Arch Microbiol 156: 15-23.

"numeric" Compound Styles

Compound styles are a variation of the **numeric** in-text style format. With these styles, popular in the field of chemistry, bibliographic entries may contain multiple references. Once a citation has defined such a bibliographic entry (e.g, "[2]"), it becomes possible to cite individual items within the entry (e.g., "[2b]"). This format is not yet supported by CSL.

Example Bibliography

1. Gidijala L, et al. (2008) BMC Biotechnol 8: 29.
2. a) Zwart KB, et al. (1983) Antonie van Leeuwenhoek 49: 369-385, b) van der Klei IJ, et al. (1991) Arch Microbiol 156: 15-23.

"label" Styles

Cites of **label** styles consist of short keys, e.g. "[GBKv2008]" and "[ZwVH1983; vaHV1991]". These keys are also included in the bibliographic entries. CSL has limited support for this format, since it currently doesn't allow for (style-specific) customisation of the key format.

Example Bibliography

[GBKv2008] Gidijala L, Bovenberg RA, Klaassen P, van der Klei IJ, Veenhuis M, et al. (2008) Production of functionally active *Penicillium chrysogenum* isopenicillin N synthase in the yeast *Hansenula polymorpha*. BMC Biotechnol 8: 29.

[vaHV1991] van der Klei IJ, Harder W, Veenhuis M (1991) Methanol metabolism in a peroxisome-deficient mutant of *Hansenula polymorpha*: a physiological study. Arch Microbiol 156: 15-23.

[ZwVH1983] Zwart KB, Veenhuis M, Harder W (1983) Significance of yeast peroxisomes in the metabolism of choline and ethanolamine. Antonie van Leeuwenhoek 49: 369-385.

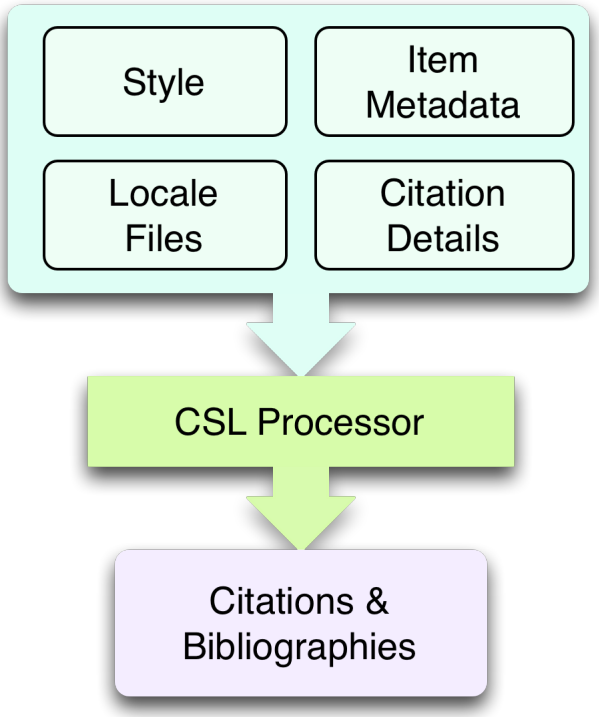
1.3.2 Note Styles

The second category of citation styles consists of **note** styles. Here a *marker*, which can be a number or a symbol, is added to the sentence when works are cited, e.g. "*[0]" and "†[0]". Each marker points to a footnote or endnote. CSL styles do not control which number formats or symbols are used for the markers, which is left to the word processor. In contrast to **in-text** citations, footnotes and endnotes typically contain all information required to identify the cited works. Some **note** styles include a bibliography to give an overview of all cited works, and to describe the works in more detail.

1.4 The CSL Ecosystem

To understand how CSL works, let's start by taking a look at the various bits and pieces of the CSL ecosystem:

[0] 'Voyage to St. Kilda' (3rd edit. 1753), p. 37.
[0] Sir J. E. Tennent, 'Ceylon,' vol. ii. 1859, p. 107.



1.4.1 Independent and Dependent Styles

Styles! Everything in the world of CSL revolves around styles. But not all CSL styles are alike. There are two types: **independent styles** and **dependent styles**.

An **independent CSL style** has two functions: first, it needs to define a citation format. What does the format look like? Is it an "author-date" style, or a "note" style? Are cites ordered alphabetically, or by date? Should bibliographic entries include DOIs? What punctuation and capitalization should be used? Does the year of publication come before or after the title? Etcetera, etcetera. Secondly, the CSL style must describe itself. We call this self-describing information **style metadata**, and it can include the title of the journal for which the CSL style was created, a link to that journal's website, the name of the creator of the CSL style, etc.

A **dependent CSL style**, on the other hand, only contains **style metadata**. Instead of providing a definition of a citation format, a dependent style simply refers to an independent CSL style (its "parent"), whose citation format will be used instead.

Dependent styles come in handy when multiple CSL styles share the same citation format. Take a publisher which uses a single citation format for all its journals. If we were limited to using independent CSL styles, every journal's CSL style would need to contain a full definition of the citation format, even though it would be the same for each journal. This would produce a collection of bulky styles that are hard to maintain. If the publisher makes a change to its citation format, we would have to update every single independent CSL style.

Dependent styles solve these problems. For example, the journals "Nature Biotechnology", "Nature Chemistry", and "Nature" all use the same citation format. We therefore created dependent CSL styles for "Nature Biotechnology" and "Nature Chemistry" that both point to our independent CSL style for "Nature". Since they don't define a citation format, dependent styles are a fraction of the size of an independent style. And, if the Nature Publishing Group ever decides to change the "Nature" citation format across its journals, we only have to correct the citation format in the "Nature" CSL style, without having to touch any of its dependents.

1.4.2 Locale Files

I have a little secret to share with you: most independent styles aren't fully independent.

Take the reference below:

> Hartman, P., Bezos, J. P., Kaphan, S., & Spiegel, J. (1999, September 28). Method and system for placing a purchase order via a communications network. Retrieved from https://www.google.com/patents/US5960411

You can describe this citation format in an independent CSL style by hard-coding all language-specific information into the style. For example, you can put the text "Retrieved from" before the URL, and use "YYYY, Month DD" as the date format. However, such a style would only be usable in US English. If you later need a German variant of this citation format, you would have to change all the translations and date formats within the style.

Fortunately, independent CSL styles can rely on the CSL **locale files** for translations of common terms, localized date formats, and grammar. For example, we can rewrite our CSL style to use the "retrieved" and "from" CSL terms, and to use the localized "text" date format. If we then set the locale of the style to US English, this style will retrieve the term translations and localized date format from the US English CSL locale file, which will produce the reference as written above. But if we switch the style locale to German, the German locale file will be used instead, producing:

> Hartman, P., Bezos, J. P., Kaphan, S., & Spiegel, J. (28. September 1999). Method and system for placing a purchase order via a communications network. Abgerufen von https://www.google.com/patents/US5960411

So with CSL locale files, it becomes possible to write CSL styles that are largely language-agnostic. As illustrated above, such styles can easily switch between different languages. However, languages are complex, and CSL's automatic localization doesn't support the peculiarities of all languages for which we have locale files. But even if you find that you need to modify a CSL style to adapt it to your language of preference, language-agnostic styles have value, since they're easier to translate.

Locale files have the added benefit that we only need to define common translations, date formats, and grammar once per language. This keeps styles compact, and makes locale data easier to maintain. Since citation formats for a given language don't always agree on a translation or date format, CSL styles can selectively overwrite any locale data that is defined in the locale files.

1.4.3 Item Metadata

Next up are the bibliographic details of the items you wish to cite: the **item metadata**.

For example, the bibliographic entry for a journal article may show the names of the authors, the year in which the article was published, the article title, the journal title, the volume and issue in which the article appeared, the page numbers of the article, and the article's Digital Object Identifier (DOI). All these details help the reader identify and find the referenced work.

Reference managers make it easy to create a library of items. While many reference managers have their own way of storing item metadata, most support common bibliographic exchange formats such as BibTeX and RIS. The citeproc-js CSL processor introduced a JSON-based format for storing item metadata in a way citeproc-js could understand. Several other CSL processors have since adopted this "CSL JSON" format (also known as "citeproc JSON").

1.4.4 Citing Details

For a given citation format, the way citations and bibliographies look not only depends on the metadata of the cited items, but also on the context in which these items are cited. We refer to this type of context-specific information as the **citing details**.

For instance, the order in which items are cited in a document can affect their order in the bibliography. And in "note" styles, subsequent cites to a previously cited item are often written in a more compact form. Another example is the use of locators, which guide the reader to a specific location within a cited work, such as the page numbers within a chapter where a certain argument is made, e.g. "(Doe 2000, pp. 43-44)".

1.4.5 CSL Processors

With CSL styles, locale files, item metadata and citing details in hand, we now need a piece of software to parse all this information, and generate citations and bibliographies in the correct format: the **CSL processor**.

Most reference managers use one of the freely available open source CSL processors, such as citeproc-js.

1.5 Understanding CSL Styles

By now you've learned what CSL is, how it can be used, and how its different parts and pieces fit together. We're now ready to dive into the CSL styles themselves, and look at their XML code.

1.5.1 XML Basics

If you're new to XML, this section gives a short overview of what you need to know about XML in order to read and edit CSL styles and locale files. For more background, just check one of the many XML tutorials online.

Let's take a look at the following dependent CSL style:

```
<?xml version="1.0" encoding="utf-8"?>
<style xmlns="http://purl.org/net/xbiblio/csl" version="1.0" default-locale="en-US">
  <!-- Generated with https://github.com/citation-style-language/utilities/tree/
↪master/generate_dependent_styles/data/asm -->
  <info>
    <title>Applied and Environmental Microbiology</title>
    <id>http://www.zotero.org/styles/applied-and-environmental-microbiology</id>
    <link href="http://www.zotero.org/styles/applied-and-environmental-microbiology"␣
↪rel="self"/>
    <link href="http://www.zotero.org/styles/american-society-for-microbiology" rel=
↪"independent-parent"/>
    <link href="http://aem.asm.org/" rel="documentation"/>
    <category citation-format="numeric"/>
    <category field="biology"/>
    <issn>0099-2240</issn>
    <eissn>1098-5336</eissn>
    <updated>2014-04-30T03:45:36+00:00</updated>
    <rights license="http://creativecommons.org/licenses/by-sa/3.0/">This work is␣
↪licensed under a Creative Commons Attribution-ShareAlike 3.0 License</rights>
  </info>
</style>
```

There are several concepts and terms you need to be familiar with. These are:

- **XML Declaration**. The first line of each style and locale file is usually the XML declaration. In most cases, this will be `<?xml version="1.0" encoding="utf-8"?>`. This declaration makes it clear that the document consists of XML, and specifies the XML version ("1.0") and character encoding ("utf-8") used.
- **Elements and Hierarchy**. Elements are the basic building blocks of XML documents. Each XML document contains a single root element (for CSL styles this is `<style/>`). If an element contains other elements, this

parent element is split into a start tag (`<style>`) and an end tag (`</style>`). In our example, the `<style/>` element has one child element, `<info/>`. This element has several children of its own, which are grandchildren of the grandparent `<style/>` element.

Element tags are always wrapped in less-than ("<") and greater-than (">") characters (e.g., `<style>`). For empty-element tags, ">" is preceded by a forward-slash (e.g., `<category/>`), while for end tags, "<" is followed by a forward-slash (e.g., `</style>`). Child elements are typically indented with spaces or tabs to show the different hierarchical levels. We use 2 spaces per level in our CSL styles and locale files.

In the rest of this primer we will use the prefix "cs:" when referring to CSL elements (e.g., `cs:style` instead of `<style/>`).

- **Attributes and Element Content**. There are two ways to add additional information to elements.

  First, XML elements can carry one or more attributes. The order of attributes on an element is arbitrary, but every attribute needs a value. For example, the `<style/>` element carries the `version` attribute, set to a value of "1.0" (this indicates that the style is compatible with the latest CSL 1.0.x release).

  Secondly, elements can store non-element content between their start and end tags. For example, the title of the style, "Applied and Environmental Microbiology", is stored as the content of the `<title/>` element.

- **Escaping**. To avoid ambiguity in defining the structure of XML files, some characters need to be substituted when used for other purposes, e.g. when used in attribute values or element content. The escape sequences are:

  - `&lt;` for <
  - `&gt;` for >
  - `&amp;` for &
  - `&apos;` for '
  - `&quot;` for "

  For example, the link `http://domain.com/?tag=a&id=4` is escaped as `<link href="http://domain.com/?tag=a&amp;id=4"/>`.

- **XML Comments**. You can use XML comments to add clarifying information to a XML file. Comments start with `<!--` and end with `-->`, and are ignored by the CSL processor.

- **Well-formedness and Schema Validity**. Unlike HTML, XML is unforgiving when it comes to markup errors. Any error, like forgetting an end tag, having more than one root element, or incorrect escaping will break the entire XML document, and prevent it from being processed.

  To make sure that a CSL style works correctly, it must follow the XML specification. An error-free XML file is called "well-formed". But to be considered "valid" CSL, a well-formed CSL style must also follow the rules specified by the CSL schema. This schema describes all the various CSL elements and attributes, and how they must be used.

  You can use a CSL validator to check a CSL style for any errors. Remember that only well-formed and valid CSL files can be expected to work properly.

1.5.2 Anatomy of a Dependent Style

As explained above, dependent CSL styles are much more compact that their independent counterparts, since they don't actually have to define a citation format. Dependent styles are also very common, and their style metadata is similar to that of independent styles, so they are a good starting point for learning CSL. Let's take a closer look at the dependent style above, line by line.

```
<?xml version="1.0" encoding="utf-8"?>
```

The XML declaration.

```
<style xmlns="http://purl.org/net/xbiblio/csl" version="1.0" default-locale="en-US">
    ...
</style>
```

The start and end tags of the `cs:style` root element. Its `xmlns` attribute specifies that all elements in the style are part of CSL, while the `version` attribute indicates CSL version compatibility. The `default-locale` attribute tells the style to generate citations and bibliographies in a certain language (in this case US English).

```
<!-- Generated with https://github.com/citation-style-language/utilities/tree/master/
↪generate_dependent_styles/data/asm -->
```

Most of our dependent styles are automatically generated from spreadsheet data. This XML comment makes it clear that this style has been generated, and contains a link to the spreadsheet.

```
<info>
  ...
</info>
```

The `cs:info` section is used to store most of the style's metadata.

```
<title>Applied and Environmental Microbiology</title>
```

The title of the style.

```
<id>http://www.zotero.org/styles/applied-and-environmental-microbiology</id>
```

The style ID, which is used by reference managers to identify styles and tell them apart.

```
<link href="http://www.zotero.org/styles/applied-and-environmental-microbiology" rel=
↪"self"/>
```

The style's "self" link. This URL links to an online copy of the style. For simplicity, we use the same URL as style ID and "self" link for our repository styles.

```
<link href="http://www.zotero.org/styles/american-society-for-microbiology" rel=
↪"independent-parent"/>
```

Dependent styles need to link to an independent parent style, whose citation format will be used. Here we use the citation format from the CSL style for the American Society for Microbiology.

```
<link href="http://aem.asm.org/" rel="documentation"/>
```

It's much easier to maintain our collection of CSL styles if each style's purpose is clear. We therefore require that all our repository styles contain at least one "documentation" link. In this case, to the journal's home page.

```
<category citation-format="numeric"/>
<category field="biology"/>
```

To help cataloguing our styles, we specify the citation format with the `citation-format` attribute on `cs:category`. Similarly, we assign each style to one or more fields of study, using the `field` attribute.

```
<issn>0099-2240</issn>
<eissn>1098-5336</eissn>
```

When a CSL styles is created for a journal, we store the journal's print ISSN and electronic ISSN in the `cs:issn` and `cs:eissn` elements, respectively.

```
<updated>2014-04-30T03:45:36+00:00</updated>
```

A time stamp to indicate when the style was last updated.

```
<rights license="http://creativecommons.org/licenses/by-sa/3.0/">This work is␣
↪licensed under a Creative Commons Attribution-ShareAlike 3.0 License</rights>
```

Last, but certainly not least, the license under which the style is released.

1.5.3 Anatomy of an Independent Style

Finally, a real independent CSL style, one that actually defines a citation format! Well, okay, maybe it's not exactly a realistic style. Most independent styles in our repository are quite a bit bigger than the simplified example style below. But our "author-date" style below is valid CSL, and still has the same overall design as any other independent style.

```
<?xml version="1.0" encoding="utf-8"?>
<style xmlns="http://purl.org/net/xbiblio/csl" class="in-text" version="1.0">
  <info>
    <title>Example Style</title>
    <id>http://www.zotero.org/styles/example</id>
    <link href="http://www.zotero.org/styles/example" rel="self"/>
    <link href="http://www.zotero.org/styles/apa" rel="template"/>
    <link href="http://www.example.com/style-guide/" rel="documentation"/>
    <author>
      <name>John Doe</name>
      <email>JohnDoe@example.com</email>
    </author>
    <contributor>
      <name>Jane Doe</name>
    </contributor>
    <contributor>
      <name>Bill Johnson</name>
    </contributor>
    <category citation-format="author-date"/>
    <category field="science">
    <updated>2014-10-15T18:17:09+00:00</updated>
    <rights license="http://creativecommons.org/licenses/by-sa/3.0/">This work is␣
↪licensed under a Creative Commons Attribution-ShareAlike 3.0 License</rights>
  </info>
  <locale xml:lang="en">
    <terms>
      <term name="no date">without date</term>
    </terms>
  </locale>
  <macro name="author">
    <names variable="author">
      <name initialize-with="."/>
    </names>
  </macro>
  <macro name="issued-year">
    <choose>
      <if variable="issued">
        <date variable="issued">
          <date-part name="year"/>
        </date>
      </if>
```

(continues on next page)

(continued from previous page)

```
      <else>
        <text term="no date"/>
      </else>
    </choose>
  </macro>
  <citation et-al-min="3" et-al-use-first="1">
    <sort>
      <key macro="author"/>
      <key macro="issued-year"/>
    </sort>
    <layout prefix="(" suffix=")" delimiter="; ">
      <group delimiter=", ">
        <text macro="author"/>
        <text macro="issued-year"/>
      </group>
    </layout>
  </citation>
  <bibliography>
    <sort>
      <key macro="author"/>
      <key macro="issued-year"/>
      <key variable="title"/>
    </sort>
    <layout suffix="." delimiter=", ">
      <group delimiter=". ">
        <text macro="author"/>
        <text macro="issued-year"/>
        <text variable="title"/>
        <text variable="container-title"/>
      </group>
      <group>
        <text variable="volume"/>
        <text variable="issue" prefix="(" suffix=")"/>
      </group>
      <text variable="page"/>
    </layout>
  </bibliography>
</style>
```

Style Structure

To understand the style above, lets first look at the child elements of the `cs:style` root element:

```
<?xml version="1.0" encoding="utf-8"?>
<style>
  <info/>
  <locale/>
  <macro/>
  <macro/>
  <citation/>
  <bibliography/>
</style>
```

Compared to a dependent style, which only has the `cs:info` child element, we see several additional elements here. In additional to `cs:info`, we see `cs:locale`, `cs:macro`, `cs:citation`, and `cs:bibliography`.

What do these elements do?

- The required `cs:info` element fulfills the same function in independent styles as it does in dependent styles: it stores the style metadata.
- The optional `cs:locale` elements can be used to overwrite the locale data from the locale files.
- The optional `cs:macro` elements can be used to store CSL code for use by `cs:citation`, `cs:bibliography`, or other `cs:macro` elements.
- The required `cs:citation` element defines the format of citations.
- The optional `cs:bibliography` element defines the format of the bibliography.

With this in mind, let's step through the style, starting with the `cs:style` element.

`cs:style` Root Element

```
<style xmlns="http://purl.org/net/xbiblio/csl" class="in-text" version="1.0">
  ...
</style>
```

We've already come across the `xmlns` and `version` attributes when we looked at the `cs:style` element of our dependent style. The `class` attribute is new. It tells the CSL processor whether it is an "in-text" or "note" style.

`cs:info` Element

The style metadata for independent styles is usually more expansive than for dependent styles:

```
<info>
  <title>Example Style</title>
  <id>http://www.zotero.org/styles/example</id>
  <link href="http://www.zotero.org/styles/example" rel="self"/>
  <link href="http://www.zotero.org/styles/apa" rel="template"/>
  <link href="http://www.example.com/style-guide/" rel="documentation"/>
  <author>
    <name>John Doe</name>
    <email>JohnDoe@example.com</email>
  </author>
  <contributor>
    <name>Jane Doe</name>
  </contributor>
  <contributor>
    <name>Bill Johnson</name>
  </contributor>
  <category citation-format="author-date"/>
  <category field="science">
  <updated>2014-10-15T18:17:09+00:00</updated>
  <rights license="http://creativecommons.org/licenses/by-sa/3.0/">This work is␣
↪licensed under a Creative Commons Attribution-ShareAlike 3.0 License</rights>
</info>
```

The title, style ID, "self" link, categories, time stamp, and license work the same, but there are differences. First, independent styles don't depend on a parent style. Instead we usually provide a "template" link to indicate which style was used as a starting point for creating the current style (CSL styles are rarely written from scratch, since it's usually much faster to adapt an existing one). In this case, the template was the APA style. We also like to include one or more "documentation" links that point to an online description of the citation format in question.

To acknowledge the creators of CSL styles, their names and contact information can be added to the style. In this case, we have one author and two contributors. Authors usually have done most of the work in creating the style, whereas contributors have provided small improvements.

`cs:citation` and `cs:macro` Elements

Let's jump down now to the macros and `cs:citation` element. The purpose of the `cs:citation` element is to describe the format of citations (or, for "note" styles, the format of footnotes or endnotes).

```
<macro name="author">
  <names variable="author">
    <name initialize-with="."/>
  </names>
</macro>
<macro name="issued-year">
  <choose>
    <if variable="issued">
      <date variable="issued">
        <date-part name="year"/>
      </date>
    </if>
    <else>
      <text term="no date"/>
    </else>
  </choose>
</macro>
<citation et-al-min="3" et-al-use-first="1">
  <sort>
    <key macro="author"/>
    <key macro="issued-year"/>
  </sort>
  <layout prefix="(" suffix=")" delimiter="; ">
    <group delimiter=", ">
      <text macro="author"/>
      <text macro="issued-year"/>
    </group>
  </layout>
</citation>
```

The code above generates citations like "(A.C. Smith et al., 2002; W. Wallace, J. Snow, 1999)". To understand how this citation format is encoded in CSL, let's first focus on the `cs:layout` element of `cs:citation`. Its `prefix` and `suffix` attributes define the parentheses around the citation, while the value of the `delimiter` attribute ("; ") separates neighboring cites. The format of each individual cite is defined by the contents of `cs:layout`, which consists of the output of the "author" and "issued-year" macros, separated by the value of the "delimiter" attribute (", ") on the `cs:group` element.

The "author" macro prints the names stored in the "author" name variable of the cited item. The `initialize-with` attribute on `cs:name` specifies that given names should appear as initials, and that each initial is followed by the attribute's value (".").

The "issued-year" macro starts with a test, defined with the `cs:choose` element. If the cited item has a date stored in its "issued" date variable, the year of this date is printed. Otherwise, the style prints the value of the "no date" term.

You might wonder why we didn't just put the CSL code from the two macros directly into the `cs:citation` element. What are the advantages of using macros? Well, in the example above, the use of macros simplifies the structure of `cs:citation`, making it easier to follow. In addition, both macros are called a total of four times in the style (twice

in `cs:citation`, and twice in `cs:bibliography`). Without macros, we'd have to repeat the CSL code of these macros multiple times. Macros thus allow for more compact styles.

We're not done yet. The `cs:citation` element carries two attributes, `et-al-min` and `et-al-use-first`. Together, they specify that if an item has three or more "author" names, only the first name is printed, followed by the value of the "et al" term.

Finally, `cs:citation` contains the `cs:sort` element, which itself contains two `cs:key` elements. This section specifies how cites within a citation are sorted. The first sorting key consists of the output of the "author" macro (CSL is smart enough to sort names by the family name first, and by initials second). Any cites with the same output for the first key are then sorted by the second sorting key, which is the output of the "issue-year" macro.

`cs:bibliography` Element

Whereas `cs:citation` is responsible for citations and cites, the `cs:bibliography` element is used to define the format of bibliographic entries.

```
<macro name="author">
  <names variable="author">
    <name initialize-with="."/>
  </names>
</macro>
<macro name="issued-year">
  <choose>
    <if variable="issued">
      <date variable="issued">
        <date-part name="year"/>
      </date>
    </if>
    <else>
      <text term="no date"/>
    </else>
  </choose>
</macro>
...
<bibliography>
  <sort>
    <key macro="author"/>
    <key macro="issued-year"/>
    <key variable="title"/>
  </sort>
  <layout suffix="." delimiter=", ">
    <group delimiter=". ">
      <text macro="author"/>
      <text macro="issued-year"/>
      <text variable="title"/>
      <text variable="container-title"/>
    </group>
    <group>
      <text variable="volume"/>
      <text variable="issue" prefix="(" suffix=")"/>
    </group>
    <text variable="page"/>
  </layout>
</bibliography>
```

The `cs:bibliography` section of our example style really only works well for a single type of items: journal articles. It generates bibliographic entries in the form of:

A.C. Smith, D. Williams, T. Johnson. 2002. Story of my life. Journal of Biographies, 12(2), 24—27. W. Wallace, J. Snow. 1999. Winter is coming. Journal of Climate Dynamics, 6(9), 97—102.

How were we able to define this format? First, the structure of `cs:bibliography` is very similar to that of `cs:citation`, but here `cs:layout` defines the format of each individual bibliographic entry. In addition to the "author" and "issued-year" macro, the bibliographic entries also show each item's "title" and "container-title" (for journal articles, the "container-title" is the title of the journal), the "volume" and "issue" in which the article was printed, and the pages ("page") on which the article appeared. The style uses the `prefix` and `suffix` attributes to wrap the journal issue number in parentheses, and relies on the `suffix` and `delimiter` attributes on the `cs:layout` and `cs:group` elements to place the rest of the punctuation.

The `cs:bibliography` element also contains a `cs:sort` element, with three keys: the "author" and "issued-year" macros, and, as a third key, the item's "title".

`cs:locale` Element

The last section of our style is `cs:locale`. As we wrote above, CSL locale files allow CSL styles to quickly translate into different languages. However, sometimes it's desirable to overwrite the default translations.

```
<locale xml:lang="en">
  <terms>
    <term name="no date">without date</term>
  </terms>
</locale>
```

The translation for the "no date" term in the CSL locale file for US English is, not very surprising, "no date". However, for our example style, I wanted to use "without date" instead. To overwrite the default translation, we can use the `cs:locale` element as shown above. For an item without an issued date, this would result in a citation like "(D. Williams, without date)".

The `xml:lang` attribute on `cs:locale` is set to "en", which tells the style to overwrite the "no date" translation whenever the style is used in English. If we used the style in German, the style would still print the translation from the German locale file ("ohne Datum").

1.6 Diving Deeper

You finished the primer. Good job! If you're interested in learning more about CSL, you're now well prepared to start reading the CSL Specification and our other documentation on the Citation Style Language website.

1.7 Feedback

Questions or feedback? Contact us on Twitter at @csl_styles.

CHAPTER 2

Guide to Translating CSL Locale Files

by Rintze M. Zelle, PhD

Table of Contents

2.1 Preface

This document describes how you can help us improve the language support of Citation Style Language (CSL) styles by translating the CSL locale file of your favorite language.

CSL styles are either bound to one particular locale (e.g. the "British Psychological Society" CSL style will always produce citations and bibliographies in British English), or they (automatically) localize, e.g. to a user-selected locale, or to the locale of the user's operating system.

All CSL styles, both those with and without a fixed locale, rely on locale files for default localization data, which consists of translated terms commonly used in citations and bibliographies, date formats, and grammar rules. Storing localization data in separate files has several benefits: translations are easier to maintain, styles are more compact (although styles can still include their own localization data to override the defaults), and styles can be (mostly) language-agnostic.

Below we will describe the structure of a locale file, give instructions on how to translate all its parts, and explain how you can submit your translations. See the CSL specification for more in-depth documentation on the structure and function of locale files.

2.2 Getting Started

The CSL locale files are kept in a GitHub repository at https://github.com/citation-style-language/locales/.

Each locale file contains the localization data for one language. Locale files are named "locales-xx-XX.xml", where "xx-XX" is a BCP 47 language code (e.g. the locale code for British English is "en-GB"). The repository wiki lists the locale code, language, and translation status of all locale files in the repository.

If you find that a locale file already exists for your language, but that its translations are inaccurate or incomplete, you can start translating that file. If there is no locale file for your language, copy the "locales-en-US.xml" file and start from there. Don't worry about finding the correct BCP 47 locale code for a new language; we'll be happy to look it up when you submit your new locale file.

Modifications to existing locale files can be made directly via the GitHub website. New locale files can be submitted as a "gist" or through a pull request. For details, see the instructions on submitting CSL styles. If you edit a locale file on your own computer, use a suitable plain text editor such as TextWrangler for OS X, Notepad++ for Windows, or jEdit.

2.3 Translating Locale Files

When translating locale files, leave the overall structure of the locale file untouched. It makes life easier for us if you don't remove existing elements, introduce new ones, or move stuff around (exceptions are discussed below).

2.3.1 xml:lang

At the top of the locale file you'll find the `locale` root element. The value of its `xml:lang` attribute should be set to the same language code used in the file name of the locale file. For the locales-en-US.xml locale file, this is "en-US":

```
<locale xmlns="http://purl.org/net/xbiblio/csl" version="1.0" xml:lang="en-US">
```

2.3.2 Locale File Metadata

Directly below the `locale` root you'll find an `info` element. Here you can list yourself as a translator using the `translator` element. Optionally you can include contact information such as a website or email address. A locale file can list multiple translators. The `rights` element indicates under which license the locale file is released. All locale files in our repository use the same Creative Commons license, so you don't have to change anything here. The `updated` element is used to keep track of when the locale file was last updated. Feel free to ignore it if the format looks too intimidating.

```
<info>
  <translator>
    <name>John Doe</name>
    <email>john.doe@citationstyles.org</email>
    <uri>http://citationstyles.org/</uri>
  </translator>
  <rights license="http://creativecommons.org/licenses/by-sa/3.0/">This work is␣
↪licensed under a Creative Commons Attribution-ShareAlike 3.0 License</rights>
  <updated>2012-07-04T23:31:02+00:00</updated>
</info>
```

2.3.3 Grammar Rules

Next up is the `style-options` element:

```
<style-options punctuation-in-quote="true"/>
```

This element is used to define localized grammar rules, as described in the Locale Options section in the CSL specification.

2.3.4 Date Formats

CSL styles can render dates in either non-localizing or localizing formats:

```
<style>

  <!-- use of non-localized date format -->
  <macro name="accessed">
    <date variable="accessed" suffix=", ">
      <date-part name="month" suffix=" "/>
      <date-part name="day" suffix=", "/>
      <date-part name="year"/>
    </date>
  </macro>

  <!-- use of localized date format -->
  <macro name="issued">
    <date variable="issued" form="text"/>
  </macro>

</style>
```

Each locale file defines two localized date formats: a numeric format (e.g. "2012/9/3"), and a textual format, where the month is written out in full (e.g. "September 3, 2012").

To localize a date format, place the date-part elements for "day", "month", and "year" in the desired order. Use the `prefix` and `suffix` attributes (on the `date-part` elements), or the `delimiter` attribute (on the `date` element) to define punctuation before, after, and between the different date-parts. When using affixes, make sure that dates that only consist of a year and a month, or of only a year, still render correctly. For example, the US English localized "text" date format,

```
<date form="text">
  <date-part name="month" suffix=" "/>
  <date-part name="day" suffix=", "/>
  <date-part name="year"/>
</date>
```

will produce dates like "September 3, 2012", "September 2012", and "2012". Compare this to

```
<date form="text">
  <date-part name="month"/>
  <date-part name="day" prefix=" "/>
  <date-part name="year" prefix=", "/>
</date>
```

which gives the same correct complete date ("September 3, 2012"), but which produces incorrect output for dates that don't have a day, or don't have a day and month ("September, 2012" and ", 2012", respectively).

To read more about customizing date formats, see the Localized Date Formats and Date-part sections in the CSL specification.

2.3.5 Terms

The `terms` element makes up the last section of the locale file, and contains all the term translations. Below we discuss the different types of terms, and how to translate them.

In its simplest form, a term consists only of a `term` element with the `name` attribute indicating the term name, and with the translation enclosed between the start and end tag:

```
<terms>
  <term name="et-al">et al.</term>
</terms>
```

See the Terms section in the CSL specification.

Abbreviations

When translating abbreviations such as "et al.", always include periods where applicable.

Plurals

Many terms have translations for both the singular and plural form. In this case, the `term` element contains a `single` (for singular) and a `multiple` (for plural) element, which enclose the translations:

```
<terms>
  <term name="edition">
    <single>edition</single>
    <multiple>editions</multiple>
  </term>
</terms>
```

Forms

Terms can also vary in their 'form', which is indicated with the "form" attribute on the `term` element. The different forms are "long" (the default), "short" (abbreviated form of "long"), "verb", "verb-short" (abbreviated form of "verb"), and "symbol". Examples of the different forms:

```
<terms>
  <term name="editor">
    <single>editor</single>
    <multiple>editors</multiple>
  </term>

  <term name="editor" form="short">
    <single>ed.</single>
    <multiple>eds.</multiple>
  </term>

  <term name="editor" form="verb">edited by</term>
  <term name="editor" form="verb-short">ed.</term>

  <term name="paragraph">
    <single>paragraph</single>
    <multiple>paragraph</multiple>
  </term>

  <term name="paragraph" form="symbol">
    <single>¶</single>
    <multiple>¶¶</multiple>
  </term>
</terms>
```

AD/BC

The "ad" and "bc" terms are used to format years before 1000. E.g. the year "79" becomes "79AD", and "-2500" becomes "2500BC".

See the AD and BC section in the CSL specification.

Punctuation

The terms "open-quote", "close-quote", "open-inner-quote", "close-inner-quote", and "page-range-delimiter" define punctuation.

When a CSL style renders a title in quotes through the use of the `quotes` attribute, it uses the "open-quote" and "close-quote" terms. When the title contains internal quotes, these are replaced by "open-inner-quote", "close-inner-quote". For example, with

```
<terms>
  <term name="open-quote">“</term>
  <term name="close-quote">”</term>
  <term name="open-inner-quote">‘</term>
  <term name="close-inner-quote">’</term>
  <term name="page-range-delimiter">–</term>
</terms>
```

styles can render titles as

```
“Moby-Dick”
“Textual Analysis of ‘Moby-Dick’”
```

The “page-range-delimiter” terms is used to connect the first and last page of page ranges, e.g. “15–18” (it’s default value is an en-dash).

See the Quotes and Page Ranges sections in the CSL specification.

Ordinals

CSL styles can render numbers (e.g., “2”) in two ordinal forms: “long-ordinal” (“second”) and “ordinal” (“2nd”). Both forms are localized through the use of terms.

The “long-ordinal” form is limited to the numbers 1 through 10 (the fallback for other numbers is the “ordinal” form). Each of these ten numbers has its own term (“long-ordinal-01” through “long-ordinal-10”).

Things are different for the “ordinal” form. Here, terms are only used to define the ordinal suffix (“nd” for “2nd”). Furthermore, terms and numbers don’t correspond one-to-one. For example, the “ordinal” term defines the default suffix, which is used for all numbers (unless, as described below, exceptions are introduced through the use of the terms “ordinal-00” through “ordinal-99”).

CSL also supports gender-specific ordinals (both for “long-ordinal” and “ordinal” forms). In languages such as French, ordinal numbers must match the gender of the target noun, which can be feminine or masculine. E.g. “1re édition” (“édition” is feminine) and “1er janvier” (“janvier” is masculine). See the relevant section below.

Terms for “ordinal” numbers

Terms for the “ordinal” form follow special rules to make it possible to render any number in the “ordinal” form (e.g., “2nd”, “15th”, “231st”), without having to define a term for each number.

The logic for defining ordinal suffixes with terms is described at Ordinal Suffixes, and won’t be revisited here. Instead, we’ll look at an example.

In English, there are four different ordinal suffixes in use: “st”, “nd”, and “rd” are used for numbers ending on 1, 2, and 3, respectively, while “th” is used for numbers ending on 0 and 4 through 9. Exceptions are numbers ending on “11”, “12”, and “13”, which also use “th”.

To capture this logic, we start by defining the “ordinal” term as “th”, which is the most common suffix. Then, we define the terms “ordinal-01”, “ordinal-02”, and “ordinal-01” as “st”, “nd”, and “rd”, respectively. By default (i.e., when the `term` elements don’t carry a `match` attribute), the terms “ordinal-00” through “ordinal-09” repeat at intervals of 10. For example, the term “ordinal-01” overrides the “ordinal” term for numbers 1, 11, 21, 31, etc. At this point, we would get “ordinal” numbers such as “1st”, “2nd”, “3rd”, “4th”, “21st”, “67th”, and “101st”, but we would also get the incorrect “11st”, “12nd” and “13rd”. For these cases, we define the terms “ordinal-11”, “ordinal-12”, and “”ordinal-13” as “th”. By default, the terms “ordinal-10” though “ordinal-99” repeat at intervals of 100. For example, the term “ordinal-11” overrides the “ordinal” and “ordinal-01” terms for numbers “11”, “111”, “211”, etc. So, in total, we need the following seven terms:

```
<terms>
  <term name="ordinal">th</term>
  <term name="ordinal-01">st</term>
  <term name="ordinal-02">nd</term>
  <term name="ordinal-03">rd</term>
  <term name="ordinal-11">th</term>
  <term name="ordinal-12">th</term>
  <term name="ordinal-13">th</term>
</terms>
```

Fortunately, many languages have simpler "ordinal" numbers. E.g., for German all "ordinal" numbers receive a period as the suffix, so it suffice to define the "ordinal" term:

```
<terms>
  <term name="ordinal">.</term>
</terms>
```

Gender-specific Ordinals

To use gender-specific ordinals, we first need to define the gender of several target nouns: the terms accompanying the number variables (it is probably sufficient to specify the gender for "edition", "issue", and "volume") and the month terms ("month-01" through "month-12", corresponding to January through December). This is done by setting the `gender` attribute on the "long" (default) form of these terms to either "masculine" or "feminine".

Secondly, we need to define "feminine" and "masculine" variants of the ordinal terms, which is done with the `gender-variant` attribute (set to "masculine" or "feminine").

A minimal example for French:

```
<?xml version="1.0" encoding="UTF-8"?>
<locale xml:lang="fr-FR">
  <terms>
    <term name="edition" gender="feminine">
      <single>édition</single>
      <multiple>éditions</multiple>
    </term>
    <term name="edition" form="short">éd.</term>
    <term name="month-01" gender="masculine">janvier</term>

    <term name="long-ordinal-01" gender-form="masculine">premier</term>
    <term name="long-ordinal-01" gender-form="feminine">première</term>
    <term name="long-ordinal-01">premier</term>
    <term name="long-ordinal-02">deuxième</term>
    <term name="long-ordinal-03">troisième</term>
    <term name="long-ordinal-04">quatrième</term>
    <term name="long-ordinal-05">cinquième</term>
    <term name="long-ordinal-06">sixième</term>
    <term name="long-ordinal-07">septième</term>
    <term name="long-ordinal-08">huitième</term>
    <term name="long-ordinal-09">neuvième</term>
    <term name="long-ordinal-10">dixième</term>

    <term name="ordinal">e</term>
    <term name="ordinal-01" gender-form="feminine" match="whole-number">re</term>
    <term name="ordinal-01" gender-form="masculine" match="whole-number">er</term>
```

(continues on next page)

(continued from previous page)

```
  </terms>
</locale>
```

In this example, the "edition" term is defined as feminine, and the "month-01" term ("janvier") is defined as masculine. For French, of the "long-ordinal" terms, only "long-ordinal-01" has gender-variants ("premier" for masculine, "première" for feminine). To cover cases where no gender is defined for the target noun (e.g., a style author might redefine a term like "edition" but forget to specify the gender), also a neuter variant of "long-ordinal-01" is defined without the `gender` attribute. The "ordinal" term defines the default suffix. The only exceptions are for the number 1 when the target noun is either feminine or masculine (with `match` set to "whole-number", the term does not repeat), e.g. "1re édition" but "11e édition".

For more information, see the Gender-specific Ordinals section in the CSL specification.

2.4 Submitting Contributions

To submit changes to an existing locale file, or to submit a new locale file, follow the submission instructions for CSL styles.

2.5 Questions?

Questions? Contact us on Twitter at @csl_styles, or create an issue on GitHub here.

CHAPTER 3

CSL 1.0.1 Specification

by Rintze M. Zelle, PhD

with contributions from Frank G. Bennett, Jr. and Bruce D'Arcus.

Table of Contents

3.1 Introduction

The Citation Style Language (CSL) is an XML-based format to describe the formatting of citations, notes and bibliographies, offering:

- An open format
- Compact and robust styles
- Extensive support for style requirements
- Automatic style localization
- Infrastructure for style distribution and updating
- Thousands of freely available styles (Creative Commons BY-SA licensed)

For additional documentation, the CSL schema, styles, and locales, visit the CSL project home, citationstyles.org.

3.1.1 Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, are to be interpreted as described in IETF RFC 2119.

3.2 Namespacing

The CSL XML namespace URI is "http://purl.org/net/xbiblio/csl". The namespace prefix `cs:` is used throughout this specification when referring to CSL elements, but is generally omitted in favor of a default namespace declaration (set with the `xmlns` attribute) on the root `cs:style` or `cs:locale` element.

3.3 File Types

There are three types of CSL files: independent and dependent styles (both types use the ".csl" extension), and locale files (named "locales-xx-XX.xml", where "xx-XX" is a language dialect, e.g. "en-US" for American English).

3.3.1 Independent Styles

Independent styles contain formatting instructions for citations, notes and bibliographies. While mostly self-contained, they rely on locale files for (default) localization data.

3.3.2 Dependent Styles

A dependent style is an alias for an independent style. Its contents are limited to style metadata, and doesn't include any formatting instructions (the sole exception is that dependent styles can specify an overriding style locale). By linking dependent styles for journals that share the same citation style (e.g., "Nature Biotechnology", "Nature Nanotechnology", etc.) to a single independent style (e.g., "Nature Journals"), there is no need to duplicate formatting instructions.

3.3.3 Locale Files

Each locale file contains a set of localization data (term translations, localized date formats, and grammar options) for a particular language dialect.

3.4 XML Declaration

Each style or locale should begin with an XML declaration, specifying the XML version and character encoding. In most cases, the declaration will be:

```
<?xml version="1.0" encoding="UTF-8"?>
```

3.5 Styles - Structure

3.5.1 The Root Element - `cs:style`

The root element of styles is `cs:style`. In independent styles, the element carries the following attributes:

`class` Determines whether the style uses in-text citations (value "in-text") or notes ("note").

`default-locale` (optional) Sets a default locale for style localization. Value must be a locale code.

`version` The CSL version of the style. Must be "1.0" for CSL 1.0-compatible styles.

In addition, `cs:style` may carry any of the *global options* and *inheritable name options*.

Of these attributes, only `version` is required on `cs:style` in dependent styles, while the `default-locale` attribute may be set to specify an overriding style locale. The other attributes are allowed but ignored.

An example of `cs:style` for an independent style, preceded by the XML declaration:

```
<?xml version="1.0" encoding="UTF-8"?>
<style xmlns="http://purl.org/net/xbiblio/csl" version="1.0" class="in-text" default-
↪locale="fr-FR"/>
```

3.5.2 Child Elements of `cs:style`

In independent styles, the `cs:style` root element has the following child elements:

`cs:info` Must appear as the first child element of `cs:style`. Contains the metadata describing the style (style name, ID, authors, etc.).

`cs:citation` Must appear once. Describes the formatting of in-text citations or notes.

`cs:bibliography` (optional) May appear once. Describes the formatting of the bibliography.

cs:macro (optional) May appear multiple times. Macros allow formatting instructions to be reused, keeping styles compact and maintainable.

cs:locale (optional) May appear multiple times. Used to specify (overriding) localization data.

In *dependent styles*, cs:style has only one child element, cs:info.

Info

The cs:info element contains the style's metadata. Its structure is based on the Atom Syndication Format.

In independent styles, cs:info has the following child elements:

cs:author and cs:contributor (optional) cs:author and cs:contributor, used to respectively acknowledge style authors and contributors, may each be used multiple times. Within these elements, the child element cs:name must appear once, while cs:email and cs:uri each may appear once. These child elements should contain respectively the name, email address and URI of the author or contributor.

cs:category (optional) Styles may be assigned one or more categories. cs:category may be used once to describe how in-text citations are rendered, using the citation-format attribute set to one of the following values:

- "author-date" - e.g. "... (Doe, 1999)"
- "author" - e.g. "... (Doe)"
- "numeric" - e.g. "... [1]"
- "label" - e.g. "... [doe99]"
- "note" - the citation appears as a footnote or endnote

cs:category may be used multiple times with the field attribute, set to one of the discipline categories (see *Appendix I - Categories*), to indicates the field(s) for which the style is relevant.

cs:id Must appear once. The element should contain a URI to establish the identity of the style. A stable, unique and dereferenceable URI is desired for publicly available styles.

cs:issn/cs:eissn/cs:issnl (optional) The cs:issn element may be used multiple times to indicate the ISSN identifier(s) of the journal for which the style was written. The cs:eissn and cs:issnl elements may each be used once for the eISSN and ISSN-L identifiers, respectively.

cs:link (optional) May be used multiple times. cs:link must carry two attributes: href, set to a URI (usually a URL), and rel, whose value indicates how the URI relates to the style. The possible values of rel:

- "self" - style URI
- "template" - URI of the style from which the current style is derived
- "documentation" - URI of style documentation

The cs:link element may contain content describing the link.

cs:published (optional) May appear once. The contents of cs:published must be a timestamp, indicating when the style was initially created or made available.

cs:rights (optional) May appear once. The contents of cs:rights specifies the license under which the style file is released. The element may carry a license attribute to specify the URI of the license.

cs:summary (optional) May appear once. The contents of cs:summary gives a (short) description of the style.

cs:title Must appear once. The contents of cs:title should be the name of the style as shown to users.

cs:title-short (optional) May appear once. The contents of cs:title-short should be a shortened style name (e.g. "APA").

cs:updated Must appear once. The contents of cs:updated must be a timestamp that shows when the style was last updated.

The cs:link, cs:rights, cs:summary, cs:title and cs:title-short elements may carry a xml:lang attribute to specify the language of the element's content (the value must be an xsd:language locale code). For cs:link, the attribute can also be used to indicate the language of the link target.

In *dependent styles*, cs:link must be used with rel set to "independent-parent", with the URI of the independent parent style set on href. In addition, cs:link may not be used with rel set to "template".

An example of cs:info for an independent style:

```
<info>
  <title>Style Title</title>
  <id>http://www.zotero.org/styles/style-title</id>
  <link href="http://www.zotero.org/styles/style-title" rel="self"/>
  <link href="http://www.example.org/instructions-to-authors#references" rel=
↪"documentation"/>
  <author>
    <name>Author Name</name>
    <email>name@example.org</email>
    <uri>http://www.example.org/name</uri>
  </author>
  <category citation-format="author-date"/>
  <category field="zoology"/>
  <updated>2011-10-29T21:01:24+00:00</updated>
  <rights license="http://creativecommons.org/licenses/by-sa/3.0/">This work
  is licensed under a Creative Commons Attribution-ShareAlike 3.0 License</rights>
</info>
```

Citation

The cs:citation element describes the formatting of citations, which consist of one or more references ("cites") to bibliographic sources. Citations appear in the form of either in-text citations (in the author (e.g. "[Doe]"), author-date ("[Doe 1999]"), label ("[doe99]") or number ("[1]") format) or notes. The required cs:layout child element describes what, and how, bibliographic data should be included in the citations (see *Layout*). cs:layout may be preceded by a cs:sort element, which can be used to specify how cites within a citation should be sorted (see *Sorting*). The cs:citation element may carry attributes for *Citation-specific Options* and *Inheritable Name Options*. An example of a cs:citation element:

```
<citation>
  <sort>
    <key variable="citation-number"/>
  </sort>
  <layout>
    <text variable="citation-number"/>
  </layout>
</citation>
```

A note to CSL processor developers In note styles, a citation is often a sentence by itself. Therefore, the first character of a citation should preferably be uppercased when there is no preceding text in the note. In all other cases (e.g. when a citation is inserted into the middle of a pre-existing footnote), the citation should be printed as is.

Bibliography

The `cs:bibliography` element describes the formatting of bibliographies, which list one or more bibliographic sources. The required `cs:layout` child element describes how each bibliographic entry should be formatted. `cs:layout` may be preceded by a `cs:sort` element, which can be used to specify how references within the bibliography should be sorted (see *Sorting*). The `cs:bibliography` element may carry attributes for *Bibliography-specific Options* and *Inheritable Name Options*. An example of a `cs:bibliography` element:

```
<bibliography>
  <sort>
    <key macro="author"/>
  </sort>
  <layout>
    <group delimiter=". ">
      <text macro="author"/>
      <text variable="title"/>
    </group>
  </layout>
</bibliography>
```

Macro

Macros, defined with `cs:macro` elements, contain formatting instructions. Macros can be called with `cs:text` from within other macros and the `cs:layout` element of `cs:citation` and `cs:bibliography`, and with `cs:key` from within `cs:sort` of `cs:citation` and `cs:bibliography`. It is recommended to place macros after any `cs:locale` elements and before the `cs:citation` element.

Macros are referenced by the value of the required `name` attribute on `cs:macro`. The `cs:macro` element must contain one or more *rendering elements*.

The use of macros can improve style readability, compactness and maintainability. It is recommended to keep the contents of `cs:citation` and `cs:bibliography` compact and agnostic of item types (e.g. books, journal articles, etc.) by depending on macro calls. To allow for easy reuse of macros in other styles, it is recommended to use common macro names.

In the example below, cites consist of the item title, rendered in italics when the item type is "book":

```
<style>
  <macro name="title">
    <choose>
      <if type="book">
        <text variable="title" font-style="italic"/>
      </if>
      <else>
        <text variable="title"/>
      </else>
    </choose>
  </macro>
  <citation>
    <layout>
      <text macro="title"/>
    </layout>
  </citation>
</style>
```

Locale

Localization data, by default drawn from the "locales-xx-XX.xml" locale files, may be redefined or supplemented with `cs:locale` elements, which should be placed directly after the `cs:info` element.

The value of the optional `xml:lang` attribute on `cs:locale`, which must be set to an xsd:language locale code, determines which languages or language dialects are affected (see *Locale Fallback*).

See *Terms*, *Localized Date Formats* and *Localized Options* for further details on the use of `cs:locale`.

An example of `cs:locale` in a style:

```
<style>
  <locale xml:lang="en">
    <terms>
      <term name="editortranslator" form="short">
        <single>ed. &amp; trans.</single>
        <multiple>eds. &amp; trans.</multiple>
      </term>
    </terms>
  </locale>
</style>
```

Locale Fallback

Locale files provide localization data for language dialects (e.g. "en-US" for American English), whereas the optional `cs:locale` elements in styles can either lack the `xml:lang` attribute, or have it set to either a language (e.g. "en" for English) or dialect. Locale fallback is the mechanism determining from which of these sources each localizable unit (a date format, localized option, or specific form of a term) is retrieved.

For dialects of the same language, one is designated the primary dialect. All others are secondaries. At the moment of writing, the available locale files include:

Primary dialect	Secondary dialect(s)
de-DE (German)	de-AT (Austria), de-CH (Switzerland)
en-US (English)	en-GB (UK)
es-ES (Spanish)	es-CL (Chile), es-MX (Mexico)
fr-FR (French)	fr-CA (Canada)
pt-PT (Portuguese)	pt-BR (Brazil)
zh-CN (Chinese)	zh-TW (Taiwan)

Locale fallback is best described with an example. If the chosen output locale is "de-AT" (Austrian German), localizable units are individually drawn from the following sources, in decreasing order of priority:

1. In-style `cs:locale` elements
   (a) `xml:lang` set to chosen dialect, "de-AT"
   (b) `xml:lang` set to matching language, "de" (German)
   (c) `xml:lang` not set
2. Locale files
   (d) `xml:lang` set to chosen dialect, "de-AT"
   (e) `xml:lang` set to matching primary dialect, "de-DE" (Standard German) (only applicable when the chosen locale is a secondary dialect)

(f) `xml:lang` set to "en-US" (American English)

If the chosen output locale is a language (e.g. "de"), the (primary) dialect is used in step 1 (e.g. "de-DE").

Fallback stops once a localizable unit has been found. For terms, this even is the case when they are defined as empty strings (e.g. `<term name="and"/>` or `<term name="and"></term>`). Locale fallback takes precedence over fallback of term forms (see *Terms*).

3.6 Locale Files - Structure

While localization data can be included in styles (see *Locale*), locale files conveniently provide sets of default localization data, consisting of terms, date formats and grammar options.

Each locale file contains localization data for a single language dialect. This locale code is set on the required `xml:lang` attribute on the `cs:locale` root element. The same locale code must also be used in the file name of the locale file (the "xx-XX" in "locales-xx-XX.xml"). The root element must carry the `version` attribute, indicating the CSL version of the locale file (must be "1.0" for CSL 1.0-compatible locale files). Locale files have the same requirements for *namespacing* as styles. The `cs:locale` element may contain `cs:info` as its first child element, and requires the child elements `cs:terms`, `cs:date` and `cs:style-options` (these elements are described below). An example showing part of a locale file:

```
<?xml version="1.0" encoding="UTF-8"?>
<locale xml:lang="en-US" version="1.0" xmlns="http://purl.org/net/xbiblio/csl">
  <style-options punctuation-in-quote="true"/>
  <date form="text">
    <date-part name="month" suffix=" "/>
    <date-part name="day" suffix=", "/>
    <date-part name="year"/>
  </date>
  <date form="numeric">
    <date-part name="year"/>
    <date-part name="month" form="numeric" prefix="-" range-delimiter="/"/>
    <date-part name="day" prefix="-" range-delimiter="/"/>
  </date>
  <terms>
    <term name="no date">n.d.</term>
    <term name="et-al">et al.</term>
    <term name="page">
      <single>page</single>
      <multiple>pages</multiple>
    </term>
    <term name="page" form="short">
      <single>p.</single>
      <multiple>pp.</multiple>
    </term>
  </terms>
</locale>
```

3.6.1 Info

The `cs:info` element may be used to give metadata on the locale file. It has the following child elements:

`cs:translator` (optional) `cs:translator`, used to acknowledge locale translators, may be used multiple times. Within the element, the child element `cs:name` must appear once, while `cs:email` and `cs:uri`

each may appear once. These child elements should contain respectively the name, email address and URI of the translator.

`cs:rights` (optional) May appear once. The contents of `cs:rights` specifies the license under which the locale file is released. The element may carry a `license` attribute to specify the URI of the license, and a `xml:lang` attribute to specify the language of the element's content (the value must be an xsd:language locale code).

`cs:updated` (optional) May appear once. The contents of `cs:updated` must be a timestamp that shows when the locale file was last updated.

3.6.2 Terms

Terms are localized strings (e.g. by using the "and" term, "Doe and Smith" automatically becomes "Doe und Smith" when the style locale is switched from English to German). Terms are defined with `cs:term` elements, child elements of `cs:terms`. Each `cs:term` element must carry a `name` attribute, set to one of the terms listed in *Appendix II - Terms*.

Terms are either directly defined in the content of `cs:term`, or, in cases where singular and plural variants are needed (e.g. "page" and "pages"), in the content of the child elements `cs:single` and `cs:multiple`, respectively.

Terms may be defined for specific forms by using `cs:term` with the optional `form` attribute set to:

- "long" - (default), e.g. "editor" and "editors" for the "editor" term
- "short" - e.g. "ed." and "eds." for the term "editor"
- "verb" - e.g. "edited by" for the term "editor"
- "verb-short" - e.g. "ed." for the term "editor"
- "symbol" - e.g. "§" and "§§" for the term "section"

If a style uses a term in a form that is undefined (even after *Locale Fallback*), there is fallback to other forms: "verb-short" first falls back to "verb", "symbol" first falls back to "short", and "verb" and "short" both fall back to "long". If no locale or form fallback is available, the term is rendered as an empty string.

The `match`, `gender`, and `gender-form` attributes can be used on `cs:term` for the formatting of number variables rendered as ordinals (e.g. "first", "2nd"). See *Ordinal Suffixes* and *Gender-specific Ordinals* below.

Term content should not contain markup such as LaTeX or HTML. Superscripted Unicode characters can be used for superscripting.

Ordinal Suffixes

Number variables can be rendered with `cs:number` in the "ordinal" form, e.g. "2nd" (see *Number*). The ordinal suffixes ("nd" for "2nd") are defined with terms.

The "ordinal" term defines the default ordinal suffix. This default suffix may be overridden for certain numbers with the following terms:

- "ordinal-00" through "ordinal-09" - by default, a term in this group is used when the last digit in the term name matches the last digit of the rendered number. E.g. "ordinal-00" would match the numbers "0", "10", "20", etc. By setting the optional `match` attribute to "last-two-digits" ("last-digit" is the default), matches are limited to numbers where the two last digits agree ("0", "100", "200", etc.). When `match` is set to "whole-number", there is only a match if the number is the same as that of the term.
- "ordinal-10" through "ordinal-99" - by default, a term in this group is used when the last two digits in the term name match the last two digits of the rendered number. When the optional `match` attribute is set to "whole-number" ("last-two-digits" is the default), there is only a match if the number is the same as that of the term.

When a number has matching terms from both groups (e.g. "13" can match "ordinal-03" and "ordinal-13"), the term from the "ordinal-10" through "ordinal-99" group is used.

Ordinal terms work differently in CSL 1.0.1 than they did in CSL 1.0. When neither the style or locale file define the "ordinal" term, but do define the terms "ordinal-01" through "ordinal-04", the original CSL 1.0 scheme is used: "ordinal-01" is used for numbers ending on a 1 (except those ending on 11), "ordinal-02" for those ending on a 2 (except those ending on 12), "ordinal-03" for those ending on a 3 (except those ending on 13) and "ordinal-04" for all other numbers.

Gender-specific Ordinals

Some languages use gender-specific ordinals. For example, the English "1st" and "first" translate in French to "1er" and "premier" if the target noun is masculine, and "1re" and "première" if the noun is feminine.

Feminine and masculine variants of the ordinal terms (see *Ordinals*) may be specified by setting the `gender-form` attribute to "feminine" or "masculine" (the term without `gender-form` represents the neuter variant). There are two types of target nouns: a) the terms accompanying the *number variables*, and b) the month terms (see *Months*). The gender of these nouns may be specified on the "long" (default) form of the term using the `gender` attribute (set to "feminine" or "masculine"). When a number variable is rendered with `cs:number` in the "ordinal" or "long-ordinal" form, the ordinal term of the same gender is used, with a fallback to the neuter variant if the feminine or masculine variant is undefined. When the "day" date-part is rendered in the "ordinal" form, the ordinal gender is matched against that of the month term.

The example below gives "1re éd." ("1st ed."), "1er janvier" ("January 1st"), and "3e édition" ("3rd edition"):

```
<?xml version="1.0" encoding="UTF-8"?>
<locale xml:lang="fr-FR">
  <terms>
    <term name="edition" gender="feminine">
      <single>édition</single>
      <multiple>éditions</multiple>
    </term>
    <term name="edition" form="short">éd.</term>
    <term name="month-01" gender="masculine">janvier</term>
    <term name="ordinal">e</term>
    <term name="ordinal-01" gender-form="feminine" match="whole-number">re</term>
    <term name="ordinal-01" gender-form="masculine" match="whole-number">er</term>
  </terms>
</locale>
```

3.6.3 Localized Date Formats

Two localized date formats can be defined with `cs:date` elements: a "numeric" (e.g. "12-15-2005") and a "text" format (e.g. "December 15, 2005"). The format is set on `cs:date` with the required `form` attribute.

A date format is constructed using `cs:date-part` child elements (see *Date-part*). With a required `name` attribute set to either `day`, `month` or `year`, the order of these elements reflects the display order of respectively the day, month, and year. The date can be formatted with *formatting* and *text-case* attributes on the `cs:date` and `cs:date-part` elements. The *delimiter* attribute may be set on `cs:date` to specify the delimiter for the `cs:date-part` elements, and *affixes* may be applied to the `cs:date-part` elements.

Note Affixes are not allowed on `cs:date` when defining localized date formats. This restriction is in place to separate locale-specific affixes (set on the `cs:date-part` elements) from any style-specific affixes (set on the calling `cs:date` element), such as parentheses. An example of a macro calling a localized date format:

```
<macro name="issued">
 <date variable="issued" form="numeric" prefix="(" suffix=")"/>
</macro>
```

3.6.4 Localized Options

There are two localized options, `limit-day-ordinals-to-day-1` and `punctuation-in-quote` (see *Locale Options*). These global options (which affect both citations and the bibliography) are set as optional attributes on `cs:style-options`.

3.7 Rendering Elements

Rendering elements specify which, and in what order, pieces of bibliographic metadata are included in citations and bibliographies, and offer control over their formatting.

3.7.1 Layout

The `cs:layout` rendering element is a required child element of `cs:citation` and `cs:bibliography`. It must contain one or more of the other rendering elements described below, and may carry *affixes* and *formatting* attributes. When used within `cs:citation`, the *delimiter* attribute may be used to specify a delimiter for cites within a citation. For example, a citation like "(1, 2)" can be achieved with:

```
<citation>
  <layout prefix="(" suffix=")" delimiter=", ">
    <text variable="citation-number"/>
  </layout>
</citation>
```

3.7.2 Text

The `cs:text` rendering element outputs text. It must carry one of the following attributes to select what should be rendered:

- `variable` - renders the text contents of a variable. Attribute value must be one of the *standard variables*. May be accompanied by the `form` attribute to select the "long" (default) or "short" form of a variable (e.g. the full or short title). If the "short" form is selected but unavailable, the "long" form is rendered instead.
- `macro` - renders the text output of a macro. Attribute value must match the value of the `name` attribute of a `cs:macro` element (see *Macro*).
- `term` - renders a term. Attribute value must be one of the terms listed in *Appendix II - Terms*. May be accompanied by the `plural` attribute to select the singular ("false", default) or plural ("true") variant of a term, and by the `form` attribute to select the "long" (default), "short", "verb", "verb-short" or "symbol" form variant (see also *Terms*).
- `value` - renders the attribute value itself.

An example of `cs:text` rendering the "title" variable:

```
<text variable="title"/>
```

`cs:text` may also carry *affixes*, *display*, *formatting*, *quotes*, *strip-periods* and *text-case* attributes.

3.7.3 Date

The `cs:date` rendering element outputs the date selected from the list of *date variables* with the required `variable` attribute. A date can be rendered in either a localized or non-localized format.

Localized date formats are selected with the optional `form` attribute, which must set to either "numeric" (for fully numeric formats, e.g. "12-15-2005"), or "text" (for formats with a non-numeric month, e.g. "December 15, 2005"). Localized date formats can be customized in two ways. First, the `date-parts` attribute may be used to show fewer date parts. The possible values are:

- "year-month-day" - (default), renders the year, month and day
- "year-month" - renders the year and month
- "year" - renders the year

Secondly, `cs:date` may have one or more `cs:date-part` child elements (see *Date-part*). The attributes set on these elements override those specified for the localized date formats (e.g. to get abbreviated months for all locales, the `form` attribute on the month-`cs:date-part` element can be set to "short"). These `cs:date-part` elements do not affect which, or in what order, date parts are rendered. *Affixes*, which are very locale-specific, are not allowed on these `cs:date-part` elements.

In the absence of the `form` attribute, `cs:date` describes a self-contained non-localized date format. In this case, the date format is constructed using `cs:date-part` child elements. With a required `name` attribute set to either `day`, `month` or `year`, the order of these elements reflects the display order of respectively the day, month, and year. The date can be formatted with *formatting* attributes on the `cs:date-part` elements, as well as several `cs:date-part`-specific attributes (see *Date-part*). The *delimiter* attribute may be set on `cs:date` to specify the delimiter for the `cs:date-part` elements, and *affixes* may be applied to the `cs:date-part` elements.

For both localized and non-localized dates, `cs:date` may carry *affixes*, *display*, *formatting* and *text-case* attributes.

Date-part

The `cs:date-part` elements control how date parts are rendered. Unless the parent `cs:date` element calls a localized date format, they also determine which, and in what order, date parts appear. A `cs:date-part` element describes the date part selected with the required `name` attribute:

"day" For "day", `cs:date-part` may carry the `form` attribute, with values:

- "numeric" - (default), e.g. "1"
- "numeric-leading-zeros" - e.g. "01"
- "ordinal" - e.g. "1st"

Some languages, such as French, only use the "ordinal" form for the first day of the month ("1er janvier", "2 janvier", "3 janvier", etc.). Such output can be achieved with the "ordinal" form and use of the `limit-day-ordinals-to-day-1` attribute (see *Locale Options*).

"month" For "month", `cs:date-part` may carry the *strip-periods* and `form` attributes. In locale files, month abbreviations (the "short" form of the month *terms*) should be defined with periods if applicable (e.g. "Jan.", "Feb.", etc.). These periods can be removed by setting *strip-periods* to "true" ("false" is the default). The `form` attribute can be set to:

- "long" - (default), e.g. "January"
- "short" - e.g. "Jan."
- "numeric" - e.g. "1"
- "numeric-leading-zeros" - e.g. "01"

"year" For "year", `cs:date-part` may carry the `form` attribute, with values:

- "long" - (default), e.g. "2005"
- "short" - e.g. "05"

`cs:date-part` may also carry *formatting*, *text-case* and `range-delimiter` (see *Date Ranges*) attributes. Attributes for *affixes* are allowed, unless `cs:date` calls a localized date format.

Date Ranges

The default delimiter for dates in a date range is an en-dash (e.g. "May–July 2008"). Custom range delimiters can be set on `cs:date-part` elements with the optional `range-delimiter` attribute. When a date range is rendered, the range delimiter is drawn from the `cs:date-part` element matching the largest date part ("year", "month", or "day") that differs between the two dates. For example,

```
<style>
  <citation>
    <layout>
      <date variable="issued">
        <date-part name="day" suffix=" " range-delimiter="-"/>
        <date-part name="month" suffix=" "/>
        <date-part name="year" range-delimiter="/"/>
      </date>
    </layout>
  </citation>
</style>
```

would result in "1-4 May 2008", "May–July 2008" and "May 2008/June 2009".

AD and BC

The "ad" term (Anno Domini) is automatically appended to positive years of less than four digits (e.g. "79" becomes "79AD"). The "bc" term (Before Christ) is automatically appended to negative years (e.g. "-2500" becomes "2500BC").

Seasons

If a date includes a season instead of a month, a season term ("season-01" to "season-04", respectively Spring, Summer, Autumn and Winter) take the place of the month term. E.g.,

```
<style>
  <citation>
    <layout>
      <date variable="issued">
        <date-part name="month" suffix=" "/>
        <date-part name="year"/>
      </date>
    </layout>
  </citation>
</style>
```

would result in "May 2008" and "Winter 2009".

Approximate Dates

Approximate dates test "true" for the `is-uncertain-date` conditional (see *Choose*). For example,

```
<style>
  <citation>
    <layout>
      <choose>
        <if is-uncertain-date="issued">
          <text term="circa" form="short" suffix=" "/>
        </if>
      </choose>
      <date variable="issued">
        <date-part name="year"/>
      </date>
    </layout>
  </citation>
</style>
```

would result in "2005" (normal date) and "ca. 2003" (approximate date).

3.7.4 Number

The `cs:number` rendering element outputs the number variable selected with the required `variable` attribute. *Number variables* are a subset of the list of *standard variables*.

If a number variable is rendered with `cs:number` and only contains numeric content (as determined by the rules for `is-numeric`, see *Choose*), the number(s) are extracted. Variable content is rendered "as is" when the variable contains any non-numeric content (e.g. "Special edition").

During the extraction, numbers separated by a hyphen are stripped of intervening spaces ("2 - 4" becomes "2-4"). Numbers separated by a comma receive one space after the comma ("2,3" and "2 , 3" become "2, 3"), while numbers separated by an ampersand receive one space before and one after the ampersand ("2&3" becomes "2 & 3").

Extracted numbers can be formatted via the optional `form` attribute, with values:

- "numeric" - (default), e.g. "1", "2", "3"
- "ordinal" - e.g. "1st", "2nd", "3rd". Ordinal suffixes are defined with terms (see *Ordinal Suffixes*).
- "long-ordinal" - e.g. "first", "second", "third". Long ordinals are defined with the *terms* "long-ordinal-01" to "long-ordinal-10", which are used for the numbers 1 through 10. For other numbers "long-ordinal" falls back to "ordinal".
- "roman" - e.g. "i", "ii", "iii"

Numbers with prefixes or suffixes are never ordinalized or rendered in roman numerals (e.g. "2E" remains "2E). Numbers without affixes are individually transformed ("2, 3" can become "2nd, 3rd", "second, third" or "ii, iii").

`cs:number` may carry *affixes*, *display*, *formatting* and *text-case* attributes.

3.7.5 Names

The `cs:names` rendering element outputs the contents of one or more *name variables* (selected with the required `variable` attribute), each of which can contain multiple names (e.g. the "author" variable contains all the author names of the cited item). If multiple variables are selected (separated by single spaces, see example below), each variable is independently rendered in the order specified, with one exception: when the selection consists of "editor" and "translator", and when the contents of these two name variables is identical, then the contents of only one name

variable is rendered. In addition, the "editortranslator" term is used if the `cs:names` element contains a `cs:label` element, replacing the default "editor" and "translator" terms (e.g. resulting in "Doe (editor & translator)"). The *delimiter* attribute may be set on `cs:names` to separate the names of the different name variables (e.g. the semicolon in "Doe, Smith (editors); Johnson (translator)").

```
<names variable="editor translator" delimiter="; ">
  <label prefix=" (" suffix=")"/>
</names>
```

`cs:names` has four child elements (discussed below): `cs:name`, `cs:et-al`, `cs:substitute` and `cs:label`. The `cs:names` element may carry *affixes*, *display* and *formatting* attributes.

Name

The `cs:name` element, an optional child element of `cs:names`, can be used to describe the formatting of individual names, and the separation of names within a name variable. `cs:name` may carry the following attributes:

`and` Specifies the delimiter between the second to last and last name of the names in a name variable. Allowed values are "text" (selects the "and" term, e.g. "Doe, Johnson and Smith") and "symbol" (selects the ampersand, e.g. "Doe, Johnson & Smith").

`delimiter` Specifies the text string used to separate names in a name variable. Default is ", " (e.g. "Doe, Smith").

`delimiter-precedes-et-al` Determines when the name delimiter or a space is used between a truncated name list and the "et-al" (or "and others") term in case of et-al abbreviation. Allowed values:

- "contextual" - (default), name delimiter is only used for name lists truncated to two or more names
  - 1 name: "J. Doe et al."
  - 2 names: "J. Doe, S. Smith, et al."
- "after-inverted-name" - name delimiter is only used if the preceding name is inverted as a result of the `name-as-sort-order` attribute. E.g. with `name-as-sort-order` set to "first":
  - "Doe, J., et al."
  - "Doe, J., S. Smith et al."
- "always" - name delimiter is always used
  - 1 name: "J. Doe, et al."
  - 2 names: "J. Doe, S. Smith, et al."
- "never" - name delimiter is never used
  - 1 name: "J. Doe et al."
  - 2 names: "J. Doe, S. Smith et al."

`delimiter-precedes-last` Determines when the name delimiter is used to separate the second to last and the last name in name lists (if `and` is not set, the name delimiter is always used, regardless of the value of `delimiter-precedes-last`). Allowed values:

- "contextual" - (default), name delimiter is only used for name lists with three or more names
  - 2 names: "J. Doe and T. Williams"
  - 3 names: "J. Doe, S. Smith, and T. Williams"
- "after-inverted-name" - name delimiter is only used if the preceding name is inverted as a result of the `name-as-sort-order` attribute. E.g. with `name-as-sort-order` set to "first":

  - "Doe, J., and T. Williams"
  - "Doe, J., S. Smith and T. Williams"
- "always" - name delimiter is always used
  - 2 names: "J. Doe, and T. Williams"
  - 3 names: "J. Doe, S. Smith, and T. Williams"
- "never" - name delimiter is never used
  - 2 names: "J. Doe and T. Williams"
  - 3 names: "J. Doe, S. Smith and T. Williams"

et-al-min / et-al-use-first Use of these two attributes enables et-al abbreviation. If the number of names in a name variable matches or exceeds the number set on `et-al-min`, the rendered name list is truncated after reaching the number of names set on `et-al-use-first`. The "et-al" (or "and others") term is appended to truncated name lists (see also *Et-al*). By default, when a name list is truncated to a single name, the name and the "et-al" (or "and others") term are separated by a space (e.g. "Doe et al."). When a name list is truncated to two or more names, the name delimiter is used (e.g. "Doe, Smith, et al."). This behavior can be changed with the `delimiter-precedes-et-al` attribute.

et-al-subsequent-min / et-al-subsequent-use-first If used, the values of these attributes replace those of respectively `et-al-min` and `et-al-use-first` for subsequent cites (cites referencing earlier cited items).

et-al-use-last When set to "true" (the default is "false"), name lists truncated by et-al abbreviation are followed by the name delimiter, the ellipsis character, and the last name of the original name list. This is only possible when the original name list has at least two more names than the truncated name list (for this the value of `et-al-use-first`/`et-al-subsequent-min` must be at least 2 less than the value of `et-al-min`/`et-al-subsequent-use-first`). An example:

```
A. Goffeau, B. G. Barrell, H. Bussey, R. W. Davis, B. Dujon, H.
Feldmann, ... S. G. Oliver
```

The remaining attributes, discussed below, only affect personal names. Personal names require a "family" name-part, and may also contain "given", "suffix", "non-dropping-particle" and "dropping-particle" name-parts. These name-parts are defined as:

- "family" - surname minus any particles and suffixes
- "given" - given names, either full ("John Edward") or initialized ("J. E.")
- "suffix" - name suffix, e.g. "Jr." in "John Smith Jr." and "III" in "Bill Gates III"
- "non-dropping-particle" - name particles that are not dropped when only the surname is shown ("de" in the Dutch surname "de Koning") but which may be treated separately from the family name, e.g. for sorting
- "dropping-particle" - name particles that are dropped when only the surname is shown ("van" in "Ludwig van Beethoven", which becomes "Beethoven")

The attributes affecting personal names:

form Specifies whether all the name-parts of personal names should be displayed (value "long", the default), or only the family name and the non-dropping-particle (value "short"). A third value, "count", returns the total number of names that would otherwise be rendered by the use of the `cs:names` element (taking into account the effects of et-al abbreviation and editor/translator collapsing), which allows for advanced *sorting*.

initialize When set to "false" (the default is "true"), given names are no longer initialized when "initialize-with" is set. However, the value of "initialize-with" is still added after initials present in the full name (e.g. with `initialize` set to "false", and `initialize-with` set to ".", "James T Kirk" becomes "James T. Kirk").

initialize-with When set, given names are converted to initials. The attribute value is added after each initial (“.” results in “J.J. Doe”). For compound given names (e.g. “Jean-Luc”), hyphenation of the initials can be controlled with the global `initialize-with-hyphen` option (see *Hyphenation of Initialized Names*).

name-as-sort-order Specifies that names should be displayed with the given name following the family name (e.g. “John Doe” becomes “Doe, John”). The attribute has two possible values:

- “first” - attribute only has an effect on the first name of each name variable
- “all” - attribute has an effect on all names

Note that even when `name-as-sort-order` changes the name-part order, the display order is not necessarily the same as the sorting order for names containing particles and suffixes (see *Name-part order*). Also, `name-as-sort-order` only affects names written in scripts where the given name typically precedes the family name, such as Latin, Greek, Cyrillic and Arabic. In contrast, names written in Asian scripts are always displayed with the family name preceding the given name.

sort-separator Sets the delimiter for name-parts that have switched positions as a result of `name-as-sort-order`. The default value is “, ” (“Doe, John”). As is the case for `name-as-sort-order`, this attribute only affects names in scripts that know “given-name family-name” order.

`cs:name` may also carry *affixes* and *formatting* attributes.

Name-part Order

The order of name-parts depends on the values of the `form` and `name-as-sort-order` attributes on `cs:name`, the value of the `demote-non-dropping-particle` attribute on `cs:style` (one of the *global options*), and the script of the individual name. Note that the display and sorting order of name-parts often differs. An overview of the possible orders:

Display order of names in “given-name family-name” scripts (Latin, etc.)

Conditions `form` set to “long”

Order

1. given
2. dropping-particle
3. non-dropping-particle
4. family
5. suffix

Example [Jean] [de] [La] [Fontaine] [III]

Conditions `form` set to “long”, name-as-sort-order active, `demote-non-dropping-particle` set to “never” or “sort-only”

Order

1. non-dropping-particle
2. family
3. given

4. dropping-particle
5. suffix

Example [La] [Fontaine], [Jean] [de], [III]

Conditions `form` set to "long", name-as-sort-order active, `demote-non-dropping-particle` set to "display-and-sort"

Order

1. family
2. given
3. dropping-particle
4. non-dropping-particle
5. suffix

Example [Fontaine], [Jean] [de] [La], [III]

Conditions `form` set to "short"

Order

1. non-dropping-particles
2. family

Example [La] [Fontaine]

Sorting order of names in "given-name family-name" scripts (Latin, etc.)

N.B. The sort keys are listed in descending order of priority.

Conditions `demote-non-dropping-particle` set to "never"

Order

1. non-dropping-particle + family
2. dropping-particle
3. given
4. suffix

Example [La Fontaine] [de] [Jean] [III]

Conditions `demote-non-dropping-particle` set to "sort-only" or "display-and-sort"

Order

1. family
2. dropping-particle + non-dropping-particle
3. given

4. suffix

Example [Fontaine] [de La] [Jean] [III]

Display and sorting order of names in "family-name given-name" scripts (Chinese, etc.)

Conditions `form` set to "long"

Order

1. family
2. given

Example [Mao Zedong]

Conditions `form` set to "short"

Order

1. family

Example [Mao]

Non-personal names lack name-parts and are sorted as is, although English articles ("a", "an" and "the") at the start of the name are stripped. For example, "The New York Times" sorts as "New York Times".

Name-part Formatting

The `cs:name` element may contain one or two `cs:name-part` child elements for name-part-specific formatting. `cs:name-part` must carry the `name` attribute, set to either "given" or "family".

If set to "given", *formatting* and *text-case* attributes on `cs:name-part` affect the "given" and "dropping-particle" name-parts. *affixes* surround the "given" name-part, enclosing any demoted name particles for inverted names.

If set to "family", *formatting* and *text-case* attributes affect the "family" and "non-dropping-particle" name-parts. *affixes* surround the "family" name-part, enclosing any preceding name particles, as well as the "suffix" name-part for non-inverted names.

The "suffix" name-part is not subject to name-part formatting. The use of `cs:name-part` elements does not influence which, or in what order, name-parts are rendered. An example, yielding names like "Jane DOE":

```
<names variable="author">
  <name>
    <name-part name="family" text-case="uppercase"/>
  </name>
</names>
```

Et-al

Et-al abbreviation, controlled via the `et-al-...` attributes (see *Name*), can be further customized with the optional `cs:et-al` element, which must follow the `cs:name` element (if present). The `term` attribute may be set to either "et-al" (the default) or to "and others" to use either term. The *formatting* attributes may also be used, for example to italicize the "et-al" term:

```
<names variable="author">
  <et-al term="and others" font-style="italic"/>
</names>
```

Substitute

The optional `cs:substitute` element, which must be included as the last child element of `cs:names`, adds substitution in case the *name variables* specified in the parent `cs:names` element are empty. The substitutions are specified as child elements of `cs:substitute`, and must consist of one or more *rendering elements* (with the exception of `cs:layout`). A shorthand version of `cs:names` without child elements, which inherits the attributes values set on the `cs:name` and `cs:et-al` child elements of the original `cs:names` element, may also be used. If `cs:substitute` contains multiple child elements, the first element to return a non-empty result is used for substitution. Substituted variables are suppressed in the rest of the output to prevent duplication. An example, where an empty "author" name variable is substituted by the "editor" name variable, or, when no editors exist, by the "title" macro:

```
<macro name="author">
  <names variable="author">
    <substitute>
      <names variable="editor"/>
      <text macro="title"/>
    </substitute>
  </names>
</macro>
```

Label in `cs:names`

The optional `cs:label` element (see *label*) must be included after the `cs:name` and `cs:et-al` elements, but before the `cs:substitute` element. When used as a child element of `cs:names`, `cs:label` does not carry the `variable` attribute; it uses the variable(s) set on the parent `cs:names` element instead. A second difference is that the `form` attribute may also be set to "verb" or "verb-short", so that the allowed values are:

- "long" - (default), e.g. "editor" and "editors" for the "editor" term
- "short" - e.g. "ed." and "eds." for the term "editor"
- "verb" - e.g. "edited by" for the term "editor"
- "verb-short" - e.g. "ed." for the term "editor"
- "symbol" - e.g. "§" and "§§" for the term "section"

3.7.6 Label

The `cs:label` rendering element outputs the term matching the variable selected with the required `variable` attribute, which must be set to "locator", "page", or one of the *number variables*. The term is only rendered if the selected variable is non-empty. For example,

```
<group delimiter=" ">
  <label variable="page"/>
  <text variable="page"/>
</group>
```

can result in “page 3” or “pages 5-7”. `cs:label` may carry the following attributes:

`form` Selects the form of the term, with allowed values:

- “long” - (default), e.g. “page”/”pages” for the “page” term
- “short” - e.g. “p.”/”pp.” for the “page” term
- “symbol” - e.g. “§”/”§§” for the “section” term

`plural` Sets pluralization of the term, with allowed values:

- “contextual” - (default), the term plurality matches that of the variable content. Content is considered plural when it contains multiple numbers (e.g. “page 1”, “pages 1-3”, “volume 2”, “volumes 2 & 4”), or, in the case of the “number-of-pages” and “number-of-volumes” variables, when the number is higher than 1 (“1 volume” and “3 volumes”).
- “always” - always use the plural form, e.g. “pages 1” and “pages 1-3”
- “never” - always use the singular form, e.g. “page 1” and “page 1-3”

`cs:label` may also carry *affixes*, *formatting*, *text-case* and *strip-periods* attributes.

3.7.7 Group

The `cs:group` rendering element must contain one or more *rendering elements* (with the exception of `cs:layout`). `cs:group` may carry the *delimiter* attribute to separate its child elements, as well as *affixes* and *display* attributes (applied to the output of the group as a whole) and *formatting* attributes (transmitted to the enclosed elements). `cs:group` implicitly acts as a conditional: `cs:group` and its child elements are suppressed if a) at least one rendering element in `cs:group` calls a variable (either directly or via a macro), and b) all variables that are called are empty. This accommodates descriptive `cs:text` elements. For example,

```
<layout>
  <group delimiter=" ">
    <text term="retrieved"/>
    <text term="from"/>
    <text variable="URL"/>
  </group>
</layout>
```

can result in “retrieved from http://dx.doi.org/10.1128/AEM.02591-07”, but doesn’t generate output when the “URL” variable is empty.

3.7.8 Choose

The `cs:choose` rendering element allows for conditional rendering of *rendering elements*. An example that renders the “issued” date variable when it exists, and the “no date” term when it doesn’t:

```
<choose>
  <if variable="issued">
    <date variable="issued" form="numeric"/>
  </if>
```

(continues on next page)

(continued from previous page)

```
  <else>
    <text term="no date"/>
  </else>
</choose>
```

`cs:choose` requires a `cs:if` child element, which may be followed by one or more `cs:else-if` child elements, and an optional closing `cs:else` child element. The `cs:if` and `cs:else-if` elements may contain any number of *rendering elements* (except for `cs:layout`). As an empty `cs:else` element would be superfluous, `cs:else` must contain at least one rendering element. `cs:if` and `cs:else-if` elements must carry one or more conditions, which are set with the attributes:

disambiguate When set to "true" (the only allowed value), the element content is only rendered if it disambiguates two otherwise identical citations. This attempt at *disambiguation* is only made when all other disambiguation methods have failed to uniquely identify the target source.

is-numeric Tests whether the given variables (*Appendix IV - Variables*) contain numeric content. Content is considered numeric if it solely consists of numbers. Numbers may have prefixes and suffixes ("D2", "2b", "L2d"), and may be separated by a comma, hyphen, or ampersand, with or without spaces ("2, 3", "2-4", "2 & 4"). For example, "2nd" tests "true" whereas "second" and "2nd edition" test "false".

is-uncertain-date Tests whether the given *date variables* contain *approximate dates*.

locator Tests whether the locator matches the given locator types (see *Locators*). Use "sub-verbo" to test for the "sub verbo" locator type.

position Tests whether the cite position matches the given positions (terminology: citations consist of one or more cites to individual items). When called within the scope of cs:bibliography, `position` tests "false". The positions that can be tested are:

- "first": position of cites that are the first to reference an item
- "ibid"/"ibid-with-locator"/"subsequent": cites referencing previously cited items have the "subsequent" position. Such cites may also have the "ibid" or "ibid-with-locator" position when:

  1. the current cite immediately follows on another cite, within the same citation, that references the same item

  or

  2. the current cite is the first cite in the citation, and the previous citation consists of a single cite referencing the same item

  If either requirement is met, the presence of locators determines which position is assigned:

  - **Preceding cite does not have a locator**: if the current cite has a locator, the position of the current cite is "ibid-with-locator". Otherwise the position is "ibid".
  - **Preceding cite does have a locator**: if the current cite has the same locator, the position of the current cite is "ibid". If the locator differs the position is "ibid-with-locator". If the current cite lacks a locator its only position is "subsequent".

- "near-note": position of a cite following another cite referencing the same item. Both cites have to be located in foot or endnotes, and the distance between both cites may not exceed the maximum distance (measured in number of foot or endnotes) set with the `near-note-distance` option (see *Note Distance*).

Whenever position="ibid-with-locator" tests true, position="ibid" also tests true. And whenever position="ibid" or position="near-note" test true, position="subsequent" also tests true.

type Tests whether the item matches the given types (*Appendix III - Types*).

variable Tests whether the default (long) forms of the given variables (*Appendix IV - Variables*) contain non-empty values.

With the exception of `disambiguate`, all conditions allow for multiple test values (separated with spaces, e.g. "book thesis").

The `cs:if` and `cs:else-if` elements may carry the `match` attribute to control the testing logic, with allowed values:

- "all" - (default), element only tests "true" when all conditions test "true" for all given test values
- "any" - element tests "true" when any condition tests "true" for any given test value
- "none" - element only tests "true" when none of the conditions test "true" for any given test value

3.8 Style Behavior

3.8.1 Options

Styles may be configured with *citation-specific options*, set as attributes on set on `cs:citation`, *bibliography-specific options*, set on `cs:bibliography`, and *global options* (these affect both citations and the bibliography), set on `cs:style`. *Inheritable name options* may be set on `cs:style`, `cs:citation` and `cs:bibliography`. Finally, *locale options* may be set on `cs:locale` elements.

Citation-specific Options

Disambiguation

A cite is ambiguous when it matches multiple bibliographic entries[1]. There are four methods available to eliminate such ambiguity:

1. Show more names
2. Expand names (adding initials or full given names)
3. Add a year-suffix
4. Render the cite with the `disambiguate` attribute of `cs:choose` conditions testing "true"

Method 2 can also be used for global *name disambiguation*, covering all cites, ambiguous and unambiguous, throughout the document.

Disambiguation methods are activated with the following optional attributes, and are always tried in the listed order:

disambiguate-add-names [Step (1)] If set to "true" ("false" is the default), names that would otherwise be hidden as a result of et-al abbreviation are added one by one to all members of a set of ambiguous cites, until no more cites in the set can be disambiguated by adding names.

disambiguate-add-givenname [Step (2)] If set to "true" ("false" is the default), ambiguous names (names that are identical in their "short" or initialized "long" form, but differ when initials are added or the full given name is shown) are expanded. Name expansion can be configured with `givenname-disambiguation-rule`. An example of cite disambiguation:

[1] The presence of uncited entries in the bibliography can make cites in the document ambiguous. To make sure such cites are disambiguated, the CSL processor should create hidden "ghost" cites for all uncited bibliographic entries and include them in the disambiguation process.

Original ambiguous cites	Disambiguated cites
(Simpson 2005; Simpson 2005)	(H. Simpson 2005; B. Simpson 2005)
(Doe 1950; Doe 1950)	(John Doe 1950; Jane Doe 1950)

If cites cannot be (fully) disambiguated by expanding the rendered names, and if `disambiguate-add-names` is set to "true", then the names still hidden as a result of et-al abbreviation after the disambiguation attempt of `disambiguate-add-names` are added one by one to all members of a set of ambiguous cites, until no more cites in the set can be disambiguated by adding expanded names.

`givenname-disambiguation-rule` Specifies a) whether the purpose of name expansion is limited to disambiguating cites, or has the additional goal of disambiguating names (only in the latter case are ambiguous names in unambiguous cites expanded, e.g. from "(Doe 1950; Doe 2000)" to "(Jane Doe 1950; John Doe 2000)"), b) whether name expansion targets all, or just the first name of each cite, and c) the method by which each name is expanded.

Expansion of Individual Names The steps for expanding individual names are:

1. If `initialize-with` is set and `initialize` has its default value of "true", then:

   (a) Initials can be shown by rendering the name with a `form` value of "long" instead of "short" (e.g. "Doe" becomes "J. Doe").

   (b) Full given names can be shown instead of initials by rendering the name with `initialize` set to "false" (e.g. "J. Doe" becomes "John Doe").

2. If `initialize-with` is *not* set, full given names can be shown by rendering the name with a `form` value of "long" instead of "short" (e.g. "Doe" becomes "John Doe").

Given Name Disambiguation Rules Allowed values of `givenname-disambiguation-rule`:

"all-names" Name expansion has the dual purpose of disambiguating cites and names. All rendered ambiguous names, in both ambiguous and unambiguous cites, are subject to disambiguation. Each name is progressively transformed until it is disambiguated. Names that cannot be disambiguated remain in their original form.

"all-names-with-initials" As "all-names", but name expansion is limited to showing initials (see step 1(a) above). No disambiguation attempt is made when `initialize-with` is not set or when `initialize` is set to "false".

"primary-name" As "all-names", but disambiguation is limited to the first name of each cite.

"primary-name-with-initials" As "all-names-with-initials", but disambiguation is limited to the first name of each cite.

"by-cite" Default. As "all-names", but the goal of name expansion is limited to disambiguating cites. Only ambiguous names in ambiguous cites are affected, and disambiguation stops after the first name that eliminates cite ambiguity.

`disambiguate-add-year-suffix` [Step (3)] If set to "true" ("false" is the default), an alphabetic year-suffix is added to ambiguous cites (e.g. "Doe 2007, Doe 2007" becomes "Doe 2007a, Doe 2007b") and to their corresponding bibliographic entries. The assignment of the year-suffixes follows the order of the bibliographies entries, and additional letters are used once "z" is reached ("z", "aa", "ab", ..., "az", "ba", etc.). By default the year-suffix is appended to the cite, and to the first year rendered through `cs:date` in the bibliographic entry, but its location can be controlled by explicitly rendering the "year-suffix" variable using `cs:text`. If "year-suffix" is rendered through `cs:text` in the scope of `cs:citation`, it is suppressed for `cs:bibliography`, unless it is also rendered through `cs:text` in the scope of `cs:bibliography`, and vice versa.

If ambiguous cites remain after applying the selected disambiguation methods described above, a final disambiguation attempt is made by rendering these cites with the `disambiguate` condition testing "true" [Step (4)] (see *Choose*).

Cite Grouping

With cite grouping, cites in in-text citations with identical rendered names are grouped together, e.g. the year-sorted "(Doe 1999; Smith 2002; Doe 2006; Doe et al. 2007)" becomes "(Doe 1999; Doe 2006; Smith 2002; Doe et al. 2007)". The comparison is limited to the output of the (first) `cs:names` element, but includes output rendered through `cs:substitute`. Cite grouping takes places after cite sorting and disambiguation. Grouped cites maintain their relative order, and are moved to the original location of the first cite of the group.

Cite grouping can be activated by setting the `cite-group-delimiter` attribute or the `collapse` attributes on `cs:citation` (see also *Cite Collapsing*).

`cite-group-delimiter` Activates cite grouping and specifies the delimiter for cites within a cite group. Defaults to ", ". E.g. with `delimiter` on `cs:layout` in `cs:citation` set to "; ", `collapse` set to "year", and `cite-group-delimiter` set to ",", citations look like "(Doe 1999,2001; Jones 2000)".

Cite Collapsing

Cite groups (author and author-date styles), and numeric cite ranges (numeric styles) can be collapsed through the use of the `collapse` attribute. Delimiters for collapsed cite groups can be customized with the `year-suffix-delimiter` and `after-collapse-delimiter` attributes:

`collapse` Activates cite grouping and collapsing. Allowed values:

- "citation-number" - collapses ranges of cite numbers (rendered through the "citation-number" variable) in citations for "numeric" styles (e.g. from "[1, 2, 3, 5]" to "[1–3, 5]"). Only increasing ranges collapse, e.g. "[3, 2, 1]" will not collapse (to see how to sort cites by "citation-number", see *Sorting*).
- "year" - collapses cite groups by suppressing the output of the `cs:names` element for subsequent cites in the group, e.g. "(Doe 2000, Doe 2001)" becomes "(Doe 2000, 2001)".
- "year-suffix" - collapses as "year", but also suppresses repeating years within the cite group, e.g. "(Doe 2000a, b)" instead of "(Doe 2000a, 2000b)".
- "year-suffix-ranged" - collapses as "year-suffix", but also collapses ranges of year-suffixes, e.g. "(Doe 2000a–c,e)" instead of "(Doe 2000a, b, c, e)".

"year-suffix" and "year-suffix-ranged" fall back to "year" when `disambiguate-add-year-suffix` is "false" (see *Disambiguation*), or when a cite has a locator (e.g. "(Doe 2000a-c, 2000d, p. 5, 2000e,f)", where the cite for "Doe 2000d" has a locator that prevents the cite from further collapsing).

`year-suffix-delimiter` Specifies the delimiter for year-suffixes. Defaults to the delimiter set on `cs:layout` in `cs:citation`. E.g. with `collapse` set to "year-suffix", `delimiter` on `cs:layout` in `cs:citation` set to "; ", and `year-suffix-delimiter` set to ",", citations look like "(Doe 1999a,b; Jones 2000)".

`after-collapse-delimiter` Specifies the cite delimiter to be used *after* a collapsed cite group. Defaults to the delimiter set on `cs:layout` in `cs:citation`. E.g. with `collapse` set to "year", `delimiter` on `cs:layout` in `cs:citation` set to ", ", and `after-collapse-delimiter` set to "; ", citations look like "(Doe 1999, 2001; Jones 2000, Brown 2001)".

Note Distance

`near-note-distance` A cite tests true for the "near-note" position (see *Choose*) when a preceding note exists that a) refers to the same item and b) does not precede the current note by more footnotes or endnotes than the value of `near-note-distance` (default value is "5").

Bibliography-specific Options

Whitespace

hanging-indent If set to "true" ("false" is the default), bibliographic entries are rendered with hanging-indents.

second-field-align If set, subsequent lines of bibliographic entries are aligned along the second field. With "flush", the first field is flush with the margin. With "margin", the first field is put in the margin, and subsequent lines are aligned with the margin. An example, where the first field is `<text variable="citation-number" suffix=". "/>`:

```
9.  Adams, D. (2002). The Ultimate Hitchhiker's Guide to the
    Galaxy (1st ed.).
10. Asimov, I. (1951). Foundation.
```

line-spacing Specifies vertical line distance. Defaults to "1" (single-spacing), and can be set to any positive integer to specify a multiple of the standard unit of line height (e.g. "2" for double-spacing).

entry-spacing Specifies vertical distance between bibliographic entries. By default (with a value of "1"), entries are separated by a single additional line-height (as set by the line-spacing attribute). Can be set to any non-negative integer to specify a multiple of this amount.

Reference Grouping

subsequent-author-substitute If set, the value of this attribute replaces names in a bibliographic entry that also occur in the preceding entry. The exact method of substitution depends on the value of the `subsequent-author-substitute-rule` attribute. Substitution is limited to the names of the first `cs:names` element rendered.

subsequent-author-substitute-rule Specifies when and how names are substituted as a result of `subsequent-author-substitute`. Allowed values:

- "complete-all" - (default), when all rendered names of the name variable match those in the preceding bibliographic entry, the value of `subsequent-author-substitute` replaces the entire name list (including punctuation and terms like "et al" and "and"), except for the affixes set on the `cs:names` element.
- "complete-each" - requires a complete match like "complete-all", but now the value of `subsequent-author-substitute` substitutes for each rendered name.
- "partial-each" - when one or more rendered names in the name variable match those in the preceding bibliographic entry, the value of `subsequent-author-substitute` substitutes for each matching name. Matching starts with the first name, and continues up to the first mismatch.
- "partial-first" - as "partial-each", but substitution is limited to the first name of the name variable.

For example, take the following bibliographic entries:

```
Doe. 1999.
Doe. 2000.
Doe, Johnson & Williams. 2001.
Doe & Smith. 2002.
Doe, Stevens & Miller. 2003.
Doe, Stevens & Miller. 2004.
Doe, Williams et al. 2005.
Doe, Williams et al. 2006.
```

With `subsequent-author-substitute` set to "—", and `subsequent-author-substitute-rule` set to "complete-all", this becomes:

```
Doe. 1999.
---. 2000.
Doe, Johnson & Williams. 2001.
Doe & Smith. 2002.
Doe, Stevens & Miller. 2003.
---. 2004.
Doe, Williams et al. 2005.
---. 2005.
```

With `subsequent-author-substitute-rule` set to "complete-each", this becomes:

```
Doe. 1999.
---. 2000.
Doe, Johnson & Williams. 2001.
Doe & Smith. 2002.
Doe, Stevens & Miller. 2003.
---, --- & ---. 2004.
Doe, Williams et al. 2005.
---, --- et al. 2006.
```

With `subsequent-author-substitute-rule` set to "partial-each", this becomes:

```
Doe. 1999.
---. 2000.
Doe, Johnson & Williams. 2001.
--- & Smith. 2002.
Doe, Stevens & Miller. 2003.
---, --- & ---. 2004.
Doe, Williams et al. 2005.
---, --- et al. 2005.
```

With `subsequent-author-substitute-rule` set to "partial-first", this becomes:

```
Doe. 1999.
---. 2000.
Doe, Johnson & Williams. 2001.
--- & Smith. 2002.
Doe, Stevens & Miller. 2003.
---, Stevens & Miller. 2004.
Doe, Williams et al. 2005.
---, Williams et al. 2005.
```

Global Options

Hyphenation of Initialized Names

`initialize-with-hyphen` Specifies whether compound given names (e.g. "Jean-Luc") should be initialized with a hyphen ("J.-L.", value "true", default) or without ("J.L.", value "false").

Page Ranges

page-range-format Activates expansion or collapsing of page ranges: “chicago” (“321–28”), “expanded” (e.g. “321–328”), “minimal” (“321–8”), or “minimal-two” (“321–28”) (see also *Appendix V - Page Range Formats*). Delimits page ranges with the “page-range-delimiter” term (introduced with CSL 1.0.1, and defaults to an en-dash). If the attribute is not set, page ranges are rendered without reformatting.

Name Particles

Western names frequently contain one or more name particles (e.g. “de” in the Dutch name “W. de Koning”). These name particles can be either kept or dropped when only the surname is shown: these two types are referred to as non-dropping and dropping particles, respectively. A single name can contain particles of both types (with non-dropping particles always following dropping particles). For example, “W. de Koning” and the French name “Jean de La Fontaine” can be deconstructed into:

```
{
    "author": [
        {
            "given": "W.",
            "non-dropping-particle": "de",
            "family": "Koning"
        },
        {
            "given": "Jean",
            "dropping-particle": "de",
            "non-dropping-particle": "La",
            "family": "Fontaine"
        }
    ]
}
```

When just the surname is shown, only the non-dropping-particle is kept: “De Koning” and “La Fontaine”.

In the case of inverted names, where the family name precedes the given name, the dropping-particle is always appended to the family name, but the non-dropping-particle can be either prepended (e.g. “de Koning, W.”) or appended (after initials or given names, e.g. “Koning, W. de”). For inverted names where the non-dropping-particle is prepended, names can either be sorted by keeping the non-dropping-particle together with the family name as part of the primary sort key (sort order A), or by separating the non-dropping-particle from the family name and have it become (part of) a secondary sort key, joining the dropping-particle, if available (sort order B):

Sort order A: non-dropping-particle not demoted

- primary sort key: “La Fontaine”
- secondary sort key: “de”
- tertiary sort key: “Jean”

Sort order B: non-dropping-particle demoted

- primary sort key: “Fontaine”
- secondary sort key: “de La”
- tertiary sort key: “Jean”

The handling of the non-dropping-particle can be customized with the `demote-non-dropping-particle` option:

demote-non-dropping-particle Sets the display and sorting behavior of the non-dropping-particle in inverted names (e.g. "Koning, W. de"). Allowed values:

- "never": the non-dropping-particle is treated as part of the family name, whereas the dropping-particle is appended (e.g. "de Koning, W.", "La Fontaine, Jean de"). The non-dropping-particle is part of the primary sort key (sort order A, e.g. "de Koning, W." appears under "D").
- "sort-only": same display behavior as "never", but the non-dropping-particle is demoted to a secondary sort key (sort order B, e.g. "de Koning, W." appears under "K").
- "display-and-sort" (default): the dropping and non-dropping-particle are appended (e.g. "Koning, W. de" and "Fontaine, Jean de La"). For name sorting, all particles are part of the secondary sort key (sort order B, e.g. "Koning, W. de" appears under "K").

Some names include a particle that should never be demoted. For these cases the particle should just be included in the family name field, for example for the French general Charles de Gaulle:

```
{
    "author": [
        {
            "family": "de Gaulle",
            "given": "Charles"
        }
    ]
}
```

Inheritable Name Options

Attributes for the `cs:names` and `cs:name` elements may also be set on `cs:style`, `cs:citation` and `cs:bibliography`. This eliminates the need to repeat the same attributes and attribute values for every occurrence of the `cs:names` and `cs:name` elements.

The available inheritable attributes for `cs:name` are `and`, `delimiter-precedes-et-al`, `delimiter-precedes-last`, `et-al-min`, `et-al-use-first`, `et-al-use-last`, `et-al-subsequent-min`, `et-al-subsequent-use-first`, `initialize`, `initialize-with`, `name-as-sort-order` and `sort-separator`. The attributes `name-form` and `name-delimiter` correspond to the `form` and `delimiter` attributes on `cs:name`. Similarly, `names-delimiter` corresponds to the `delimiter` attribute on `cs:names`.

When an inheritable name attribute is set on `cs:style`, `cs:citation` or `cs:bibliography`, its value is used for all `cs:names` elements within the scope of the element carrying the attribute. If an attribute is set on multiple hierarchical levels, the value set at the lowest level is used.

Locale Options

limit-day-ordinals-to-day-1 Date formats are defined by the `cs:date` element and its `cs:date-part` child elements (see *Date*). By default, when the `cs:date-part` element with `name` set to "day" has `form` set to "ordinal", all days (1 through 31) are rendered in the ordinal form, e.g. "January 1st", "January 2nd", etc. By setting `limit-day-ordinals-to-day-1` to "true" ("false" is the default), the "ordinal" form is limited to the first day of each month (other days will use the "numeric" form). This is desirable for some languages, such as French: "1er janvier", but "2 janvier", "3 janvier", etc.

punctuation-in-quote For `cs:text` elements rendered with the `quotes` attribute set to "true" (see *Formatting*), and for which the output is followed by a comma or period, `punctuation-in-quote` specifies whether this punctuation is placed outside (value "false", default) or inside (value "true") the closing quotation mark.

3.8.2 Sorting

`cs:citation` and `cs:bibliography` may include a `cs:sort` child element before the `cs:layout` element to specify the sorting order of respectively cites within citations, and bibliographic entries within the bibliography. In the absence of `cs:sort`, cites and bibliographic entries appear in the order in which they are cited.

The `cs:sort` element must contain one or more `cs:key` child elements. The sort key, set as an attribute on `cs:key`, must be a variable (see *Appendix IV - Variables*) or macro name. For each `cs:key` element, the sort direction can be set to either "ascending" (default) or "descending" with the `sort` attribute. The attributes `names-min`, `names-use-first`, and `names-use-last` may be used to override the values of the corresponding `et-al-min`/`et-al-subsequent-min`, `et-al-use-first`/`et-al-subsequent-use-first` and `et-al-use-last` attributes, and affect all names generated via macros called by `cs:key`.

Sort keys are evaluated in sequence. A primary sort is performed on all items using the first sort key. A secondary sort, using the second sort key, is applied to items sharing the first sort key value. A tertiary sort, using the third sort key, is applied to items sharing the first and second sort key values. Sorting continues until either the order of all items is fixed, or until the sort keys are exhausted. Items with an empty sort key value are placed at the end of the sort, both for ascending and descending sorts.

An example, where cites are first sorted by the output of the "author" macro, with overriding settings for et-al abbreviation. Cites sharing the primary sort key are subsequently sorted in descending order by the "issued" date variable.

```
<citation>
  <sort>
    <key macro="author" names-min="3" names-use-first="3"/>
    <key variable="issued" sort="descending"/>
  </sort>
  <layout>
    <!-- rendering elements -->
  </layout>
</citation>
```

The sort key value of a variable or macro can differ from the "normal" rendered output. The specifics of sorting variables and macros:

Sorting Variables

The sort key value for a variable called by `cs:key` via the `variable` attribute consists of the string value, without rich text markup. Exceptions are name, date and numeric variables:

names: *Name variables* called via the `variable` attribute (e.g. `<key variable="author"/>`) are returned as a name list string, with the `cs:name` attributes `form` set to "long", and `name-as-sort-order` set to "all".

dates: *Date variables* called via the `variable` attribute are returned in the YYYYMMDD format, with zeros substituted for any missing date-parts (e.g. 20001200 for December 2000). As a result, less specific dates precede more specific dates in ascending sorts, e.g. "2000, May 2000, May 1st 2000". Negative years are sorted inversely, e.g. "100BC, 50BC, 50AD, 100AD". Seasons are ignored for sorting, as the chronological order of the seasons differs between the northern and southern hemispheres. In the case of date ranges, the start date is used for the primary sort, and the end date is used for a secondary sort, e.g. "2000–2001, 2000–2005, 2002–2003, 2002–2009". Date ranges are placed after single dates when they share the same (start) date, e.g. "2000, 2000–2002".

numbers: *Number variables* called via the `variable` attribute are returned as integers (`form` is "numeric"). If the original variable value only consists of non-numeric text, the value is returned as a text string.

Sorting Macros

The sort key value for a macro called via `cs:key` via the `macro` attribute generally consists of the string value the macro would ordinarily generate, without rich text markup (exceptions are discussed below).

For name sorting, there are four advantages in using the same macro for rendering and sorting, instead of sorting directly on the name variable. First, substitution is available (e.g. the "editor" variable might substitute for an empty "author" variable). Secondly, et-al abbreviation can be used (using either the `et-al-min`/`et-al-subsequent-min`, `et-al-use-first`/`et-al-subsequent-use-first`, and `et-al-use-last` options defined for the macro, or the overriding `names-min`, `names-use-first` and `names-use-last` attributes set on `cs:key`). When et-al abbreviation occurs, the "et-al" and "and others" terms are excluded from the sort key values. Thirdly, names can be sorted by just the surname (using a macro for which the `form` attribute on `cs:name` is set to "short"). Finally, it is possible to sort by the number of names in a name list, by calling a macro for which the `form` attribute on `cs:name` is set to "count". As for names sorted via the `variable` attribute, names sorted via `macro` are returned with the `cs:name` attribute `name-as-sort-order` set to "all".

Number variables rendered within the macro with `cs:number` and *date variables* are treated the same as when they are called via `variable`. The only exception is that the complete date is returned if a date variable is called via the `variable` attribute. In contrast, macros return only those date-parts that would otherwise be rendered (respecting the value of the `date-parts` attribute for localized dates, or the listing of `cs:date-part` elements for non-localized dates).

3.8.3 Range Delimiters

Collapsed ranges for the "citation-number" and "year-suffix" variables are delimited by an en-dash (e.g. "(1–3, 5)" and "(Doe 2000a–c,e)").

The "locator" variable is always rendered with an en-dash replacing any hyphens. For the "page" variable, this replacement is only performed if the `page-range-format` attribute is set on `cs:style` (see *Page Ranges*).

3.8.4 Formatting

The following formatting attributes may be set on `cs:date`, `cs:date-part`, `cs:et-al`, `cs:group`, `cs:label`, `cs:layout`, `cs:name`, `cs:name-part`, `cs:names`, `cs:number` and `cs:text`:

`font-style` Sets the font style, with values:

- "normal" (default)
- "italic"
- "oblique" (i.e. slanted)

`font-variant` Allows for the use of small capitals, with values:

- "normal" (default)
- "small-caps"

`font-weight` Sets the font weight, with values:

- "normal" (default)
- "bold"
- "light"

`text-decoration` Allows for the use of underlining, with values:

- "none" (default)

- "underline"

vertical-align Sets the vertical alignment, with values:

- "baseline" (default)
- "sup" (superscript)
- "sub" (subscript)

3.8.5 Affixes

The affixes attributes `prefix` and `suffix` may be set on `cs:date` (except when `cs:date` defines a localized date format), `cs:date-part` (except when the parent `cs:date` element calls a localized date format), `cs:group`, `cs:label`, `cs:layout`, `cs:name`, `cs:name-part`, `cs:names`, `cs:number` and `cs:text`. The attribute value is either added before (`prefix`) or after (`suffix`) the output of the element carrying the attribute, but affixes are only rendered if the element produces output. With the exception of affixes set on `cs:layout`, affixes are outside the scope of *formatting*, *quotes*, *strip-periods* and *text-case* attributes set on the same element (as a workaround, these attributes take effect on affixes when set on a parent `cs:group` element).

3.8.6 Delimiter

The `delimiter` attribute, whose value delimits non-empty pieces of output, may be set on `cs:date` (delimiting the date-parts; `delimiter` is not allowed when `cs:date` calls a localized date format), `cs:names` (delimiting name lists from different *name variables*), `cs:name` (delimiting names within name lists), `cs:group` and `cs:layout` (delimiting the output of the child elements).

3.8.7 Display

The `display` attribute (similar the "display" property in CSS) may be used to structure individual bibliographic entries into one or more text blocks. If used, all rendering elements should be under the control of a display attribute. The allowed values:

- "block" - block stretching from margin to margin.
- "left-margin" - block starting at the left margin. If followed by a "right-inline" block, the "left-margin" blocks of all bibliographic entries are set to a fixed width to accommodate the longest content string found among these "left-margin" blocks. In the absence of a "right-inline" block the "left-margin" block extends to the right margin.
- "right-inline" - block starting to the right of a preceding "left-margin" block (behaves as "block" in the absence of such a "left-margin" block). Extends to the right margin.
- "indent" - block indented to the right by a standard amount. Extends to the right margin.

Examples

1. Instead of using `second-field-align` (see *Whitespace*), a similar layout can be achieved with a "left-margin" and "right-inline" block. A potential benefit is that the styling of blocks can be further controlled in the final output (e.g. using CSS for HTML, styles for Word, etc.).

```
<bibliography>
  <layout>
    <text display="left-margin" variable="citation-number"
        prefix="[" suffix="]"/>
    <group display="right-inline">
```

(continues on next page)

(continued from previous page)

```
      <!-- rendering elements -->
    </group>
  </layout>
</bibliography>
```

2. A per-author publication listing. With `subsequent-author-substitute` (see *Reference Grouping*) set to an empty string, the block with the author names is only rendered once for items by the same authors.

```
<bibliography subsequent-author-substitute="">
  <sort>
    <key variable="author"/>
    <key variable="issued"/>
  </sort>
  <layout>
    <group display="block">
      <names variable="author"/>
    </group>
    <group display="left-margin">
      <date variable="issued">
        <date-part name="year" />
      </date>
    </group>
    <group display="right-inline">
      <text variable="title"/>
    </group>
  </layout>
</bibliography>
```

The output of this example would look like:

Author1	
year-publication1	title-publication1
year-publication2	title-publication2
Author2	
year-publication3	title-publication3
year-publication4	title-publication4

3. An annotated bibliography, where the annotation appears in an indented block below the reference.

```
<bibliography>
  <layout>
    <group display="block">
      <!-- rendering elements -->
    </group>
    <text display="indent" variable="abstract" />
  </layout>
</bibliography>
```

3.8.8 Quotes

The `quotes` attribute may set on `cs:text`. When set to "true" ("false" is default), the rendered text is wrapped in quotes (the quotation marks used are terms). The localized `punctuation-in-quote` option controls whether an

adjoining comma or period appears outside (default) or inside the closing quotation mark (see *Locale Options*).

3.8.9 Strip-periods

The `strip-periods` attribute may be set on `cs:date-part` (but only if `name` is set to "month"), `cs:label` and `cs:text`. When set to "true" ("false" is the default), any periods in the rendered text are removed.

3.8.10 Text-case

The `text-case` attribute may be set on `cs:date`, `cs:date-part`, `cs:label`, `cs:name-part`, `cs:number` and `cs:text`. The allowed values:

- "lowercase": renders text in lowercase
- "uppercase": renders text in uppercase
- "capitalize-first": capitalizes the first character of the first word, if the word is lowercase
- "capitalize-all": capitalizes the first character of every lowercase word
- "sentence": renders text in sentence case
- "title": renders text in title case

Sentence Case Conversion

Sentence case conversion (with `text-case` set to "sentence") is performed by:

1. For uppercase strings, the first character of the string remains capitalized. All other letters are lowercased.
2. For lower or mixed case strings, the first character of the first word is capitalized if the word is lowercase. The case of all other words stays the same.

CSL processors don't recognize proper nouns. As a result, strings in sentence case can be accurately converted to title case, but not vice versa. For this reason, it is generally preferable to store strings such as titles in sentence case, and only use `text-case` if a style desires another case.

Title Case Conversion

Title case conversion (with `text-case` set to "title") for English-language items is performed by:

1. For uppercase strings, the first character of each word remains capitalized. All other letters are lowercased.
2. For lower or mixed case strings, the first character of each lowercase word is capitalized. The case of words in mixed or uppercase stays the same.

In both cases, stop words are lowercased, unless they are the first or last word in the string, or follow a colon. The stop words are "a", "an", "and", "as", "at", "but", "by", "down", "for", "from", "in", "into", "nor", "of", "on", "onto", "or", "over", "so", "the", "till", "to", "up", "via", "with", and "yet".

Non-English Items

As many languages do not use title case, title case conversion (with `text-case` set to "title") only affects English-language items.

If the `default-locale` attribute on `cs:style` isn't set, or set to a locale code with a primary language tag of "en" (English), items are assumed to be English. An item is only considered to be non-English if its metadata contains a `language` field with a non-nil value that doesn't start with the "en" primary language tag.

If `default-locale` is set to a locale code with a primary language tag other than "en", items are assumed to be non-English. An item is only considered to be English if the value of its `language` field starts with the "en" primary language tag.

3.9 Appendix I - Categories

- anthropology
- astronomy
- biology
- botany
- chemistry
- communications
- engineering
- generic-base - used for generic styles like Harvard and APA
- geography
- geology
- history
- humanities
- law
- linguistics
- literature
- math
- medicine
- philosophy
- physics
- political_science
- psychology
- science
- social_science
- sociology
- theology
- zoology

3.10 Appendix II - Terms

3.10.1 Locators

- book
- chapter
- column
- figure
- folio
- issue
- line
- note
- opus
- page
- paragraph
- part
- section
- sub verbo
- verse
- volume

3.10.2 Months

- month-01
- month-02
- month-03
- month-04
- month-05
- month-06
- month-07
- month-08
- month-09
- month-10
- month-11
- month-12

3.10.3 Ordinals

- ordinal
- ordinal-00 through ordinal-99
- long-ordinal-01
- long-ordinal-02
- long-ordinal-03
- long-ordinal-04
- long-ordinal-05
- long-ordinal-06
- long-ordinal-07
- long-ordinal-08
- long-ordinal-09
- long-ordinal-10

3.10.4 Quotation marks

- open-quote
- close-quote
- open-inner-quote
- close-inner-quote

3.10.5 Roles

- author
- collection-editor
- composer
- container-author
- director
- editor
- editorial-director
- editortranslator
- illustrator
- interviewer
- original-author
- recipient
- reviewed-author
- translator

3.10.6 Seasons

- season-01
- season-02
- season-03
- season-04

3.10.7 Miscellaneous

- accessed
- ad
- and
- and others
- anonymous
- at
- available at
- bc
- by
- circa
- cited
- edition
- et-al
- forthcoming
- from
- ibid
- in
- in press
- internet
- interview
- letter
- no date
- online
- presented at
- reference
- retrieved
- scale
- version

3.11 Appendix III - Types

- article
- article-magazine
- article-newspaper
- article-journal
- bill
- book
- broadcast
- chapter
- dataset
- entry
- entry-dictionary
- entry-encyclopedia
- figure
- graphic
- interview
- legislation
- legal_case
- manuscript
- map
- motion_picture
- musical_score
- pamphlet
- paper-conference
- patent
- post
- post-weblog
- personal_communication
- report
- review
- review-book
- song
- speech
- thesis
- treaty

- webpage

3.12 Appendix IV - Variables

3.12.1 Standard Variables

abstract abstract of the item (e.g. the abstract of a journal article)

annote reader's notes about the item content

archive archive storing the item

archive_location storage location within an archive (e.g. a box and folder number)

archive-place geographic location of the archive

authority issuing or judicial authority (e.g. "USPTO" for a patent, "Fairfax Circuit Court" for a legal case)

call-number call number (to locate the item in a library)

citation-label label identifying the item in in-text citations of label styles (e.g. "Ferr78"). May be assigned by the CSL processor based on item metadata.

citation-number index (starting at 1) of the cited reference in the bibliography (generated by the CSL processor)

collection-title title of the collection holding the item (e.g. the series title for a book)

container-title title of the container holding the item (e.g. the book title for a book chapter, the journal title for a journal article)

container-title-short short/abbreviated form of "container-title" (also accessible through the "short" form of the "container-title" variable)

dimensions physical (e.g. size) or temporal (e.g. running time) dimensions of the item

DOI Digital Object Identifier (e.g. "10.1128/AEM.02591-07")

event name of the related event (e.g. the conference name when citing a conference paper)

event-place geographic location of the related event (e.g. "Amsterdam, the Netherlands")

first-reference-note-number number of a preceding note containing the first reference to the item. Assigned by the CSL processor. The variable holds no value for non-note-based styles, or when the item hasn't been cited in any preceding notes.

genre class, type or genre of the item (e.g. "adventure" for an adventure movie, "PhD dissertation" for a PhD thesis)

ISBN International Standard Book Number

ISSN International Standard Serial Number

jurisdiction geographic scope of relevance (e.g. "US" for a US patent)

keyword keyword(s) or tag(s) attached to the item

locator a cite-specific pinpointer within the item (e.g. a page number within a book, or a volume in a multi-volume work). Must be accompanied in the input data by a label indicating the locator type (see the *Locators* term list), which determines which term is rendered by `cs:label` when the "locator" variable is selected.

medium medium description (e.g. "CD", "DVD", etc.)

note (short) inline note giving additional item details (e.g. a concise summary or commentary)

original-publisher original publisher, for items that have been republished by a different publisher

original-publisher-place geographic location of the original publisher (e.g. “London, UK”)

original-title title of the original version (e.g. “ ”, the untranslated Russian title of “War and Peace”)

page range of pages the item (e.g. a journal article) covers in a container (e.g. a journal issue)

page-first first page of the range of pages the item (e.g. a journal article) covers in a container (e.g. a journal issue)

PMCID PubMed Central reference number

PMID PubMed reference number

publisher publisher

publisher-place geographic location of the publisher

references resources related to the procedural history of a legal case

reviewed-title title of the item reviewed by the current item

scale scale of e.g. a map

section container section holding the item (e.g. “politics” for a newspaper article)

source from whence the item originates (e.g. a library catalog or database)

status (publication) status of the item (e.g. “forthcoming”)

title primary title of the item

title-short short/abbreviated form of “title” (also accessible through the “short” form of the “title” variable)

URL Uniform Resource Locator (e.g. “http://aem.asm.org/cgi/content/full/74/9/2766”)

version version of the item (e.g. “2.0.9” for a software program)

year-suffix disambiguating year suffix in author-date styles (e.g. “a” in “Doe, 1999a”)

Number Variables

Number variables are a subset of the *Standard Variables*.

chapter-number chapter number

collection-number number identifying the collection holding the item (e.g. the series number for a book)

edition (container) edition holding the item (e.g. “3” when citing a chapter in the third edition of a book)

issue (container) issue holding the item (e.g. “5” when citing a journal article from journal volume 2, issue 5)

number number identifying the item (e.g. a report number)

number-of-pages total number of pages of the cited item

number-of-volumes total number of volumes, usable for citing multi-volume books and such

volume (container) volume holding the item (e.g. “2” when citing a chapter from book volume 2)

3.12.2 Date Variables

accessed date the item has been accessed

container ?

event-date date the related event took place

issued date the item was issued/published

original-date (issue) date of the original version

submitted date the item (e.g. a manuscript) has been submitted for publication

3.12.3 Name Variables

author author

collection-editor editor of the collection holding the item (e.g. the series editor for a book)

composer composer (e.g. of a musical score)

container-author author of the container holding the item (e.g. the book author for a book chapter)

director director (e.g. of a film)

editor editor

editorial-director managing editor ("Directeur de la Publication" in French)

illustrator illustrator (e.g. of a children's book)

interviewer interviewer (e.g. of an interview)

original-author ?

recipient recipient (e.g. of a letter)

reviewed-author author of the item reviewed by the current item

translator translator

3.13 Appendix V - Page Range Formats

The page abbreviation rules for the different values of the `page-range-format` attribute on `cs:style` are:

"chicago" Page ranges are abbreviated according to the Chicago Manual of Style-rules:

First number	Second number	Examples
Less than 100	Use all digits	3–10; 71–72
100 or multiple of 100	Use all digits	100–104; 600–613; 1100–1123
101 through 109 (in multiples of 100)	Use changed part only, omitting unneeded zeros	107–8; 505–17; 1002–6
110 through 199 (in multiples of 100)	Use two digits, or more as needed	321–25; 415–532; 11564–68; 13792–803
4 digits	If numbers are four digits long and three digits change, use all digits	1496–1504; 2787–2816

"expanded" Abbreviated page ranges are expanded to their non-abbreviated form: 42–45, 321–328, 2787–2816

"minimal" All digits repeated in the second number are left out: 42–5, 321–8, 2787–816

"minimal-two" As "minimal", but at least two digits are kept in the second number when it has two or more digits long.