Citation Style Language Documentation

Release 1.0.1-dev

Rintze M. Zelle

November 05, 2016

Contents

1 CSL 1.0 Specification 1

2 CSL 1.0 Release Notes 39

CSL 1.0 Specification

by Rintze M. Zelle, PhD

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

Table of Contents

- *CSL 1.0 Specification*
  - *Introduction*
  - *CSL Styles - Basic Structure*
    - *Namespacing*
    - *XML Declaration*
    - *The Root Element -* `cs:style`
    - *Child Elements of* `cs:style`
      - *Info*
      - *Citation*
      - *Bibliography*
      - *Macro*
      - *Locale*
      - *Locale Prioritization*
    - *Dependent Styles*
  - *Locale Files - Basic Structure*
    - *Localized Terms*
    - *Localized Dates*
    - *Localized Options*
  - *Rendering Elements*
    - *Layout*
    - *Text*
    - *Date*
      - *Date-part*
      - *Date Ranges*
      - *AD and BC*
      - *Seasons*
      - *Uncertain Dates*
    - *Number*
    - *Names*
      - *Name*
      - *Name-part Order*
      - *Name-part Formatting*
      - *Et-al*
      - *Substitute*
      - *Label in* `cs:names`
    - *Label*
    - *Group*
    - *Choose*
  - *Style Behavior*
    - *Options*
      - *Citation-specific Options*
      - *Disambiguation*
      - *Citation Collapsing*
      - *Note Distance*
      - *Bibliography-specific Options*
      - *Whitespace*
      - *Reference Grouping*
      - *Global Options*
      - *Hyphenation of Initialized Names*
      - *Page Ranges*
      - *Name Particles*
      - *Inheritable Name Options*
      - *Locale Options*
    - *Sorting*
      - *Sorting Variables*
      - *Sorting Macros*
    - *Formatting*
    - *Affixes*

1.1 Introduction

The Citation Style Language (CSL) is an XML format for describing the formatting of in-text citations, notes and bibliographies. CSL offers:

- An open format that may be used by any application
- The ability to write compact and robust styles
- Extensive support for style requirements
- Automatic style localization
- Easy distribution and updating of styles
- A fast growing library with thousands of freely available styles

This document is meant as a complete and accurate specification of CSL 1.0. Additional documentation, such as the CSL schema, CSL styles, and information on how to add CSL support to applications, can be found at the official home of CSL, citationstyles.org.

1.2 CSL Styles - Basic Structure

1.2.1 Namespacing

All elements in CSL are namespaced. The recommended prefix `cs` is attached to element names throughout this specification, but is usually omitted from CSL styles when the default namespace is declared in the root `cs:style` element.

The CSL namespace "http://purl.org/net/xbiblio/csl"

1.2.2 XML Declaration

It is highly recommended to initialize each CSL style with an XML declaration, specifying the version of XML used as well as the character encoding. In most cases, the declaration will be:

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

1.2.3 The Root Element - `cs:style`

The root element of a CSL style is `cs:style`. This element carries the following arguments:

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

`default-locale` (optional) Fixes style localization to the locale code specified. This is desirable for most journal styles.

`version` Indicates with which version of the CSL schema the style is compatible. Should have a value of "1.0" for CSL 1.0-compatible styles.

`xmlns` (optional) The namespace declaration that binds the elements in the style to the given namespace URI. CSL elements in the style don't need individual namespace declarations if this attribute is set to "http://purl.org/net/xbiblio/csl".

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

An example of a style preamble:

```
<?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">
```

1.2.4 Child Elements of `cs:style`

All independent CSL styles share the same basic structure, with five possible types of child elements in the `cs:style` root element. The roles of each of these elements (which will be described in more detail later) are:

`cs:info` The `cs:info` element must be included as the first child element of `cs:style`. It contains the metadata that describes the style (the name of the style, a unique style identifier, the style authors, etc.).

`cs:citation` This required element describes the formatting instructions of in-text citations or notes.

`cs:bibliography` (optional) This optional element describes the formatting instructions of the bibliography.

`cs:macro` (optional) Styles may include one or more `cs:macro` elements. Macros allow for reuse of formatting instructions, which helps keeping styles compact and maintainable.

`cs:locale` (optional) Styles may include one or more `cs:locale` elements. These elements allow styles to override the default localization data (terms, date formats and formatting options) on a per-locale basis.

Info

The `cs:info` element contains all the style's metadata, many elements of which are borrowed from the Atom Syndication Format. Although it has no influence on how citations are formatted, complete and correct metadata is important if styles are made publicly available. Below is an example of a `cs:info` element, followed by a description of all possible elements.

```
<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"/>
 <author>
  <name>Author Name</name>
  <email>name@domain.com</email>
  <uri>http://www.domain.com/name</uri>
 </author>
 <category citation-format="author-date"/>
 <category field="zoology"/>
 <updated>2008-10-29T21:01:24+00:00</updated>
 <summary>Style for Some Journal</summary>
 <rights>This work is licensed under a Creative Commons
        Attribution-Share Alike 3.0 Unported License
        http://creativecommons.org/licenses/by-sa/3.0/</rights>
</info>
```

`cs:author` and `cs:contributor` (optional) One or more of these elements may be used to acknowledge style authors and contributors. Authorship is generally limited to those who have written a new style, or have made significant changes to existing styles, while contributorship can be assigned to those who have made small changes. Both elements require one child element, cs:name, and allow for two others, cs:email, and cs:uri, indicating respectively the name, email address and URI of the author or contributor in question.

`cs:category` **(optional)** Styles may be assigned one or more categories. This information can be used to organize style repositories. Two types of categories exist. The first category type describes how in-text citations are rendered. For this type the `citation-format` attribute is 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

The second category type indicates the fields or disciplines for which the style is relevant. For this category type the `field` attribute is set to one of the discipline *categories*.

`cs:id` This required element should contain a URI. This identifier establishes the identity of the style. A valid, stable, and unique URL that resolves to the style is desired if the style is made publicly available. Keeping the same URI is crucial for applications that support automatic style updating.

`cs:issn/cs:issnl` **(optional)** Journal-specific styles may include one or more `cs:issn` elements, containing the journal's ISSN identifiers (multiple ISSNs can be assigned to a single journal, e.g. for the print and online editions). In addition, the `cs:issnl` element may be used for the newly established ISSN-L identifier.

`cs:link` **(optional)** The `cs:link` element is used to specify a URI (usually a URL), which is set on the `href` attribute. The accompanying `rel` attribute must be set to indicate the relation of the URI to the style. The possible values of `rel`:

- "self": if the URI is that of the CSL style itself. Needed for automatic style updating.
- "independent-parent": if the URI is that of the parent CSL style, the content of which should be used for the citation formatting. Needed for *dependent styles*.
- "template": if the URI is that of the CSL style from which the current independent style is derived. May be used to indicate style parentage.
- "documentation: if the URI points to the online style documentation.

The `cs:link` element may contain textual content to describe the link, and may carry the `xml:lang` attribute to specify the language of either the link description or of the link target (the value should be a xsd:language locale code).

`cs:published` **(optional)** The contents of this element must be a timestamp. This timestamp indicates when the style was initially created or made available.

`cs:rights` **(optional)** This element specifies the license under which the style file is released. See, e.g. the Creative Commons. The element may include a `xml:lang` attribute to specify the language of the content (the value should be an xsd:language locale code).

`cs:summary` **(optional)** This element gives a summary of the style.The element may include a `xml:lang` attribute to specify the language of the content (the value should be an xsd:language locale code).

`cs:title` The contents of this required element should be the name of the style as it should be shown to users. The element may include a `xml:lang` attribute to specify the language of the content (the value should be an xsd:language locale code).

`cs:updated` The contents of this required element must be a timestamp. This timestamp is used for automatic updating of styles.

Citation

The `cs:citation` element describes the formatting of citations, which can consist of one or multiple references to bibliographic sources, and may appear in the form of either in-text citations (generally formatted as label [doe99], number [1], author [Doe] or author-date descriptors [Doe 1999]) or notes. The required `cs:layout` child element describes what, and how, bibliographic data should be included in the citations (see the chapter on the *Layout element*). The `cs:citation` element may carry attributes for citation-specific formatting options and inheritable name options (see the *Citation-specific Options* and *Inheritable Name Options* sections). Finally, the optional `cs:sort` element, which should precede the `cs:layout` element, specifies how citations consisting of multiple references should be sorted (see the chapter on *Sorting*). An example of a `cs:citation` element:

```
<citation option="option-value">
  <sort>
    <!-- sort keys -->
  </sort>
  <layout>
    <!-- rendering elements -->
  </layout>
</citation>
```

A note to developers of CSL processors Note styles are unique in that citations may effectively become full sentences. Because of this, the first character of the output should be uppercased when a citation is footnoted without any additional text. By contrast, if the citation occurs within a pre-existing footnote, and is preceded by non-citation text, then it should be printed as is.

Bibliography

The `cs:bibliography` element describes the formatting of bibliographies, and is used in a similar way as `cs:citation`: the required `cs:layout` child element describes how each reference should be formatted in the bibliography, while the optional `cs:sort` element (which should precede the `cs:layout element) specifies the sorting order of the references in the bibliography. In addition, the ``cs:bibliography` element may carry attributes for bibliography-specific formatting options and inheritable name options (see the *Bibliography-specific Options* and *Inheritable Name Options* sections).

```
<bibliography option="option-value">
  <sort>
    <!-- sort keys -->
  </sort>
  <layout>
    <!-- rendering elements -->
  </layout>
</bibliography>
```

Macro

Macros, which are defined using `cs:macro` elements, can contain the same set of *rendering elements* that are available within `cs:layout` inside `cs:citation` or `cs:bibliography`. Macros allow formatting instructions to be reused, both within the same style (e.g. the same macro could be used in both `cs:citation` and `cs:bibliography`) as well as between styles. It is therefore recommended to use common macro names as much as possible. Correct use of macros can greatly improve the readability, compactness and maintainability of styles. Ideally, the contents of `cs:citation` and `cs:bibliography` should be kept compact and agnostic of resource types (i.e. books, journal articles, etc.), depending mainly on macro calls.

By convention, macros are placed after any `cs:locale` elements and before the `cs:citation` element. The `cs:macro` element must carry the `name` attribute (the value of which is used to identify the macro), and contain one or more *rendering elements*. Once defined, macros can be called by *rendering elements* in `cs:citation` or `cs:bibliography` (from within `cs:layout`), or by the *rendering elements* in other macros.

The following example shows a style that call the "title" macro. This macro outputs the contents of the title variable, applying italics when the resource 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

CSL supports localization of terms, date formats and formatting options. Default localization data for several tens of locales is provided through "locales-xx-XX.xml" files ("xx-XX" represents the locale code, e.g. "en-US" for US English). Localization data can also be included in styles, using one or more of the optional `cs:locale` elements, which by convention are included directly after the `cs:info` element. For each `cs:locale` element, the relevant locale can be indicated with the `xml:lang` attribute (set to an xsd:language locale code). If the attribute is absent, the `cs:locale` element's localization data will apply to all locales.

See *Localized Terms*, *Localized Dates* and *Localized Options* in the *Locale Files - Basic Structure* section for more details on the use of `cs:locale`.

An example illustrating the use of `cs:locale` in a CSL 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 Prioritization

Locale codes can indicate either the language (e.g. "en" for English) or the language dialect (e.g. "en-US" for American English and "en-GB" for British English). While the "locales-xx-XX.xml" files are only maintained for language dialects, the (optional) `xml:lang` attribute on `cs:locale` in styles may be set to languages as well as language dialects. The "locales-xx-XX.xml" files must contain the full set of localization data. The `cs:locale` elements are typically only used in styles to redefine the localization data provided via the "locales-xx-XX.xml" files, and these may include only the localization data that should be redefined. The existence of two types of locale codes (languages and language dialects), and the ability to define localization both in "locales-xx-XX.xml" files and in styles, requires prioritization of localization data. This prioritization can best be illustrated with an example. If a CSL processor is asked to localize the style output to "de-AT" (Austrian German), the priority of the locale data is as follows:

Localization data specified in styles using `cs:locale`

1. `xml:lang` set to "de-AT" (Austrian German)
2. `xml:lang` set to "de" (German)
3. `xml:lang` not set (all locales)

Localization data stored in "locales-xx-XX.xml" files

4. `xml:lang` set to "de-AT" (Austrian German)
5. `xml:lang` set to "de-DE" (Standard German)
6. `xml:lang` set to "en-US" (American English)

Thus, when a style does not include a `cs:locale` element for the "de-AT" locale, or when it exists but is incomplete, the missing localization data is retrieved from the `cs:locale` set to "de" (if present). If the set of localization data is still incomplete, the `cs:locale` element without a `xml:lang` element is used (if present). The localization data is completed with the data stored in the "locales-xx-XX.xml" files. If the file for "de-AT" does not exist, fallback locales consist of "de-DE" and, as a last resort, "en-US". Note that locale substitution is only activated when a term is not defined. It does not occur when a term is defined, but consists of an empty string (e.g. <term name="and"/> or <term name="and"></term>).

1.2.5 Dependent Styles

In addition to independent styles, which are self-contained, CSL also supports dependent styles, which function like aliases or shortcuts. Dependent styles can be used when multiple journals share the same style format. In such a case, it is sufficient to create a single independent master style for the format (e.g. "Nature Journals"). Dependent styles, which only contain a `cs:info` element, can then be added for all journals that use this format (e.g. "Nature Biotechnology", "Nature Nanotechnology", etc.). With this approach, style repositories can show entries for the individual journals, without the need to duplicate formatting instructions. If the common format has to be modified, it is sufficient to change the independent master style, which makes style maintainance simpler and faster.

The `cs:info` element of dependent styles should provide the metadata of the individual journals. A `cs:link` element which points to the independent master style must be included (for this the `rel` attribute on the relevant `cs:link` element should be set to "independent-parent", see also *Info*).

N.B. Dependent styles cannot be used to indicate changes compared to the independent master style. If there is any difference in formatting between two styles, however small, separate independent styles have to be created.

1.3 Locale Files - Basic Structure

CSL ships with a number of "locales-xx-XX.xml" files (the "xx-XX" is the locale code, e.g. "en-US" for US English). While localization data can also be specified in styles (see *Locale*), locale files conveniently provide complete sets of default localization data (terms, dates and formatting options).

The locale files, which like styles are written in XML, each contain the localization data for a single locale. The `cs:locale` root element require two attributes: `xml:lang`, to specify the locale of the data, and `version`, to indicates with which version of the CSL schema the locale file is compatible (this attribute should have a value of "1.0" for CSL 1.0-compatible styles). The root element also typically carries a `xmlns` namespace declaration, set to the CSL namespace ("http://purl.org/net/xbiblio/csl"). The `cs:locale` element has three required child elements, which are described in the sections below: `cs:terms`, `cs:date` and `cs:style-options`. An example of the (incomplete) contents of a locale file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<locale xml:lang="en-US" version="1.0" xmlns="http://purl.org/net/xbiblio/csl">
  <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>
  <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>
  <style-options punctuation-in-quote="true"/>
</locale>
```

1.3.1 Localized Terms

Terms are localized strings. For example, if a style specifies that the term "and" should be used, the string that appears in the style output depends on the locale: "and" for English, "und" for German. Terms are defined using `cs:term` elements, child elements of `cs:terms`, itself a child element of `cs:locale`. Terms are identified by the value of the `name` attribute of `cs:term`. Two types of terms exist: simple terms, where the content of the `cs:term` is the localized string, and compound terms, where `cs:term` includes the two child elements `cs:single` and `cs:multiple`, which respectively contain the singular and plural variant of the term (e.g. "page" and "pages"). Some terms are defined for multiple forms. In these cases, multiple `cs:term` element share the same value of `name`, but differ in the value of the optional `form` attribute. The different forms are:

- "long" - the default, e.g. "editor" and "editors" for the term "editor"
- "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. "§" for the term "section"

Examples of how terms are defined have been given above (*Locale Files - Basic Structure*). The complete list of terms can be found in *Appendix III - Terms*.

1.3.2 Localized Dates

Styles can use either localized or non-localized date formats. Localized date formats are defined with the `cs:date` element as child element of `cs:locale`. The required `form` attribute on `cs:date` must be set to either "numeric" (for numeric date formats, e.g. "12-15-2005") or to "text" (e.g. "December 15, 2005"). A date format is then defined by the child elements of `cs:date`, the `cs:date-part` elements. These must carry the `name` attribute, set to `day`, `month` or `year`. The order of the `cs:date-part` elements is also the display order. Additional formatting can be achieved by setting *formatting* attributes on the `cs:date` and `cs:date-part` elements, as well as a number of attributes that are specific to `cs:date-part` (see *Date-part*). In addition, a *delimiter* may be set on `cs:date` to delimit the `cs:date-part` elements, and *affixes* may be applied to the `cs:date-part` elements.

N.B. Affixes are not allowed on `cs:date` when used as a child element of `cs:locale`. This helps in separating locale-specific affixes (which should be set on the `cs:date-part` elements) from any style-specific affixes (such as parentheses, which should be set on the `cs:date` rendering element). E.g. a macro could specify:

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

1.3.3 Localized Options

CSL 1.0 includes a single localized global option (affecting both citation and bibliography output), `punctuation-in-quote` (see *Locale Options*). This option is set as an attribute on `cs:style-options`, a child element of `cs:locale`.

1.4 Rendering Elements

Rendering elements are used to specify which, and in what order, bibliographic data should be included in citations and bibliographies. Rendering elements also partly control the formatting of this data.

1.4.1 Layout

As discussed in the *citation* and *bibliography* sections, `cs:layout` is a required child element of both `cs:citation` and `cs:bibliography`. All the rendering elements that should appear in the citations and bibliography should be nested inside the `cs:layout` element. Itself a rendering element, `cs:layout` accepts both *affixes* and *formatting* attributes. When used in the `cs:citation` element, a *delimiter* can be set to separate multiple bibliographic items in a single citation. For example, citations like "(1, 2)" can be produced with:

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

1.4.2 Text

The `cs:text` element is used to output text, which can originate from different sources. The source-type is indicated with an attribute, and the attribute value acts as an identifier within the source-type. For example,

```
<text variable="title" form="short" font-style="italic"/>
```

indicates that the source-type is a variable, and that the variable that should be displayed is the italicized short form of "title". The different source-types are:

- `variable` - the text contents of a variable (see *Standard Variables*). The optional `form` attribute can be set to either "long" (the default) or "short" to select the long or short forms of variables, e.g. the full and short title.
- `macro` - the text generated by a macro. The value of `macro` should correspond to the value of the `name` attribute of the desired `cs:macro` element.
- `term` - the text of a localized term (see *Appendix III - Terms* and *Locale*). The `plural` attribute can be set to choose either the singular (value "false", the default) or plural variant (value "true") of a term. In addition, the `form` attribute can be set to select the desired term form ("long" [default], "short", "verb", "verb-short" or "symbol"). If for a given term the desired form does not exist, another form may be used: "verb-short" reverts to "verb", "symbol" reverts to "short", and "verb" and "short" both revert to "long".
- `value` - used to output verbatim text, which is set via the value of `value` (e.g. value="some text")

In all cases the attributes for *affixes*, *display*, *formatting*, *quotes*, *strip-periods* and *text-case* may be applied to `cs:text`.

1.4.3 Date

The `cs:date` element is used to output dates, in either a localized or a non-localized format. The desired date variable (see *Date Variables*) is selected with the `variable` attribute.

Localized date formats are selected with the `form` attribute. This attribute can be set to "numeric" (for numeric date formats, e.g. "12-15-2005"), or to "text" (for date formats with a non-numeric month, e.g. "December 15, 2005"). Localized dates can be customized in two ways. First, the `date-parts` attribute may be used to specify which `cs:date-part` elements are shown. The possible values are:

- "year-month-day" - default, displays year, month and day
- "year-month" - displays year and month
- "year" - displays year only

Secondly, `cs:date` may include one or more `cs:date-part` elements (see *Date-part*). The attributes set on these elements override those originally specified for the localized date formats (e.g. the `form` attribute of the month-`cs:date-part` element can be set to "short" to get abbreviated month names in all locales.). Note that the use of `cs:date-part` elements for localized dates does not affect which, and in what order, the `cs:date-part` elements are included in the rendered date. Also, the `cs:date-part` elements may not carry the attributes for *affixes*, as these are considered to be locale-specific.

Non-localized date formats are self-contained: the date format is entirely controlled by `cs:date` and its `cs:date-part` children. In contrast to localized dates, `cs:date` is used without the `form` and `date-parts` attributes. Only the included `cs:date-part` elements will be rendered, in the order in which they are specified. The `cs:date-part` elements may carry attributes for both *affixes* and *formatting*, while `cs:date` may carry a *delimiter* (delimiting the various `cs:date-part` elements).

For both localized and non-localized dates, *affixes*, *display* and *formatting* attributes may be specified for the `cs:date` element.

Date-part

The `cs:date-part` element is used to control how the different date parts of the date variable specified in the parent `cs:date` element are rendered. The date parts are identified by the value of the `name` attribute, which can be:

`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"

month For `month`, `cs:date-part` may carry the *strip-periods* and `form` attributes. Abbreviated months (e.g. "Jan.", "Feb.") are *localized terms* and include periods by default (if applicable). These periods are removed when *strip-periods* is set 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"

All `cs:date-part` elements may carry the *formatting* and *text-case* attributes. Attributes for *affixes* are also allowed, except when `cs:date` is used to call a localized date format. Finally, the `cs:date-part` elements may carry the `range-delimiter` attribute (see *Date Ranges*).

Date Ranges

By default, date ranges are delimited by an en-dash (e.g. "May–July 2008"). The `range-delimiter` attribute can be used to specify custom date range delimiters. The attribute value set on the largest date-part ("day", "month" or "year") that differs between the two dates of the date range will then be used instead of the en-dash. For example,

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

would result in "May–July 2008" and "May 2008/June 2009".

AD and BC

The terms `ad` and `bc` (Anno Domini and Before Christ) are automatically appended to years: `bc` is added to negative years (e.g. 2500BC), while `ad` is added to positive years of less than four digits (79AD).

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) will substituted the month date-part. 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".

Uncertain Dates

Uncertain dates can receive special formatting by using the `is-uncertain-date` conditional (see *Choose*) and the "circa" term. The conditional tests "true" when a date is flagged as uncertain. For example,

```
<style>
  <citation>
    <layout delimiter="; ">
      <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 certain date) and "ca. 2003" (uncertain date).

1.4.4 Number

The `cs:number` element can be used to output any of the following variables (selected with the `variable` attribute):

- "edition"
- "volume"
- "issue"
- "number"
- "number-of-volumes"

Although these variables can also be rendered with `cs:text`, `cs:number` has the benefit of offering number-specific formatting via the `form` attribute, with values:

- "numeric" (default) - e.g. "1", "2", "3"
- "ordinal" - e.g. "1st", "2nd", "3rd"
- "long-ordinal" - e.g. "first", "second", "third"
- "roman" - e.g. "i", "ii", "iii"

If a variable displayed with `cs:number` contains a mixture of numeric and non-numeric text, only the first number encountered is used for rendering (e.g. "12" when the entire string is "12th edition"). If a variable only contains non-numeric text (e.g. "special edition"), the entire string is rendered, as if *cs:text* were used instead. Fields can be tested for containing numeric content with the `is-numeric` conditional, e.g. “12th edition” would test “true” while “third edition” would test “false” (see *Choose*).

The `cs:number` element may carry any of the *affixes*, *display*, *formatting* and *text-case* attributes.

1.4.5 Names

The `cs:names` element can be used to display the contents of one or more *name variables*, each of which can contain multiple names (e.g. the “author” variable will contain all the cited item’s author names). The variables to be displayed are set with the `variable` attribute. If multiple variables are selected (separated by single spaces, see example below), each variable is independently rendered in the order specified, with one exception: if the value of `variable` consists of “editor” and “translator” (in either order), and if the contents of the two name variables is identical, then the contents of only one name variable is rendered. In addition, the “editor-translator” term is used if the `cs:names` element contains a `cs:label` element, replacing the default “editor” and “translator” terms (e.g., this might result in “Doe (editor & translator)”. The *delimiter* attribute may be set on `cs:names` to delimit the names of the different name variables (e.g. the semicolon in “Doe (editor); Johnson (translator)”).

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

There are four child elements associated with the `cs:names` element: `cs:name`, `cs:et-al`, `cs:substitute` and `cs:label` (all discussed below). In addition, the `cs:names` element may carry the attributes for *affixes*, *display* and *formatting*.

Name

The `cs:name` element is a required child element of `cs:names`, and describes both how individual names are formatted, and how names within a name variable are separated from each other. The attributes that may be used on `cs:name` are:

`and` This attribute specifies the delimiter between the second to last and the last name of the names in a name variable. The value of the attribute may be either “text”, which selects the “and” term, or “symbol”, which selects the ampersand (&).

`delimiter` Specifies the text string to separate names of a name variable. The default value is “, ” (“J. Doe, S. Smith”).

`delimiter-precedes-last` Determines in which cases the delimiter used to delimit names is also used to separate the second to last and the last name in name lists. The possible values are:

- “contextual” (default): the delimiter is only included for name lists with three or more names
  - 2 names: “J. Doe and T. Williams,”
  - 3 names: “J. Doe, S. Smith, and T. Williams”
- “always”: the delimiter is always included
  - 2 names: “J. Doe, and T. Williams”
  - 3 names: “J. Doe, S. Smith, and T. Williams”
- “never”: the delimiter is never included
  - 2 names: “J. Doe and T. Williams,”
  - 3 names: “J. Doe, S. Smith and T. Williams”

et-al-min / et-al-use-first Together, these attributes control et-al abbreviation. When the number of names in a name variable matches or exceeds the number set on `et-al-min`, the rendered name list is truncated at the number of names set on `et-al-use-first`. If truncation occurs, the "et-al" term is appended to the names rendered (see also *Et-al*). With a single name (et-al-use-first="1"), the "et-al" term is preceded by a space (e.g. "Doe et al."). With multiple names, the "et-al" term is preceded by the name delimiter (e.g. "Doe, Smith, et al.").

et-al-subsequent-min / et-al-subsequent-use-first The (optional) `et-al-min` and `et-al-use-first` attributes take effect for all cites and bibliographic entries. With the `et-al-subsequent-min` and `et-al-subsequent-use-first` attributes divergent et-al abbreviation rules can be specified for subsequent cites (cites referencing earlier cited items).

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. The roles of these name-parts, which are delimited by single spaces in rendered names, are:

- "family": the surname minus any particles and suffixes
- "given": the given names, which may be 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 last name is shown ("de" in the Dutch surname "de Koning") but which may be treated as a separate object 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")

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

initialize-with If this attribute is set, given names are converted to initials. The attribute value specifies the suffix that is included after each initial ("." results in "J.J. Doe"). Note that the global `initialize-with-hyphen` option controls how compound given names (e.g. "Jean-Luc") are hyphenated when initialized (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 may have one of the two values:

- "first": name-as-sort-order applies to the first name in each name variable
- "all": name-as-sort-order applies to all names

Note that the sort order of names may differ from the display order for names containing particles and suffixes (see *Name-part order*). Also, this attribute only affects names written in the latin or Cyrillic alphabet. Names written in other alphabets (e.g. Asian scripts) are always shown 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 written in the latin or Cyrillic alphabet.

The `cs:name` element may also carry any of the attributes for *affixes* and *formatting*.

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 alphabet of the individual name. Note that the display and sorting order of name-parts often differs. An overview of the different orders:

Display order of latin/Cyrillic names

Conditions `form` set to "long"

Order

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

Example [Gérard] [de] [la] [Martinière] [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] [Martinière], [Gérard] [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 [Martinière], [Gérard] [de] [la], [III]

Conditions `form` set to "short"

Order

1. non-dropping-particles
2. family

Example [la] [Martinière]

Sorting order of latin/Cyrillic names

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

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

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

Example [la Martinière] [de] [Gérard] [III]

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

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

Example [Martinière] [de la] [Gérard] [III]

Display and sorting order of non-latin/Cyrillic names

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 include one or two `cs:name-part` child elements. These child elements accept the *formatting* and *text-case* attributes, which allows for separate formatting of the different name parts (e.g. "Jane DOE", see example below). The required `name` attribute on `cs:name-part` specifies which name-parts are affected: when set to "given", the formatting only acts on the "given" name-part. When set to "family", the formatting acts on the "family", "dropping-particle" and "non-dropping-particle" name-parts (the "suffix" name-part is not subject to any name-part formatting). The order of the `cs:name-part` elements does not affect which, and in what order, the name-parts are rendered.

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

Et-al

Et-al abbreviation, controlled via the et-al attributes on `cs:name` (see *Name*), can be further customized with the optional `cs:et-al` element, which should be included directly after the `cs:name` element. The `term` attribute of this element can be set to either "et-al" (default) or to "and others" to use either term (with this different et-al terms can be used for citations and the bibliography). In addition, attributes for *affixes* and *formatting* can be used, for example to italicize the et-al term:

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

Substitute

The optional `cs:substitute` element, which should be included as the last child element of `cs:names`, controls 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 can consist of any of the standard *rendering elements* (with the exception of `cs:layout`). It is also possible to use a shorthand version of `cs:names`, which doesn't allow for any child elements, and uses the attributes values set on the `cs:name` and `cs:et-al` child elements of the original `cs:names` element. If `cs:substitute` contains multiple child elements, the first element to return a non-empty result is used for substitution. Substituted variables are repressed 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">
    <name/>
    <substitute>
      <names variable="editor"/>
      <text macro="title"/>
    </substitute>
  </names>
</macro>
```

Label in `cs:names`

The `cs:label` element, used to output text terms whose pluralization depends on the contents of another variable (e.g. "(editors)" in "Doe and Smith (editors)"), is discussed in detail in the *label* section. It should be included after the `cs:name` and `cs:et-al` elements, but before the `cs:substitute` element. When used within `cs:names`, the `variable` attribute should be omitted, as the value set on the parent `cs:names` element is used.

1.4.6 Label

The Citation Style Language includes several variables that have matching terms. The `cs:label` element can be used to render one of these terms, while matching the term plurality with that of the corresponding variable. The variable/term combination is selected with the `variable` attribute, which can be set to either "page" or "locator". When `cs:label` is used as a child element of `cs:names`, the value of the `variable` attribute is automatically inherited from the parent `cs:names` element. The example below displays the "page" variable, using the singular form of the "page" term for a single page ("page 5"), or the plural form for a page range ("pages 5-7").

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

The `cs:label` element may carry attributes for *affixes*, *formatting*, *text-case* and *strip-periods*, as well as:

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

- "long": the default, e.g. "editor"/"editors" for the "editor" term
- "verb": e.g. "edited by" for the "editor" term
- "short": e.g. "ed"/"eds" for the "editor" term
- "verb-short": e.g. "ed" for the "editor" term
- "symbol": e.g. "§" for the singular "section" term

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

- "contextual": the default, pluralization is dependent on the variable contents, e.g. "page 1" and "pages 1-3"
- "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"

1.4.7 Group

The `cs:group` element may contain one or more *rendering elements* (not `cs:layout`). `cs:group` itself may carry the *delimiter* attribute (to delimit the enclosed elements) and the attributes for *affixes* (applied to the group output as a whole), *display* and *formatting* (formatting settings are transmitted to the enclosed elements). Note that `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 behavior exists to accommodate descriptive cs:text elements. For example,

```
<layout>
  <group prefix="(" suffix=")">
    <text value="Published by: "/>
    <text variable="publisher"/>
  </group>
</layout>
```

results in "(Published by: Company A)" when the "publisher" variable is set to "Company A", but doesn't generate output when the "publisher" variable is empty.

1.4.8 Choose

Similarly to the conditional statements encountered in programming languages, the `cs:choose` element allows for the conditional rendering of *rendering elements*. An example is shown below:

```
<choose>
  <if type="book thesis" match="any">
    <text variable="title" font-style="italic">
  </if>
  <else-if type="chapter">
    <text variable="title" quotes="true">
  </else-if>
  <else>
    <text variable="title">
  </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 each hold at least one condition, which are expressed as attributes. The different types of conditions available are:

disambiguate The contents of an <if disambiguate="true"> block is only rendered if it disambiguates two otherwise identical citations. This attempt at disambiguation will only be made when all other disambiguation methods have failed to uniquely identify the target source.

is-numeric Tests whether the given variables (*Appendix I - Variables*) contain numeric data.

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

locator Tests whether the locator matches the given locator variable subtype (see *Locators*).

position Tests whether the position of the item cite matches the given positions (when called within cs:bibliography, this condition will always test "false"). The different positions are (note on terminology: a *citation* refers to a citation group, which contains one or more *cites* to individual items):

- "first": the position of a cite that is the first to reference an item
- "ibid"/"ibid-with-locator"/"subsequent": a cite that references an earlier cited item always has the "subsequent" position. In special cases cites may have the "ibid" or "ibid-with-locator" position. These positions are only assigned 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 includes a single cite that references 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 the position is "subsequent".

- "near-note": the position of a cite following another cite that references 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*).

Note that each cite can have multiple position values. Whenever position="ibid-with-locator" is true, position="ibid" is also true. And whenever position="ibid" or position="near-note" is true, position="subsequent" is also true.

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

`variable` Tests whether the given variables (*Appendix I - 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 include the `match` attribute to control the testing logic, with possible values:

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

1.5 Style Behavior

1.5.1 Options

Styles can be extensively configured with (optional) options, which are set as attributes. *Citation-specific options* are set on `cs:citation`, while *bibliography-specific options* are set on `cs:bibliography`. *Global options*, which affect both citations and the bibliography, are 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

Disambiguation can be achieved in five ways:

1. The number of names shown can be increased.
2. A given name can be added.
3. Initialized given names can be expanded.
4. A year-suffix can be included.
5. The cite can be rendered with the `disambiguate` attribute of `cs:choose` conditions testing "true".

Note that the term "disambiguation" in the statement above is itself ambiguous. Steps (1), (4) and (5) aim solely to disambiguate cites that otherwise would be the same. Steps (2) and (3), however, are different. In addition to the strict purpose of disambiguating *cites*, the adding or expansion of given names may be used for the broader purpose of disambiguating *names* throughout the document. In the description below, this difference is referred to as "the scope of names transformation".

The five potential steps to disambiguation are activated with the attributes described in this section, and are always performed, if at all, in the order listed below.

`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, until either the target reference is uniquely identified, or all names are shown.

`disambiguate-add-givenname` [Steps (2) & (3)] If set to "true" ("false" is the default), given names are added or expanded. For example:

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

Note that the value of the `givenname-disambiguation-rule` attribute (the default is "all-names") determines a) the precise method of name expansion, and b) whether or not cites that are not themselves ambiguous but do contain the ambiguous name(s) are affected by this type of disambiguation.

`givenname-disambiguation-rule` [Steps (2) & (3) supplemental] This attribute accepts one of five possible values, which vary in three respects: the scope of names transformation within the document; the steps included in the disambiguation attempt; and the names within a cite that are affected.

The scope of names transformation With a value of "all-names", "all-names-with-initials", "primary-name", or "primary-name-with-initials", disambiguation is performed for all relevant names, without regard to ambiguity in individual cites. Transformations governed by these rules apply to all cites throughout the document. Disambiguation of cites is in this case incidental to the disambiguation of names.

With a value of "by-cite", only the names within ambiguous cites are transformed, as required to discriminate between references. Cites that are not ambiguous are not affected.

Transformation steps All five types of given name disambiguation follow the same general transformation steps (the specific steps applied depend on the value of `givenname-disambiguation-rule`).

1. If `initialize-with` is set, then:

   (a) A `form` value of "short" can be incremented to "long" (e.g. "Doe" becomes "J. Doe").

   (b) `initialize-with` can be ignored (e.g. "J. Doe" becomes "John Doe").

2. If `initialize-with` is *not* set, then the `form` value of "short" can be immediately incremented to "long" (e.g. "Doe" becomes "John Doe").

Given name disambiguation rules The effect of each given name disambiguation rule is described below. In all cases, transformations that do not contribute to disambiguation are omitted, and any names added by `disambiguate-add-names` that follow the name that results in success are discarded.

"all-names" The default value. If a name is rendered the same in different cites (e.g. "Doe 2000" and "Doe 2001"), the name is progressively transformed until it can be distinguished from the others (e.g. "A. Doe 2000" and "B. Doe 2001"), or until the transformation steps are exhausted.

"all-names-with-initials" Same as "all-names", but limited to step 1(a). If `initialize-with` is not set, no disambiguation attempt is made.

"primary-name" Same as "all-names", but ambiguity is only checked for the first-listed name, and only first-listed names are affected by the transformation.

"primary-name-with-initials" Same as "primary-name", but limited to step 1(a). If `initialize-with` is not set, no disambiguation attempt is made.

"by-cite" Same as "all-names", but the transformation is limited to ambiguous cites. The appearance of the names transformed will not be affected in other cites.

`disambiguate-add-year-suffix` **[Step (4)]** If set to "true" ("false" is the default), a year-suffix is added to cites that are otherwise identical (e.g. "Doe 2007, Doe 2007" becomes "Doe 2007a, Doe 2007b"). The placement of the year-suffix, which by default is appended to each cite, can be controlled by explictly rendering the "year-suffix" variable using `cs:text`.

If ambiguous cites remain after the above steps have been exhausted, a final attempt at disambiguation is performed with the `disambiguate` test value on any `cs:choose` conditions testing "true" [Step (5)]. If this results in successful disambiguation, any names added by `disambiguate-add-names` are discarded.

Citation Collapsing

`collapse` The collapse option activates citation collapsing. Note that "year-suffix" and "year-suffix-ranged" both fall back to "year" when the `disambiguate-add-year-suffix` attribute is not set to "true" (see *Disambiguation*). Its possible values are:

- "citation-number": collapses numeric citation ranges (e.g. from "[1, 2, 3, 5]" to "[1-3, 5]"). Note that only increasing ranges are collapsed, e.g. "[3, 2, 1]" will not collapse (to see how numeric styles can sort citations by the `citation-number` variable, see *Sorting*).
- "year": when the names stored in the rendered name variables are the same for two subsequent cites, the latter cite is collapsed to only the year, e.g. from "(Doe 2000, Doe 2001)" to "(Doe 2000, 2001)".
- "year-suffix": collapses as "year", but also collapses identical years, e.g. "(Doe 2000a, b)" instead of "(Doe 2000a, 2000b)".
- "year-suffix-ranged": collapses as "year-suffix", but also collapses ranges of year-suffix markers, e.g. "(Doe 2000a-c,e)" instead of "(Doe 2000a, b, c, e)".

`year-suffix-delimiter` Specifies the delimiter for year-suffix elements. For example, citations like "(Smith 1999a,b; 2000; Jones 2001)" are obtained when the `collapse` attribute is set to "year-suffix", the `delimiter` on `cs:layout` in `cs:citation` is set to "; ", and the `year-suffix-delimiter` is set to ",". When the `year-suffix-delimiter` attribute is not set, year-suffixes are delimited with the delimiter set on `cs:layout` in `cs:citation`.

`after-collapse-delimiter` Specifies the cite delimiter that should be used *after* a group of collapsed cites. For example, citations like "(Smith 1999a, b, 2000; Jones 2001, Brown 2007)" are obtained when the `collapse` attribute is set to "year-suffix", the `delimiter` on `cs:layout` in `cs:citation` is set to ", " and `after-collapse-delimiter` is set to "; ".

Note Distance

`near-note-distance` The "near-note" position (see *Choose*) tests "true" if a preceding note exists that: a) refers to the same item and b) has a distance (measured in footnotes or endnotes) to the current item that does not exceed the value of `near-note-distance`. This attribute defaults to 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 to “flush”, subsequent lines of each bibliography entry are aligned with the beginning of the second field. If set to “margin”, the first field is put in the margin and all subsequent lines of text are aligned with the margin (as in the IEEE style). An example showing the alignment, if the first field is `<text variable="citation-number" suffix=".  "/>`:

```
1. Adams, D. (2002). The Ultimate Hitchhiker's Guide to the
   Galaxy (1st ed.).
```

`line-spacing` Specifies a formatting hint for vertical line distance. Defaults to “1” (single-spacing), and can be set to any non-negative integer to specify a multiple of the standard unit of line height (e.g. “2” for double-spacing).

`entry-spacing` Specifies a formatting hint for 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` The value of the `subsequent-author-substitute` attribute (which may be any string) is used to replace the names in a bibliographic entry, when it shares these names with the preceding bibliographic entry. Note that only the first `cs:names` element rendered is affected. E.g., with `subsequent-author-substitute` set to “—”:

```
Asimov. Foundation, 1951.
---. Foundation and Empire, 1952.
---. Second Foundation, 1953.
```

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”) or without (“J.L.”, value “false”). Defaults to “true”.

Page Ranges

`page-range-format` The value of this attribute determines how page ranges are formatted. Available values: “expanded” (e.g. “321-328”), “minimal” (“321-8”), and “chicago” (“321-28”) (see *Appendix IV - Page Abbreviation Rules* for the Chicago Manual of Style page range collapsing rules). If the attribute is not specified, the content of the page-field is left unchanged.

Name Particles

Many Western names include one or more name particles (e.g. “de” in the Dutch name “W. de Koning”). However, not all particles are equal: name particles can be either maintained or dropped when only the surname is shown (from now on we will refer to these two types as non-dropping-particle and dropping-particle, respectively). A single name can contain particles of both types (with the non-dropping-particle always following the dropping-particle). For example, the French name “Gérard de la Martinière” can be deconstructed into:

```
{
    "author": {
        "given": "Gérard",
        "dropping-particle": "de",
        "non-dropping-particle": "la",
        "family": "Martinière"
    },
    {
        "given": "W.",
        "non-dropping-particle": "de",
        "family": "Koning"
    }
}
```

When just the surname is shown, only the non-dropping-particle is kept: "La Martinière".

Whereas the dropping-particle is always treated the same, styles vary in how the non-dropping-particle is handled in case of inverted names, where the family name precedes the given name. First, the non-dropping-particle can be either prepended to the family name (e.g. "de Koning, W.") or appended (after initials or given names, e.g. "Koning, W. de"). Note that the dropping-particle is always appended in inverted names. Secondly, if the choice has been made to prepend the non-dropping-particle to the family name for inverted names, the author sort order can differ. Either the non-dropping-particle remains part of the family name (as part of the primary sort key; sort order A), or it may be separated from the family name and 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 Martinière"
- secondary sort key: "de"
- tertiary sort key: "Gérard"

Sort order B: non-dropping-particle demoted

- primary sort key: "Martinière"
- secondary sort key: "de la"
- tertiary sort key: "Gérard"

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"
        }
    }
```

The handling of particles for inverted names is set 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"). The possible values are:

- "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 Martinière, Gérard 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 (see sort order B, e.g. "de Koning, W." appears under "K").
- "display-and-sort" (default): the dropping and non-dropping-particle are appended to the rest of the name (e.g. "Koning, W. de" and "Martinière, Gérard de la"). When names are sorted, all particles are part of the secondary sort key (see sort order B, e.g. "Koning, W. de" appears under "K").

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-last, et-al-min, et-al-use-first, et-al-subsequent-min, et-al-subsequent-use-first, initialize-with, name-as-sort-order and sort-separator. The attributes name-form and name-delimiter accompany the form and delimiter attributes on cs:name. Similarly, names-delimiter, the only inheritable attribute available for cs:names, accompanies 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 element carrying the attribute. When an element lower in the hierarchy includes the same attribute with a different value, this latter value will override the value(s) specified higher in the hierarchy.

Locale Options

punctuation-in-quote Determines whether punctuation (commas and periods) is placed inside (value "true") or outside (default, value "false") quotation marks added with the quotes attribute (see *Formatting*).

1.5.2 Sorting

The sort order for citations and the bibliography can be set with the cs:sort element in cs:citation and cs:bibliography. If a style does not include sorting instructions, references are listed in the order cited.

The cs:sort element must contain one or more cs:key child elements. The sort key, set as an attribute on cs:key, can be a variable (see *Appendix I - Variables*) or a macro. The cs:key element may carry the sort attribute, with possible values of "ascending" (default) or "descending", to indicate the sort order. The names-min and names-use-first attributes (which affect all names generated by macros called by cs:key) can be used to (further) constrain the number of names used in the sort, overriding the values of the corresponding et-al-min and et-al-use-first and et-al-subsequent options.

Sort keys are evaluated one by one. The primary sort is performed using the first sort key. A secondary sort (using the second sort key) is performed on those items which share the first sort key. A tertiary sort (using the third sort key) is performed on those items which share the first and second sort key. This process continues until either the order of all items is fixed, or until the sort keys are exhausted. Items for which a sort key is empty are placed at the end of the sort (both for ascending and descending sorts).

An example, where citations are first sorted by the output of the author macro with overriding settings for et-al abbreviation. Entries that share the same author macro output are further sorted in reverse order by date of issue.

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

The values returned for variables and macros called in cs:sort may differ from the "ordinary" rendered values. These differences are detailed below.

Sorting Variables

When variables are referenced in `cs:key` via the `variable` attribute, the string value is returned, without formatting decorations. Exceptions are name, date and numeric variables, which are returned as follows:

names: *Name variables* can be set directly on `cs:key` using the `variable` attribute (e.g. `<key variable="author"/>`). In this case, the name-list from the variable will be returned as a string in the "long" `form` of `cs:name`, formatted with `name-as-sort-order` set to "all".

dates: *Date variables* that are set directly on `cs:key` using the `variable` attribute are returned to `cs:key` in the YYYYMMDD format, with zeros substituted for any missing date-parts (e.g. 20001200 for December 2000). As a result, dates with more date-parts will come after those with fewer date-parts, e.g. (2000, May 2000, May 1st 2000). Note that 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. Date ranges consist of a start date and an end date. The start date is used for comparison with single dates. However, for items with the same (start) date, the items with date ranges are placed after those with single dates, e.g. (1999, 2000, 2000-2002, 2001, 2001-2003). In addition, date ranges are subjected to a secondary sort based on the end date, e.g. (2000, 2000-2001, 2000-2004, 2000-2005, 2001).

numbers: If the `variable` attribute is used, numeric values 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

A macro called via `cs:key` returns whatever string value the macro would ordinarily generate, with a few exceptions. In all cases, rich text markup is removed from the sort key.

For name sorting, it is generally preferable to use the same macro that is used to render the names in the context (`cs:citation` or `cs:bibliography`) to which the sort applies. The first benefit of using macros is that substitution logic becomes 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` and `et-al-use-first` or et-al-subsequent options defined within the macro, or the overriding `names-min` and `names-use-first` attributes set on `cs:key`). Note that the "et-al" and "and others" terms are not included in the sort key when et-al abbreviation occurs. The third benefit is that names can be sorted by just the family name and name particles, 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 names-list, by calling a macro in which the `form` attribute of `cs:name` is set to "count". In this case a count value of "3" would be obtained for a name variable that would otherwise return "Jones, Smith, Doe". For name sorting, the `name-as-sort-order` attribute on `cs:name` elements is set to "all".

Number variables (rendered with `cs:number`) and date variables are treated the same as when they were called via `variable`. The only exception is that if a date variable is called by the `variable` attribute, the complete date is returned. 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).

1.5.3 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)

1.5.4 Affixes

The affixes attributes `prefix` and `suffix` may be set on `cs:date` (except when `cs:date` is used within `cs:locale`), `cs:date-part` (except when the parent `cs:date` element calls a localized date format), `cs:et-al`, `cs:group`, `cs:label`, `cs:layout`, `cs:name`, `cs:names`, `cs:number` and `cs:text`. The attribute value is included either before (`prefix`) or after (`suffix`) the displayed text. Affixes are generally insensitive to the formatting attributes acting on the calling element: the only exception to this rule are affixes set on `cs:layout`. In cases where formatting of affixes is desired, separate `cs:text` elements can be used instead, with a `value` attribute to output verbatim text.

1.5.5 Delimiter

The `delimiter` attribute can be used to specify a delimiting string for `cs:date` (delimiting the date-parts; not allowed when `cs:date` calls a localized date format), `cs:names` (delimiting multiple *name variables*), `cs:name` (delimiting names in name lists), `cs:group` and `cs:layout` (both delimiting the direct child elements).

1.5.6 Display

Many of the anticipated output formats for CSL 1.0 (RTF, LaTeX, XML dialects such as XHTML) allow styling to be applied to individual blocks of text in order to control their appearance and position. The `display` attribute can be used in `cs:bibliography` to allocate styling to particular text blocks for this purpose [1]. The attribute may be set on `cs:date`, `cs:group`, `cs:names`, `cs:number` and `cs:text` elements, and accepts one of the following values:

- "block": A block stretching from margin to margin.
- "left-margin": A block of fixed width starting at the left margin (all "left-margin" blocks in a bibliography share the same width, set according to the maximum number of characters appearing in any one such block).

[1] N.B. if `display` attributes are used, make sure all rendering elements are under the control of exactly one display attribute.

- “right-inline”: A block directly to the right of any immediately preceding “left-margin” block, and extending to the right margin.
- “indent”: Block indented to the right by a standard amount.

Examples

1. A similar effect as with `second-field-align` can be achieved with [2]:

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

---

2. A per-author publication listing can be formatted as follows [3]:

```
<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>
```

which would result in

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

---

3. An annotated bibliography with the annotation block-indented below the reference can be formatted as follows:

[2] The styling definitions in the target application (CSS for HTML, styles for Word etc.) can be adjusted to achieve special effects, such as floating the labels into the margin.

[3] The effect of the empty `subsequent-author-substitute` attribute is to render the author name only once, at the top of the list of each author’s publications.

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

1.5.7 Quotes

The `quotes` attribute may set on `cs:text`. When set to "true" ("false" is default), the rendered text is wrapped in quotes. The quote-symbols are defined as (localized) terms. The localized `punctuation-in-quote` option controls whether punctuation appears inside or outside the quotes (see *Locale Options*).

1.5.8 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.

1.5.9 Text-case

The `text-case` attribute may be set on `cs:date-part`, `cs:label`, `cs:name-part`, `cs:number` and `cs:text` and can be used to control the text case of the rendered text. The possible values are:

- "lowercase": displays all text in lowercase
- "uppercase": displays all text in uppercase
- "capitalize-first": capitalizes the first character; the case of other characters is not affected
- "capitalize-all": capitalizes the first character of every word; other characters are displayed lowercase
- "title": displays text in title case (the *Chicago Manual of Style* calls this "headline style")
- "sentence": displays text in sentence case ("sentence style")

1.6 Appendices

1.6.1 Appendix I - Variables

Standard Variables

- abstract
- annote
- archive
- archive_location
- archive-place
- authority

- call-number
- chapter-number
- citation-label
- citation-number
- collection-title
- container-title
- DOI
- edition
- event
- event-place
- first-reference-note-number
- genre
- ISBN
- issue
- jurisdiction
- keyword
- locator
- medium
- note
- number
- number-of-pages
- number-of-volumes
- original-publisher
- original-publisher-place
- original-title
- page
- page-first
- publisher
- publisher-place
- references
- section
- status
- title
- URL
- version
- volume

- year-suffix

Date Variables

- accessed
- container
- event-date
- issued
- original-date

Name Variables

- author
- editor
- translator
- recipient
- interviewer
- publisher
- composer
- original-publisher
- original-author
- container-author (to be used when citing a section of a book, for example, to distinguish the author proper from the author of the containing work)
- collection-editor (use for series editor)

1.6.2 Appendix II - Types

These are the different item types available within CSL:

- article
- article-magazine
- article-newspaper
- article-journal
- bill
- book
- broadcast
- chapter
- 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

1.6.3 Appendix III - Terms

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

Locators

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

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

Ordinals

- ordinal-01
- ordinal-02
- ordinal-03
- ordinal-04
- 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

Quotation marks

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

Roles

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

Seasons

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

Miscellaneous

- accessed
- ad
- and
- and others
- anonymous
- at
- bc
- by
- circa
- cited
- edition
- et-al
- forthcoming
- from
- ibid

- in
- in press
- internet
- interview
- letter
- no date
- online
- presented at
- reference
- retrieved

1.6.4 Appendix IV - Page Abbreviation Rules

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

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

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

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

Table: **Chicago Manual of Style page range abbreviation 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

CHAPTER 2

CSL 1.0 Release Notes

by Rintze M. Zelle, PhD

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

Table of Contents

- *CSL 1.0 Release Notes*
  - *Preface*
  - *Project Home(s)*
  - *Documentation*
  - *Schema*
    - *Split Schema*
    - *Validation*
  - *Styles*
    - *Updating CSL 0.8 Styles*
      - *Using upgrade.xsl*
    - *Version Number*
    - *Options as Attributes*
    - *Style Metadata*
      - *Links*
      - *Formats and Fields*
      - *ISSN and ISSN-L*
    - *Localization*
      - *Localized Dates*
      - *Localized Options*
      - *Default Locale*
    - *Formatting*
      - *Citation Collapsing*
      - *Dates*
      - *AD and BC*
      - *Date Ranges*
      - *Seasons*
      - *Uncertain Dates*
      - *Names*
      - *delimiter-precedes-last*
      - *Editorial Director*
      - *Editor/Translator Name Collapsing*
      - *Et al.*
      - *Formatting Name Parts*
      - *Inheritable Name Options*
      - *Hyphenation of Compound Given Names*
      - *Name Particles*
      - *Ordinal Numbers*
      - *Original Publisher*
      - *Pages*
      - *Pluralization*
      - *Quotes*
      - *Quote Substitution*
      - *Flipflopping*
      - *Punctuation around Quotes*
      - *Rich Text Markup within Fields*
      - *second-field-align*
      - *Stripping Periods*
      - *Year Suffix*
      - *Layout Control with the Display Attribute*
    - *Disambiguation*
    - *Sorting*
    - *Back-References in Note Styles*
    - *Variables*
    - *Terms*

2.1 Preface

Citation Style Language 1.0 represents a significant update of the Citation Style Language (CSL), an open XML language to describe citation styles. This document describes the changes compared to CSL 0.8 (released in March 2009).

Nota Bene These notes include many XML examples to demonstrate new CSL features. Please note that these examples are reduced to a minimum, and often don't constitute valid CSL styles. In addition, when references are made to the XML elements of CSL, the CSL namespace ("cs") is always attached as a prefix (e.g. `cs:citation` for the `<citation/>` element). However, for CSL styles it is customary to declare the default CSL namespace on the root `cs:style` element (`<style xmlns="http://purl.org/net/xbiblio/csl"/>`), which eliminates the need to include this namespace for each element.

2.2 Project Home(s)

The first change is that CitationStyles.org is the new home of the CSL project (the old home was located at http://xbiblio.sourceforge.net/csl/). The new website will be the central location to find CSL schemas, documentation and styles.

For CSL development the SourceForge xbiblio mailing list will be kept in use. Code hosting has moved around a bit. A switch from SourceForge.net to Bitbucket in 2010 has been followed by a move to GitHub in 2011.

2.3 Documentation

Prior to the release of CSL 1.0, CSL documentation was rather scarce. Most of what was available was hosted on the Zotero wiki. With CSL 1.0, we improved on this situation. In addition to these upgrade notes, a full specification has been made available. As before, the schema itself is also a source of information, although reading the schema requires some understanding of RELAX NG Compact, the XML schema language in which the CSL schema is written.

2.4 Schema

2.4.1 Split Schema

Whereas the CSL 0.8 schema consists of a single file, the CSL 1.0 schema has been split into the following files:

- csl.rnc - The core of the CSL schema. This file contains most of the schema logic and calls the other schema files
- csl-categories.rnc
- csl-terms.rnc
- csl-types.rnc
- csl-variables.rnc

The main advantage of splitting up the schema is that the schema will be easier to maintain. If you wish to validate styles against the CSL schema, make sure all the files are located in the same directory, and validate against csl.rnc.

2.4.2 Validation

The CSL 1.0 schema has been extended with two Schematron rules to make sure styles don't use `cs:text` and `cs:key` elements that call non-existing `cs:macro` elements. Note that not all validators support embedded Schematron code (e.g. Jing just ignores the rules). In addition, the CSL 1.0 schema can now be used to validate the locales files (e.g. "locales-en-US.xml"), which contain localizations of terms, date formats and style options.

2.5 Styles

2.5.1 Updating CSL 0.8 Styles

CSL 1.0 is backward incompatible with CSL 0.8, which means that CSL 0.8 styles don't work with CSL 1.0 processors. Fortunately, it is possible to (automatically) update CSL 0.8 styles to the CSL 1.0 format using the upgrade.xsl XSLT stylesheet. This conversion has been performed for all the styles in the Zotero Style Repository. However, if you use (custom) CSL styles that aren't included in this style repository, you might need to do this yourself.

Using upgrade.xsl

First, check whether the styles that you wish to update validate against the CSL 0.8.1 schema.

Then use an XSLT processor to update the styles. Available options are the command-line tools Saxon and xsltproc. Alternatively, one of the (more user-friendly) online converters, such as the one offered by www.shell-tools.net can be used. For the latter tool, the instructions are:

1. Paste the contents of upgrade.xsl into the "xslt" text box at http://www.shell-tools.net/index.php?op=xslt
2. Paste the contents of the CSL 0.8 style into the "xml" text box
3. Click the "Submit Query"-button
4. Copy the text from the "output" text box to a suitable text editor (e.g. Notepad on Windows) and save the file with a .csl-extension.

Finally, after the conversion, it is recommended to validate the converted style, this time against the CSL 1.0 schema.

2.5.2 Version Number

Starting with CSL 1.0, styles (and locales files) must indicate the CSL version with which they are compatible. All CSL 1.0 styles should include the `version` attribute with the value "1.0" on the `cs:style` element, e.g.:

```
<style version="1.0" class="in-text"/>
```

For "locale-xx-XX.xml" files this attribute should be set on the root `cs:locale` element.

2.5.3 Options as Attributes

In CSL 0.8, citation- and bibliography-specific style options were set with `cs:option` elements. In CSL 1.0 this element is no longer used. Instead, options are set using attributes. CSL 1.0 includes global options, which affect the output of both the `citation` and `bibliography` sections, and which are set as attributes on the `cs:style` element. Options that are citation-specific are now set on the `cs:citation` element, while bibliography-specific options are set on the `cs:bibliography` element.

2.5.4 Style Metadata

Links

The role of the `cs:link` element is to store URLs (using the `href` attribute) while indicating how these URLs are related to the style (using the `rel` attribute). The use of `rel` has been slightly modified in CSL 1.0 to increase clarity. First, every `cs:link` element should carry the `rel` attribute. Secondly, the available values of `rel` have changed to:

- "self". The URI of the CSL style itself (only for independent styles). In CSL 0.8 "self" was implicit when `ref` wasn't used.
- "independent-parent". Renamed from "source", this indicates the URI of the independent parent style (only for dependent styles).
- "template". Can be used to link to the style from which the current style is derived.
- "documentation". Links to documentation, e.g. the style guide. Note that "documentation" has replaced "homepage".

Formats and Fields

The `cs:category` element has two purposes: to indicate for which field(s) of study a style is relevant (e.g. "biology") and to indicate the format of the style (e.g. "author-date"). In CSL 0.8 the `term` attribute was used for both cases. With CSL 1.0, `term` has been replaced with two attributes: `citation-format`, to indicate the citation format, and `field`, to indicate the field of study. An example:

```
<style>
  <info>
    <category citation-format="author-date"/>
    <category field="biology"/>
  </info>
</style>
```

In CSL 0.8 the possible citation formats were: "author-date", "label", "note", "numeric" and "in-text". In CSL 1.0 "in-text" has been replaced with "author" (a format that only shows author names in in-text citations, like the MLA style).

ISSN and ISSN-L

ISSN-identifiers unambiguously identify journals. While CSL 0.8 allowed only a single ISSN identifier to be included in the style metadata section, CSL 1.0 now supports multiple ISSNs (e.g. the ISSNs of the print and online editions of a journal), as well as the relatively new ISSN-L identifier. For example:

```
<style>
  <info>
    <issn>0099-2240</issn>
    <eissn>1098-5336</eissn>
    <issnl>0099-2240</issnl>
  </info>
</style>
```

2.5.5 Localization

The "locales-xx-XX.xml" files (with "xx-XX" indicating the locale, e.g. "en-US" for "English - United States") that are part of CSL previously had only the role of supplying localized terms. With CSL 1.0, these locale files also contain localized style options and localized date formats. Because of this, some changes have been made to the XML format of these files: `cs:locale` has replaced `terms` as the root element, the `xml:lang` and `xmlns` attributes are now applied to the `cs:locales` element, and several new elements have been introduced for the localization of dates and options (`cs:date`, `cs:date-part` and `cs:style-options`).

As before, it is possible to use the `cs:locale` element in styles to override any content of the "locales-xx-XX.xml" files. The `cs:locale` element can be used with or without the `xml:lang` attribute. If `xml:lang` is not set, the contents of the `cs:locale` element will be used for all locales. If `xml:lang` is set to a locale code, the content of the `cs:locale` element will override the content of the specified locale. N.B. a `cs:locale` element with the `xml:lang` attribute takes priority over a `cs:locale` element without the attribute. For example,

```
<style>
  <locale>
    <terms>
      <term name="et-al">et alii</term>
    </terms>
  </locale>
  <locale xml:lang="en">
    <style-options punctuation-in-quote="true" />
    <terms>
      <term name="et-al">and others</term>
    </terms>
    <date form="text">
      <date-part name="month" suffix=" " form="short"/>
      <date-part name="day" suffix=", "/>
      <date-part name="year"/>
    </date>
  </locale>
</style>
```

with regard to the "et-al" term, this will result in the use of "and others" for the English locales, and of "et alii" for all other locales.

Localized Dates

CSL 1.0 introduces support for date localization. This feature is optional: styles can still define dates in the usual non-localized format. To use a localized date, all you need to do is use `cs:date` with the `form` attribute set to either `text` (for dates like 'April 21, 2008') or `numeric` (e.g. '4/21/08'). As demonstrated in the example below, it is not necessary to specify any `cs:date-part` elements for localized dates:

```
<style>
  <bibliography>
    <layout>
      <!-- old-fashioned, unlocalized date -->
      <date variable="accessed">
        <date-part name="year"/>
        <date-part name="month" form="numeric" prefix="-"/>
        <date-part name="day" prefix="-"/>
      </date>
      <!-- default localized date -->
      <date variable="accessed" form="numeric"/>
    </layout>
  </bibliography>
</style>
```

The format of localized dates (i.e. punctuation and the order of the date-parts) is specified in the "locales-xx-XX.xml". As with terms, localized date formats can be overridden within styles. In addition, localized dates can be customized within a style via two options. First, the `date-parts` attribute can be added to `cs:date` to control which date-parts are shown. With the default value of `year-month-day` the whole date is shown. With `year-month` and `year` only the year/month and year date-parts are shown, respectively. The second option is the ability to redefine how one or more `cs:date-part` element are formatted. Note that the order of `cs:date-part` elements for a localized date within `cs:layout` doesn't affect the rendering order of the date-parts (this in contrast with non-localized dates or dates specified within `cs:locale`, where the order of the `cs:date-part` elements does control the rendering order). Neither does the presence or absence of `cs:date-part` elements affect which date-parts are shown (this is controlled via the `date-parts` attribute described above). Instead, `cs:date-part` elements allow you to override specific properties of the localized date-parts (e.g. the `form` attribute of the month-date-part can be set to "short"). Note that changes made in this way affect all locales. An example illustrating the different options:

```
<style>
  <!-- a modified date format for the English locale -->
  <locale xml:lang="en">
    <date form="text">
      <date-part name="month" suffix=" " form="short"/>
      <date-part name="day" suffix=", "/>
      <date-part name="year"/>
    </date>
  </locale>
  <bibliography>
    <layout>
      <!-- localized date that only shows the year and month -->
      <date form="text" date-parts="year-month"/>
      <!-- localized date in numeric format with leading zeros -->
      <date form="numeric">
        <date-part name="month" form="numeric-leading-zeros"/>
        <date-part name="day" form="numeric-leading-zeros"/>
      </date>
    </layout>
  </bibliography>
</style>
```

In developing CSL 1.0 it was recognized that robust date localization requires a clear distinction between style-specific and locale-dependent formatting of dates. As a result, some limitations have been placed on the use of `cs:date` when used for localized dates. First, affixes (prefixes and suffixes) on `cs:date` are considered style-specific formatting (e.g. parentheses around the date: "(2000)"). It is therefore not allowed to apply affixes to `cs:date` when this element is used within `cs:locale` (in both styles and "locales-xx-XX.xml" files). Instead, all locale-specific affixes should be applied to the `cs:date-part` elements. Conversely, it is not allowed to apply affixes to `cs:date-part` elements when the parent `cs:date` calls a localized date. Secondly, `cs:date` may not carry the `delimiter` attribute when used in a style to call a localized date. In CSL 1.0 this attribute can be used to specify a delimiter for the date-parts, which is considered locale-specific formatting.

N.B. When creating a localized date format, consider graceful scaling of dates when applying affixes to the `cs:date-part` elements. As an example, consider the date format "May 1, 2008". By using the following arrangement of affixes, correct dates are obtained for any value of the `date-parts` attribute:

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

`date-parts` value	date
"year-month-day"	"May 1, 2008"
"year-month"	"May 2008"
"year"	"2008"

Localized Options

In addition to localized dates and terms, CSL 1.0 now also supports localized options (although for now, there is only one such an option, `punctuation-in-quote`). The default value of localized options is set for each locale in the "locales-xx-XX.xml" files, but these values can be overridden using the `cs:style-options` element within `cs:locale` in a CSL style. An example:

```
<style>
  <locale xml:lang="en">
    <style-options punctuation-in-quote="true"/>
  </locale>
</style>
```

Default Locale

To prevent localization of styles (which might be desirable for journal-specific styles) the `default-locale` attribute can be included on the `cs:style` element (this attribute already existed in CSL 0.8, but was not supported by Zotero). Its value should be a locale code (e.g. "fr-FR" for French). An example:

```
<style default-locale="fr-FR"/>
```

N.B. With CSL 0.8 there was some confusion about the use of `default-locale`, and some style authors included the `xml:lang` attribute instead. In CSL 1.0 `xml:lang` is no longer allowed as an attribute on `cs:style`.

2.5.6 Formatting

Citation Collapsing

CSL 1.0 offers finer control of citation collapsing. First, two new options have been introduced, both of which are set as attributes on `cs:citation`: `year-suffix-delimiter`, which defines the delimiter for subsequent year suffixes (e.g. the comma in "Doe 2000a,b, Smith 1999"), and `after-collapse-delimiter`, which defines the delimiter between a group of collapsed citations and the subsequent citation (e.g. the semicolon in "Doe 2000a, b; Smith 1999, Williams 2002"). Both attributes default to the delimiter set on the `cs:layout` element within `cs:citation`. Secondly, "year-suffix-ranged" has been added as a possible value of the `collapse` attribute of `cs:citation`. If `collapse` is set this value, citations are collapsed as with "year-suffix", but ranges of year-suffixes are collapsed as well (e.g. "Doe 2000a,b,c,e" would become "Doe 2000a-c,e"). An example of how these attributes are set:

```
<style>
  <citation collapse="year-suffix-ranged" year-suffix-delimiter="," after-collapse-delimiter=";">
    <layout delimiter=", " />
  </citation>
</style>
```

Dates

AD and BC

CSL 1.0 includes two new terms, `ad` and `bc` (Anno Domini and Before Christ). These terms are automatically appended to years: `bc` is added to negative years (e.g. 2500BC), while `ad` is added to positive years of less than four digits (79AD).

Date Ranges

CSL 1.0 adds support for date ranges. By default, date ranges are delimited by an en-dash (e.g. May–July 2008). Custom delimiters can be set on the `cs:date-part` elements with the new `range-delimiter` attribute. The attribute value set on the largest date-part ("day", "month" or "year") that differs between the two dates of the date range will then be used instead of the en-dash. For example,

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

would result in "May–July 2008" and "May 2008/June 2009".

Seasons

CSL 1.0 includes four new season terms, `season-01` to `season-04` (respectively Spring, Summer, Autumn and Winter). If a date includes a season instead of a month, the season term will substituted the month date-part. 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".

Uncertain Dates

Two new features of CSL 1.0 allow for special formatting of uncertain dates. First, CSL 1.0 introduces the `is-uncertain-date` conditional. This conditional tests "true" when a date is flagged as uncertain. The second addition is the new "circa" term. For example,

```
<style>
  <citation>
    <layout delimiter="; ">
      <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" (certain date) and "ca. 2003" (uncertain date).

Names

delimiter-precedes-last

The `delimiter-precedes-last` attribute on `cs:names` controls the use of the name delimiter between the last and next-to-last name in name lists. In CSL 0.8, this attribute could be set to either `always` or `never`. To include the delimiter for lists of three or more names ("Doe, Smith, and Williams") and to exclude it for lists of only two names ("Doe and Smith"), you would have to leave out the attribute. Now, in CSL 1.0, it is also possible to explicitly set the last behavior by using the value "contextual".

Editorial Director

CSL 1.0 includes a new name variable, `editorial-director`. This addition is mostly specific to French, where the "Directeur de la publication" role is common.

Editor/Translator Name Collapsing

If a `cs:names` element has its `name` attribute set to "editor translator" (or "translator editor"), CSL 1.0 collapses both name lists when their contents is identical. If a label is specified, and collapsing occurs, the newly added `editortranslator` term is used. For example,

```
<names variable="editor translator">
  <name />
  <label form="short" prefix=" (" suffix=")" />
</names>
```

could result in "John Doe (ed. & trans.)".

Et al.

A new element, `cs:et-al`, can now be included within `cs:names`. This adds two important features. First, formatting can now be set independently for the `et-al` term. For example,

```
<names variable="author">
  <name/>
  <et-al font-style="italic" prefix=" "/>
</names>
```

results in "Doe *et al.*". Secondly, it is now possible to use two different `et-al` terms within a single style (e.g. one for in-text citations and one for the bibliography). The desired term, "et-al" (the default) or "and others", is set with the `name` attribute on the `cs:et-al` element. For example,

```
<names variable="author">
  <name/>
  <et-al term="and others" prefix=" "/>
</names>
```

would yield "Doe and others" (note that both terms are localized).

Formatting Name Parts

CSL 1.0 introduces the ability to separately format given and family names. Formatting is specified via the new `cs:name-part` element, a child of `cs:name`. The `name` attribute of this element should be set to either "family" or "given". Note that the order of `name-part` elements does not affect the order in which the name parts are shown. An example, resulting in names like "John SMITH":

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

Inheritable Name Options

In CSL 0.8, any attribute used for name formatting had to be included for each occurrence of the `cs:names` element, even if names were identically formatted for all these elements. To reduce the need for duplication, CSL 1.0 introduces inheritable options: the attributes `and`, `delimiter-precedes-last`, `initialize-with`, `name-as-sort-order` and `sort-separator` can now also be set on `cs:style`, `cs:citation` and `cs:bibliography`. The attributes `form` and `delimiter` have been made available as `name-form` and `name-delimiter`, respectively, as the original attribute names have different uses when set on `cs:style`, `cs:citation` and `cs:bibliography`. Similarly, the `names-delimiter` attribute has been introduced as a companion of the `delimiter` attribute on `cs:names`.

When a name attribute is set on `cs:style`, `cs:citation` or `cs:bibliography`, its value is used for all `cs:names` elements within the element carrying the attribute. However, when an element lower in the hierarchy carries the same attribute with a different value, this value will override the value(s) specified higher in the hierarchy.

In addition to these changes, CSL 1.0 also includes more fine-grained control for et-al settings. The attributes `et-al-min`, `et-al-use-first`, `et-al-subsequent-min`, `et-al-subsequent-use-first` now behave like any other `cs:name` attribute, and thus can be set on `cs:style`, `cs:citation`, `cs:bibliography` and `cs:name`.

Hyphenation of Compound Given Names

A new attribute, `initialize-with-hyphen`, can be set on `cs:style` to control hyphenation of compound given names (e.g. "Jean-Luc Picard"). When set to "true" (the default), a hyphen is added when the given name is initialized ("J.-L. Picard"). With "false" the hyphen is left out ("J.L. Picard").

Name Particles

Many Western names consist not only of a given and a family name, but also of one or more name particles. To control how these particles are handled, CSL 1.0 introduces a new option, `demote-non-dropping-particle`, which can be set as an attribute on `cs:style`. To understand how this option works, it is important to recognize that not all particles are equal: name particles can be either kept or dropped when only the surname is shown (from now on we will refer to these two types as non-dropping-particle and dropping-particle, respectively). A single name can contain particles of both types (in this case the non-dropping-particle always comes after the dropping-particle). For example, the French name "Gérard de la Martinière" can be deconstructed into:

"Gérard"	given name
"de"	dropping-particle
"la"	non-dropping-particle
"Martinière"	family name

When only the surname is shown, only the non-dropping-particle is kept: "La Martinière". However, when names are inverted (with the family name preceding the given name), styles often differ in name particle handling. First, the non-dropping-particle can be either prepended to the family name (e.g. "de Koning, W.") or appended (after initials or given names, e.g. "Koning, W. de"). Note that the dropping-particle is always appended in inverted names. Secondly, if the non-dropping-particle is prepended to the family name, names can be sorted in two ways: the non-dropping-particle can remain part of the family name (as part of the primary sort key; example A), or it may be separated from the family name and become (part of) a secondary sort key, joining the dropping-particle, if available (example B). The different sort orders are illustrated below:

Sort order A: non-dropping-particle not demoted

- primary sort key: "la Martinière"
- secondary sort key: "de"
- tertiary sort key: "Gérard"

Sort order B: non-dropping-particle demoted

- primary sort key: "Martinière"
- secondary sort key: "de la"
- tertiary sort key: "Gérard"

The `demote-non-dropping-particle` attribute can be set to the following 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 Martinière, Gérard de"). The non-dropping-particle is part of the primary sort key (example A, e.g. "de Koning, W." appears under "D").
- "sort-only": as "never", with the exception that non-dropping-particle is demoted to a secondary sort key (see example B, e.g. "de Koning, W." appears under "K").
- "display-and-sort" (default): the dropping and non-dropping-particle are appended to the rest of the name (e.g. "Koning, W. de" and "Martinière, Gérard de la"). When names are sorted, both particles are part of the secondary sort key (see example B, e.g. "Koning, W. de" appears under "K").

Ordinal Numbers

To allow for localization of ordinal numbers, CSL 1.0 includes the new terms `ordinal-01` to `ordinal-04`. For the en-US locale, these terms have the values "st", "nd", "rd" and "th" (resulting in ordinal numbers of "1st", "2nd", "3rd", "4th", etc.). In addition, support for long ordinals has been introduced with the terms `long-ordinal-01` to `long-ordinal-10` ("first", "second", ..., "tenth"). Long ordinals can be selected by using `cs:number` and setting the `form` attribute to "long-ordinal".

Original Publisher

Sometimes (older) books are republished by a different publisher. To indicate the original publisher, and the location of the original publisher, CSL 1.0 adds two new variables, `original-publisher` and `original-publisher-place`. Note that CSL 0.8 already included the name variable `original-publisher`, which could only be used with `cs:names`. The variables `original-publisher` and `original-publisher-place` in CSL 1.0 are 'normal' variables, and can be used with `cs:text`.

Pages

CSL 1.0 introduces two new page variables: "page-first" and "number-of-pages". The existing variable "pages" is still used for page ranges (e.g. of journal articles and book chapters). The variable "page-first" holds the first page of the page range. The variable "number-of-pages" is used to indicate the total number of pages of an item (e.g. a book or thesis).

In addition, a new global (non-localized) option, `page-range-format`, has been added to control the collapsing of page ranges. This attribute, set on `cs:style`, can have the values "expanded" (e.g. "321-328"), "minimal" ("321-8"), and "chicago" ("321-28", which follows the collapsing rules of the Chicago Manual of Style). When the attribute isn't present, the content of the "page"-variable is shown as is. An example:

```
<style page-range-format="chicago">
  <bibliography>
    <layout>
      <text variable="page"/>
    </layout>
  </bibliography>
</style>
```

Pluralization

In CSL 1.0 small changes have been made to the use of `plural`. As with CSL 0.8, this attribute can be set on `cs:text` and `cs:label`. When used on `cs:text`, `plural` can still be set to "false" to use the singular form of a term (the default), or to "true" to use the plural form. But when used on `cs:label`, different values are now available. With "contextual" (the default value), the plurality of the variable determines whether the singular or plural form of the term is used (e.g. "page 43" and "pages 3-5"). With "never" and "always" respectively the singular or plural form of the term is used, regardless of the plurality of the variable. An example:

```
<number variable="edition" form="ordinal"/>
<text term="edition" plural="false"/>
<group>
  <label variable="page" plural="always"/>
  <text variable="page"/>
</group>
```

N.B. The `plural` attribute was one of the few cases where the implementation in Zotero did not follow the CSL 0.8 schema.

Quotes

CSL 1.0 introduces new terms for inner ("open-inner-quote" and "close-inner-quote", e.g.   and) and outer quotes ("open-quote" and "close-quote", e.g.   and). Together with the new `punctuation-in-quote`-option (see *Punctuation around Quotes*), quotes applied with the `quotes` attribute are now fully localized.

Quote Substitution

If a field (e.g. a variable) contains a matching set of quotation marks (", ', or the quotation marks defined by the `open-inner-quote` and `close-inner-quote` terms), then these quotation marks are replaced by those defined by the `open-quote` and `close-quote` terms. For example:

```
<text value="Voyage of 'The Beagle'"/>
```

will render as: Voyage of The Beagle

Flipflopping

Flipflopping occurs when a field (e.g. a variable) contains a matching set of quotation marks (", ', or the quotation marks defined by the `open-quote`, `close-quote`, `open-inner-quote` and `close-inner-quote` terms), or when it contains markup for italics or boldfacing (see *Rich Text Markup within Fields*). For example:

```
<text prefix="Speak, " value="'friend'"  suffix=", and enter" quotes="true"/>
```

will render as Speak friend, and enter. Quotes flipflop between inner (`open-inner-quote` and `close-inner-quote`) and outer (`open-quote` and `close-quote`) quotes. Italics flipflop between italics and the normal font-style, and boldface between bold and the normal font-weight.

Punctuation around Quotes

The localized option `punctuation-in-quote` is used to specify whether punctuation (commas and periods) should appear within ("true", e.g. for American English) or outside quotation marks that have been applied by the style ("false" (default value), e.g. for British English). As such, it can toggle the style output between

> Douglas Adams, "The Hitchhiker's Guide to the Galaxy," 1979.

and

> Douglas Adams, "The Hitchhiker's Guide to the Galaxy", 1979.

Rich Text Markup within Fields

Although not part of the CSL 1.0 specification, the new citeproc-js CSL processor used by Zotero supports an exciting new feature: the ability to use rich text markup within item fields. This markup is applied with a small set of HTML(-like) tags:

- `<b>` - bold
- `<i>` - italics
- `<sc>` - small-caps
- `<sub>` - subscript
- `<sup>` - superscript

E.g. if a Zotero item has the title "Ca<sup>2+</sup> levels in <i>Homo sapiens</i>", this will render as "Ca^{2+}levels in *Homo sapiens*". Rich text markup can also be used with the `value` attribute of `cs:text`, but here special XML characters ("<", ">") have to be escaped, e.g.:

```
<text value="&lt;b&gt;some bold text&lt;/b&gt;"/>
```

In contrast, markup used with the `prefix` and `suffix` attributes is not recognized. Finally, note that bold and italics markup are subject to *flipflopping*.

second-field-align

The `second-field-align` attribute can be used to align any subsequent lines of a bibliography entry with the beginning of the second field. In CSL 0.8 the value of this attribute could be set to "true" or "margin" to place the first field respectively in the margin, or flush against it. In CSL 1.0 "true" has been renamed to "flush".

Stripping Periods

A new attribute, `strip-periods`, can now be set on `cs:date-part` (only for name=”month”), `cs:label` and `cs:text`. The attribute is inactive when set to “false” (the default value), but if set to “true”, any periods are stripped from the variable contents. `strip-periods` replaces the `include-period` attribute that was part of CSL 0.8.

`strip-periods` is especially useful for journal abbreviations. There are plans to improve support for journal abbreviations in future versions of CSL (e.g. by using lookup lists to find the correct journal abbreviation given a certain journal title), but for now it is recommended that users include periods for journal abbreviations in their (Zotero) libraries. With the help of `strip-periods`, styles can then either use the journal abbrevation as is, or use a version without periods. An example:

```
<text variable="container-title" form="short" strip-periods="true"/>
```

would output “Appl Environ Microbiol” if the journal abbreviation for the Zotero item is “Appl. Environ. Microbiol.”.

Year Suffix

Year-suffixes are included automatically when the `disambiguate-add-year-suffix` attribute on `cs:citation` is set to “true”. However, some styles desire special markup of year-suffixes, such as italics (e.g. “`2000`*a*, *b*”). For this CSL 1.0 introduces the `year-suffix` variable, which can be used to explicitly specify the location and formatting of year-suffixes. An example:

```
<style>
  <citation>
    <layout delimiter=", ">
      <date variable="issued">
        <date-part name="year"/>
      </date>
      <text variable="year-suffix" font-style="italic"/>
    </layout>
  </citation>
</style>
```

Layout Control with the Display Attribute

CSL 0.8 included a `display` attribute, with possible values of “block” or “inline-block”, intended to provide some control over the layout of bibliography entries. However, it remained unimplemented by any known processor, and was not used in any known styles. In CSL 1.0, the `display` attribute has been refined and extended: it is now restricted to rendering-element children of `cs:layout` under `cs:bibliography`, and has possible values of “block”, “left-margin”, “right-inline”, and “indent”.

By leveraging the styling features of the target rendering platform (HTML, a word processor, a document processing system), the enhancements to `display` permit the implementation of sophisticated formatting effects, such as publication listings headed by the name of each author. See the CSL 1.0 Specification for further details on the use of this attribute.

2.5.7 Disambiguation

The disambiguation algorithm specified in CSL 0.8 followed the Chicago Manual of Style. CSL 1.0 supports additional disambiguation methods through the addition of a new `givenname-disambiguation-rule` attribute, which can be used in combination with the existing `disambiguate-add-names` and `disambiguate-add-givenname` attributes.

The `givenname-disambiguate-rule` option accepts values of "all-names", "all-names-with-initials", "primary-name", "primary-name-with-initials", and "by-cite" *[0]. The first value specifies the Chicago Manual of Style method, which assures that all names included in citations uniquely identify the relevant author. The second does the same, but will not expand initialized names. The third and fourth values specify analogous methods, but here the transformation of names is limited to the first-listed name. The last option transforms names only as necessary to uniquely identify references listed in the bibliography. A more detailed discussion of the disambiguation options can be found in the CSL 1.0 Specification.

2.5.8 Sorting

CSL 1.0 includes several new features to allow for more complex reference sorting. The first change is that the `form` attribute on `cs:name` can now be set to "count". With this value, the enclosing `cs:names` returns the number of names in the name variable instead of the names themselves. When used for a sort key, this makes it possible to sort according to the number of authors (or any other kind of contributor). An example:

```xml
<style>
  <macro name="author">
    <names variable="author">
      <name form="count"/>
    </names>
  </macro>
  <bibliography>
    <sort>
      <key macro="author"/>
    </sort>
    <layout/>
  </bibliography>
</style>
```

The second change consists of two new attributes for the `cs:key` element, `names-min` and `names-use-first`. These attributes, when set, override the values of `et-al-min` and `et-al-min-first`, respectively. The following example shows how with these attributes a bibliography can be sorted alphabetically, while only taking the first author into account:

```xml
<style>
  <macro name="author">
    <names variable="author">
      <name/>
    </names>
  </macro>
  <bibliography>
    <sort>
      <key macro="author" names-min="1" names-use-first="1"/>
    </sort>
    <layout/>
  </bibliography>
</style>
```

2.5.9 Back-References in Note Styles

CSL 1.0 adds two features related to back-referencing in note styles. First, the `position` conditional supports a new value, "near-note". It tests true when an item has been previously cited, and the distance between the current and most recent use (measured in number of footnotes or endnotes) does not exceed the value of the new

[0] A hat tip to user komrade of the Zotero forums, whose review of the major style guides led to this set of disambiguation options.

`near-note-distance` attribute. This attribute, which has a default value of 5, may be set on `cs:citation`. An example:

```
<style class="note">
  <citation near-note-distance="3">
    <layout>
      <choose>
        <if position="near-note">
          ...
        </if>
        <else>
          ...
        </else>
      </choose>
    </layout>
  </citation>
</style>
```

N.B. The value of “near-note” is *always* false for references that are not in a footnote/endnote.

The second feature is the new `first-reference-note-number` variable. When an item has been previously cited, this variable holds the number of the first note to cite the item.

2.5.10 Variables

CSL 1.0 introduces a number of new variables:

- “event-date”. Replaces “event” date variable.
- “first-reference-note-number”. See *Back-References in Note Styles*
- “jurisdiction”. The geographic unit for which a resource (such as legislation) is relevant.
- “number-of-pages”. See *Pages*.
- “page-first”. See *Pages*.
- “year-suffix”. See *Year Suffix*.

2.5.11 Terms

CSL 1.0 also introduces a number of new terms:

- “ad” and “bc”. See *AD and BC*.
- “author”. This term can be applied as a label to author names. For most styles, this term will consist of an empty string (“”).
- “by”.
- “editortranslator”. See *Editor/Translator Name Collapsing*.
- “ordinal-01‘‘ to “ordinal-04” and “ordinal-01” to “ordinal-10”. See *Ordinal Numbers*.
- “original-publisher” and “original-publisher-place”. See *Original Publisher*.
- “reference”. This term replace the “references” term from CSL 0.8.
- “season-01” to “season-04”. These terms map to respectively spring, summer, fall and winter. See *Seasons*.