EVT Editor Manual

For EVT v. 1.3

Introduction
  About EVT
  How it Works
  Main Features
  A Quick Guide to EVT
0. Installation
1. Images
  1.1 Single Side Images
  1.2 Double Side Images
  1.3 Hotspot Images
  1.4 Other Images
2. TEI XML Transcription Files
  2.0 TEI Header
  2.1 Facsimile
  2.2 Parallel Transcription
  2.3 Embedded Transcription
  2.4 Support for named entities
  2.5 Notes
    2.5.1 Stand-off notes
    2.5.2 Critical notes
  2.6 Tables and columns
  2.7 Edition levels
  2.8 Translations
  2.9 Stage directions
  2.10 Other languages
  2.11 Chronological index
3. VisColl: describing and visualizing the manuscript quire structure
  3.1 The VisColl data model
4. Configuring EVT
  4.1 The Configuration File
    Image extension
    Web site of the project
    Edition/Project title and Badge
    Image frame (on/off)
    Book Reader view (on/off)
    Regesto (on/off)
    Information about the current document (on/off)
    Manuscript description (on/off)

- Information about the project (on/off)
- Edition levels
- Translation
- Frame content showed at first loading of the edition
- Image-text linking button (on/off)
- Hot-spot button (on/off)
- Magnifying lens button (on/off)
- Thumbnails button (on/off)
- Viscoll button (on/off)
- Path to the XML file containing the VisColl Collation Model
- Path to the XML file containing the VisColl Image List
- Edition level selector
- Page selector position (left/right)
- Page grouping based on document (on/off)
- Tooltip for page selector
- Search and virtual keyboard (on/off)
- Navigation bar (on/off and initial status)
- Document navigation button (on/off)
- Prose/Lines visualization button (on/off)
- Language tooltip (on/off)
- Person list (on/off)
- Place list (on/off)
- Organization list (on/off)
- Term list (on/off)
- Gloss list (on/off)
- Chronological index for texts (on/off)
- Customizable filters
- Final remarks

4.2 Fine-tuning the transformation results, custom templates

- Bookreader view
- The keyboard_config.xml file
- The language files
- The evt_builder-custom-templates.xsl file
- The evt_builder-custom-styles.css file
- The structure.xml file

4.3 Testing an EVT-based edition

5. Step by step instructions

Known bugs (and workarounds)

Warnings

References

Introduction

This guide is aimed at the editor interested in creating an image-based edition using EVT, and not at the reader/user of such edition. To use EVT, the editor must have a good knowledge of the TEI XML markup, and must be able to apply XSLT style sheets to an XML document. EVT main configuration is done by editing a single .xsl file (see section *3. Configuring EVT*), note however that the software can be customized as desired modifying the relevant XSLT and CSS style sheets, or adding new ones (f.i. for a different kind of text visualization).

About EVT

EVT (Edition Visualization Technology) is a software for creating and browsing digital editions of manuscripts based on text encoded according to the TEI XML schemas and Guidelines. This tool was born as part of the DVB (Digital Vercelli Book) project in order to allow the creation of a digital edition of the Vercelli Book, a parchment codex of the late tenth century, now preserved in the Archivio e Biblioteca Capitolare of Vercelli and regarded as one of the four most important manuscripts of the Anglo-Saxon period as regards the transmission of poetic texts in the Old English language.

To ensure that it will be working on all the most recent web browsers, and for as long as possible on the World Wide Web itself, the edition-browsing component is built on open and standard web technologies such as HTML, CSS and JavaScript. Specific features, such as the magnifying lens, are entrusted to jQuery plugins, again chosen among the open source and best supported ones to reduce the risk of future incompatibilities. The general architecture of the software, in any case, is modular, so that any component which may cause trouble or turn out to be not completely up to the task can be replaced easily.

How it Works

The basic idea of EVT is very similar to the *modus operandi* which is commonly used to convert TEI XML documents into HTML: when the main style-sheet is applied to the document, it starts a processing which ends with a website containing the digital edition of the manuscript. Our ideal goal, in fact, is to have a simple, very user-friendly drop-in tool, requiring little work and/or knowledge of anything beyond XML from the editor. To reach this goal, EVT is based on a modular structure where a single style-sheet (`evt_builder.xsl`) starts a chain of XSLT 2.0 transformations calling in turn all the other modules; the latter belong to two general categories: those devoted to building the HTML site, and the XML processing ones, which extract the edition text (the content of each folio which lies between `<pb/>` elements, or the content of each `<surface>` when the Embedded Transcription method is used) and format it according to the edition level. All XSLT modules live inside the `builder_pack` folder, in order to have a clean and well organized directory hierarchy.

Main Features

At the present moment EVT can be used to create image-based editions with two possible edition levels: diplomatic and diplomatic-interpretative. This means that a transcription encoded using elements of the TEI ***transcr*** module (see chapter 11 *Representation of Primary Sources* in the TEI *Guidelines*) should already be compatible with EVT, or require only minor changes to be made compatible. The Vercelli Book transcription is following the standard TEI schemas with no custom elements or attributes added: our tests with similarly encoded texts showed a high grade of compatibility. A critical edition level is being studied and will be added to the next version of EVT.

On the image side, several features such as a magnifying lens, a general zoom, image-text linking and more are already available. The image-text feature is inspired by Martin Holmes' Image Markup Tool software and was implemented in XSLT and CSS by one of the students collaborating to the project; all other features are achieved by using jQuery plugins.

A Quick Guide to EVT

To create an image-based edition EVT requires both images and TEI XML documents as input data. For the XSLT style sheets to work properly, all the input data must follow specific rules:

- images have to be named according to a strict naming convention and have to be available in a few different sizes/resolutions;
- TEI XML documents are likewise bound not only by the encoding schema, which must include the ***transcr*** module, but also by the use of several attributes whose values must follow rigid conventions.

All these conventions, which will be explained in full detail in the following sections, are necessary because they build a layer of rich metadata information which is accessed by the different XSLT style sheets to find all the relevant data to be extracted, processed and finally presented to the user.

After preparing all the image and textual data according to the conventions that will be explained below (sections *1. Images* and *2. TEI XML Transcription Files*), the user will be required to check and if necessary modify the EVT configuration file to suit her/his needs (see section *3. Configuring EVT*), and finally follow a step-by-step path to put everything in place and start the transformation chain that will result in a web-based edition (see section *4. Step by Step Instructions*).

This guide tries to include examples for each case described, however remember that the zipped archive available on the EVT home page and on SourceForge includes all text and image files necessary for the sample editions it holds, so that you can inspect them in case of doubt. In the `builder_pack/doc` folder you will find a first draft of the technical documentation, automatically generated thanks to the XSLTdoc system.

The main file for the transformation is `evt_builder.xsl`, which calls all the other `evt_builder-*.xsl` modules using XInclude.

As a first step of the process, the XSL style sheets that we consider "basic templates" are called into action: these are used to generate the index page in HTML and the different outputs depending on the edition levels specified by the editor. This is also the phase where all the parameters and variables that we defined in the configuration file are used by the transformation chain, and the structure for the page-by-page navigation is generated as well.

After the structural generation part is over, it is now the turn of the files designed for the implementation of specific features, such as text-image linking or the magnifying lens. The style sheets in the `modules/elements` folder are those defining the visual aspect of the edition text for the different edition levels.

0. Installation

EVT installation is quite easy: you only have to download the zipped archive from SourceForge and unpack it in some suitable folder on your hard disk. Everything is already in place, first of all you should browse the included examples clicking on the index.html document. You can also browse the available documents in the data/input_data folder, especially the TEI XML files, to understand what the building system requires; if you feel particularly adventurous you're welcome to browse the XSLT style sheets in the `builder_pack` and `builder_pack/modules` folders, note however that you should try to modify these files only if you know what you are doing!

The data/input_data folder is where your own data will go before you can use EVT to produce a web edition of your manuscript.

1. Images

As of version 1.1 EVT supports digitized images of the manuscript(s) saved in any of the most popular standards: JPEG, PNG, TIFF etc. Note, however, that

1. the actual image format must be stated using the corresponding extension in the main configuration file (see section *3. Configuring EVT*), e.g. `<xsl:param name="imageExt">jpg</xsl:param>`;
2. all images used in an edition must belong to the same format: if you choose JPEG, all of your images must be JPEGs and have the .jpg extension, the same is true for PNG, etc.

We suggest using one of the most popular and effective standards, such as PNG or JPEG, to reduce loading times in the web-edition.

Images should be provided in three different types:

- single side images
- double side images
- detail (hot spot) images

and the first two types must be made available in three different resolutions:

- standard
- high (mandatory for the Magnifier tool to work properly)
- thumbnail (only for single images, not yet used for double side images)

Note that, although only single side images are actually necessary, it is recommended to provide double side ones too, so that the Bookreader view is available to browse the manuscript showing verso-recto folios. If for some reason the double side images aren't available, however, the Bookreader view can be disabled in the configuration file (see below).

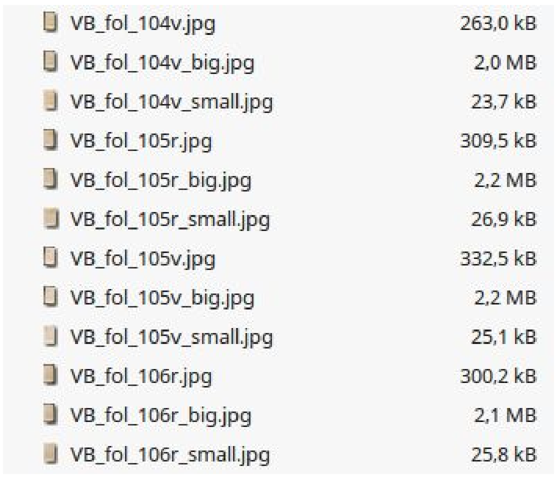
1.1 Single Side Images

These are single page/folio images of the manuscript:

- **standard** resolution: each image must be named "[folio_number].jpg" (e.g. "104v.jpg"), where "[folio_number]" is the value of the `xml:id` attribute of the `<pb/>` element (if using the Parallel Transcription method) or the `<surface>` element (if using the Embedded Transcription method) that is used to encode the corresponding folio;
- **high** resolution images: these are necessary for the magnifier (the proportion must be the same as the normal ones) and are named "[folio_number]_big.jpg" (e.g. "104v_big.jpg"), where "[folio_number]" is the name of the corresponding image in standard resolution (note the "**_big**" suffix);
- **thumbnails** of the standard images must be named "[folio_number]_small.jpg" (e.g. "104v_small.jpg"), where "[folio_number]" is the name of the corresponding image (note the "**_small**" suffix).

Important: note that it is recommended to use a meaningful prefix including, f.i., a reference to the manuscript (e.g. "VB_fol_104v", which means "folio 104v of the Vercelli Book manuscript"), useful when dealing with more than one manuscript; since at the present moment EVT can only handle one manuscript, this is not essential to its proper functioning.

Examples:



```
VB_fol_104v.jpg          263,0 kB
VB_fol_104v_big.jpg        2,0 MB
VB_fol_104v_small.jpg     23,7 kB
VB_fol_105r.jpg          309,5 kB
VB_fol_105r_big.jpg        2,2 MB
VB_fol_105r_small.jpg     26,9 kB
VB_fol_105v.jpg          332,5 kB
VB_fol_105v_big.jpg        2,2 MB
VB_fol_105v_small.jpg     25,1 kB
VB_fol_106r.jpg          300,2 kB
VB_fol_106r_big.jpg        2,1 MB
VB_fol_106r_small.jpg     25,8 kB
```

Note that you will have to find a suitable compromise with regard to image size/resolution and actual performance of the edition on the Web: bigger images may be better for detail inquiries, but they take longer to load and to navigate, especially on low speed Internet connections. The current beta2 of the Digital Vercelli Book is geared more on navigation speed than image detail.

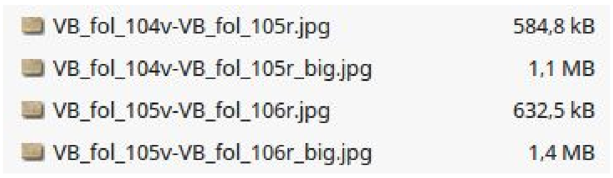
1.2 Double Side Images

These are double side images of the manuscript, showing the verso of a folio and the recto of the following one:

- **standard** resolution double folio images: these images are necessary for the functioning of the Bookreader view and must be named "[left_folio_number]-[right_folio_number].jpg" (e.g. "104v-105r.jpg"), where "[left_folio_number]" and "[right_folio_number]" are the names of images of the single facing folios, respectively the "verso" (on the left hand side) and the "recto" (on the right hand side); the double image name is therefore composed by the names assigned to the image of each single folio;
- **high** resolution double folios images: these images are necessary to activate the magnifier in the Bookreader view (the proportion must be the same as the normal ones) and must be named "[left_folio_number]-[right_folio_number]**_big.**jpg" (e.g. "104v-105r_big.jpg"), where "[left_folio_number]-[right_folio_number]" is the name of the corresponding double folio normal image.

Note that for the first and last page of the manuscript and for all the pages that do not have a "partner", it is still mandatory to have the images for the double view; however, it is sufficient that they be named as usual: [left_folio_number].jpg (e.g. “135v.jpg”) if it doesn't have the right partner, and [right_folio_number].jpg (e.g. “103r.jpg”) if it doesn't have the left partner. The same rule applies to the high resolution images, they must be named [left_folio_number]_big.jpg (e.g. “135v_big”) and [right_folio_number]_big.jpg (e.g. “103v_big.jpg”).

Examples:



VB_fol_104v-VB_fol_105r.jpg	584,8 kB
VB_fol_104v-VB_fol_105r_big.jpg	1,1 MB
VB_fol_105v-VB_fol_106r.jpg	632,5 kB
VB_fol_105v-VB_fol_106r_big.jpg	1,4 MB

1.3 Hotspot Images

These alternative images for specific areas of a manuscript folio, e.g. a more detailed version produced thanks to virtual restoration, possibly to be accompanied by a textual note. The only requirement is that they are named in the same way as they are referred to in the XML file, it is recommended that they are included in the <back> section of a text:

```
<back>
   <div type="hotspot">
     <div facs="VB_hs_106r_01" xml:id="VB_div_hs_106r_01">
       <p>The scribe here wrote a word, 'þrowode', that he
        later corrected in 'wolde' by erasing the first
        three letters and adding an 'l'. In this image you
        can see the contours of the erased letters.</p>
       <figure>
         <graphic url="VB_106r_HS_01.jpg"/>
       </figure>
     </div>
   </div>
 </back>
```

1.4 Other Images

Depending on specific project goals, it is possible that you have a set of images created for a specific purpose: in the Codice Pelavicino Digitale project, for example, all the notary graphic symbols have been collected to form a *signa tabellionis* database, and have been placed in a separate `images/signum` folder. Again this is supported by EVT, provided that the correct links to images are provided in the XML documents.

2. TEI XML Transcription Files

2.0 TEI Header

The <teiHeader> element at the top of the XML file must be compiled according to the standard TEI *Guidelines*, providing all the necessary information about the project and its curators. Note that if you decide to include one or more <msDesc> elements these will be used to show information pertaining to the corresponding manuscripts in the image frame, where the digitized manuscript images are showed.

Please also note that if you wish to take advantage of the support for named entities, this is where you should insert the lists (e.g. <listPerson>), this is explained in more detail in the *2.4 Support for named entities* section.

2.1 Facsimile

The <facsimile> element is where all the information necessary for image-text linking is saved, both for hot-spot and for continuous linking (e.g. page or line level) purposes. Note that you can do without this element in the transcription document, EVT will work and show your texts even if there are no images: to avoid having an empty image frame, though, you may want to set the image_frame variable in the configuration file (see below) to false() since EVT doesn't check the existence of this element (yet) and generates an image frame if that variable is left to true().

A <facsimile> holds as many <surface>s as there are single side images. Each <surface> includes:

- a corresp attribute pointing to the xml:id of the corresponding <pb/> element

```
<surface xml:id="VB_surf_104v" corresp="#VB_fol_104v">
```

- a <graphic> element: this can include an url attribute pointing to a specific file location, but at the present moment only height and width are used by EVT;

```
<graphic width="1200px" height="1800px"
         url="../images/Vercelli-Book_104V_S_300dpi.jpg"/>
```

- a number of <zone> elements: these are optionally used to record the coordinates of image areas corresponding to text elements and other metadata information, to do so they require use of several attributes:

◦ **ulx**, **uly**, **lrx**, **lry**	cartesian coordinates of the image area
◦ **rendition**	two possible values: **Line** (for line-by-line text-image linking) and **HotSpot** (for image areas to be used in hotspots)
◦ **xml:id**	the unique ID for the current `<zone>`
◦ **corresp**	points to the corresponding `<lb/>` ID (i.e. line) in the text transcription (OPTIONAL)

```
<zone corresp="#VB_lb_104v_01" lrx="1052" lry="211"
      rend="visible" rendition="Line" ulx="261" uly="156"
      xml:id="VB_line_104v_01"/>
```

Complete example:

```
<facsimile xml:id="VB_fac_dotr">
  <surface xml:id="VB_surf_104v" corresp="#VB_fol_104v">
     <graphic height="1800px" width="1200px"
              url="immagini\Vercelli-Book_104V_S_300dpi.jpg"/>
     <zone corresp="#VB_lb_104v_01" lrx="1052" lry="211"
          rend="visible" rendition="Line" ulx="261" uly="156"
          xml:id="VB_line_104v_01"/>
     <zone corresp="#VB_lb_104v_02" lrx="1072" lry="263"
          rend="visible" rendition="Line" ulx="257" uly="209"
          xml:id="VB_line_104v_02"/>
      [...]
     <zone corresp="#VB_lb_104v_23" lrx="1084" lry="1318"
          rend="visible" rendition="Line" ulx="262" uly="1269"
          xml:id="VB_line_104v_23"/>
     <zone corresp="#VB_lb_104v_24" lrx="1104" lry="1372"
          end="visible" rendition="Line" ulx="260" uly="1316"
          xml:id="VB_line_104v_24"/>
  </surface>
```

For more information about digital facsimiles in TEI XML see chapter 11 Representation of Primary Sources of the *Guidelines*.

2.2 Parallel Transcription

The Parallel Transcription method is the most popular and recommended way to couple a (semi-)diplomatic transcription with digitized manuscript images. This is what EVT expects to find in a TEI XML document created according to this method:

- **`<text>`** follows the **`<facsimile>`** element and holds the following items:

  - `<front>` front matter (information about a text, regesto (diplomatic text summary), etc.)
  - `<body>` actual text
  - `<back>` back matter (notes, comments, etc.)

- EVT also support the **`<group>`** element, so that you can have more than one <text> in a single TEI document, which is perfect for miscellaneous manuscripts:

```
<text>
    <front> [ front matter for the whole document ] </front>
    <group>
        <text>
            <front> [ front matter for the first text ] </front>
            <body> [ body of the first text ]           </body>
            <back> [ back matter for the first text ]   </back>
        </text>
        <text>
            <front> [ front matter for the second text] </front>
            <body> [ body of the second text ]          </body>
            <back> [ back matter for the second text ]  </back>
        </text>
        ... [ more texts ] ...
    </group>
    <back> [ back matter for the whole document ] </back>
</text>
```

Note that you can also **XInclude** to keep the texts in separate TEI documents:

```
<text>
  <group>

    <xi:include href="vb/VB-21-SoulI.xml"
        xmlns:xi="http://www.w3.org/2001/XInclude"
        xpointer="VB_text_SoulI">
      <xi:fallback>
        <body>
          <p>
            <emph>FIXME: MISSING XINCLUDE CONTENT</emph>
          </p>
        </body>
      </xi:fallback>
    </xi:include>

    <xi:include href="vb/VB-23-DOTR.xml"
        xmlns:xi="http://www.w3.org/2001/XInclude"
        xpointer="VB_text_DOTR">
      <xi:fallback>
```

```
        <body>
          <p>
            <emph>FIXME: MISSING XINCLUDE CONTENT</emph>
          </p>
        </body>
      </xi:fallback>
    </xi:include>

  </group>
</text>
```

- To provide a safe starting point for the chain of XSLT transformations it is essential that each `<text>` includes an **`xml:id`** attribute holding an unique ID for the text. Also, while not mandatory, it is highly recommended to add an **`n`** attribute holding the text title, so that it can be showed in the appropriate selector in the web edition; if `n` is not available, `xml:id` will be used removing the underscore characters (in other words, results may be acceptable but not always pretty …). Other elements may be used as necessary per specific needs, f.i. `type` to record if the text is in prose or verse, but they aren't relevant for EVT. Example:

  ```
  <text n="Dream of the Rood" type="verse" xml:id="DOTR">
  ```

  Note that adding `subtype="edition_text"` is not necessary anymore, since all `<text>`s are now automatically sorted out, be they single texts or part of a larger `<group>`.

Important: in EVT 1.0 the most usual container where to put the necessary metadata described above was a simple `<div>` holding the actual body of a given text (e.g. `<div n="Dream of the Rood" subtype="edition_text" type="verse" xml:id="DOTR">`). This isn't possible anymore in EVT 1.1, but as you can see for existing documents the solution is simple: just copy the relevant attributes in the corresponding `<text>` element.

- Starting from version 1.1 EVT supports the case when there is some prefatory or introductory matter before the main body of text in the original document, for which digitized images are possibly available to be showed together with the edition text. Since all prefatory matter, both present in the original document and/or added by the editor, is encoded in the **`<front>`**, it is therefore necessary to distinguish which is which:

  - EVT 1.3 now supports the **`<titlePage>`** element, so all prefatory matter already present in the original document should go in a `<titlePage>` and/or a `<div>` both having a **`type="document_front"`** attribute;

- the `<titlePage>` must be the first element in the `<front>`;
- the first `<pb/>` must be **the first node** inside the `<titlePage>` or the `<div>`;
- the `<div type="document_front">` can hold sub-`<div>`s, so that you can structure the document prefatory matter as you prefer;
- before adding the modern prefatory matter, you must "close" the sequence of `<pb/>`s of the original front matter by adding a **`<pb type="end"/>`**;
- after this last `<pb/>` all other `<div>`s are going to be treated as text added by a modern editor, and will be inserted in the "Info" or "Regesto" text frame.

The text-image linking is done exactly in the same way as for the main text as described above, but there is a caveat: if the prefatory text ends exactly in the same page where the main text begins, it is necessary to add two `<pb/>`s:

- one at the end of the `<front>` with an `xml:id` ending with -front;
- one at the beginning of the `<body>` with the same `xml:id` but without the -front suffix.

Example:

```
<front>

  <titlePage type="document_front">
    <pb n="000" xml:id="G_vol1_I"/>
    <lb/>
    <docTitle>
      <titlePart type="main">LE<lb/>THEATRE<lb/>ITALIEN<lb/></titlePart>
      <titlePart type="sub">DE</titlePart>
      <lb/>
      <titlePart type="main">GHERARDI</titlePart>
      <lb/>
      <titlePart type="sub">TOMO I</titlePart>
    </docTitle>
  </titlePage>

  <div type="document_front">
    <pb n="327" xml:id="G_vol1_327"/>

    <head type="main">COLOMBINE <lb/>AVOCAT <lb/>POUR ET CONTRE.</head>
    <head rend="italic" type="sub">COMEDIE EN TROIS ACTES.</head>
    <lb/>

    <!--<performance>-->
    <p rend="italic">Mise au Theâtre par Monsieur D*** et representée pour
    la premiere fois par les Comediens Italiens du Roy dans leur Hotel de
    Bourgogne, le huitiéme jour de Juin 1685.</p>
```

```
      <div type="set">
        <p rend="italic">La Scene est à Paris, tantôt chez le Docteur, tantôt
        dans un Cabaret.</p>
      </div>
    </div>

    <pb type="end"/>

    <div>
      <head type="main">Colombine avocat pour et contre</head>

      <head rend="italic">L'histoire</head>
      <p>Au grand désespoir de son amant Cinthio, Isabelle, fille du Docteur
  est résignée à épouser le marquis Sbrufadelli (Arlequin) à qui son père la
  destine; celui, ayant hérité d'un oncle richissime, veut se dédire de sa
  promesse de mariage à Colombine. Pasquariel, déguisé en Espagnol, raconte à
  Arlequin le désespoir de Colombine, dont selon lui tout Venise parle. Cette
  dernière, également déguisée en Espagnole, vient parler à Arlequin avant de
  se démasquer et de maudire son fiancé volage, qui croit voir vu un fantôme.
  Cinthio de son côté, tente en vain de soudoyer Scaramouche pour communiquer
  avec Isabelle, qui tente de dissuader son père de conclure le mariage.
  Arlequin se fait valoir auprès d'Isabelle en lui parlant de sa familiarité
  avec la cour, et de sa charge de colonel nouvellement acquise. Lorsque
  Colombine travestie vient se présenter à Isabelle pour solliciter un emploi
  de chambrière, Arlequin, séduit, tente de la débaucher et lui promet de
  l'installer chez lui comme maîtresse --- de nouveau, elle se dévoile et
  maudit Arlequin, terrorisé.</p>
      <p>[...]</p>
    </div>

  </front>
```

Important: at the present moment this feature only works for unitary TEI documents, i.e. documents having a single <text> and **not** the <text> <group> <text>s hierarchy. Also note that this only applies to the single <pb/> present both in the <front> and in the <body> of a text, all other <pb/>s in the <front> do **not** need a -front suffix.

- **`<pb/>`** elements are used to mark up folio sides: they must include the `n` and `xml:id` attributes, respectively to show the correct folio number and to enable text-image linking at the page level (see above *1.1 Single Side Images at standard resolution*); more in detail, the **`n`** is the label you will see in the page selector in the web interface, so you are relatively free to choose the characters that you put in it, while the **`xml:id`** is a unique identifier used to recover the text and image of each page (you can't have two identical values for `xml:id` in your file, while you can have two or more identical `n` attributes). For the Bookreader view to work properly, the `n` attribute values must end with "r" (for "recto") o "v" (for "verso").

```
<pb n="104v" xml:id="VB_fol_104v"/>
```

Important: if a document starts on a new page/folio, then the corresponding `<pb/>` must be the first node inside the `<body>` or the first node of the first `<div>` inside the `<body>`. In other words, always start a text with the relevant `<pb/>`.

- **`<lb/>`** elements are used to mark up manuscript lines: they must include the `n` and `xml:id` attributes, respectively for line numbering and text-image linking at the line level; `xml:id` only serves the purpose of text-image linking so you can omit it if you're not going to take advantage of this feature.

  ```
  <lb facs="#VB_line_104v_07" n="7" xml:id="VB_lb_104v_07"/>
  ```

  `<lb/>`s with no attributes (or an `<lb rend="empty"/>` element) will be rendered as empty lines. Note that you can have numbered empty lines, but you will have to add not only the `n` but also the `rend="empty"` attribute:

  ```
  <lb n="1" rend="empty"/>
  <lb n="2" rend="empty"/>
  <lb n="3" rend="empty"/>
  ```

- **`<head>`** elements are used for document headings and titles, starting from version 1.3 EVT supports several combinable styles by means of the `rend` and `type` attributes, these are the values currently processed and the corresponding visualization:

  `<head rend="">`

● **right**	text alignment: right
● **left**	text alignment: left
● **center**	text alignment: center
● **uppercase**	text style: upper case
● **bold**	text style: bold
● **italic**	text style: italic

`<head type="">`

● **main**	text size: 140%
● **sub**	text size: 120%

The two attributes and their values are combinable, so you can have for instance

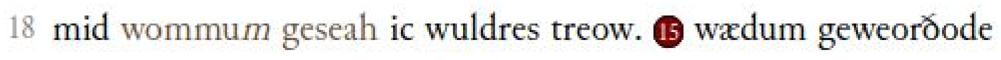
```
<head rend="italic bold right" type="main">Main title</head>
```

which will produce a bold italic text of size 140% aligned on the right side of the text frame.

- Starting from version 1.3, EVT supports the case when a word is interrupted at the end of a line: using **`<w>`** to mark the whole word and an `<lb/>` to mark where the new line begins, it will appear as written by the scribe in the diplomatic level, but it will be automatically recomposed in the interpretative level. F.i. this markup

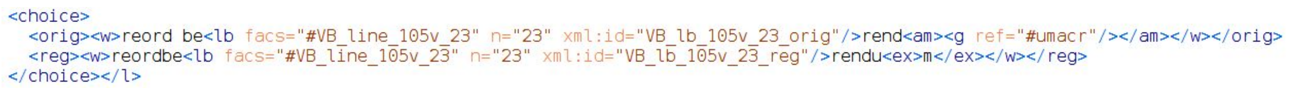
  ```
  <w>geweor<lb facs="#VB_line_104v_19" n="19" xml:id="VB_lb_104v_19"/>ðode</w>
  ```

  will be rendered as follows in the interpretative level:

  18 mid wommum geseah ic wuldres treow. 15 wædum geweorðode

  Note that, just like `<lb/>`s, the `<w>` element will have to be used twice within <choice> sub-elements:

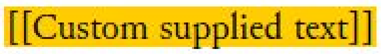
  ```
  <choice>
    <orig><w>reord be<lb facs="#VB_line_105v_23" n="23" xml:id="VB_lb_105v_23_orig"/>rend<am><g ref="#umacr"/></am></w></orig>
    <reg><w>reordbe<lb facs="#VB_line_105v_23" n="23" xml:id="VB_lb_105v_23_reg"/>rendu<ex>m</ex></w></reg>
  </choice></l>
  ```

- Starting from version 1.2 EVT supports use of the **`<supplied>`** element according to two different `reason` values: `<supplied reason="omitted">` (= text which has been omitted by the scribe, usually one or more characters, will be inserted in angle brackets < >) and `<supplied reason="illegible">` (= text which is currently illegible because of damages to the manuscript or textual transmission problems, will be inserted in square brackets [ ]). If no `reason` is specified, the default is the second value ("illegible") and text supplied by the editor will be inserted into square brackets.
  It is possible to customize the appearance of `<supplied>` elements for other `reason` values: you just have to insert your own CSS rules in the `config/evt_builder-custom-styles.css` file (see below) similar to the following:

  ```
  /* layout for the element <supplied reason="custom-reason">  */
  span.interp-supplied[data-reason='custom-reason'] {
     background: #0000;
  }
  /* content before the element <supplied reason="custom-reason">  */
  span.interp-supplied[data-reason='custom-reason']::before {
     content: "[[";
  ```

```
}
/* content after the element <supplied reason="custom-reason">  */
span.interp-supplied[data-reason='custom-reason']::after {
   content: "]]";
}
```

Following these rules, all elements such as, for instance, `<supplied reason="custom-reason">Custom supplied text</supplied>` will be formatted in this way:



[[Custom supplied text]]

- **`<ref>`** elements with a `target` including the “www” or “http” strings will be transformed into HTML hyperlinks.

2.3 Embedded Transcription

The Embedded Transcription method, for which EVT provides experimental support thanks to an EADH small grant, is an alternative way to attach a transcription to an image, be it on parchment or other types of support. This is what EVT expects to find in a TEI XML document created according to this method:

- the TEI document must be based on one or more `<sourceDoc>` elements, with no `<facsimile>` or `<text>`;
- cartesian coordinates for text-image link at line level are once again placed in the relevant attributes of `<zone>` elements (see the preceding section);
- folio sides marked with the `<surface>` element that are direct children of `<sourceDoc>` or `<surfaceGrp>` must include the `xml:id` attribute. If you want to activate the Bookreader view, the `xml:id` attribute values end with "r" (for "recto") or "v" (for "verso");
- transcription text goes into `<line>` elements within or at the same hierarchy level of `<zone>` elements: we recommend to put them inside the corresponding `<zone>`s, alternatively they may be linked to `<zone>`s by means of their `xml-id`; empty `<line>`s will be ignored.

Full example:

```
<sourceDoc xml:id="DOTR_ET">
  <surface xml:id="VB_fol_104v" n="104v">
    <graphic height="1800px" url="immagini\Vercelli-Book_104V_S_300dpi.jpg"
          width="1200px"/>
    <zone lrx="1108" lry="560" rend="visible" rendition="Line" ulx="210"
          uly="459">
      <line n="7" xml:id="VB_txtline_104v_07"><hi rend="init3.1">
```

```
          <g ref="#Hunc"/></hi><hi rend="cap">W</hi>æt ic <g
          ref="#slong"/>wefna c<g ref="#ydot"/><g ref="#sins"/>t secgan
          wylle
        <choice>
          <sic>hæt</sic>
          <corr resp="Grein">hwæt</corr>
        </choice>
        <choice>
          <orig>mege mætte</orig>
          <reg>me gemætte</reg>
        </choice></line>
      <line n="8" xml:id="VB_txtline_104v_08">to midre nihte syðþan <choice>
        <orig>reord b<g ref="#eenl"/>r<g ref="#eenl"/>nd</orig>
        <reg>reordberend</reg>
      </choice> reste wunedon<orig><pc type="metrical">.</pc></orig></line>
    </zone>
    <zone corresp="#VB_txtline_104v_09" lrx="1031" lry="610" rend="visible"
          rendition="Line" ulx="257" uly="558" xml:id="VB_msline_104v_09"/>
    <line facs="#VB_msline_104v_09" n="9" xml:id="VB_txtline_104v_09">þuhte me
          þæt ic <choice>
          <orig>ge <g ref="#sins"/>awe</orig>
          <reg>gesawe</reg>
        </choice>
        <g ref="#sins"/>yllicre treow<choice>
          <orig>onlyft</orig>
          <reg>on lyft</reg>
        </choice></line>

    […]

    <zone corresp="#VB_txtline_105r_32" lrx="987" lry="1476" rend="visible"
          rendition="Line" ulx="121" uly="1422" xml:id="VB_msline_105r_32"/>
    <line facs="#VB_msline_105r_32" n="32"
          xml:id="VB_txtline_105r_32"><choice>
          <orig>ge namon</orig>
          <reg>genamon</reg>
        </choice>
        <damage type="faded">hie</damage> þær ælmihtigne <choice>
            <orig>god</orig>
            <reg>God</reg>
          </choice> <g ref="#aacute"/>hofon hine of ðam</line>
  </surface>

</sourceDoc>
```

2.4 Support for named entities

Starting from version 0.2.0 EVT supports named entities. To best take advantage of this feature, it is necessary to prepare lists that will be included in an appropriate location:

- the `<sourceDesc>` element within the `<teiHeader>`;
- the `<back>` section of a `<text>`.

Note that according to the TEI Guidelines these lists can be added in many other parts of a document, EVT supports these two locations; the `<back>` is probably more precise on the semantic level, but there is no difference when it comes to the generated web-edition.

One likely candidate is a list of persons, as in this example (selected from the Codice Pelavicino digital edition):

```
<listPerson>
  <person xml:id="AccursetusVitalis">
    <persName>
      <forename>Accursetus</forename>
      <surname>quondam Vitalis de castro Sarzane</surname>
    </persName>
    <sex>M</sex>
  </person>
  <person xml:id="AcoltusBonavite">
    <persName>
      <forename>Acoltus</forename>
      <surname>quondam Bonavite de Trebiano</surname>
    </persName>
    <sex>M</sex>
  </person>
  <person xml:id="Adiutus">
    <persName>
      <forename>Adiutus/Aiutus</forename>
    </persName>
    <sex>M</sex>
  </person>
  <person xml:id="Adiutuscap">
    <persName>
      <forename>Adiutus / Aiutus</forename>
    </persName>
    <sex>M</sex>
    <occupation n="2">presbiter, capellanus domini Guilielmi</occupation>
  </person>
  <person xml:id="Adiutusnot">
    <persName>
      <forename>Adiutus / Aiutus</forename>
    </persName>
    <sex>M</sex>
    <occupation n="1">sacri palatii notarius</occupation>
  </person>
  <person xml:id="Adornellusnot">
    <persName>
      <forename>Adornellus</forename>
      <surname>domini Tranchedi comitis de Advocatis de Luca</surname>
    </persName>
```

```
      <sex>M</sex>
      <occupation n="2">iudex ordinarius, notarius</occupation>
    </person>
  [...]
  </listPerson>
```

Note that every `<person>` item relates to an individual, in this case historical figures, but they might as well be fictional characters. Another useful list may one of places (again from the CPD edition):

```
  <listPlace>
    <place xml:id="Aciliano">
      <settlement type="località">Aciliano</settlement>
    </place>
    <place xml:id="Acola">
      <settlement type="località">Acola</settlement>
    </place>
    <place xml:id="Agina">
      <settlement type="località">Agina</settlement>
    </place>
    <place xml:id="Alione">
      <settlement type="località">Alione</settlement>
    </place>
    <place xml:id="Alpes">
      <settlement type="monte">Alpes</settlement>
      <placeName type="new">Alpi Apuane</placeName>
    </place>
  [...]
  <listPlace>
```

At the present moment, EVT supports the following list elements for the purpose of named entity annotation and linking:

- `<listPerson>`
- `<listPlace>`
- `<listOrg>`

Note that the TEI schemas offer other specific list elements, such as `<listNym>` for the canonical form of names, or `<listEvent>` for historical events, but it is also possible to use the generic `<list>` element to provide for other needs (you only have to specify a value for `type` to make explicit what kind of items are gathered in a list), provided that all the lists are related to individual "objects". Future versions of EVT may support other types of list, including the generic one, but at the moment only those specified above are supported.

Once the lists are ready, you can link each occurrence of an item to the corresponding list entry using the `ref` attribute for elements such as `<persName>` (→ `<listPerson>`) and

<placeName> (→ <listPlace>). As a value for each ref you will have to use the xml:id of the corresponding item in a list:

```
<p xml:id="CCLXXXXII_p_003" n="3"><ptr target="CCLXXXXII_st_001"
facs="#st_279r_001"/> Ego <persName ref="#Adiutusnot">Adiutus,
<roleName>sacri palacii notarius</roleName></persName>, hiis interfui et
de mandato suprascripti domini episcopi hanc cartam abreviavi et in publicam
formam redegi, signum et nomen proprium apponendo.</p>
```

2.5 Notes

2.5.1 Stand-off notes

Starting from version 1.2, it is possible to add notes not in the usual inline fashion, but in a separate part of the TEI document, typically grouping them in the <back> element; this is particularly useful when the same <note> may refer to more than one text segment in the document, to avoid redundancy.

To take advantage of this feature, you will have to add an **xml:id** to each <note> element, and insert a <ptr/> with a **type="noteAnchor"** attribute and a **target** attribute whose value will be the corresponding <note>'s ID (as usual preceded by the "**#**" character).

Example:

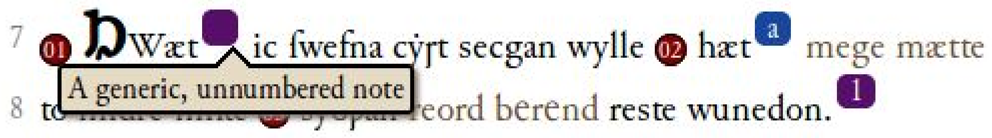
```
<lb/>Lucchese de <placeName>sancto<lb/>Luca</placeName></persName>.
<ptr target="#SM_note_1" type="noteAnchor"/>

...

<note n="1" xml:id="SM_note_1">Per la chiesa di San Luca, nell'area
extramuraria orientale, esterna alla <foreign xml:lang="lat">civitas</foreign>
altomedievale, si veda <ref target="#Garzella1991">Garzella 1991, pp. 179,
188</ref> e <emph>passim</emph>.</note>
```

2.5.2 Critical notes

Starting from version 1.1 EVT lets the editor distinguish between critical notes and commentary notes: the former are encoded as <note n="a" type="**critical**">, the latter as <note n="1" type="**comment**">; if the type attribute is missing, the note will be considered a comment. The n attribute is required to have a distinction after the transformation is done: if you use n with alphabetical values for critical notes and n with numerical values for comment notes (or vice versa, if you prefer) you will see note exponents on dark red and blue background, respectively; feature introduced in EVT 1.3. If you omit the n attribute you will go back to the old, smallish dot. For example:

```
<l n="1"><lb facs="#VB_line_104v_07" n="7" xml:id="VB_lb_104v_07"/><hi
    rend="init3.1">
  <g ref="#Hunc"/></hi><hi rend="cap">W</hi>æt<note>A generic, unnumbered
    note</note> ic
  <g ref="#slong"/>wefna c<g ref="#ydot"/><g ref="#sins"/>t secgan
    wylle</l>
<l n="2"><choice>
    <sic>hæt</sic>
    <corr resp="Grein">hwæt</corr>
  </choice><note type="critical" n="a">A critical note</note>
  <choice>
    <orig>mege mætte</orig>
    <reg>me gemætte</reg>
  </choice><lb facs="#VB_line_104v_08" n="8" xml:id="VB_lb_104v_08"/>to midre
    nihte</l>
<l n="3">syðþan <seg type="kenning"><choice>
    <orig>reord b<g ref="#eenl"/>r<g ref="#eenl"/>nd</orig>
    <reg>reordberend</reg>
  </choice></seg> reste wunedon<orig><pc type="metrical">.</pc></orig><note
    type="comment" n="1">A comment note</note></l>
```

will be transformed into



7 01 ƿWæt ic ſwefna cẏſt secgan wylle 02 hæt a mege mætte

A generic, unnumbered note

8 to ... reord berēnd reste wunedon. 1

and the note content will be shown as a pop-up window clicking on the exponent.

2.6 Tables and columns

Starting from version 1.2, EVT supports the use of the **`<table>`** element, possibly with a layout of two or more columns. A table doesn't require special precautions compared to the normal TEI usage. The use of **`<cb/>`** to mark columns, on the other hand, needs more attention: you will have to add both an `xml:id` and a `rend` attribute to each `<cb/>` marking the beginning of a column; the latter must have a value corresponding to the actual column layout in the document, e.g. "**2col_layout**" for two columns, "**3col_layout**" for three, etc. Also it is necessary to add a final **`<cb type="end"/>`** at the end of the column sequence. This feature is currently being worked upon, see the "Known bugs" section for a caveat.

Full example:

```
<p xml:id="CCCXLVI_p_002" n="2">Sunt ulterius episcopi:
  <pb n="305v" xml:id="fol_305v"/>
  <cb n="1" rend="2col_layout" xml:id="CCCXLVI_cb_003"/>
  <table rows="1" cols="2">
    <head>§ In <placeName ref="#Graçano">Graçano</placeName></head>
    <row role="data">
    <cell><persName ref="#Ventura">Ventura</persName></cell>
```

```
      <cell><measure type="Imperiales" quantity="1" unit="denari">denarium
      I<note xml:id="CCCXLVI_n_006" n="f" place="foot"><term
      xml:lang="latin">Ventura denarium I</term>: <emph>nel testo</emph>
      <term xml:lang="latin">denarium I Ventura</term>.</note></measure></cell>
      </row>
  …
    <cb n="2" rend="2col_layout" xml:id="CCCXLVI_cb_004"/>
      <row role="data">
      <cell><persName ref="#PerasonusAventia">Perasonus de <placeName
      ref="#Avenza">Aventia</placeName></persName></cell>
      <cell><measure type="Imperiales" quantity="4" unit="denari">denarii
      IIII</measure></cell>
      </row>
      ...
      <row role="data">
      <cell>Filie <persName ref="#Lanfranchinus">Lanfranchini</persName></cell>
      <cell><measure type="Imperiales" quantity="1" unit="denari">denarium
      I</measure></cell>
      </row>
    </table>
    <cb type="end"/>
  </p>
```

2.7 Edition levels

Different **edition levels** in the same TEI document are managed through a combination of transcriptional and editorial elements:

- the **diplomatic** level is encoded using the `<damage>`, `<hi>`, `<abbr>`+`<am>` and `<orig>` elements; at the character level, if a `<charDecl>` is present, using the `<mapping type="diplomatic">` character values inside each <char> (or <glyph>) element;
- the **interpretative** level is encoded using the `<supplied>`, `<expan>`+`<ex>` and `<reg>` elements; at the character level, if a `<charDecl>` is present, using the `<mapping type="normalized">` character values inside each `<char>` (or `<glyph>`) element; `<sic>`/`<corr>` pairs are possible, but not necessary if a full critical edition is envisaged.

Note that all the editorial elements are usually inserted in <choice> elements: this is mandatory for word-level pairs.

Full example:

```
 <text>
  <body>
    <div n="DOTR" subtype="edition_text" type="verse" xml:id="DOTR">
     <pb n="104v" xml:id="VB_fol_104v"/>
      <l n="1"><lb facs="#VB_line_104v_07" n="7" xml:id="VB_lb_104v_07"/>
            <hi rend="init3.1"><g ref="#Hunc"/></hi>
            <hi rend="cap">W</hi>æt ic
            <g ref="#slong"/>wefna c<g ref="#ydot"/>
            <g ref="#sins"/>t secgan wylle</l>
      <l n="2"><choice>
            <sic>hæt</sic>
                  <corr resp="Grein">hwæt</corr>
               </choice>
                <choice>
                  <orig>mege mætte</orig>
                  <reg>me gemætte</reg>
                </choice>
            <lb facs="#VB_line_104v_08" n="8" xml:id="VB_lb_104v_08"/>
            to midre nihte</l>
             <l n="3">syðþan <choice>
                  <orig>reord b<g ref="#eenl"/>r<g ref="#eenl"/>nd</orig>
                  <reg>reordberend</reg>
                </choice> reste wunedon<orig><pc type="metrical">.</pc></orig></l>
              <l n="4"><lb facs="#VB_line_104v_09" n="9"
      xml:id="VB_lb_104v_09"/>þuhte me þæt ic <choice>
                  <orig>ge <g ref="#sins"/>awe</orig>
                  <reg>gesawe</reg>
                </choice>
                <g ref="#sins"/>yllicre treow</l>

      […]

          <l n="156">ælmihtig <name type="religion"><choice>
              <orig>god</orig>
              <reg>God</reg>
            </choice></name> þær hi<g ref="#sins"/> eðel wæ<g ref="#sins"/>
          <g ref="#colmidcomposit"/></l>
      </div>
    </body>
  </text>
```

2.8 Translations

In EVT 1.3 you can also add one or more translations to your diplomatic / interpretative edition. The translation text(s) must be added to the **<back>** section of your TEI document, in a `<div>` using the following attributes:

● **`n`**	a title for your translation
● **`type="translation"`**	this is **mandatory** to recognize the translation text properly
● **`xml:id`**	the unique ID for the current `<div>`
● **`xml:lang`**	translation language

For instance:

```
<div n="Il sogno della Croce" type="translation" xml:id="DOTR_trans_it"
xml:lang="ita">
```

It is possible to have translations in more than one language, just remember to have each translated text in a `<div type="translation">` in the `<back>` section, and to specify the language used for each translation using **`xml:lang`**, for instance:

```
<div n="Il sogno della Croce" type="translation" xml:id="DOTR_trans_it"
     xml:lang="ita">

      [...]

</div>

<div n="The Dream of the Rood" type="translation"
     xml:id="DOTR_trans_en" xml:lang="eng">

      [...]

</div>
```

If you have several `<text>`s because you based your hierarchy on the `<group>` element just use the `<back>` of each `<text>`, EVT will take care of the rest:

```
<text>
  <group>
    <text xml:id="text_1">
      <front>...</front>
      <body>...</body>
      <back>
        <div type="translation">[TRANSLATION OF DOCUMENT 1]</div>
      </back>
    </text>

    <text xml:id="text_2">
      <front>...</front>
      <body>...</body>
      <back>
        <div type="translation">[TRANSLATION OF DOCUMENT 2]</div>
      </back>
    </text>
  </group>
</text>
```

The translation will be shown as a single HTML document, but you can add a synchronization at the page level splitting the text in `<div>`s with `type="page"` attribute and a `corresp` pointing to the `xml:id` of the corresponding `<pb/>` element in the original text:

```
<body>
  <pb xml:id="page1" n="1" />
  <div><p>Original text in English</p></div>
  <!-- ... -->
</body>

<back>
  <div type="translation" xml:lang="ita">
    <div corresp="#page1">
      <p>Traduzione italiana del testo originale in inglese</p>
    </div>
  </div>
  <div type="translation" xml:lang="fra">
    <div corresp="#page1">
      <p>Traduction française du texte original en anglais</p>
    </div>
  </div>
<back>
```

Note that using <div>s for pages will inevitably lead to multiple hierarchies problems, it is suggested that you split the paragraphs as explained in the relevanta chapter of the TEI *Guidelines* (20.3 Fragmentation and Reconstitution of Virtual Elements).

You will also have to enable support for a translation in the configuration file (see below). After the XSLT transformation your translation(s) will be accessible from the edition level selector. Textual search within the translated text works, but only if it is inside `<p>` or `<l>` elements, which must include a `n` or `xml:id` attribute each.

2.9 Stage directions

In dramatic texts, stage directions are useful to reconstruct all those non-verbal actions that the characters perform on stage. Using the `type` attribute for each **`<stage>`** element in the encoded text, EVT can be configured to make these values appear in the entity selector as a way to highlight the various scene indications in the text. The following table summarizes the currently managed values that will be displayed in the entity selector:

• `<stage type="`**`setting`**`">`	describes the setting of a scene
• `<stage type="`**`entrance`**`">`	describes an entrance
• `<stage type="`**`exit`**`">`	describes an exit
• `<stage type="`**`business`**`">`	describes a generic activity on stage
• `<stage type="`**`novelistic`**`">`	is a narrative, motivating stage direction
• `<stage type="`**`delivery`**`">`	describes how a character speaks

● `<stage type="modifier">`	gives some detail about a character.
● `<stage type="location">`	describes a location.
● `<stage type="mixed">`	more than one of the above

See *Customizable filters* in section 4. Configuring EVT for more information about the entity selector.

2.10 Other languages

Just like stage directions, all elements that make use of the `xml:lang` attribute can be highlighted in the text by means of the entity selector. To activate this feature you have to set the **`lang_tooltip`** variable to **`true()`** in the configuration file. It is also required that you add the corresponding values to the language files in `config/langpack` so that the right tags will be displayed in the entity selector, to do that you will have to add the usual string couplet in the form

```
"LANGUAGE_NAME": "Language_tag"
```

So if you have text fragments (quotations and the like) in Latin in your text, you can mark those as `<quote` **`xml:lang="latin"`**`>` or `<seg` **`xml:lang="latin"`**`>`, and after you add

```
"LATIN": "Latin"
```

in `config/langpack/en.js`. Those text fragments will be highlighted in the text choosing **Latin** from the entity selector.

2.11 Chronological index

Starting from version 1.3 we added support for a chronological index for editions with multiple `<text>`s. Information about the date must be encoded in the `<front>` in a **`<date>`** element inside a **`<docDate>`**, the date value in standard format must be inserted within a **`when`** attribute in `<date>`:

```
<docDate xml:id="CI_docDate">
  <date when="1212-04-02">1212 aprile 2</date>-<date
   when="1212-04-03">3</date>,
  <placeName>Sarzanello <emph>(<term xml:lang="latin">in palatio castri
   de Sarzana</term>)</emph>.</placeName>
</docDate>
```

Don't forget to enable the corresponding option in the configuration file.

3. VisColl: describing and visualizing the manuscript quire structure

EVT 1.3 introduces a new and very interesting feature: support for VisColl, a software tool aiming at building models of the physical collation of manuscripts, and then visualizing them in various ways. While not based on the TEI schemas and guidelines, VisColl is based on XML documents to describe the manuscript quire structure and on XSLT style sheets to create dynamic representations of such structure using SVG pictures combined with manuscript folios miniatures. Since VisColl is open source software, the EVT development team managed to adapt the XSLT style sheets so that they could be integrated in EVT build chain.

3.1 The VisColl data model

To show the quire structure of a manuscript using VisColl and EVT you have to prepare two different files:

- an XML file describing such structure according to the VisColl data model 1 (Collation Model);
- an XML file holding a list of the manuscript images (Image List).

Both quire description and image list can be prepared by hand, VisColl makes use of a custom XML schema which is quite simple, see the examples below. As an alternative, you can follow the instructions available on its home page, see steps 1 (Create a Collation Model) and 2 (Create an Image List).

This is an example of a manuscript quire description (Collation Model) based on the Vercelli Book codex:

```xml
<?xml version="1.0"?>
<manuscript>
  <url>http://vbd.humnet.unipi.it/</url>
  <title>Vercelli Book</title>
  <shelfmark>CXVII</shelfmark>
  <quire n="14">
    <leaf n="1" mode="original" single="false" folio_number="99"
        conjoin="8" position="1" opposite="8"/>
    <leaf n="2" mode="original" single="false" folio_number="100"
        conjoin="7" position="2" opposite="7"/>
    <leaf n="3" mode="missing" single="false" folio_number="x"
        conjoin="6" position="3" opposite="6"/>
    <leaf n="4" mode="original" single="false" folio_number="101"
        conjoin="5" position="4" opposite="5"/>
    <leaf n="5" mode="original" single="false" folio_number="102"
        conjoin="4" position="5" opposite="4"/>
    <leaf n="6" mode="original" single="false" folio_number="103"
        conjoin="3" position="6" opposite="3"/>
    <leaf n="7" mode="missing" single="false" folio_number="x"
```

```
        conjoin="2" position="7" opposite="2"/>
    <leaf n="8" mode="original" single="false" folio_number="104"
        conjoin="1" position="8" opposite="1"/>
  </quire>
</manuscript>
```

While this is the corresponding image list:

```
<?xml version="1.0"?>
<imageList>
  <imageRV>
    <image val="99r"
    url="data/input_data/images/single/VB_fol_099r.jpg">99r</image>
    <image val="99v"
    url="data/input_data/images/single/VB_fol_099v.jpg">99v</image>
  </imageRV>
  <imageRV>
    <image val="100r"
  url="data/input_data/images/single/VB_fol_100r.jpg">100r</image>
    <image val="100v"
  url="data/input_data/images/single/VB_fol_100v.jpg">100v</image>
  </imageRV>
  <imageRV>
    <image val="x"
      url="data/input_data/images/single/x.jpg">x</image>
    <image val="x"
      url="data/input_data/images/single/x.jpg">x</image>
  </imageRV>
  <imageRV>
    <image val="101r"
   url="data/input_data/images/single/VB_fol_101r.jpg">101r</image>
    <image val="101v"
  url="data/input_data/images/single/VB_fol_101v.jpg">101v</image>
  </imageRV>
  <imageRV>
    <image val="102r"
  url="data/input_data/images/single/VB_fol_102r.jpg">102r</image>
    <image val="102v"
  url="data/input_data/images/single/VB_fol_102v.jpg">102v</image>
  </imageRV>
  <imageRV>
    <image val="103r"
  url="data/input_data/images/single/VB_fol_103r.jpg">103r</image>
    <image val="103v"
  url="data/input_data/images/single/VB_fol_103v.jpg">103v</image>
  </imageRV>
  <imageRV>
    <image val="x"
      url="data/input_data/images/single/x.jpg">x</image>
    <image val="x"
      url="data/input_data/images/single/x.jpg">x</image>
  </imageRV>
  <imageRV>
    <image val="104r"
  url="data/input_data/images/single/VB_fol_104r.jpg">104r</image>
    <image val="104v"
  url="data/input_data/images/single/VB_fol_104v.jpg">104v</image>
  </imageRV>
</imageList>
```

The quire structure description (e.g. `VB-quireStructure.xml`) and the image list (e.g. `VB-imageList.xml`) should be copied into a viscoll directory within `data/input_data/text` and the complete path has to be specified in the configuration file (see below) setting the following parameters:

1. **viscoll_button**: must be set to `"true()"`
2. **viscoll_scheme_path**: full path to the file with the VisColl collation model (see above) starting from the `data/input_data/` folder, e.g. `"text/viscoll/CP_quireStructure.xml"`;
3. **viscoll_image_list_path:** full path to the image list file (see above), again having as a starting point the `data/input_data/ folder`, e.g. `"text/viscoll/CP_20-21-imageList.xml"`.

After EVT has been configured and the relevant XML files are in the appropriate place, the execution of the VisColl stylesheet has been integrated in the general XSLT 2 transformation chain, which means that you will get the SVG diagrams automatically and the VisColl button in the navigation bar when the Web edition is generated.

Besides showing the physical structure of a manuscript binding, VisColl images can be used as an alternative navigation tool: if you click on a thumbnail, you will be carried to that manuscript folio.

4. Configuring EVT

4.1 The Configuration File

If you open the `evt_builder-conf.xsl` file that is inside the `config` folder, you will be able to configure the existing parameters to customize the layout and functionalities of the resulting web-edition

---

An **experimental web-app** to configure the same options is available at this URL: http://evt.labcd.unipi.it/evt1-config/; it is recommended, however, that you read the description of each option which is included in this document, also note that EVT customization can go further than modifying the configuration file (see section 3.2).

The parameters that you will be able to configure are the following:

Image extension

```
<xsl:param name="imageExt"/>jpg</xsl:param>
```

Web site of the project

```
<xsl:param name="webSite" select="'http://www.yoursite.com/'"/>
```

Edition/Project title and Badge

```
<!-- EN: Index title -->
<xsl:param name="index_title" select="'Codex Viewer'"/>
<xsl:param name="badge_text" select="'DIGITAL'"/>

<!-- EN: Hide/Show badge -->
<xsl:param name="badge" select="true()"/>
```

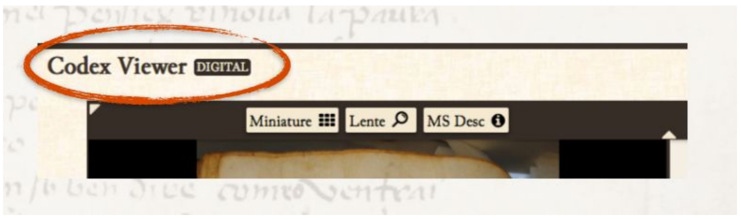
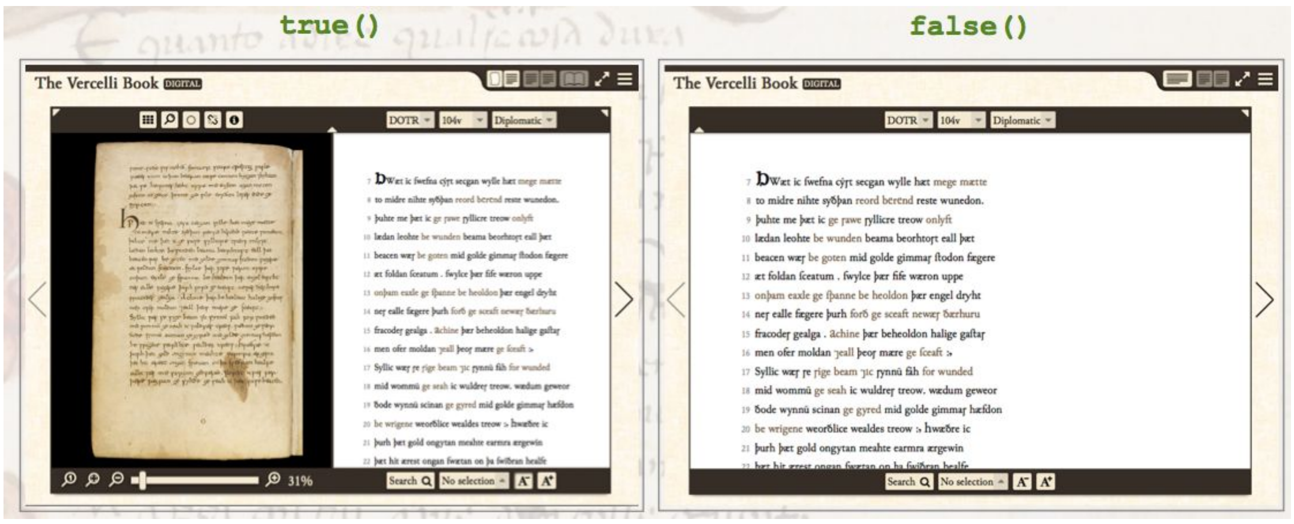


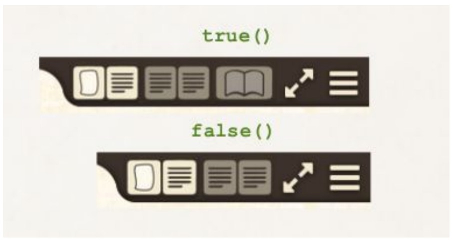
Image frame (on/off)

```
<xsl:param name="image_frame" select="true()"/>
<xsl:param name="image_frame" select="false()"/>
```



Ƿæt ic ſweſna cẏꞃt secgan wylle hæt mege mætte
to midre nihte syðþan reord berēnd reste wunedon.
þuhte me þæt ic ge ꞃawe ꞃyllicre treow onlyſt
lædan leohte be wunden beama beorhtoꞃt eall þæt
beacen wæꞃ be goten mid golde gimmaꞃ ſtodon fægere
æt foldan ſceatum . ſwylce þær ſiſe wæron uppe
onþam eaxle ge ſpanne be heoldon þær engel dryht
neꞃ ealle fægere þurh forð ge sceaft newæꞃ ðærhuru
fracodeꞃ gealga . ac hine þær beheoldon halige gaſtaꞃ
men ofer moldan ⁊eall þeoꞃ mære ge ſceaſt :•
Syllic wæꞃ ꞃe ꞃige beam ⁊ic ꞃynnū fāh for wunded
mid wommū ge seah ic wuldreꞃ treow. wædum geweor
ðode wynnū scinan ge gyred mid golde gimmaꞃ hæſdon
be wrigene weorðlice wealdes treow :• hwæðre ic
þurh þæt gold ongytan meahte earmra ærgewin
þæt hit ærest ongan ſwætan on þa ſwiðran healfe

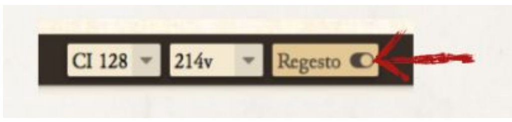
Book Reader view (on/off)



```
<xsl:param name="double_view" select="true()"/>
<xsl:param name="double_view" select="false()"/>
```

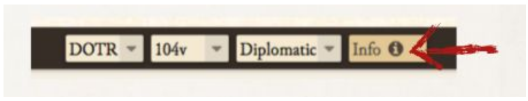
Regesto (on/off)

```
<xsl:param name="regesto" select="true()"/>
```



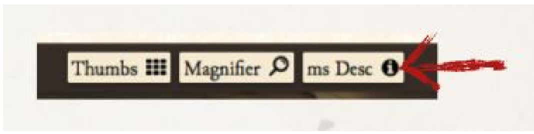
Information about the current document (on/off)

```
<xsl:param name="frontInfo" select="true()"/>
```



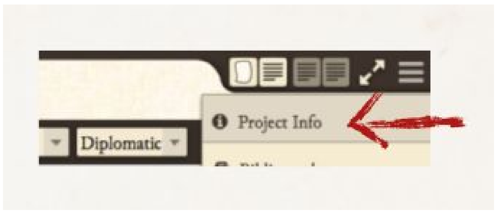
Manuscript description (on/off)

```
<xsl:param name="msDesc" select="true()"/>
```



Information about the project (on/off)

```
<xsl:param name="headerInfo" select="true()"/>
```



Bibliography (on/off)

```
<xsl:param name="bibliography" select="true()"/>
```

Edition levels

```
<xsl:variable name="edition_array" as="element()*">
   <edition>Diplomatic</edition>
   <!-- NB: Leave it empty if you don’t have a diplomatic
        edition: <edition></edition> -->
   <edition>Interpretative</edition>
   <!-- NB: Leave it empty if you don’t have an interpretative
        edition: <edition></edition> -->
</xsl:variable>
```

Translation

If you have added a translation to your text as explained above, remember to enable this option.

```
<xsl:variable name="translation" select="true()">
```

Frame content showed at first loading of the edition

Left frame: possible values are

- *image*: if you want to show the digitized ms images
- *info*: if you want to show manuscript information

```
<xsl:variable name="left_frame_default_content" select="'image'" />
```

Right frame: possible values are

- *text*: if you want to show the edition text
- *info*: if you want to show information about the current text

```
<xsl:variable name="right_frame_default_content" select="'text'" />
```

Image-text linking button (on/off)

```
<xsl:param name="txtimg_link_button" select="true()"/>
```

Hot-spot button (on/off)

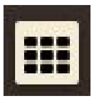
```
<xsl:param name="hs_button" select="true()"/>
```

Magnifying lens button (on/off)

```
<xsl:param name="mag_button" select="true()"/>
```

Thumbnails button (on/off)

```
<xsl:param name="thumbs_button" select="true()"/>
```



Starting from EVT 1.3 the thumbnail navigation button is located in the navigation bar, not anymore in the upper toolbar in the image frame. If you want to enable thumbnails, however, you will also have to make sure that the navigation bar is enabled as well (see below).

Viscoll button (on/off)

```
<xsl:param name="viscoll_button" select="true()"/>
```

This button is a new EVT 1.3 and appears in the navigation bar: see section 3 for a description of this feature and see below about how to activate the navigation bar.

Path to the XML file containing the VisColl Collation Model

If you need to use an online resource, you can put the complete URL (e.g: `http://www.mysite.com/viscollScheme.xml`) here. Otherwise put the file in data/input_data/text folder and just write the relative path starting from that folder.

```
<xsl:param name="viscoll_scheme_path">text/viscoll/VB-structure.xml</xsl:param>
```

Path to the XML file containing the VisColl Image List

If you need to use an online resource, you can put the complete URL (e.g: `http://www.mysite.com/viscollImagelist.xml`) here. Otherwise put the file in data/input_data/text folder and just write the relative path starting from that folder.

```
<xsl:param name="viscoll_image_list_path">text/viscoll/VB-imglist.xml</xsl:param>
```

Edition level selector

```
<xsl:param name="edition_level_selector" select="true()"/>
```

Page selector position (left/right)

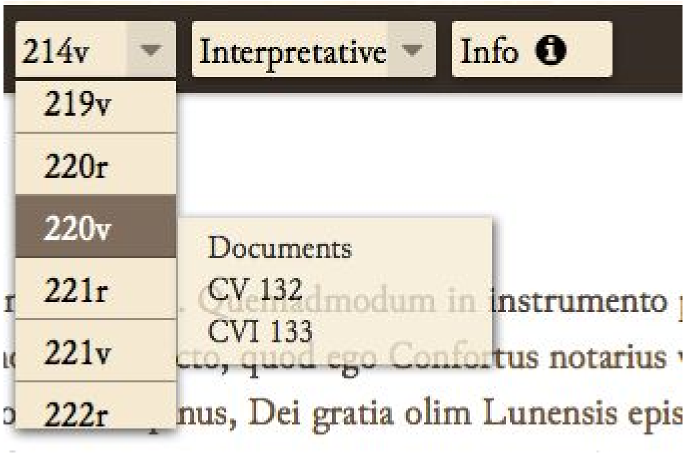
```
<xsl:param name="pp_selector_pos" select="'right'"/>
```

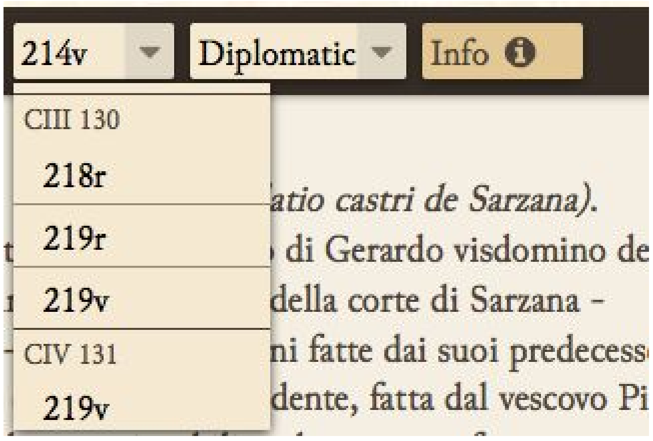
Page grouping based on document (on/off)

```
<xsl:param name="pp_selector_doc_grouping" select="false()"/>
```

Tooltip for page selector

```
<xsl:param name="pp_selector_doc_tooltip" select="true()"/>
```





Search and virtual keyboard (on/off)

```
<xsl:param name="search" select="true()"/>
<xsl:param name="virtual_keyboard_search" select="true()"/>
```

Navigation bar (on/off and initial status)

```
<xsl:param name="bottom_navbar" select="true()"/>
<xsl:param name="bottom_navbar_initial_status" select="'collapsed'"/>
```

Document navigation button (on/off)

```
<xsl:param name="document_navigation" select="true()"/>
```

Prose/Lines visualization button (on/off)

```
<xsl:param name="prose_verses_toggler" select="true()"/>
```

Language tooltip (on/off)

```
<xsl:param name="lang_tooltip" select="true()"/>
```

Person list (on/off)

```
<xsl:param name="list_person" select="true()"/>
```

Place list (on/off)

```
<xsl:param name="list_place" select="true()"/>
```

Organization list (on/off)

```
<xsl:param name="list_org" select="true()"/>
```

Term list (on/off)

```
<xsl:param name="list_term" select="true()"/>
```

Gloss list (on/off)

```
<xsl:param name="list_gloss" select="true()"/>
```

Chronological index for texts (on/off)

```
<xsl:param name="list_doc" select="true()"/>
```

Customizable filters

EVT versions up to 1.2 (included) handle element filters, aiming at highlighting specific XML elements in the text by means of a selector in the lower toolbar (text frame).
Starting from **EVT 1.3 a new, more flexible method** has been implemented: it is now possible to group the desired elements according to a custom typology, and to deactivate / hide a whole typology or a single element without having to delete it. To configure this feature:

- add a `<group>` element with an attribute `active="true"` for each desired element group or category;
- add all the elements that belong to that group inside it as empty elements (f.i. `<persName/>`);
- for each element add an `active` attribute, if you want that element to be included in the selector set `"true"` as value, if you want to hide it set it to `"false"` (f.i. `<persName active="true"/>`);
- a `<group>` will be shown only if it contains at least one `active="true"` element.

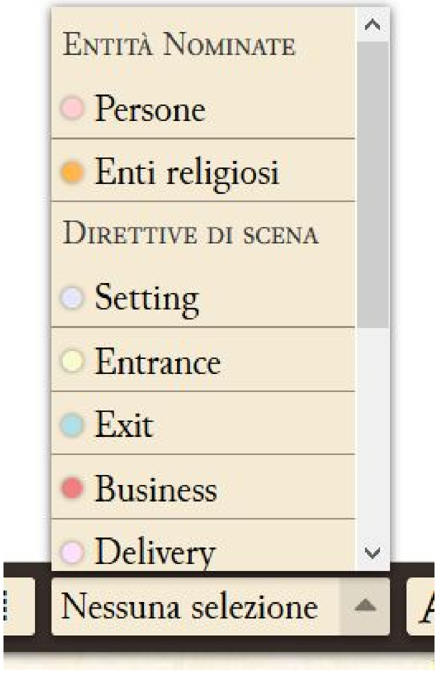
This way it is possible to show / hide whole groups and/or single elements in the selector. Note, however, that a group will be shown only if it contains at least a single active element. Example:

```
<xsl:variable as="element()*" name="lists">
     <group active="true" label="NAMED_ENTITIES">
          <persName active="true"/>
          <placeName active="false"/>
          <orgName active="true"/>
     </group>
     <group active="false" label="INTERESTING_ELEMENTS">
          <roleName active="true"/>
          <measure active="true"/>
          <date active="true"/>
          <foreign active="true"/>
     </group>
     <group active="true" label="STAGES">
          <setting active="true"/>
          <entrance active="true"/>
          <exit active="true"/>
          <business active="true"/>
          <delivery active="true"/>
          <modifier active="true"/>
          <novelistic active="true"/>
          <mixed active="true"/>
     </group>
     <group active="true" label="OTHERS"> </group>
</xsl:variable>
```

The group label (`label` attribute) must contain a string with no spaces, which will become a new entry in the localization files, so that it can be translated in all supported languages. Example:

`it.json`

```
{
"NAMED_ENTITIES": "Entità nominate",
"INTERESTING_ELEMENTS": "Elementi interessanti",
"OTHERS": "Altri elementi"
}
```



You can also add more elements, and a custom `label` for each one, to those already included in the available `evt_builder-config.xsl` file: note that, since these elements are going to be integrated into the current XSLT transformation chain, it will be necessary to configure them in full, just like the standard ones are pre-configured in EVT, and that it is possible that not all new elements will work perfectly.

First of all, add the desired element to your `evt_builder-config.xsl` file assigning it a meaningful `label`, f.i.

```
evt_builder-config.xsl
<group label="OTHERS" active="true">
    <name active="true" label="NAMES"/>
</group>
```

After which you will have to translate it in the localization file(s):

```
it.json
{ "NAMES": "Nomi" }
```

If you forget to add a label, or don't want to, the element name will be used and shown in capital letters, but it will still be necessary to add a translation, f.i.:

`evt_builder-config.xsl`
```
<group label="OTHERS" active="true">
<l active="true"/>
</group>
```

`it.json`
```
{ "L": "Versi" }
```

It is now necessary to define the CSS rules to highlight both the edition text and the label in the text toolbar selector. To do this, edit the **`config/evt_builder-custom-styles.css`** file using the template shown below (just remember to change `"name"` with whichever element name you added in the preceding step):

```
.option[data-value="name"] .filter_color {
    color: #00ccff;
}
.doc.current span.name.list_active,
.doc.current span.name.list_active span.trigger,
.doc.current span.name.over > span.trigger,
.doc.current span.name.opened > span.trigger {
    background: #00ccff;
}
```

It may happen that, in spite of the CSS rules, the desired element doesn't appear in the UI selector: in that case it will be necessary to also add a custom XSLT template to make sure that the element is transformed correctly in order to be shown. This template has to be added to the config/evt_builder-custom-templates.xsl file as follows:

```
<xsl:template match="tei:name" mode="interp dipl #default">
    <span class="name">
            <xsl:apply-templates mode="#current"/>
    </span>
</xsl:template>
```

This way the element will be transformed in an HTML element having as class the TEI element name, which will be used to highlight it and show it in the selector.

Important: since there are XSLT templates for many TEI elements already, and those are transformed in a specific way each, adding a custom template for one of those elements may conflict con the existing rules and .

If you assign a higher priority to your custom template for a TEI element already included in the XSLT transformation chain, you will overwrite the basic template, but this also means that you **will lose** all transformations and functionality applied by EVT in relation to such element.

To avoid losing the basic EVT processing of such element and have it included in the highlighted elements list you can

- search the element template in the current EVT XSLT style sheets and modify it so that the output includes the element name;

Warning: you should follow this path only if you know XSLT programming well, because there is a high risk of breaking the building process.

- add a CSS rule for each edition level, distinguishing each rule adding the edition level prefix to the element class:

  ```
  .doc.current span.interp-name.list_active,
  .doc.current span.interp-name.list_active span.trigger,
  .doc.current span.interp-name.over > span.trigger,
  .doc.current span.interp-name.opened > span.trigger
  ```

  This is necessary because after the transformation our special elements are going to have the edition level, followed by the element name, as their class value.

Advanced customization. It is also possible to rely on specific attributes and their values to identify the elements to be highlighted in the text. To do this, besides all that has been explained above, it is required to add a couple attribute-value for each corresponding TEI XML encoding to be processed. For instance, if I have marked metaphors in a text by means of `<seg type="metaphor">` and I want to highlight those using the named entities / special elements selector I will have to add this to the `evt_builder-config.xsl` file:

```
<seg type="metaphor" active="true" label="METAPHOR"/>
```

Since I may be using more than one `type` attribute for my `<seg>` elements, I want to customize the element tag to be displayed inn the selector using the `label` attribute, as you can see from the example above. Again, the corresponding translation will have to be added to the localization files.

In case I forget to add the label attribute, the element tag will be created automatically on the base of the element name and the value of the `type` attribute, f.i.

```
<seg type="metaphor" active="true"/>
```

will result in a translation string `"SEG_TYPE_METAPHOR"`. Just like for the examples discussed above, it will be necessary to define the CSS rules, adding not only the class selectors, but also the attribute ones:

```
.option[data-value="seg"][data-type="metaphor"] .filter_color {
```

```
    color: #00ff00;
}
.doc.current span.seg[data-type="metaphor"].list_active,
.doc.current              span.seg[data-type="metaphor"].list_active
span.trigger,
.doc.current span.seg[data-type="metaphor"].over > span.trigger,
.doc.current span.seg[data-type="metaphor"].opened > span.trigger {
    background: #00ff00;
}
```

It is again possible that the specific element we want to highlight hasn't been processed as we expected, in the which case you must create a custom XSLT template making sure that the generated HTML has not only the element name as class, but also that it specifies as **`data-*`** *`attribute`* the specific attribute used with the specific value used for the purpose of selection. con il particolare valore che vogliamo usare per la selezione.
To transform all attributes belonging to a node it is also possible to use the `"dataAttributesFromAttributes"` template defined in **`evt_builder-general.xsl`**, just recall it with this instruction:

```
<xsl:call-template name="dataAttributesFromAttributes"/>
```

Final remarks

All available options are described by means of comments in the file itself and should be understandable for anyone having a little experience with XML/XSLT documents. The experimental web-app allows to save and/or upload a configuration file, which of course can also be modified by hand once it has been saved on your computer.

4.2 Fine-tuning the transformation results, custom templates

Bookreader view

The Bookreader navigation needs the identification of page pairs, which is based on the `n` values of the `<pb/>` / `<surface>` elements. If you want EVT Builder to properly recognize the page pairs, you need to use numbers instead of letters for such values, followed by "v" for *verso* and "r" for *recto*. We do realize that this is a bit restrictive, but we are working on a better way to recognize the pairs. It is possible, in any case, to manually modify the `structure.xml` file (in the `output_data` folder) to correct possible mistakes: you'll find a `<pages>` element inside that file, with `<pair>`(s) containing two `<pb>`(s), one for the image in the left-hand-side, one for the image in the right-hand-side. Note that every time you transform the XML file with EVT Builder, the structure file will be overwritten and you will lose your changes.

The keyboard_config.xml file

In the `config` folder you will find a `keyboard_config.xml` file which will let you configure and use, provided that you set the corresponding switch to `true()` in the `evt_builder-conf.xsl` file, a virtual keyboard to insert special characters in the search field. Simple instructions are included in the file itself.

The language files

In the `config/langpack` folder there are a few files with a .js extension that are used to localize the User Interface language, these correspond to the languages you can choose from the Settings menu (first item on the main menu starting from right). You can change the language values to correct mistakes or to add a new language, in the latter case please send us the new .js file!

The `evt_builder-custom-templates.xsl` file

If you just need an XSLT template to handle a specific TEI element not included in the transformation chain, you can add it in this file, `evt_builder-custom-templates.xsl` in the `config` folder. When adding your templates to this file don't forget to add one of the three available modes (interp, dipl) so that the template will be included in the transformation chain to operate in the appropriate mode. If you want your template to be applied in all available modes it is necessary to add all possible values for `mode`; it is also advised to set a very high priority for your template, so that it will be executed avoiding possible conflicts with EVT's standard templates.

Example:

```
<xsl:template match="tei:title" mode="interp dipl" priority="9">
      ...
</xsl:template>
```

Beware of possible conflicts if you try to modify the processing of a node already included in the standard templates.

The `evt_builder-custom-styles.css` file

This configuration file has exactly the same purpose as the .xsl one described above: this is where you can add custom CSS rules for specific elements.

`evt_builder-custom-styles.css` can be found in the `config` folder. You can use it together with the custom XSLT templates file, but again be careful if trying to modify standard rules.

The structure.xml file

For immediate fixes or temporary changes to the file and directory structure of the web edition you can also modify this file (in the `output_data` folder), note however that all changes will be lost when a new version of the edition is generated!

4.3 Testing an EVT-based edition

In the past, we always suggested that you use Firefox to test an EVT-based edition on your computer, but during 2019 things have changed: starting from version 67 Firefox developers adopted the same security-conscious policy chosen by developers of Chrome and other Web browsers, that is forbidding loading local files (= documents available on the user's computer drive) in the browser as a result of the execution of Javascript programs. The goal is to improve global security when browsing the Web, but the unpleasant collateral effect is that of preventing the loading of digital editions based on EVT, or similar software, from local folders. Fortunately there are several workarounds that can be used to test EVT editions that are located on your hard drive:

- option no. 1: download and install **Firefox ESR v. 60**: this version predates the new security policy adopted in FF v. 67 and, furthermore, it can be installed in parallel with any other version of Firefox;
- option no. 2: launch Chrome from the command line with the `--allow-file-access-from-files` parameter, after that you can press CTRL+O to open the `index.html` file, or you can just drag and drop it on Chrome's window;
- option no. 3: install an extension providing a local web server on Firefox or Chrome, f.i. there is this one available for Chrome:

  https://chrome.google.com/webstore/detail/web-server-for-chrome/ofhbbkphhbklhfoeikjpcbhemlocgigb

This problem, however, **only affects local testing**, after the edition has been uploaded on a server there are no problems in accessing it with any of the major browsers.

5. Step by step instructions

How to use the EVT builder pack, step by step:

1. After unpacking the zipped archive on your hard disk, put the images files of every manuscript folio in the `data/input_data/images` folder. The images must be organized as explained above:

- single side images go in the `data/input_data/images/single` folder
- double side images go in the `data/input_data/images/double` folder
- hotspot images go in the `data/input_data/images/hotspot` folder

2. Put the TEI XML file of the transcription and all other related files if necessary (e.g. schema, entities) in the `data/input_data/text` folder. You can use a sub-folder e.g. for the schema, provided that the correct path to the schema is included in the TEI XML documents.

3. Delete, or move into another folder on your hard drive, all content of the `data/output_data/` folder and the `index.html` file. Note that this is just a precaution and/or a way to backup your previous results, the transformation should overwrite the existing folder in any case.

4. Configure the software as explained in section *3. Configuring EVT*.

5. Start the edition creation process by applying the `evt_builder.xsl` stylesheet, which is in the `builder_pack` folder, to the TEI XML transcription document. To perform the XSLT transformation using Oxygen XML Editor:

- Open the XML file with Oxygen;
- Select the "Document/Transformation/Configure a transformation scenario" menu voice;
- Choose the "XML transformation with XSLT" scenario;
- Type or select the path to the `evt_builder.xsl` file;
- Choose one of the 9.x versions of the Saxon XSLT transformer engine;
- Save and apply the transformation scenario.

Known bugs (and workarounds)

No software tool is usually totally bug-free after a certain complexity threshold has been crossed, and surely EVT is no exception under this regard. All the known bugs and quirks will be detailed here, please make sure that you send us an email about any problem you may face, so that if it is a new bug we have a chance to try to reproduce it and then fix it.

Support for `<cb/>` elements is not yet perfect: to make it work properly, you may have to add a final `<cb type="end"/>` at the end of the columnar layout area of the original document (see section 2.6).

Likewise, support for page synchronization is a very quick addition to translation support, and is based on using a `<div>` for each `<pb/>`: as noted above, this is very likely to cause multiple hierarchies conflicts for `<p>` elements of the translation text, which are easily solved splitting the paragraphs in two, however (see 20.3 Fragmentation and Reconstitution of Virtual Elements).

When marking up slanted text lines in a folio by means of `<zone>` elements it is possible to use the `rotate` attribute to specify an inclination angle so that the rectangular highlight in the image frame follows the text more accurately. Unfortunately this attribute won't accept negative values or decimal numbers because that will make the TEI document non valid. A simple workaround is to comment those `<zone>`s that have such values, and remove the comment marks just before doing a transformation.

The open source plugin used for textual search, Tipue Search (v. 3.1), is limited in the number of results that will present to the user for each "container" element (e.g. `<p>` or `<l>`): only the first one will be shown in the result list. Note, however, that the yellow highlighting works as expected, all results are highlighted in the text.

Warnings

Note that, while a web-edition created with EVT is ready to be copied on a web server for publication, you may want to prevent access to part of the directory structure: this is particularly the case of the `data/input_data/images` directory, where normal and high resolution images of the edited manuscript can be found.

In EVT 1.1 or later the starting point for the XSLT transformation chain is now each <text> element in your TEI document: this marks a difference with EVT 1.0, so please read the appropriate section (2.2 Parallel Transcription, third item in the list) especially if you have TEI documents that already work in EVT 1.0 and you want to update them so that v. 1.1 or later can be used.

Important: in Oxygen 19 the default version of the Saxon XSLT transformation engine (v. 9.7) doesn't work with EVT's style sheets. This problem has been solved in later versions, but for v. 19 the Oxygen developers suggestion is to downgrade the Saxon software to v. 9.6, here you will find all the necessary instructions:
https://www.oxygenxml.com/doc/versions/19.0/ug-editor/topics/saxon-issues.html.

References

EVT home page
EVT release page
EVT development repository
Mail contact

Cacioli, Giulia. 2019. *Filologia e Information Retrieval: progettazione e sviluppo di un motore di ricerca per EVT*. MA thesis. Pisa: ETD - Electronic theses and dissertations repository, Università di Pisa. http://etd.adm.unipi.it/theses/available/etd-01152019-191842/.

Codice Pelavicino. Edizione digitale, a cura di E. Salvatori, E. Riccardini, L. Balletto, R. Rosselli del Turco, C. Alzetta, C. Di Pietro, C. Mannari, R. Masotti, A. Miaschi, 2014 http://pelavicino.labcd.unipi.it ISBN 978-88-902289-0-2; DOI 10.13131/978-88-902289-0-2.

Di Pietro, Chiara. 2016. *EVT per le edizioni critiche digitali: progettazione e sviluppo di una nuova GUI basata sullo schema progettuale MVC*. MA thesis. Pisa: ETD - Electronic theses and dissertations repository, Università di Pisa.

Di Pietro, Chiara and Roberto Rosselli Del Turco. 2018. Between Innovation and Conservation: The Narrow Path of User Interface Design for Digital Scholarly Editions. In *Digital Scholarly Editions as Interfaces*, edited by Roman Bleier, Martina Bürgemeister, Helmut W. Klug, Frederike Neuber, and Gerlinde Schneider. *Schriften Des Instituts Für Dokumentologie Und Editorik* 12: 133–63. BoD, Norderstedt. https://kups.ub.uni-koeln.de/9085/.

Leclerc, Élise. 'Le risorse dell'italianista 2.0: Edition Visualization Technology.' Blog post: *Fonte gaia blog* 2015/08/25. URL: http://fontegaia.hypotheses.org/1239.

Martignano, C. 2017. Progettazione e sviluppo di un apparato critico modellato sulla tradizione a stampa in EVT. MA thesis. Pisa: ETD - Electronic theses and dissertations repository, Università di Pisa.

Rosselli Del Turco, Roberto. 'EVT development: an update (and quite a bit of history).' Rapporto tecnico sul blog *Edition Visualization Technology*, 26 gennaio 2014. URL: http://visualizationtechnology.wordpress.com/2014/01/26/evt-development-an-update-and-quite-a-bit-of-history/.

Rosselli Del Turco, Roberto et al. 2015. "Edition Visualization Technology: A Simple Tool to Visualize TEI-Based Digital Editions." *Journal of the Text Encoding Initiative* Issue 8. URL: http://jtei.revues.org/1077; DOI: 10.4000/jtei.1077.

Rosselli Del Turco, Roberto. 2019. "Designing an advanced software tool for Digital Scholarly Editions: The inception and development of EVT (Edition Visualization Technology)." *Textual Cultures* 12.2: 91–111. URL: https://scholarworks.iu.edu/journals/index.php/textual/article/view/27690; DOI: 10.14434/textual.v12i2.27690.

TEI Consortium, eds. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. 3.6.0. Last updated on 16th July 2019. TEI Consortium. http://www.tei-c.org/Guidelines/P5/.

The Digital Vercelli Book. A facsimile edition of Vercelli, Biblioteca Capitolare, CXVII, a cura di R. Rosselli Del Turco; trascrizione e codifica a cura di R. Rosselli Del Turco, R. Cioffi, F. Goria; software EVT creato da C. Di Pietro, J. Kenny, R. Masotti, R. Rosselli Del Turco. *Collane@unito.it,* 2017 https://www.collane.unito.it/oa/items/show/11 ISBN 9788875901073.

VisColl: https://github.com/leoba/viscoll.