

# QXmlEdit: the manual

## Introduction

Why another editor for XML? QXmlEdit purpose is edit a complex XML document without copying and pasting elements with textual editors. Complex documents can be a nightmare to handle as text, even with syntax colored editors. Last, I wanted to test Qt framework as a viable multi platform solution for this kind of utilities.

## Main functionality

Menus are divided in main functionality group; most used commands are kept visible in main dialog as a button on the right.

File: XML document creation and saving

Edit: clipboard handling; find command

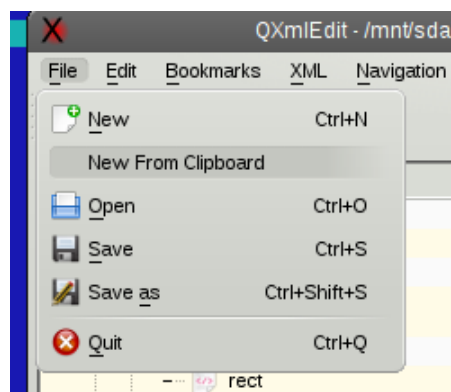
XML: XML tree manipulation (insertion, deletion, etc)

Navigation: XML tree navigation

View: view commands and options

## File

File contains usual actions, with an exception: “Create from clipboard”, that imports an XML document from the clipboard, if possible. This feature is handy for inspecting resources copied from other programs without exporting them in a temporary file.



*Illustrazione 1: File Menu*

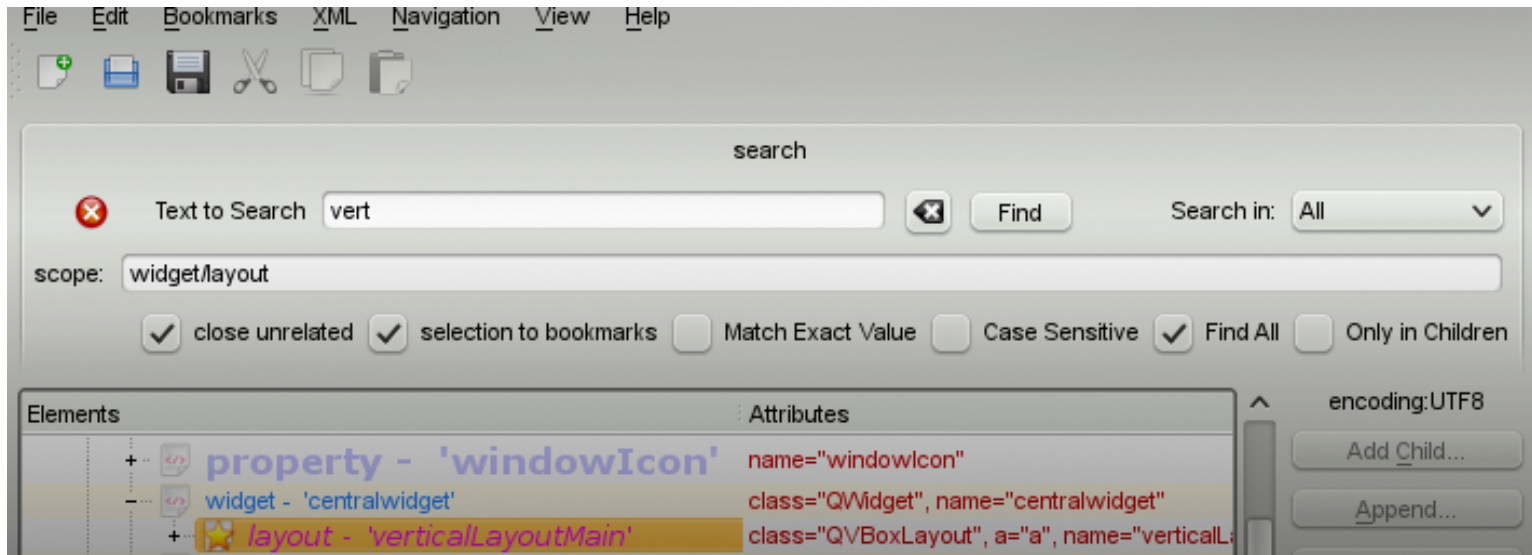
## Edit

The usual clipboard operations act on an internal clipboard regarding all that concerns XML nodes handling (i. e. it is not possible to paste text from an extern program into QXmlEdit), but each time an operation is done, a text value of the selection is copied as text in system clipboard, so XML can be exported as text to other programs.

The clipboard operations act on the whole selected node subtree, including any child.

## Find

Find panel gives you the option to search a string.



Find options explanations:

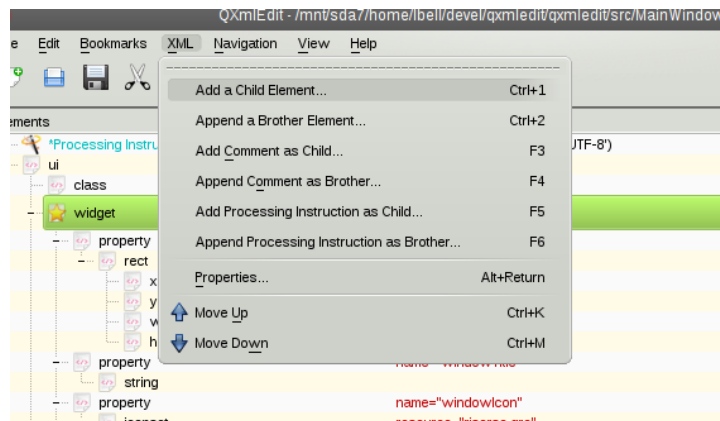
- **“Search in”** defines where the search is performed: in element tags, attribute names, attribute values, element text or in all scopes.
- **“close unrelated”** close all the branches that does not contain any occurrence of the search text.
- **“selection to bookmarks”** adds all occurrences found to bookmarks collection.
- **“match exact value”** executes the search of the exact value of text (opposed to regard the text as a substring)
- **“case sensitive”** does what the name suggest
- **“find all”** mark all the occurrences of the search pattern with a background pattern to ease the reading, elsewhere the search will stop to first match.
- **“only children”** the search can be limited only to children of the selected item or to the whole tree.
- **“scope”** can limit the search to a set of elements or attributes. This field can contain a path in XPath like syntax. For example to search a value in the “id” elements children of “resource”, the field must contains “resource/id”. If the search has to be performed only on “name” attribute of the “window” element, write “[window/@name](#)”. To include any elements between “window” and “widget” tags, simply omit it as in “window//widget”

## XML menu

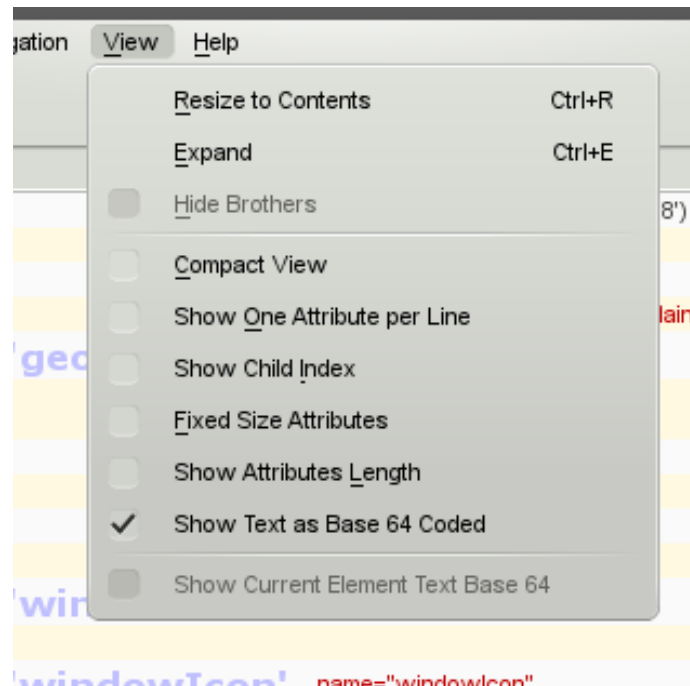
This menu contains commands that perform XML tree manipulation, allowing inserting of elements, processing instructions and comments relative to the actual selection.

Each item has two options: one to create a child item of the selected one and another to create a new item as a brother (same level) of the selected one.

Each item can be moved relative to its brothers via the “Move Up” or “Move Down” commands. Edit menu commands (Cut and Paste) commands are the only way to move an item into another hierarchy.



## View menu



View menu has three commands.

- **Expand:** expands the whole tree at once
- **Resize to Contents:** resize columns to fit the text
- **Hide Brothers:** close all the brothers of the selected element; useful when the element is contained in a large set. If any element is closed, an ellipsis sign “...” appears. To restore normal behavior reselect same option.

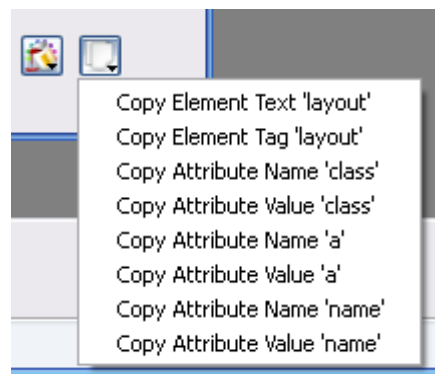
Other items are options relative to appearance:

- **Compact View:** show the element text and comments in a single line, attributes are shown in columns if “Show One Attribute per Line” is selected. If the text element has no attributes, its text is shown in second column.

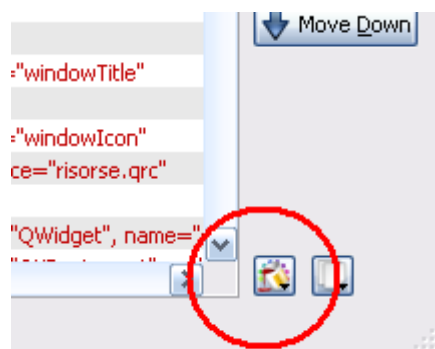
- **Show One Attribute per Line:** displays each attribute in a separate line.
- **Show Child Index:** next to each item icon is shown its ordinal position in the hierarchy.
- **Show Attributes Length:** shows the attribute value length next to the value. Useful to verify data when the attributes value is used to carry data for electronic processing.
- **Fixed Size Attribute:** this item activates the display of the length using a fixed picture and activates the use of a fixed width (non proportional) font in the attribute column.
- **Show Text as Base 64:** show all elements text translated from base 64 in last column
- **Show Current Element Text Base 64:** same of the previous, but only for an element.

## Copy special

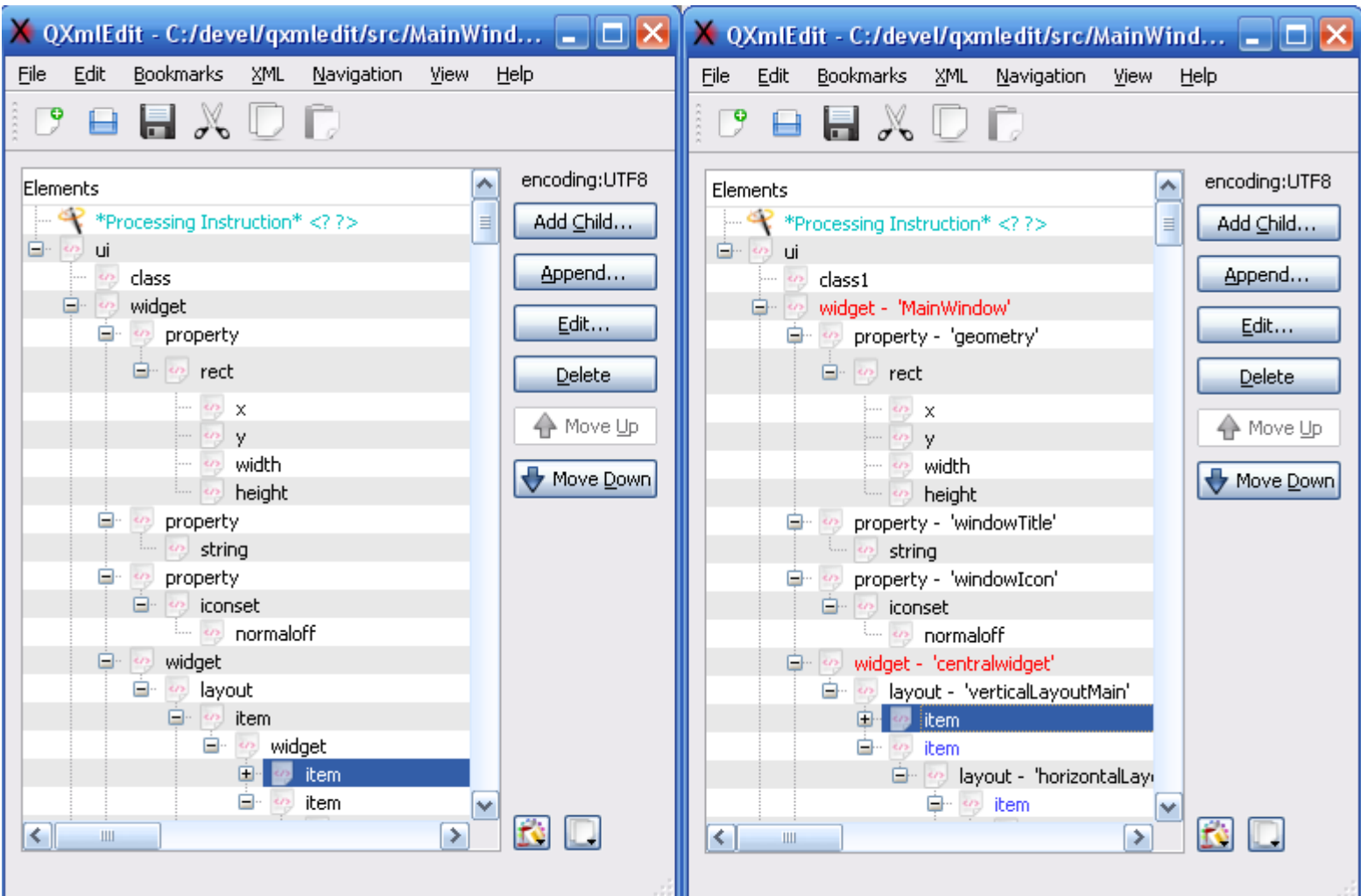
Copy special is a feature, activated via tool button near the bottom right corner of the main window, that fills the clipboard with the content of the current tag name, current element value, or current element name or attributes value.



## Styles



Styles, activated using the button encircled in previous figure, are a mean to highlight relevant informations in XML structure. Styles are useful in two ways: to apply a distinct visual style on selected tag names and to present directly in the first view column, important attributes values (like name or id) to better identify tags. The following image shows how styles can ease information lookup in complex files.



A style consist of a series of attribute names that, if found, will be printed near to element names. For example when encountered a “name” attribute whose value is “Printer” for the element “window”, the text  
window “Printer”  
will be shown.

Style files are read at the program start. Their location can be chosen using 'configure' dialog, and have '.style' extension. File format is described in appendix. It is very easy to write a style file.

**Technical Informations**

**Style File format**

This section describes the structure of a style file.  
A style file is an XML file with the following structure:

**root tag:**

Tag name	attributes	Child elements
style:	<ul style="list-style-type: none"><li>“name” style name as shown in</li></ul>	<ul style="list-style-type: none"><li>“keywords”</li></ul>

	the user interface <ul style="list-style-type: none"> <li>• “description”: a description</li> </ul>	<ul style="list-style-type: none"> <li>• “styles”</li> <li>• “ids”</li> </ul>
--	---	---

### ***Element “styles”***

This element is simply a collection of “style” elements

### ***Element “style”***

Tag name	attributes	Child elements
style	<ul style="list-style-type: none"> <li>• “id” unique identifier of the style. It is a string</li> <li>• “color”: hexadecimal representation of a color used to paint the text. Example: “FF0000”</li> <li>• “family”: font family if different than default.</li> <li>• “size”: font size if different from default.</li> <li>• “bold” set to 'true' or a numeric value different from zero to force bold style on font</li> <li>• “italic”: same of bold, but for italic style</li> </ul> <p>The only mandatory attribute is id, the others are activated if set.</p>	

### ***Element “keywords”***

This element is simply a collection of “keyword” elements

### ***Element “keyword”***

A keyword, with associated style. Each element tag in the data file that is enrolled in this section will be printed with the indicated style

Tag name	attributes	Child elements
keyword	<ul style="list-style-type: none"> <li>• “keyword” the name of the element to mark</li> <li>• “idStyle”: the identifier of the associated style</li> </ul>	

## Element “ids”

This element is simply a collection of “id” elements

## Element “id”

This element contains the name of attributes whose content will be printed next to element tag

Tag name	attributes	Child elements
id	<ul style="list-style-type: none"><li>• “id” name of the attribute that is considered as an identifier.</li><li>• “alpha”: if true, force the enclosing of the value between quotes.</li></ul>	

This is a complete sample style file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- this is a sample QXmlEdit style file -->
<style name="Bluish style" description="this is a simple style in blu">

<keywords>
  <keyword keyword="resource" idStyle="1"/>
  <keyword keyword="widget" idStyle="1"/>
  <keyword keyword="class" idStyle="1"/>
  <keyword keyword="layout" idStyle="2"/>
  <keyword keyword="window" idStyle="5"/>
  <keyword keyword="property" idStyle="3"/>
</keywords>

<styles>
  <style id="1" color="2080FF" />
  <style id="2" color="FF00FF" family="Lucida" size="10" bold=""
italic="true" />
  <style id="3" color="C0C0FF" family="Verdana" size="14" bold="true"
italic="" />
  <style id="5" color="0000FF" italic="true" />
</styles>

<ids>
  <id id="name" alpha="true"/>
  <id id="id" alpha=""/>
  <id id="buildId" alpha=""/>
</ids>

</style>
```

## Installation of new styles

Given that styles are so simple, they can be created by the user with a simple text editor. Styles are searched in a directory configured via 'Configure...' menu.