

# Main concept of the Web frontend

**Petr Novák**

19. 11. 2004

The Web frontend is intended to manage the XML configuration during the interactive Web interface.

## 1 Requires

The Web frontend must be able to work correctly with several types of Netopeer schemas (in the RELAX NG format). If the user can edit some configuration, the Web frontend generates the HTML forms from the Netopeer schema. The version of these forms is the same like the version of the Netopeer RELAX NG schema.

## 2 Design

The Web frontend is based on HTML forms with CSS. The page is designed like *NetScreen* or *Extreme Networks* - there is a box with the main menu in the top and there is a box in the left part of the screen to select which part user can edit. There is the status bar in the bottom indicating which version user edits, whether the version is valid, saved, committed, etc. The rest of the page is the HTML form for edit the XML configuration. If some error is occurred, the section is marked by red color and blinking (if possible).

## 3 Concept

Web frontend consists of two main parts: the XSLT style-sheet for generating the HTML forms and the Web application for processing the data from these forms.

### 3.1 Part one: the XSLT style-sheet

XSLT style-sheet generates the forms from selected RELAX NG schema and from the actual configuration (it fills the HTML boxes with the actual values from the configuration). The result is the HTML form designed to the selected RELAX

[menu](#) [validate](#) [save](#) [load](#) [commit](#) [checkout](#) [support](#) [logout](#)

**description**

**logging-options**

**enabled**

**log-destination**

log-file: /var/log/netopeer.log

**log-classes**

info  
 warning  
 error  
 fatal  
 trace  
 remote  
 auth

**debug-options**

**debug-protocol**

Revision: 1.2.3 NOT CHANGED / COMMITTED

**Figure 1:** Example - the part of the System section

NG schema with values from the actual XML configuration. The creating of the HTML forms needs two steps: in the first step it is created the XSLT style-sheet for current schema:

```
$ xsltproc main.xsl netopeer.rng > temp.xsl.
```

The second step is applying this XSLT style-sheet to the XML configuration:

```
$ xsltproc temp.xsl configuration.xml > form.html
```

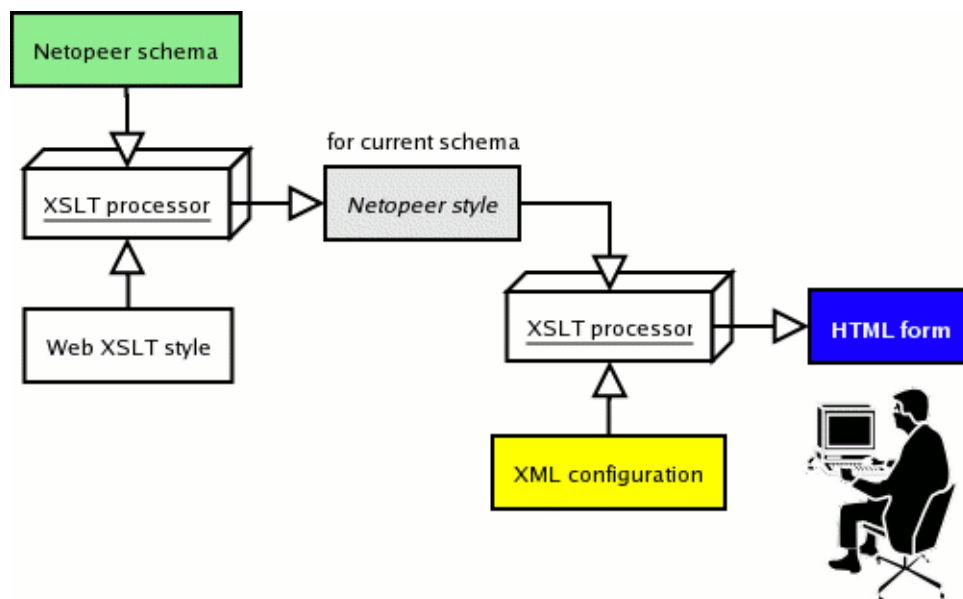
(this is very simple, in the fact you must initialize some variable here) All of these transformations will be executed from the Web frontend CGI application during the API of the XSLT processor.

The forms are in the follow format: the name of even input box is the absolute XPath path to the node of the current configuration and the value is the value of that item. For example:

```
<input type="text" name="/configuration[1]/system[1]/box-id[1]" value="" />
```

This part is written in XSLT 1.0 and is designed to be used with *libxslt xsltproc* XSLT processor.

If there is some "table in the table" in the form, it is written as a hypertext link in the first table, because the "table in table" is not good human readable. After



**Figure 2:** Generating the HTML forms from the Netopeer schema and data from configuration

clicking to this link it is created a new window with the second table. For all those links it is written out absolute XPath path to them during processing the XSLT style-sheet.

If you have got a XSLT style-sheet generated for the actual RELAX NG schema, you can generate some parts of HTML forms - you only need to set the absolute XPath path to the nodes which you can process (and some variables - see later).

For example for generating the HTML form for processing `/configuration[1]/system[1]`, you have to use:

```

$ xsltproc main.xsl netopeer.rng > temp.xsl
$ xsltproc toProcess "/netopeer:configuration[1]/netopeer:system[1]"
temp.xsl configuration.xml > form.html
  
```

(see "Variables for creating the HTML form from the XSLT style-sheet" section for the list of all available variables).

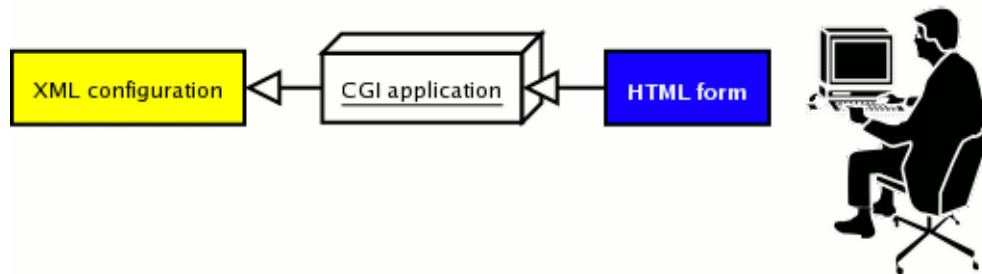
### 3.1.1 Some notes for status in this time

- If you need to generate the HTML form for processing some part of the schema, you must set some variables - see "Variables for creating the HTML forms from the XSLT style-sheet" section.
- There is a problem with creating the output namespace `xmlns:netopeer`. It is possible to do it by "deprecated" way - create the output style-sheet as

a text (not as a XML) and xmlns:netopeer write as a text too. The second way is to give to all elements the namespace - this is easy but the output is very unreadable and the style-sheet spends more space (and it is slower).

### 3.2 Part two: the Web application

The purpose of the Web application is to process the data from the forms. The main function of the Web application is to read data from forms and write it to the XML configuration file.



**Figure 3:** Sending data from HTML forms to the XML configuration

This application is designed to use the *FastCGI* library. Simply it gets the names and the values from the forms and saves them to the XML configuration. There are some special "key words" for specify the actions over the configuration as a deleting some node etc. All these "key words" and its values are defined in *main.xsl* XSLT file as a variables.

## 4 Implementation of the XSLT style-sheet

XSLT style-sheet consists of two parts. One part is the *main.xsl* style-sheet, which generates all the templates of the result XSLT style-sheet. For elements it creates two templates: one is in format `<xsl:template match="...">` and is used if element exists in the XML configuration (this template switches the context node to the processing element). It contains only calling the second template: `<xsl:call-template name="...">`. The second template is the template in the form `<xsl:template name="...">` where the name is the *id* for the element. It is called always - if the element exists it is called by the `<xsl:template match="...">` template and if the element doesn't exist it is called by its parent. The name of the template for the element contains the name of the parent(s) too (because there are many elements with the same name in the Netopeer RELAX NG schema and it is necessary to different them). For attributes it is only created a template in the format `<xsl:template name="...">` where name is the *id* of the attribute.

The second part is RELAX NG parser, which parses the Netopeer RELAX NG schema and creates the proper HTML forms from it. It contains many files with prefix "relaxng". Note that it is not parser for any RELAX NG, it implements only parts important for Netopeer RELAX NG (see documentation in the *relaxng-core.xsl* for more).

#### **4.1 Variables in the templates**

There are some variables in every template. There are some differences between the variables in the templates for elements and for attributes. See documentation in the *main.xsl* for more.

#### **4.2 Variables for creating the HTML forms from the XSLT style-sheet**

You must initialize some variables during the second step of the generating the HTML forms from the RELAX NG schema. There are the following variables:

- `toProcess` - contains the absolute XPath path which indicates which part of the Netopeer schema the HTML form will be generated from.
- `toProcessPath` - is the string which contains the prefix of the generated part of the Netopeer schema. It depends on "toProcess" variable.
- `error` - is the string which is the absolute XPath path to the node in which the error is occurred.
- `revision` - is the string with the number of the revision of the editing configuration.
- `changed` - is the boolean XPath expression which indicates whether the editing configuration was changed.
- `committed` - is the boolean XPath expression which indicates whether the editing configuration was changed.