Table of Contents

Introduction

This part of the documentation is intended for anyone who creates and publishes web pages using a Roxen

It is assumed that the reader is familiar with HTML and has some experience with XML. Most of Roxen WebServer's functions are available as RXML tags, easily learned by anyone who knows HTML or XML.

The Manual

As of now all tag documentation can be found within Roxen WebServers's source code. This manual is somewhat unique as it uses a system that automatically imports the documentation into each tag's file.

The purpose of this system is to:

- keep documentation redundancy at a minimum.
- make updating the documentation as painless a task as possible.
- provide a help system through <*help*>.

End of /roxen/2.1/creator/introduction/index.xml

Concepts

Start of /roxen/2.1/creator/introduction/concepts.xml

The Roxen WebServer is a modular web server.

RXML

Content on web pages is written using HTML, a text format with mark-up in the form of tags telling the browser how the content should be displayed. The Roxen WebServer comes with its own macro language, RXML, that uses tags like the ones in HTML. RXML is never sent to the browser though, as all the RXML tags are converted into HTML using the RXML parser.

RXML can be used for a number of things, such as creating graphical headings and diagrams, connecting to databases, creating dynamic pages or all of the above. The bulk of this manual describes the various RXML tags and how they can be combined. The key to RXML is that each tag solves a separate task. Hence several tags can be combined to perform an even greater task.

RXML Parser

RXML is parsed and converted to HTML by the RXML Parser. The RXML parser is an advanced engine fully integrated with the that helps in converting all RXML tags into HTML using very advanced methods.

Modules

Roxen WebServer uses a system of modules. The different functions of Roxen WebServer are handled by different modules. Modules are enabled and configured through the administration interface by the administrator. Many modules handle different RXML tags, therefore, which RXML tags can be used depends on how the administrator has configured the virtual server. If a tag is not available, the administrator can add the proper module to the server to enable the tag.

Modules that handle RXML tags can be written by third-party developers or any programmer with sufficient knowledge of Pike and Roxen. It is also possible to create packages of RXML tags in RXML, for use with the <*use*> tag.

XML

XML, Extensive Markup Language, is a markup language for documents containing structured information. The XML specification defines a standard way to add markup to documents. The Roxen WebServer supports XML and its rigorous demands on document formatting.

End of /roxen/2.1/creator/introduction/concepts.xml

Changes

Start of /roxen/2.1/creator/introduction/changes.xml

Changes: 001114

- Major markup overhaul to enable PDF-generation.
- Tag/page references.
- Module - tag relationship shown.
- Fixed a lot of small bugs.

New Features in Roxen 2.1

A new type of tag has been introduced into the RXML Parser, XML compliant processing instruction tag, i.e. `<?tagname ... ?>`. The first tags to make use of this construction are <*?pike ?*> and <*?perl ?*> that makes it possible to run Pike and Perl code embedded in RXML pages.

New and changed tags:

- `<wash-html>...</wash-html>` do some common text to html tasks.
- `<?comment ..... ?>`
- `<if sizeof=.../>`
- `<sprintf [spitemt=separator] format=...>`values,...`</sprintf>`
- `<sscanf variables=a,b,... format=...>`data`</sscanf>`
- _.scopename will be the name of the default scope, if scopename is not used as a variable in that scope.
- `<dice type='3D6+D4'>`
- `<fsize>` now prints the size in a somewhat less user-unfriendly way.
- `<gbutton>` can now have images above or below the text (for toolbar style images).
- `<cimg>` checks mtime on the source file (if any).
- `<date strftime=format>`
- `<vform>`, with which you can create self-verified forms. Uses the same widgets as the configuration variables.
- fonts - A list of all fonts available.
- directory - Directory listings.
- path - Separate a path on '/'.

- values - Traverse a list, e.g. the response from a multivalue select.
- cimg - Like the <cimg> tag, but entity based. Includes _.type, _.src, _.file-size, _.xsize, _.ysize and _.data
- ldap - LDAP search query.

Emit can now sort the result. It is possible to select a sub-range of the result.

Changes from Roxen 1.3

Many changes have occured within the webserver since Roxen Challenger 1.3 was released. A total redesign of RXML has been made. The intention was to make RXML into a XML-compliant markup language. XML is now the preferable document format. However, full compatability with earlier versions of Roxen Challenger is ensured.

To improve usability the consistency in RXML has been greatly improved. Names for tags, attributes and arguments now have a similar name convention and are more easy to remember.

The RXML redesign includes the introduction of Scopes into RXML. This has made it is possible to reach a lot of the Roxen WebServer's internal data easily. Gathering statistics and putting them on a web site has never been easier. All variables in RXML can now be reached with a scope.

No more quoting problems. In Roxen Challenger 1.3 it was awkward to write complex applications. RXML tags like <sqloutput> and <formoutput> had serious quoting problems and nestling tags was a pain. This has now been remedied with the scope system and a new intelligent output tag; the <emit> tag. This tag has functionality to help parsing quotes i.e. in SQL-queries. Also, together with scopes, different quoting-characters is not necessary anymore.

With the general redesign of RXML, its parser has a more advanced caching engine with a better memory management to prevent from unnecessary parsing.

WAP and WML support is now part of the Roxen WebServer. To help building advanced WAP applications a special set of WML tags has been designed.

Compatibility with Older Versions

Websites built with earlier versions of Roxen Challenger won't operate properly or not at all under Roxen WebServer unless certain issues are solved. Read more: *Compatibility.*

End of /roxen/2.1/creator/introduction/changes.xml

Publishing Web Pages

To publish web pages using Roxen WebServer there are a few things the web page creator ought to know. This chapter will explain the basics.

Roxen WebServer

To publish an HTML page through Roxen WebServer, the web page creator only needs to know in what directory the files should be placed and on what URL the pages will be found. This is configured by the administrator of the server.

The site must have a file system module enabled. The file system module can mount a directory from a normal file system as a virtual file system. When referring to other files, the path to the file must be in the virtual file system. Both absolute and relative paths can be used. An absolute path is static and begins at the root-level of the file system or mounted directory. A relative path however is dynamic and relative to the page from where the path is linked.

```
Our file's path: /home/eric/html/my_site/travel/
Greece/animals.html

Absolute path to link an image from animals.html:
<img src="/home/eric/html/my_site/img/
grasshopper.jpg" />

Relative path to link an image from animals.html:
<img src="../../img/grasshopper.jpg" />

Relative path to link a "facts" page from animals.h
tml:
<a href="../../animals/insects/
grasshoppers.html">More about grasshoppers</a>
```

As in the example, a website might have a dedicated directory for images in the site's root-directory (/home/eric/html/my_site/). This dedicated directory becomes a place where all images for the site might be stored and is easily accessed from the whole site. Using relative links to these images and other documents on the site makes it possible to move the entire site elsewhere without disrupting the links on the site.

To display the contents of a directory, a directory parsing module is needed. If a file called `index.html` is found in a directory that is mounted by a file system module, it will be shown when pointing a browser to the URL of that directory. By default the server will look for the following files: `index.html`, `index.xml`, `index.htm`, `index.pike` and `index.cgi`, in that order. The names and the order can be configured by the administrator.

End of /roxen/2.1/creator/publishing/index.xml

Content Types

Start of /roxen/2.1/creator/publishing/content_types.xml

Each file fetched through a web server contains a MIME that identifies what type of file it is. Thus an HTML file has the content-type *text/html*, while a GIF image has the content-type *image/gif*.

In the Roxen WebServer, the file extension determines the content type of that file. Usually *.html* or *.htm* files are given the content-type *text/html* while *.gif* files are given the content-type *image/gif*.

As a user, you usually don't have to bother with content-types. If you just give your files their standard extensions everything will work. But sometimes, when you try out new plugins that use their own file format, the extension and content-type that you want to use is not handled by the server. Then the administrator for the server has to change the configurations for the *content-types* module.

Some extensions might be handled by the web server itself. The most common use is to run files through the *Main RXML parser* module. This makes it possible to use RXML tags on such pages. Depending on the policy of your site this might be done for all *.html* files, or only for special *.rxml* files.

End of /roxen/2.1/creator/publishing/content_types.xml

RXML

RXML, RoXen Macro Language, is a functional serverside XML compliant scripting language that is built into Roxen WebServer.

RXML is made to be easy to learn, especially for people with skills in HTML, as its syntax and visible semantics is similar to HTML.

RXML consists of over one hundred different tags that in themselves are much more complicated than HTML-tags, but as simple to use as HTML-tags. RXML makes it possible to create new tags by writing modules in Java or Pike, or by making wrappers around, i.e. CGI-scripts with the `<define tag>` tag.

In this way the webdesigner's and the developer's work can be separated. The developer can bind code to RXML-tags which then can be used by the webdesigners. If taken into account that webdesigners often are better than developers at designing webpages than writing code and vice versa, it can be assumed that the developer never has to hard wire HTML code into the code.

RXML was designed with security in mind, thus it is a blessing to write complex and secure applications.

RXML Tags

RXML consists of many often simple tags, that when combined can create very advanced and dynamic webpages. Most chapters following this will discuss these tags.

Since the Roxen WebServer is highly modular (all functions, tags, etc are built into modules) it is possible to customize every aspect of the Roxen WebServer. Hence, if a tag isn't working, it is very possible that its module hasn't been loaded. The tag `<help/>` gives a listing of all tags enabled as well as the latest documentation available for those tags. Note that the `<help/>` tag only will give a listing of those tags written properly as modules, not those made with the `<define>` tag and similar tags.

RXML Naming Convention

In case of planning an upgrade from a Roxen 1.3 product to Roxen 2.1 this section should be taken into consideration.

In RXML 1.3, many attributes had different names despite them performing the same tasks and some tags had strange names that didn't reflect their functionality. Due to these and other internal inconsistencies a RXML naming convention has been created.

If a tag or an attribute has changed name it most probably is because of this convention. The tags that still don't conform to the naming convention have not yet been 'newstyled', i.e. converted to make full use of the RXML 2 parser.

1. The tag and attribute name reflects the functionality as much as possible.

   E.g. although <pr> is easy to remember it does not in any way hint its functionality to the user, hence the change to <roxen>

2. The tag and attribute name should be as easy to remember as possible. Attributes have the same name as their HTML counterparts. Attribute names are now more consistent in RXML tags.

   E.g. although the *bg* argument to <*gtext*> is easy to understand and remember it differs from *bgcolor* in HTML and other RXML tags.

3. Tag and attribute names that consist of several words have been concatenated without a separator. If there for some reason is a need to use a separator, the "-" symbol has been used, for instance the "set cookie"-tag has been named <set-cookie> instead of e.g. <set_cookie>.

4. RXML now uses assignment attributes instead of atomic attributes as it might be necessary to introduce more attributes in the future that are not compatible with the old ones.

   E.g. the case description is now <tag case="upper"> instead of <tag upper> since the latter made it possible to write <tag upper lower>, which isn't compliant with XML and used to produce a situation with undefined result for the user.

End of /roxen/2.1/creator/rxml/index.xml

Parsing

Start of /roxen/2.1/creator/rxml/parse.xml

This chapter will describe how tags, scopes, entities and their attributes are parsed. To make it easier to understand how parsing in RXML works many examples will be provided.

When RXML was redesigned to be XML compliant, many of the flaws were removed. Security issues have hopefully been removed as well as many odd quirks and behaviours.

XML Compliancy

To be XML compliant, a set of requirements has to be fulfilled. At this moment the XML standard is not fully supported. Roxen is able to understand and parse all XML compliant code, but doesn't have any support for DTD-parsing. Also, the parser is very forgiving when it comes to demanding that the code follows the correct XML-syntax. The table below shows a subset of the most common differences between old RXML and the new XML compliant RXML.

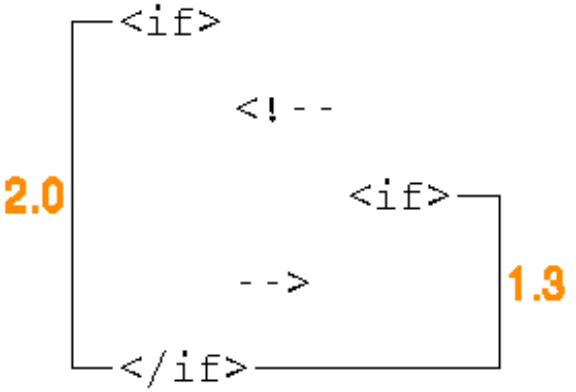
RXML 1.3	RXML 2
`<accessed>`	`<accessed/>`
`<p>`	`<p>content</p>`
`<tablify nice>`	`<tablify nice="nice"/>` or `<tablify nice=""/>`
`<if supports=cookies>`	`<if supports="cookies"/ >`

Note that if a site has older RXML code it can still survive the move to the new system. By adding the RXML compatibility module the parser will understand the old syntax.

RXML Parsing

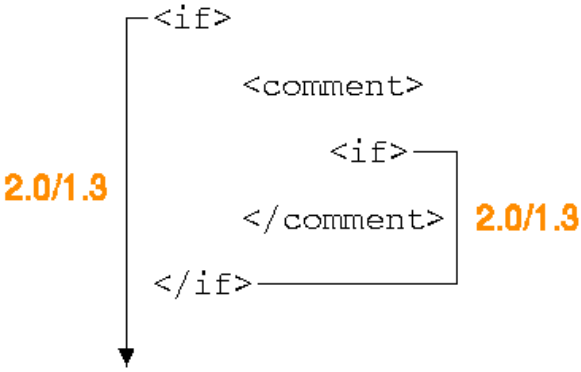
The Roxen 2 RXML parser utilizes a top-down parsing approach. This means that the parser now uses preparsing instead of postparsing as in RXML 1.3. Repetitive parsing in output tags used to be a performance hog in RXML 1.3. The new parser now makes one analyze pass when doing output from a tag, instead of rescanning the output tag in every loop. This feature makes tags like `<formoutput>` unneccessary.

The images below show some of the differences in RXML parsing between Roxen WebServer 2.1 and Roxen Challenger 1.3.



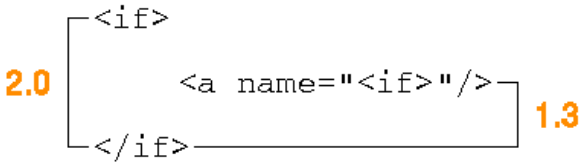
Parsing '<!--' comments

The old 1.3 RXML parser didn't recognize '<!--' comments and thus the `<if>` inside the comment is parsed and the first ignored. The RXML parser 2 consistently ignores code within comments. However, if the comment is a RXML comment, problems might arise while the parser are scanning the page for tags.



Locating the closing tag

When the parser comes upon a container tag it does a scanpass, without performing any parsing, to locate the closing tag. If the parser as shown below, finds another `<if>` inside a RXML comment it automatically looks for its closing tag thus believing the closing tag for the first `<if>` is further down the page. As the RXML parser 2 can't find the last `<if>` tag it decides there are unmatching tags in the code and returns an unbalanced tag error message.



Evaluating tags

When the parser scans a page for tags it automatically evaluates the tag that it recognizes and stores the results for later use. Recognized tags are tags that are registered within the parser, mostly RXML-tags. Tags which the parser doesn't recognize, like HTML-tags, won't be parsed but their attributes will be stored. The 1.3 parser however, isn't that smart and will declare all unknown tags as text and hence the `<if>` tag will be found and parsed.

End of /roxen/2.1/creator/rxml/parse.xml

Variables, Scopes & Entities

An entity is the collective name for a variable that in RXML 2 has been given an all accessible name, which mean it is available everywhere whenever the module defining it is loaded. A more correct name might be *variable-enitity*, but in RXML 2 it is referred to as an *entity*.

An entity mostly contains information fetched from within the Roxen WebServer, but it can also be declared by a tag, making it a carrier of website specific information.

A variable always belongs to a scope, i.e. a group of variables providing information about something specific, e.g. a *client* or a *page*. A list of variables belonging to a scope can be made available by this RXML code:

```
<pre>
<insert variables="full" scope="the scope"/>
</pre>
```

Entity syntax

The syntax that specifies a variable as an entity is "scope.variable", e.g. page.filename.

Quoting entities

When outputting information from a source that contains entities that should be parsed after the output is parsed, it is necessary to quote these entities else they will be parsed in advance. Quoting entities is done by introducing an extra "." between the scope and the variable, e.g. scope..variable

Scopes

The most common scopes that handles variables are the *Var* and *Form* scopes. The *Var Scope* is always empty when the page parsing begins and should be used for temporary variables. The *Form Scope* contains all returned results from forms.

There is no default scope on the page top level, i.e. outside of all tags that creates scopes, unless you are running in compatibility mode. Then the *Form Scope* will be your default scope. The scope "_" (underscore) is always the present scope, e.g. in <*emit sql*> where sql.x = _.x.

```
<emit source="sql" query=" ... " scope="outer">
 &_.name; = &outer.name;
  <emit source="sql" query=" ... " scope="inner">
   &_.name; = &inner.name; != &outer.name;
  </emit>
</emit>
```

This is an example on how entities works inside nestled <*emit*> tags. Notice how the present scope "_" (underscore) changes from the first `<emit>` to the second.

Entities in RXML 2 are handled like any variable in RXML 1.3. Examples:

RXML	Result
`<set variable="var.foo" value="bar"/>`	var.foo = bar

RXML	Result
`<set variable="var" scope="foo" value="bar"/>`	foo.var = bar
`<if variable="var.foo = bar">gazonk</if>`	Returns gazonk if var.foo = bar.
`<if match="var.foo = *test">gazonk</if>`	Returns gazonk if var.foo ends in test.

Splice

The splice attribute-operand '::', has a similar function in RXML as in most programming languages. Splice in RXML is used for expanding what is inside a variable, i.e. put what is stored in the variable into for instance a tag before parsing it. It might for instance be a list of attributes stored in the variable or an URL, etc. The splice operand is necessary as the RXML-parser (RXML 2) only will make one pass when parsing a page while the old parser (RXML 1.3) sometimes made several passes while parsing a tag.

```
<define variable='var.foo'>quant='20' gamma='0.5'</
define>
<cimg src='../img/testimage.jpg' ::='var.foo'/>
```

The really useful thing with :: is that it is possible to give attributes, which are unknown, to a tag in advance, which makes it is possible to store an entire list of attributes in a variable.

```
<!-- before -->
<sandwich ::="bread='rye' &var.stuffings;" butter="
low-fat"/>

<!-- after -->
<sandwich bread='rye' lettuce="" ham="" tomato="" b
utter="low-fat"/>
```

Encoding

An entity can be placed anywhere in a page and will have its content inserted during the parsing. The content is encoded per default so that "<" will be output as "<" in HTML pages. You can choose another scheme by using the *scope.variable:scheme* syntax, e.g. form.html:none. *[More about encoding]*

The scopes are:

End of /roxen/2.1/creator/entity/index.xml

Encoding

Start of /roxen/2.1/creator/entity/encoding.xml

All variables in RXML 2 are accessed through entities, i.e. var.foo.

By default, an entity will be HTML encoded, that is, < will be inserted as <, > as > and & as & However, there are instances when that is not what you want, for example, when inserting enitities into SQL queries. Therefore, the encoding can be controlled by applying another encodingscheme on the entity, scope.entity:*scheme*.

Available Encoding Schemes

- No quoting. This is dangerous and should never be used unless you have total control over the contents of the variable. If the variable contains an RXML tag, the tag will be parsed.
- The default quoting, for inserting into regular HTML or RXML. Encoded characters are "&", "<", ">", "\"", "\'" and "\000".
- For inserting variables into URLs. Encoded characters are "\000", " ", "\t", "\n", "\r", "%", "'", "\"", "#", "&", "?", "=", "/" and ":".
- Uses a subset of the URL encoding scheme. Characters "&" and "?" are not encoded as it would make inserting i.e. variables into http-strings impossible. Encoded characters are "\000", " ", "\t", "\n", "\r", "%", "'" and "\"".
- Uses a subset of the URL encoding scheme. Only the characters "=", ",", ";" and "%" are encoded.
- For inserting into Pike strings, for use with the `<pike>` tag. Encoded characters are "\"", "\\" and "\n".
- For inserting into Javascript strings. Encoded characters are "\b", "\014", "\n", "\r", "\t", "\\", "'" and "\"".
- For inserting into MySQL SQL queries. Encoded characters are "\"", "'" and "\\".
- For inserting into MySQL SQL queries in pike strings. Encoded characters are "\"", "'", "\\" and "\n".
- For inserting into SQL queries. Encoded characters are "'".

End of /roxen/2.1/creator/entity/encoding.xml

Client

This scope contains information specific to the client/browser that is accessing the page.

&client.Fullname; The full user agent string, i.e. name of the client and additional info like; operating system, type of computer, etc. E.g. "Mozilla/4.7 [en] (X11; I; SunOS 5.7 i86pc)".

&client.accept-language; The client prefers to have the page contents presented in this language.

&client.accept-languages; The client prefers to have the page contents presented in this language but these additional languages are accepted as well.

&client.authenticated; Returns the name of the user logged on to the site, i.e. the login name, if any exists.

&client.fullname; The full user agent string, i.e. name of the client and additional info like; operating system, type of computer, etc. E.g. "mozilla/4.7 [en] (x11; i; sunos 5.7 i86pc)".

&client.height; The presentation area height in pixels. For WAP-phones.

&client.host; The host name of the client, if possible to resolve.

&client.ip; The client is located on this IP-address.

&client.javascript; Returns the highest version of javascript supported.

&client.language; The clients most preferred language.

&client.languages; An ordered list of the clients most preferred.

&client.name; The name of the client, i.e. "Mozilla/4.7".

&client.password;

&client.referrer; Prints the URL of the page on which the user followed a link that brought her to this page. The information comes from the referrer header sent by the browser.

&client.robot; Returns the name of the webrobot. Useful if the robot requesting pages is to be served other contents than most visitors. Use client.robot together with <*if*>.

Possible webrobots are: ms-url-control, architex, backrub, checkbot, fast, freecrawl, passagen, gcreep, getright, googlebot, harvest, alexa, infoseek, intraseek, lycos, webinfo, roxen, altavista, scout, slurp, url-minder, webcrawler, wget, xenu and yahoo.

&client.user; Returns the name the user used when he/she tried to log on the site, i.e. the login name, if any exists.

&client.width; The presentation area width in pixels. For WAP-phones.

Cookie

This scope contains the cookies sent by the client. Adding, deleting or changing in this scope updates the clients cookies. There are no predefined entities for this scope. When adding cookies to this scope they are automatically set to expire after two years.

Form

This scope contains the form variables, i.e. the answers to HTML forms sent by the client. There are no predefined entities for this scope.

Page

This scope contains information specific to this page.

&page.accessed; Generates an access counter that shows how many times the page has been accessed. Needs the accessed module.

&page.author; Returns the author's name.

&page.author-id; Returns the author's identification number.

&page.author-name; Returns the author's handle, i.e. login name.

&page.content-editor; Returns the full internal SiteBuilder URL to the page in the Content Editor, focusing on the file.

&page.description; Returns the description of the file, set in the metadata.

&page.filename; Returns the filename.

&page.filesize; Returns the filesize.

&page.filesize; This file's size, in bytes.

&page.keywords; Returns the keywords of the file, set in the metadata.

&page.language; What language the contens of this file is written in. The language must be given as metadata to be found.

&page.languages; All languages present in the file returned as a comma-separated string.

&page.last-true; Is "1" if the last <if>-statement succeeded, otherwise 0. (<*true*> and <*false*> is considered as <if>-statements here) See also: *If Tags*.

&page.modification-date; Returns the date when the file was last modified.

&page.modification-time; Returns the time when the file was last modified.

&page.path; Absolute path to this file in the virtual filesystem.

&page.pathinfo; The "path info" part of the URL, if any. Can only get set if the "Path info support" module is installed. For details see the documentation for that module.

&page.permission; Returns the string "read" or "write" depending on whether the viewer has permission to write in the file or not.

&page.query; The query part of the page URI.

&page.realfile; Path to this file in the file system.

&page.revision; Returns the file's current revision number.

&page.scope; The name of the current scope, i.e. the scope accessible through the name "_".

&page.selectable; For showing whether an XSL template file is selectable within the "Edit metadata" wizard. Returns either "n/a" if the file is of an other content-type than "site-builder/xsl-template", "yes" if the file is selectable or "no" if the file isn't selectable.

&page.selected; Whether this file is the current file or if this directory is a directory within the path to the current file. Only one entry will be selected.

&page.self; The name of this file.

&page.site-status; Returns the file's status in the site's workarea.

&page.ssl-strength; The strength in bits of the current SSL connection.

&page.stationery; Returns "yes" if the page is marked as a stationery, "no" otherwise.

&page.status-img; Returns the full internal SiteBuilder URL to the page's current status image. The status image tells if the page has been modified etc. from the viewer's point of view. This is the same status icon as used by the Content Editor.

&page.template; Returns the current page's selected template.

&page.title; Returns the title of the file, set in the metadata.

&page.type; Returns the file's content-type.

&page.type-img; Returns the internal SiteBuilder URI to where the file's content-type image is stored.

&page.url; The absolute path for this file from the web server's root or point of view including query variables

&page.user-status; Returns the file's status in the user's workarea.

&page.virtroot; The root of the present virtual filesystem.

&page.visible; Returns "yes" if the page is currently externally visible, "no" otherwise.

&page.visible-from; Returns the date and time, in iso format, when the page starts to be visible. If external visibility is set to *Never*, "never" will be returned. If external visibility is set to *Visible util a specified time*, "now" will be returned.

&page.visible-to; Returns the date and time, in iso format, when the page ends to be visible. If external visibility is set to *Never*, "never" will be returned. If external visibility is set to *Visible after a specified time*, "indefinite" will be returned.

&page.workarea; Returns the name of the workarea where the viewed page is stored.

&page.workarea-id; Returns the unique id of the workarea. Useful when doing web applications.

Roxen

This scope contains information specific to this Roxen WebServer. It is not possible to write any information to this scope

&roxen.domain; The domain name of this site.

&roxen.hits; The number of hits, i.e. requests the webserver has accumulated since it was last started.

&roxen.hits-per-minute; The number of hits per minute, in average.

&roxen.pike-version; The version of Pike the webserver is using.

&roxen.sent; The total amount of data the webserver has sent.

&roxen.sent-kbit-per-second; The average amount of data the webserver has sent, in Kibibits.

&roxen.sent-mb; The total amount of data the webserver has sent, in Mebibits.

&roxen.sent-per-minute;

&roxen.server; The URL of the webserver.

&roxen.ssl-strength; How many bits encryption strength are the SSL capable of

&roxen.time; The current posix time.

&roxen.uptime; The total uptime of the webserver, in seconds.

&roxen.uptime-days; The total uptime of the webserver, in days.

&roxen.uptime-hours; The total uptime of the webserver, in hours.

&roxen.uptime-minutes; The total uptime of the webserver, in minutes.

&roxen.version; Which version of Roxen WebServer that is running.

Var

This scope is empty when the page parsing begins. There are no predefined entities for this

WML Support Tags

The Roxen WebServer WAP/WML support allows for fast and easy development of pages for WAP-enabled phones.

As different WAP phones have different browsers that function differently and support different versions of WML the Roxen WebServer WML support includes a system that handles all phones individually. The WML support also allows for automatic conversion of contents between the WML 1.0 and 1.1 open standards. This has great impacts as it regards WAP content management/ application development, which will become very much simplified. Further, Roxen WebServer automatically converts any image (standard Web image like GIF/JPEG/PNG, or others) to the specific b/w format (WBMP) that WAP phones understand.

The automatic conversion for different WAP phones and different Web browsers is handled by the "Roxen Browser Feature Supports" database. This holds the information of what features each browser can handle, and that is used to automatically customize the output sent.

When developing WAP-pages the developer only needs to know which WML-standard to support.

End of /roxen/2.1/creator/wml/index.xml

<wml></wml>

Provided by module: *WAP WML helper*

Processes the wml tag and adapts the contents to better suit the client. The contents is always preparsed. No attributes are required.

Attributes

from="{1.0, 1.1}" (1.1)
Tells what version of wml is used.

to="{1.0, 1.1}"
Force conversion to this version of wml.

noheader
If used, no xml and doctype tags will be added to the document.

mime="string"
Sets the mime-type of the document.

Information Tags

Information tags are simple tags that provide information about the client, the server or some external event. Examples are <*accessed*>, that counts accesses to the page and <*modified*>, which shows when the page was last updated.

End of /roxen/2.1/creator/information/index.xml

<accessed/>

Provided by module: *Accessed counter*

Generates an access counter that shows how many times the page has been accessed. A file, AccessedDB, in the logs directory is used to store the number of accesses to each page. By default the access count is only kept for files that actually contain an accessed-tag, but can also be configured to count all files of a certain type.

```
<accessed/>
```

Attributes

add="number"
Increments the number of accesses with this number instead of one, each time the page is accessed.

addreal
Prints the real number of accesses as an HTML comment. Useful if you use the cheat attribute and still want to keep track of the real number of accesses.

case="{upper, lower, capitalize}"
Sets the result to upper case, lower case or with the first letter capitalized.

cheat="number"
Adds this number of accesses to the actual number of accesses before printing the result. If your page has been accessed 72 times and you add `<accessed cheat='100'>` the result will be 172.

database
Works like the since attribute, but counts from the day the first entry in the entire accessed database was made.

factor="percent"
Multiplies the actual number of accesses by the factor. E.g. `<accessed factor='50'>` displays half the actual value.

file="filename"
Shows the number of times the page filename has been accessed instead of how many times the current page has been accessed. If the filename does not begin with "/", it is assumed to be a URL relative to the directory containing the page with the accessed tag. Note, that you have to type in the full name of the file. If there is a file named tmp/index.html, you cannot shorten the name to tmp/, even if you've set Roxen up to use index.html as a default page. The filename refers to the virtual filesystem.
One limitation is that you cannot reference a file that does not have its own `<accessed>` tag. You can use `<accessed silent='1'>` on a page if you want it to be possible to count accesses to it, but don't want an access counter to show on the page itself.

lang="langcodes"
Will print the result as words in the chosen language if used together with type=string.

```
<accessed type="string"/>
```

```
<accessed type="string" lang="sv"/>
```

per="{second, minute, hour, day, week, month, year}"
Shows the number of accesses per unit of time.

```
<accessed per="week"/>
```

prec="number"
Rounds the number of accesses to this number of significant digits. If prec=2 show 12000 instead of 12148.

reset
Resets the counter. This should probably only be done under very special conditions, maybe within an `<if>` statement. This can be used together with the file argument, but it is limited to files in the current- and sub-directories.

silent
Print nothing. The access count will be updated but not printed. This option is useful because the access count is normally only kept for pages with actual `<access>` on them. `<accessed file='filename'>` can then be used to get the access count for the page with the silent counter.

since
Inserts the date that the access count started. The language will depend on the *lang* attribute, default is English. All normal date related attributes can be used. Also see: <*date*>.

```
<accessed since=""/>
```

type="{number, string, roman, iso, discordian, stardate, mcdonalds, linus, ordered}"
Specifies how the count are to be presented. Some of these are only useful together with the since attribute.

```
<accessed type="roman"/>
```

```
<accessed since="" type="iso"/>
```

```
<accessed since="" type="discordian"/>
```

```
<accessed since="" type="stardate"/>
```

```
<accessed type="mcdonalds"/>
```

```
<accessed type="linus"/>
```

```
<accessed type="ordered"/>
```

minlength="number"
Defines a minimum length the the resulting string should have. If it is shorter it is padded from the left with the padding value. Only values between 2 and 10 are valid.

padding="character" (0)
The padding that the minlength function should use.

<countdown/>

Provided by module: *Countdown*

This tag can count days, minutes, months, etc. from a specified date or time. It can also give the time to or from a few special events. See below for a full list.

Attributes

Time:

year="number"
Sets the year.

month="{number, month_name}"
Sets the month.

day="{number, day_name}"
Sets the weekday.

mday="number"
Sets the day of the month.

hour="number"
Sets the hour.

minute="number"
Sets the minute.

second="number"
Sets the second.

iso="year-month-day"
Sets the year, month and day all at once. (YYYY-MM-DD, YYYYMMDD or YYYY-MMM-DD).

```
<countdown iso='2020-FEB-12'/>
```

event="easter,gregorian-easter,julian-easter,christmas,christmas-day,christmas-eve"
Sets the time of an evet to count down to.

years="number"
Add this number of years to the result.

months="number"
Add this number of months to the result.

weeks="number"
Add this number of weeks to the result.

days="number"
Add this number of days to the result.

hours="number"
Add this number of hours to the result.

beats="number"
Add this number of beats to the result.

minutes="number"
Add this number of minutes to the result.

seconds="number"
Add this number of seconds to the result.

now="year-month-day"
Sets the 'present' time, if other than really present time. (YYYY-MM-DD, YYYYMMDD or YYYY-MMM-DD)

```
<countdown now="1999-12-
24" year="2000" display="days"/>
```

Presentation:

<date/>

Provided by module: *RXML 2 tags*

Inserts the time and date. Does not require attributes.

Attributes

unix-time="number"
Display this time instead of the current. This attribute uses the specified Unix 'time_t' time as the starting time, (which is *01:00, January the 1st, 1970*) instead of the current time. This is mostly useful when the `<date>` tag is used from a Pike-script or Roxen module.

```
<date unix-time='120'/>
```

timezone="{local, GMT}" (local)
Display the time from another timezone.

years="number"
Add this number of years to the result.

```
<date date='' years='2'/>
```

months="number"
Add this number of months to the result.

```
<date date='' months='2'/>
```

weeks="number"
Add this number of weeks to the result.

```
<date date='' weeks='2'/>
```

days="number"
Add this number of days to the result.

hours="number"
Add this number of hours to the result.

```
<date time='' hours='2' type='iso'/>
```

beats="number"
Add this number of beats to the result.

```
<date time='' beats='10' type='iso'/>
```

minutes="number"
Add this number of minutes to the result.

seconds="number"
Add this number of seconds to the result.

adjust="number"
Add this number of seconds to the result.

brief
Show in brief format.

```
<date brief=''/>
```

time
Show only time.

```
<date time=''/>
```

date
Show only date.

```
<date date=''/>
```

type="{string, ordered, iso, discordian, stardate, number}"
Defines in which format the date should be displayed in. Discordian and stardate only make a difference when not using part. Note that type=stardate has a separate companion attribute, prec, which sets the precision.

type=discordian	
type=iso	
type=number	
type=ordered	
type=stardate	
type=string	

part="{year, month, day, wday, date, mday, hour, minute, second, yday, beat, week, seconds}"

Defines which part of the date should be displayed. Day and wday is the same. Date and mday is the same. Yday is the day number of the year. Seconds is unix time type. Only the types string, number and ordered applies when the part attribute is used.

part=year	Display the year.
part=month	Display the month.
part=day	Display the weekday, starting with Sunday.
part=wday	Display the weekday. Same as 'day'.
part=date	Display the day of this month.
part=mday	Display the number of days since the last full month.
part=hour	Display the numbers of hours since midnight.
part=minute	Display the numbers of minutes since the last full hour.
part=second	Display the numbers of seconds since the last full minute.
part=yday	Display the number of days since the first of January.
part=beat	Display the number of beats since midnight Central European Time(CET). There is a total of 1000 beats per day. The beats system was designed by Swatch as a means for a universal time, without time zones and day/night changes.
part=week	Display the number of the current week.
part=seconds	Display the total number of seconds this year.

strftime="string"

If this attribute is given to date, it will format the result according to the argument string.

%%	Percent character
%a	Abbreviated weekday name, e.g. "Mon"
%A	Weekday name
%b	Abbreviated month name, e.g. "Jan"
%B	Month name
%c	Date and time, e.g. "%a %b %d %H:%M:%S %Y"
%C	Century number, zero padded to two charachters.
%d	Day of month (1-31), zero padded to two characters.
%D	Date as "%m/%d/%y"
%e	Day of month (1-31), space padded to two characters.
%H	Hour (24 hour clock, 0-23), zero padded to two characters.
%h	See %b
%I	Hour (12 hour clock, 1-12), zero padded to two charcters.
%j	Day numer of year (1-366), zero padded to three characters.
%k	Hour (24 hour clock, 0-23), space padded to two characters.
%l	Hour (12 hour clock, 1-12), space padded to two characters.
%m	Month number (1-12), zero padded to two characters.
%M	Minute (0-59), zero padded to two characters.
%n	Newline
%p	"a.m." or "p.m."
%r	Time in 12 hour clock format with %p

%R	Time as "%H:%M"
%S	Seconds (0-61), zero padded to two characters.
%t	Tab
%T	Time as "%H:%M:%S"
%u	Weekday as a decimal number (1-7), 1 is Sunday.
%U	Week number of year as a decimal number (0-53), with sunday as the first day of week 1, zero padded to two characters.
%V	ISO week number of the year as a decimal number (1-53), zero padded to two characters.
%w	Weekday as a decimal number (0-6), 0 is Sunday.
%W	Week number of year as a decimal number (0-53), with sunday as the first day of week 1, zero padded to two characters.
%x	Date as "%a %b %d %Y"
%X	See %T
%y	Year (0-99), zero padded to two characters.
%Y	Year (0-9999), zero padded to four characters.

```
<date strftime="%Y%m%d"/>
```

lang="langcode"
Defines in what language a string will be presented in. Used together with *type=string* and the *part* attribute to get written dates in the specified language.

```
<date part='day' type='string' lang='de'>
```

case="{upper, lower, capitalize}"
Changes the case of the output to upper, lower or capitalize.

```
<date date='' lang='&client.language;' case='upper'
/>
```

prec="number"
The number of decimals in the stardate.

<help/>

Provided by module: *RXML 2 parser*

Gives help texts for tags. If given no arguments, it will list all available tags. By inserting `<help/>` in a page, a full index of the tags available in that particular Roxen WebServer will be presented. If a particular tag is missing from that index, it is not available at that moment. All tags are available through modules, hence that particular tags' module hasn't been added to the Roxen WebServer. Ask an administrator to add the module.

Attributes

for="tag"
Gives the help text for that tag.

```
<help for='roxen'/>
```

<modified/>

Provided by module: *RXML 2 tags*

Prints when or by whom a page was last modified, by default the current page.

Attributes

by
Print by whom the page was modified. Takes the same attributes as *<user>*. This attribute requires a userdatabase.

```
This page was last modified by <modified by=''
 realname=''/>.
```

date
Print the modification date. Takes all the date attributes in *<date>*.

```
This page was last modified <modified date=''
 case='lower' type='string'/>.
```

file="path"
Get information from this file rather than the current page.

realfile="path"
Get information from this file in the computers filesystem rather than Roxen Webserver's virtual filesystem.

<number/>

Provided by module: *RXML 2 parser*

Prints a number as a word.

Attributes

num="number"
Print this number.

```
<number num='4711'/>
```

language="langcodes"
The language to use.

```
Mitt favoritnummer %r <number num='11' language='sv
'/>.
```

```
Il mio numero preferito &egrave; <number num='15' l
anguage='it'/>.
```

type="{number, ordered, roman, memory}" (number)
Sets output format.

```
It was his <number num='15' type='ordered'/
> birthday yesterday.
```

```
Only <number num='274589226' type='memory'/
> left on the Internet.
```

```
Spock Garfield <number num='17' type='roman'/
> rests here.
```

<roxen/>

Provided by module: *RXML 2 tags*

Returns a nice Roxen logo.

Attributes

size="{small, medium, large}" (medium)
Defines the size of the image.

```
<roxen size='small'/> <roxen/> <roxen size='large'/
>
```

color="{black, white}" (white)
Defines the color of the image.

```
<roxen color='black'/>
```

alt="string" ("Powered by Roxen")
The image description.

border="number" (0)
The image border.

class="string"
This cascading style sheet (CSS) definition will be applied on the img element.

target="string"
Names a target frame for the link around the image.
All other attributes will be inherited by the generated img tag.

<user/>

Provided by module: *RXML 2 tags*

Prints information about the specified user. By default, the full name of the user and her e-mail address will be printed, with a mailto link and link to the home page of that user.
The `<user>` tag requires an authentication module to work.

Attributes

email
Only print the e-mail address of the user, with no link.

```
Email: <user name='foo' email=''/>
```

link
Include links. Only meaningful together with the realname or email attribute.

name
The login name of the user. If no other attributes are specified, the user's realname and email including links will be inserted.

```
<user name='foo'/>
```

nolink
Don't include the links.

nohomepage
Don't include homepage links.

realname
Only print the full name of the user, with no link.

```
<user name='foo' realname=''/>
```

Text Tags

Text tags are container tags that process their contents somehow. Examples are *<sort>*, that sorts its contents and *<tablify>*, that creates good looking tables from tab separated text files.

End of /roxen/2.1/creator/text/index.xml

<ai></ai>

Provided by module: *Indirect href*

Makes it possible to use a database of links. Each link is referred to by a symbolic name instead of the URL.

The database is updated through the configuration interface. The tag is available through the *Indirect href* module.

Attributes

name="string"

Which link to fetch from the database. There is a special case, *name='random'* that will choose a random link from the database.

```
<ai name='roxen'>Roxen Platform</ai>
```

<autoformat></autoformat>

Provided by module: *RXML 2 tags*

Replaces newlines with
:s'.

```
<autoformat>
It is almost like
using the pre tag.
</autoformat>
```

Attributes

p

Replace empty lines with <p>:s.

```
<autoformat p=''>
It is almost like

using the pre tag.
</autoformat>
```

nobr

Do not replace newlines with
:s.

class="string"

This cascading style sheet (CSS) definition will be applied on the p elements.

<case></case>

Provided by module: *RXML 2 parser*

Alters the case of the contents.

Attributes

case="{upper, lower, capitalize}"

Changes all characters to upper or lower case letters, or capitalizes the first letter in the content.

```
<case upper=''>upper</case>

<case lower=''>lower</case>

<case capitalize=''>capitalize</case>
```

<comment></comment>

Provided by module: *RXML 2 parser*

The enclosed text will be removed from the document. The difference from a normal SGML (HTML/XML) comment is that the text is removed from the document, and can not be seen even with *view source* in the browser.

Note that since this is a normal tag, it requires that the content is properly formatted. Therefore it's ofter better to use the <?comment ... ?> processing instruction tag to comment out arbitrary text (which doesn't contain '?>').

Just like any normal tag, the <comment> tag nests inside other <comment> tags. E.g:

```
<comment> a <comment> b </comment> c </comment>
```

Here 'c' is not output since the comment starter before 'a' matches the ender after 'c' and not the one before it.

Attributes

preparse

Parse and execute any RXML inside the comment tag. This is useful to do stuff without producing any output in the response.

<?comment ?>

Provided by module: *RXML 2 parser*

Processing instruction tag for comments. This tag is similar to the RXML <comment> tag but should be used when commenting arbitrary text that doesn't contain '?>'.

```
<?comment
  This comment will not be shown.
?>
```

<default></default>

Provided by module: *RXML 2 tags*

Makes it easier to give default values to "<select>" or "<checkbox>" form elements.

The <default> container tag is placed around the form element it should give a default value.

This tag is particularly useful in combination with database tags.

Attributes

value="string"
The value to set.

separator="string" (,)
If several values are to be selected, this is the string that separates them.

name="string"
Only affect form element with this name.

```
<default name='my-select' value='&form.preset;'>
   <select name='my-select'>
     <option value='1'>First</option>
     <option value='2'>Second</option>
     <option value='3'>Third</option>
   </select>
</default>
```

<doc></doc>

Provided by module: *RXML 2 tags*

Eases documentation by replacing "{", "}" and "&" with "<", ">" and "&". No attributes required.

Attributes

quote
Instead of replacing with "{" and "}", "<" and ">" is replaced with "<" and ">".

```
<doc quote=''>
<table>
 <tr>
    <td> First cell </td>
    <td> Second cell </td>
 </tr>
</table>
</doc>
```

pre
The result is encapsulated within a <pre> container.

```
<doc pre=''>
{table}
 {tr}
    {td} First cell {/td}
    {td} Second cell {/td}
 {/tr}
{/table}
</doc>
```

class="string"
This cascading style sheet (CSS) definition will be applied on the pre element.

<foldlist></foldlist>

Provided by module: *Folding lists*

This tag is used to build folding lists, that are like <dl> lists, but where each element can be unfolded. The tags used to build the lists elements are <ft> and <fd>.

Attributes

unfolded
Will make all the elements in the list unfolded by default.

<ft></ft>

Provided by module: *Folding lists*

This tag is used within the foldlist tag. The contents of this container, that is not within an fd, tag will be visible both when the element is folded and unfolded.

Attributes

folded
Will make this element folded by default. Overrides an unfolded attribute set in the foldlist tag.

unfolded
Will make this element unfolded by default.

<fd></fd>

Provided by module: *Folding lists*

The contents of this container will only be visible when the element it is written in is unfolded.

Attributes

```
<foldlist>
  <ft>
    Heading1
    <fd>Contents 1</fd>
  </ft>
  <ft>
    Heading2
    <fd>Contents 2</fd>
  </ft>
</foldlist>
```

<obox></obox>

Provided by module: *Outlined box*

This tag creates an outlined box.

Attributes

align="{left, right}"
Vertical alignment of the box.

bgcolor="color"
Color of the background and title label.

fixedleft="number"

Fixed length of line on the left side of the title. The unit is the approximate width of a character.

fixedright="number"
Fixed length of line on the right side of the title. The unit is the approximate width of a character.

left="number"
Length of the line on the left of the title.

outlinecolor="color"
Color of the outline.

outlinewidth="number"
Width, in pixels, of the outline.

right="number"
Length of the line on the right of the title.

spacing="number"
Width, in pixels, of the space in the box.

style="{caption, groupbox}"
Style of the box. Groupbox is default.

textcolor="color"
Color of the text inside the box.

title="string"
Sets the title of the obox.

titlecolor="color"
Color of the title text.

width="number"
Width, in pixels, of the box.
Note that the left and right attributes are constrained by the width argument. If the title is not specified in the argument list, you can put it in a `<title>` container in the obox contents.

```
<obox align='left' outlinewidth='5' outlinecolor='g
reen' width='200'>
<title>Sample box</title>

This is just a sample box.

</obox>
```

<random></random>

Provided by module: *RXML 2 tags*

Randomly chooses a message from its contents.

Attributes

separator="string"
The separator used to separate the messages, by default newline.

```
<random separator='#'>
Roxen#Pike#Foo#Bar#roxen.com
</random>
```

<replace></replace>

Provided by module: *RXML 2 tags*

Replaces strings in the contents with other strings.

Attributes

from="string"
String or list of strings that should be replaced.

to="string"
String or list of strings with the replacement strings. Default is the empty string.

separator="string" (,)
Defines what string should separate the strings in the from and to attributes.

type="{word, words}" (word)
Word means that a single string should be replaced. Words that from and to are lists.

<smallcaps></smallcaps>

Provided by module: *RXML 2 tags*

Prints the contents in smallcaps. If the size attribute is given, font tags will be used, otherwise big and small tags will be used.

```
<smallcaps>Roxen WebServer</smallcaps>
```

Attributes

space
Put a space between every character.

```
<smallcaps space=''>Roxen WebServer</smallcaps>
```

class="string"
Apply this cascading style sheet (CSS) style on all elements.

smallclass="string"
Apply this cascading style sheet (CSS) style on all small elements.

bigclass="string"
Apply this cascading style sheet (CSS) style on all big elements.

size="number"
Use font tags, and this number as big size.

small="number" (size-1)
Size of the small tags. Only applies when size is specified.

```
<smallcaps size='6' small='2'>Roxen WebServer</
smallcaps>
```

<sort></sort>

Provided by module: *RXML 2 tags*

Sorts the contents.

```
<sort>
 1
 Hello
 3
 World
 Are
```

```
 2
 We
 4
 Communicating?
</sort>
```

Attributes

separator="string"
Defines what the strings to be sorted are separated with. The sorted string will be separated by the string.

```
<sort separator='#'>
 1#Hello#3#World#Are#2#We#4#Communicating?
</sort>
```

reverse
Reversed order sort.

```
<sort reverse=''>
 1
 Hello
 3
 World
 Are
 2
 We
 4
 Communicating?
</sort>
```

<strlen></strlen>

Provided by module: *RXML 2 parser*

Returns the length of the contents.

```
There are <strlen>foo bar gazonk</
strlen> characters
 inside the tag.
```

<tablify></tablify>

Provided by module: *Tablify*

Transforms texts into tables. No attributes required.

Attributes

rowseparator="string" (newline)
Defines the rowseparator.

cellseparator="string" (tab)
Defines the cellseparator.

```
<tablify cellseparator=','>
Country, Population
Sweden, 8 911 296
Denmark, 5 356 845
Norway, 4 438 547
Iceland, 272 512
Finland, 5 158 372
</tablify>
```

border="number"
Defines the width of the border. Default is 2 in nice and nicer modes. Otherwise undefined.

cellspacing="number"
Defines the cellspacing attribute. Default is 0 in nice and nicer modes. Otherwise undefined.

cellpadding="number"
Defines the cellpadding attribute. Default is 4 in nice and nicer modes. Otherwise undefined.

interactive-sort
Makes it possible for the user to sort the table with respect to any column.

sortcol="number"
Defines which column to sort the table with respect to. The leftmost column is number 1. Negative value indicate reverse sort order.

min="number"
Indicates which of the input rows should be the first to be displayed. The first row is number 1.

max="number"
Indicates which of the input rows should be the last to be displayed.

negativecolor="color" (#ff0000)
The color of negative values in economic fields.

cellalign="{left, center, right}"
Defines how the cell contents should be align by default.

cellvalign="{top, middle, bottom}"
Defines how the cell contents should be verically aligned.

width="number"
Defines the width of the table.

nice
Add some extra layout to the table. All attributes below only applies in nice or nicer mode.

```
<tablify nice='' cellseparator=',' modulo='2'>
Country, Population
Sweden, 8 911 296
Denmark, 5 356 845
Norway, 4 438 547
Iceland, 272 512
Finland, 5 158 372
</tablify>
```

grid="number"
Draws a grid with the thickness given.

notitle
Don't add a title to each column.

bordercolor="color" (#000000)
The color of the border.

titlebgcolor="color" (#112266)
The background color of the title.

titlecolor="color" (#ffffff)
The color of the title.

modulo="number"
Defines how many rows in a row should have the same color.

oddbgcolor="color" (#ffffff)
The first background color.

evenbgcolor="color" (#ddeeff)
The second background color.

nicer
Add some extra extra layout to the table. All attributes below only applies in nicer mode. Nicer requires the gtext module.

noxml
Don't terminate the gifs with slashes.

font="text" (lucida)
Gtext font to write the column titles with.

scale="float" (0.36)
Size of the gtext font to write the column titles with.

textcolor="color" (#000000)
The color of the text. This will also work with economic fields in any mode.

size="number" (2)
The size of the table text.

font="string" (helvetica,arial)
The font of the table text.

```
<tablify nicer='' cellseparator=',' font='andover'
scale='1.0'>
Country, Population
Sweden, 8 911 296
Denmark, 5 356 845
Norway, 4 438 547
Iceland, 272 512
Finland, 5 158 372
</tablify>
```

<fields></fields>

Provided by module: *Tablify*

The container 'fields' may be used inside the tablify container to describe the type of contents the fields in a column has. Available fields are:

- text (default)
- left
- center
- right
- num
- int
- economic-int
- float
- economic-float

All fields except text overrides the cellvalign attribute.

Attributes

separator="string"
Defines the field type separator.
The fields types are separated by

1. The value given in the separator attribute to fields.
2. The value given in the cellseparator attribute to tablify.
3. Tab.

<trimlines></trimlines>

Provided by module: *RXML 2 tags*

Removes all empty lines from the contents.

```
<trimlines>


 Are


 We

 Communicating?


</trimlines>
```

<wash-html></wash-html>

Provided by module: *HTML washer*

This tag is mostly useful for turning user freetext input from a form into HTML intelligently, by turning sections of the text separated by more than one newline into paragraphs , filtering out or explicitly allowing some HTML tags in the input and creating <a>anchor-links out of potential www-addresses.

Attributes

keep-all
Leave all tags containing info intact. Overrides the value of keep-tags and keep-containers. This attribute is useful together with the attributes *unparagraphify* and *unlink*.

```
<wash-html keep-all=''>
  Some text, <i>italic</i>, <b>bold</
b>, <i><b>bold italic</b></i>.

  <hr>A little image:<img src='/internal-roxen-
next'>.
</wash-html>
```

keep-tags="list"
Comma-separated array of empty element <tags> not to filter. Quote all other empty element tags i.e. transform "<", ">" and "&" to "<", ">" and "&".

```
<wash-html keep-tags='hr'>
  Some text, <i>italic</i>, <b>bold</
b>, <i><b>bold italic</b></i>.

  <hr />A litle image:<img src='/internal-roxen-
next'>.
</wash-html>
```

keep-containers="list"
Comma-separated array of <container>...</> tags not to filter. Quote all other container tags e.i. transform "<", ">" and "&" to "<", ">" and "&".

```
<wash-html keep-containers='b'>
  Some text, <i>italic</i>, <b>bold</
b>, <i><b>bold italic</b></i>.
```

```
  <hr>A little image:<img src='/internal-roxen-
next'>.
</wash-html>
```

linkify

Makes text that looks like it might be useful as a link, e g http://www.roxen.com/, into a link. Text that starts with "http://", "https://", "ftp://", "www." or "http." will be converted to a clickable link with the text as the link label.

```
<wash-html linkify='' keep-containers='a' keep-
tags='br'>
  <a href="http://docs.roxen.com">Roxen docs</
a><br />
  http://pike.roxen.com<br />
  www.roxen.com
</wash-html>
```

unlinkify

Undo a linkify-conversion. Only the links that has the same label as address will be converted to plain text.

```
<wash-html unlinkify='' keep-tags='br' keep-
containers='a'>
  <a href="http://www.roxen.com">http://
www.roxen.com</a><br />
  <a href="http://www.roxen.com">Roxen IS</a>
</wash-html>
```

paragraphify

If more than one newline exists between two text elements, this attribute automatically makes the next text element into a paragraph.

```
<wash-html paragraphify=''>
A Paragraph

An other paragraph.
And some more text to the same paragraph.
</wash-html>
```

unparagraphify

Turn paragraph breaks into double newlines instead.

```
<pre><wash-html unparagraphify=''>
<p>A Paragraph<p>

<p>An other paragraph.
And some more text to the same paragraph.</p>
</wash-html></pre>
```

The `<pre>` is only used in the example for layout-purposes.

close-tags

Terminate all tags with an ending slash, making them XML-compliant.

Variable Tags

Variable tags are mostly used to create new variables or change a variable's contents in some manner. Some of the tags can also be used to create new tags or insert various contents of variables, files, etc into a webpage.

Variables are mostly created with tags like *<set>* and *<define>* and are reached through entities (scope.variable). *[More about variables]*

End of /roxen/2.1/creator/variable/index.xml

<append/>

Provided by module: *RXML 2 tags*

Appends a value to a variable. The variable attribute and one more is required.

Attributes

variable="string"
The name of the variable.

value="string"
The value the variable should have appended.

```
 <set variable='var.ris' value='Roxen'/>
 <append variable='var.ris' value=' Internet Softwa
re'/>
 &var.ris;
```

from="string"
The name of another variable that the value should be copied from.

<dec/>

Provided by module: *RXML 2 tags*

Subtracts 1 from a variable.

Attributes

variable="string"
The variable to be decremented.

value="number" (1)
The value to be subtracted.

<define></define>

Provided by module: *RXML 2 parser*

Defines variables, tags, containers and if-callers.

Attributes

name="name"
Sets the defaultvalue of various tagattributes. See *<tablist>* for examples.

variable="name"
Sets the value of the variable to the contents of the container.

tag="name"
Defines a tag that outputs the contents of the container.

```
<define tag="hi">Hello &_.name;!</define>
<hi name="Martin"/>
```

container="name"
Defines a container that outputs the contents of the container.

if="name"
Defines an if-caller that compares something with the contents of the container.

trimwhites
Trim all white space characters from the begining and the end of the contents.

preparse="preparse"
Sends the definition through the RXML parser when defining. (Without this attribute, the definition is only RXML parsed when it is invoked.)

The values of the attributes given to the defined tag are available in the scope created within the define tag.

&_.args; The full list of the attributes, and their arguments, given to the tag.

&_.contents; The containers contents.

&_.rest-args; A list of the attributes, and their arguments, given to the tag, excluding attributes with default values defined.

<attrib></attrib>

Provided by module: *RXML 2 parser*

When defining a tag or a container the container `<attrib>` can be used to define default values of the attributes that the tag/container can have.

Attributes

name="name"
The name of the attribute which default value is to be set.

<contents/>

Provided by module: *RXML 2 parser*

As the contents entity, but unquoted.

<inc/>

Provided by module: *RXML 2 tags*

Adds 1 to a variable.

Attributes

variable="string"
The variable to be incremented.

value="number" (1)
The value to be added.

<insert/>

Provided by module: *RXML 2 tags*

Inserts a file, variable or other object into a webpage.

Attributes

quote="{html, none}"
How the inserted data should be quoted. Default is "html", except for href and file where it's "none".

<insert/>

Provided by module: *RXML 2 tags*

Inserts the contents of a file. It reads files in a way similar to if you fetched the file with a browser, so the file may be parsed before it is inserted, depending on settings in the RXML parser. Most notably which kinds of files (extensions) that should be parsed. Since it reads files like a normal request, e.g. generated pages from location modules can be inserted. Put the tag <*eval*> around `<insert>` if the file should be parsed after it is inserted in the page. This enables RXML defines and scope variables to be set in the including file (as opposed to the included file). You can also configure the file system module so that files with a certain extension can not be downloaded, but still inserted into other documents.

Attributes

file="string"
The virtual path to the file to be inserted.

```
<eval><insert file='html_header.inc'/></eval>
```

<insert/>

Provided by module: *Additional RXML tags*

Inserts the contents at that URL. This function has to be enabled in the *Additional RXML tags* module in the Roxen WebServer configuration interface.

Attributes

href="string"
The URL to the page that should be inserted.

<insert/>

Provided by module: *RXML 2 tags*

Inserts a raw, unparsed file. The disadvantage with the realfile plugin compared to the file plugin is that the realfile plugin needs the inserted file to exist, and can't fetch files from e.g. an arbitrary location module.

Attributes

realfile="string"
The virtual path to the file to be inserted.

<insert/>

Provided by module: *RXML 2 tags*

Inserts a listing of all present scopes.

Attributes

scopes="{full, plain}"
Sets how the output should be formatted.

```
<insert scopes='plain'/>
```

<insert/>

Provided by module: *RXML 2 tags*

Inserts the value of a variable.

Attributes

variable="string"
The name of the variable.

scope="string"
The name of the scope, unless given in the variable attribute.

index="number"
If the value of the variable is an array, the element with this index number will be inserted. 1 is the first element. -1 is the last element.

split="string"
A string with which the variable value should be splitted into an array, so that the index attribute may be used.

<insert/>

Provided by module: *RXML 2 tags*

Inserts listing of all variables in a scope. Note that it is possible to create a scope with an infinite number of variables set. In this case the programme of that scope decides which variables that should be listable, i.e. this will not cause any problem except that all variables will not be listed. It is also possible to hide variables so that they are not listed with this tag.

Attributes

variables="{full, plain}"
Sets how the output should be formatted.

```
<pre>
<insert variables='full' scope='roxen'/>
</pre>
```

scope
The name of the scope that should be listed, if not the present scope.

<scope></scope>

Provided by module: *RXML 2 tags*

Creates a new variable scope. Variable changes inside the scope container will not affect variables in the rest of the page.

Attributes

extend="name" (form)
If set, all variables in the selected scope will be copied into the new scope. NOTE: if the source scope is "magic", as e.g. the roxen scope, the scope will not be copied, but rather linked and will behave as the original scope. It can be useful to create an alias or just for the convinience of refering to the scope as "_".

scope="name" (form)
The name of the new scope, besides "_".

<set/>

Provided by module: *RXML 2 tags*

Sets a variable.

Attributes

variable="string"
The name of the variable.

```
<set variable='var.foo' value='bar'/>
```

value="string"
The value the variable should have.

expr="string"
An expression whose evaluated value the variable should have.

from="string"
The name of another variable that the value should be copied from.

split="string"
The value will be splitted by this string into an array.
If none of the above attributes are specified, the variable is unset. If debug is currently on, more specific debug information is provided if the operation failed. See also: <*append*> and <*debug*>.

<sprintf></sprintf>

Provided by module: *Additional RXML tags*

Prints out variables with the formating functions availble in the Pike function sprintf. Refer to the Pike reference manual for a complete description.

Attributes

format="string"
The formatting string.

split="charater"
If used, the tag content will be splitted with the given string.

```
<sprintf format='#%02x%02x%02x' split=','>250,0,33<
/sprintf>
```

<sscanf></sscanf>

Provided by module: *Additional RXML tags*

Extract parts of a string and put them in other variables. Refer to the sscanf function in the Pike reference manual for a complete description.

Attributes

variables="list"
A comma separated list with the name of the variables that should be set.

```
<sscanf variables='form.year,var.month,var.day'
format='%4d%2d%2d'>19771003</sscanf>
&form.year;-&var.month;-&var.day;
```

scope="name"
The name of the fallback scope to be used when no scope is given.

```
<sscanf variables='form.year,month,day' scope='var'
 format='%4d%2d%2d'>19801228</sscanf>
&form.year;-&var.month;-&var.day;
```

return="name"
If used, the number of successfull variable 'extractions' will be available in the given variable.

<undefine/>

Provided by module: *RXML 2 parser*

Removes a definition made by the define container. One attribute is required.

Attributes

variable="name"
Undefines this variable.

```
<define variable='var.hepp'>hopp</define>
&var.hepp;
<undefine variable='var.hepp'/>
&var.hepp;
```

tag="name"
Undefines this tag.

container="name"
Undefines this container.

if="name"
Undefines this if-plugin.

<unset/>

Provided by module: *RXML 2 tags*

Unsets a variable, i.e. removes it.

Attributes

variable="string"
The name of the variable.

```
<set variable='var.jump' value='do it'/>
&var.jump;
<unset variable='var.jump'/>
&var.jump;
```

<use></use>

Provided by module: *RXML 2 parser*

Reads tags, container tags and defines from a package or file. Everything defined in the package is local for the page the package or file is used from.

Attributes

packageinfo
Show a all available packages.

package="name"
Reads all tags, container tags and defines from the given package. Packages are files located by default in *../rxml_packages/*.

file="path"
Reads all tags and container tags and defines from the file.
This file will be fetched just as if someone had tried to fetch it with an HTTP request. This makes it possible to use Pike script results and other dynamic documents. Note, however, that the results of the parsing are heavily cached for performance reasons. If you do not want this cache, use `<insert file='...' nocache=''>` instead.

info
Show a list of all defined tags/containers and if arguments in the file.

The `<use>` tag is much faster than <*insert*>, since the parsed definitions is cached.

<vform></vform>

Provided by module: *Verified form*

Creates a self verifying form. You can use all standard HTML-input widgets in this container as well.

```
<vform>
  <vinput name='mail' type='email'>&_.warning;</
vinput>
  <input type='hidden' name='user' value='&form.use
rid;' />
  <input type='submit' />
</vform>
<then><redirect to='other_page.html' /></then>
<else>No, this form is still not valid</else>
```

Attributes

hide-if-verified
Hides the form if it is verified

<clear/>

Provided by module: *Verified form*

Resets all the widgets to their initial values.

Attributes

value="string"
The text in the button.

<if vform-failed></if>

Provided by module: *Verified form*

If used with empty argument this will be true if the complete form is failed, otherwise only if the named field failed.

<if vform-verified></if>

Provided by module: *Verified form*

If used with empty arguemnt this will be true if the complete form so far is verified, otherwise only if the named field was successfully verified.

<reload/>

Provided by module: *Verified form*

Reload the page without variable checking.

Attributes

value="string"
The text on the button.

<verify-fail/>

Provided by module: *Verified form*

If put in a vform tag, the vform will always fail.This is useful e.g. if you put the verify-fail tag in an if tag.

<vinput></vinput>

Provided by module: *Verified form*

Creates a self verifying input widget.

Attributes

fail-if-failed="name"
The verification of this variable will always fail if the verification of a named variable also failed.

ignore-if-false
Don't verify if the false flag i set.

ignore-if-failed="name"
Don't verify if the verification of a named variable failed.

ignore-if-verified="name"
Don't verify if the verification of a named variable succeeded.

name="string"
The name of the variable that should be set.

value="anything"
The default value of this input widget.

scope="name" (vinput)
The name of the scope that is created in this tag.

trim
Trim the variable before verification.

type="{int, float, email, date, text, string}"
Set the type of the data that should be input, and hence what widget should be used and how the input should be verified.

minlength="number"
Verify that the variable has at least this many characters. Only available when using the type string or text.

maxlength="number"
Verify that the variable has at most this many characters. Only available when using the type string or text.

is="empty"
Verify that the variable is empty. Pretty useless... Only available when using the type string or text.

glob="pattern"
Verify that the variable match a certain glob pattern. Only available when using the type string or text.

regexp="pattern"
Verify that the variable match a certain regexp pattern. Only available when using the type string or text.

case="{upper, lower}"
Verify that the variable is all uppercased (or all lowercased). Only available when using the type string or text.

equal="string"
Verify that the variable is equal to a given string. Pretty useless... Only available when using the type string or text.

disable-domain-check
Only available when using the email type. When set the email domain will not be checked against a DNS to verify that it does exists.

mode="{before, after, complex}"
Select how to treat the contents of the vinput container. Before puts the contents before the input tag, and after puts it after, in the event of failed verification. If complex, use one tag `<verified>` for what should be outputted in the event of successful verification tag `<failed>` for every other event.

```
<table>
<tr><td>upper</
td><vinput name='a' case='upper' mode='complex'>
<verified><td bgcolor=green></verified>
<failed><td bgcolor=red></failed>&_.input:none;</
td>
</vinput></tr>
<tr><td><input type='submit' /></td></tr>
</table>
```

min="number"
Check that the number is at least the given. Only available when using the type int or float.

max="number"
Check that the number is at most the given. Only available when using the type int or float.

optional
Indicates that the variable should only be tested if it does contain something.

&_.input; The input tag, in complex mode.

&_.warning; May contain a explaination of why the test failed.

<failed></failed>

Provided by module: *Verified form*

The content will only be shown if the variable failed to verify, in complex mode.

<verified></verified>

Provided by module: *Verified form*

The content will only be shown if the variable was verfied, in complex mode.

HTTP Tags

HTTP tags are tags that somehow use or manipulate the URL or HTTP headers. Among other things they manipulate:

Prestate options
Prestate options are a way to present options in the URL, that will be persistent for a user over several pages. A prestate for the options *txt* and *en* would be stored as *http://www.roxen.com/(en,txt)/my.page* in the URL. If you use prestate options you must only use relative URLs in your links.

Cookies
Cookies are a way for a web site to store a small amount of information in the users browsers. It is a much better way than prestates to handle information that should be persistent for a user over several pages. Read more about cookies in RFC 2965.

Authentication
HTTP can be used to transmit a user name and a password through HTTP. Note that it isn't possible to use HTTP to end an authenticated session, since no HTTP-command exist that enables a user to logout from a server. To effectively be logged out from a server the user must kill her browser. Read more about HTTP authentication in RFC 2617.

Expire
It is possible to tell the browser, and any proxy on the way to it, how long it is to cache a page.

[More about HTTP]

End of /roxen/2.1/creator/http/index.xml

<aconf></aconf>

Provided by module: *RXML 2 tags*

Creates a link that can modify the persistent states in the cookie RoxenConfig. In practice it will add <keyword>/ right after the server, i.e. if you want to remove bacon and add egg the first "directory" in the path will be <-bacon,egg>. If the user follows this link the WebServer will understand how the RoxenConfig cookie should be modified and will send a new cookie along with a redirect to the given url, but with the first "directory" removed. The presence of a certain keyword in can be controlled with <*if config*>.

Attributes

href="uri"
Indicates which page should be linked to, if any other than the present one.

add="string"
The "cookie" or "cookies" that should be added, in a comma separated list.

drop="string"
The "cookie" or "cookies" that should be droped, in a comma separated list.

class="string"
This cascading style sheet (CSS) class definition will apply to the a-element.
All other attributes will be inherited by the generated a tag.

<apre></apre>

Provided by module: *RXML 2 tags*

Creates a link that can modify prestates. Prestates can be seen as valueless cookies or toggles that are easily modified by the user. The prestates are added to the URL. If you set the prestate "no-images" on "http://www.demolabs.com/index.html" the URL would be "http://www.demolabs.com/(no-images)/". Use <*if prestate*> to test for the presence of a prestate. `<apre>` works just like the `<a href='...'>` container, but if no "href" attribute is specified, the current page is used.

Attributes

href="uri"
Indicates which page should be linked to, if any other than the present one.

add="string"
The prestate or prestates that should be added, in a comma separated list.

drop="string"
The prestate or prestates that should be droped, in a comma separated list.

class="string"
This cascading style sheet (CSS) class definition will apply to the a-element.

<auth-required/>

Provided by module: *RXML 2 tags*

Adds an HTTP auth required header and return code (401), that will force the user to supply a login name and password. This tag is needed when using access control in RXML in order for the user to be prompted to login.

Attributes

realm="string"
The realm you are logging on to, i.e "Demolabs Intranet".

message="string"
Returns a message if a login failed or cancelled.

<expire-time/>

Provided by module: *RXML 2 tags*

Sets client cache expire time for the document by sending the HTTP header "Expires".

Attributes

now
Notify the client that the document expires now. The headers "Pragma: no-cache" and "Cache-Control: no-cache" will be sent, besides the "Expires" header.

years="number"
Add this number of years to the result.

months="number"
Add this number of months to the result.

weeks="number"
Add this number of weeks to the result.

days="number"
Add this number of days to the result.

hours="number"
Add this number of hours to the result.

beats="number"
Add this number of beats to the result.

minutes="number"
Add this number of minutes to the result.

seconds="number"
Add this number of seconds to the result.
It is not possible at the time to set the date beyond year 2038, since a unix time_t is used.

<header/>

Provided by module: *RXML 2 tags*

Adds a HTTP header to the page sent back to the client. For more information about HTTP headers please steer your browser to chapter 14, 'Header field definitions' in RFC 2616, available at Roxen Community.

Attributes

name="string"
The name of the header.

value="string"
The value of the header.

<killframe/>

Provided by module: *Kill frame*

This tag adds some JavaScript that will prevent others from putting the page in a frame. It can also strip any occurrences of index files, like *index.html*, from the end of the URL.

Attributes

killindex
Removes trailing index.html from the URL

<redirect/>

Provided by module: *RXML 2 tags*

Redirects the user to another page by sending a HTTP redirect header to the client.

Attributes

to="string"
Where the user should be sent to.

add="string"
The prestate or prestates that should be added, in a comma separated list.

drop="string"
The prestate or prestates that should be dropped, in a comma separated list.

text="string"
Sends a text string to the browser, that hints from where and why the page was redirected. Not all browsers will show this string. Only special clients like Telnet uses it.
Arguments prefixed with "add" or "drop" are treated as prestate toggles, which are added or removed, respectively, from the current set of prestates in the URL in the redirect header (see also *<apre>*). Note that this only works when the to=... URL is absolute, i.e. begins with a "/", otherwise these state toggles have no effect.

<remove-cookie/>

Provided by module: *RXML 2 tags*

Sets the expire-time of a cookie to a date that has already occured. This forces the browser to remove it. This tag won't remove the cookie, only set it to the empty string, or what is specified in the value attribute and change it's expire-time to a date that already has occured. This is unfortunutaly the only way as there is no command in HTTP for removing cookies. We have to give a hint to the browser and let it remove the cookie.

Attributes

name
Name of the cookie the browser should remove.

value="text"
Even though the cookie has been marked as expired some browsers will not remove the cookie until it is shut down. The text provided with this attribute will be the cookies intermediate value.
Note that removing a cookie won't take effect until the next page load.

<return/>

Provided by module: *RXML 2 tags*

Attributes

code
The HTTP status code to return (an integer).

text

The HTTP status message to set. If you don't provide one, a default message is provided for known HTTP status codes, e g "No such file or directory." for code 404.

<set-cookie/>

Provided by module: *RXML 2 tags*

Sets a cookie that will be stored by the user's browser. This is a simple and effective way of storing data that is local to the user. If no arguments specifying the time the cookie should survive is given to the tag, it will live until the end of the current browser session. Otherwise, the cookie will be persistent, and the next time the user visits the site, she will bring the cookie with her.

Attributes

name="string"
The name of the cookie.

seconds="number"
Add this number of seconds to the time the cookie is kept.

minutes="number"
Add this number of minutes to the time the cookie is kept.

hours="number"
Add this number of hours to the time the cookie is kept.

days="number"
Add this number of days to the time the cookie is kept.

weeks="number"
Add this number of weeks to the time the cookie is kept.

months="number"
Add this number of months to the time the cookie is kept.

years="number"
Add this number of years to the time the cookie is kept.

persistent
Keep the cookie for two years.

domain
The domain for which the cookie is valid.

value="string"
The value the cookie will be set to.

path="string"
The path in which the cookie should be available.

If persistent is specified, the cookie will be persistent until year 2038, otherwise, the specified delays are used, just as for *<expire-time>*.

Note that the change of a cookie will not take effect until the next page load.

<throttle/>

Provided by module: *Throttling control tags*

This tag determines a request's allocated bandwidth.

Attributes

not
Disables all and any throttling for the current request. Implies the 'final' arg.

add="rate"
Adds 'rate' bytes/sec to the current rate for the current request.

subtract="rate"
Subtracts 'rate' bytes/sec from the current rate for the current request.

multiply="float"
Multiplies this requests' bandwidth by 'float'.

divide="float"
Divides this requests' bandwidth by 'float'.

rate="value"
Sets this request's bandwidth to 'value'.

final
No subsequent modifications will be done to this request's bandwidth after the current one.

If Tags

If-tags make it possible to make dynamic pages that show different content based on conditions. Authenticated users can get confidential information and pages can be optimized for all browsers. They also makes it possible to program web applications in RXML, without using any programming language.

Learn how to use the if tags from scratch in our *If tags* tutorial.

End of /roxen/2.1/creator/if/index.xml

<else></else>

Provided by module: *RXML 2 parser*

Show the contents if the previous <*if*> tag didn't, or if there was a <*false*> tag above. This tag also detects if the page's truthvalue has been set to false. <*emit*> is an example of a tag that may change a page's truthvalue.

The result is undefined if there has been no <*if*>, <*true*> or <*false*> tag above.

<elseif></elseif>

Provided by module: *RXML 2 parser*

Same as the <*if*>, but it will only evaluate if the previous `<if>` returned false.

<false/>

Provided by module: *RXML 2 parser*

Internal tag used to set the return value of *If Tags*. It will ensure that the next <*else*> tag will show its contents. It can be useful if you are writing your own <*if*> lookalike tag.

<if></if>

Provided by module: *RXML 2 parser*

`<if>` is used to conditionally show its contents.The `<if>` tag is used to conditionally show its contents. <*else*> or <*elseif*> can be used to suggest alternative content.

It is possible to use glob patterns in almost all attributes, where * means match zero or more characters while ? matches one character. * Thus t*f?? will match trainfoo as well as * tfoo but not trainfork or tfo. It is not possible to use regexp's together with any of the if-plugins.

The if tag itself is useless without its plugins. Its main functionality is to provide a framework for the plugins.

It is mandatory to add a plugin as one attribute. The other attributes provided are and, or and not, used for combining plugins or logical negation.

```
  <if variable='var.foo > 0' and='' match='var.bar
is No'>
    ...
  </if>
```

```
  <if variable='var.foo > 0' not=''>
    &var.foo; is lesser than 0
  </if>
  <else>
    &var.foo; is greater than 0
  </else>
```

Operators valid in attribute expressions are: '=', '==', 'is', '!=', '<' and '>'.

The If plugins are sorted according to their function into five categories: Eval, Match, State, Utils and SiteBuilder.

The Eval category is the one corresponding to the regular tests made in programming languages, and perhaps the most used. They evaluate expressions containing variables, entities, strings etc and are a sort of multi-use plugins. All If-tag operators and global patterns are allowed.

```
  <set variable='var.x' value='6'/>
  <if variable='var.x > 5'>More than one hand</if>
```

The Match category contains plugins that match contents of something, e.g. an IP package header, with arguments given to the plugin as a string or a list of strings.

```
  Your domain <if ip='130.236.*'> is </if>
  <else> isn't </else> liu.se.
```

State plugins check which of the possible states something is in, e.g. if a flag is set or not, if something is supported or not, if something is defined or not etc.

```
   Your browser
  <if supports='javascript'>
   supports Javascript version &client.javascript;
  </if>
  <else>doesn't support Javascript</else>.
```

Utils are additonal plugins specialized for certain tests, e.g. date and time tests.

```
  <if time='1700' after=''>
    Are you still at work?
  </if>
  <elseif time='0900' before=''>
     Wow, you work early!
  </elseif>
  <else>
   Somewhere between 9 to 5.
  </else>
```

SiteBuilder plugins requires a Roxen Platform SiteBuilder installed to work. They are adding test capabilities to web pages contained in a SiteBuilder administrated site.

Attributes

not
Inverts the result (true->false, false->true).

or
If any criterion is met the result is true.

and
If all criterions are met the result is true. And is default.

<if accept></if>

Provided by module: *RXML 2 parser*

Returns true if the browser accepts certain content types as specified by it's Accept-header, for example image/jpeg or text/html. If browser states that it accepts */* that is not taken in to account as this is always untrue. Accept is a *Match* plugin.

Attributes

accept="type1[,type2,...]"

<if client></if>

Provided by module: *RXML 2 parser*

Compares the user agent string with a pattern. Client is an *Match* plugin.

Attributes

client=""

<if clientvar></if>

Provided by module: *RXML 2 parser*

Evaluates expressions with client specific values. Clientvar is an *Eval* plugin.

Attributes

clientvar="variable [is value]"
Choose which variable to evaluate against. Valid operators are '=', '==', 'is', '!=', '<' and '>'.

Available variables are:

<if config></if>

Provided by module: *RXML 2 parser*

Has the config been set by use of the *<aconf>* tag? Config is a *State* plugin.

Attributes

config="name"

<if cookie></if>

Provided by module: *RXML 2 parser*

Does the cookie exist and if a value is given, does it contain that value? Cookie is an *Eval* plugin.

Attributes

cookie="name[is value]"

<if date></if>

Provided by module: *RXML 2 parser*

Is the date yyyymmdd? The attributes before, after and inclusive modifies the behavior. Date is a *Utils* plugin.

Attributes

date="yyyymmdd"
Choose what date to test.

after
The date after todays date.

before
The date before todays date.

inclusive
Adds todays date to after and before.

```
<if date='19991231' before='' inclusive=''>
   - 19991231
</if>
<else>
  20000101 -
</else>
```

<if defined></if>

Provided by module: *RXML 2 parser*

Tests if a certain RXML define is defined by use of the *<define>* tag. Defined is a *State* plugin.

Attributes

defined="define"
Choose what define to test.

<if domain></if>

Provided by module: *RXML 2 parser*

Does the user's computer's DNS name match any of the patterns? Note that domain names are resolved asynchronously, and that the first time someone accesses a page, the domain name will probably not have been resolved. Domain is a *Match* plugin.

Attributes

domain="pattern1[,pattern2,...]"
Choose what pattern to test.

<if exists></if>

Provided by module: *RXML 2 parser*

Returns true if the file path exists. If path does not begin with /, it is assumed to be a URL relative to the directory containing the page with the `<if>`-statement. Exists is a *Utils* plugin.

Attributes

exists="path"
Choose what path to test.

<if expr></if>

Provided by module: *RXML 2 tags*

This plugin evaluates expressions. The arithmetic operators are "+, - and /". The last main operator is "%"(per cent). The allowed relationship operators are "<. >, ==, <= and >=".
All integers(characters 0 to 9) may be used together with "." to create floating point expressions.

```
Hexadecimal expression: (0xff / 5) + 3
```

To be able to evaluate hexadecimal expressions the characters "a to f and A to F" may be used.

```
Integer conversion: ((int) 3.14)
Floating point conversion: ((float) 100 / 7)
```

Conversion between int and float may be done through the operators "(int)" and "(float)". The operators "&"(bitwise and), "|"((pipe)bitwise or), "&&"(logical and) and "||"((double pipe)logical or) may also be used in expressions. To set prioritizations within expressions the characters "(and)" are included. General prioritization rules are:

1. (int), (float)
2. *, /, %
3. +, -
4. <, >, <=, >=
5. ==
6. &, |
7. &&, ||

```
Octal expression: 045
```

```
Calculator expression: 3.14e10 / 3
```

Expressions containing octal numbers may be used. It is also possible to evaluate calculator expressions.
Expr is an *Eval* plugin.

Attributes

expr="expression"
Choose what expression to test.

<if false></if>

Provided by module: *RXML 2 parser*

This will always be true if the truth value is set to be false. Equivalent with <*else*>. False is a *State* plugin.

Attributes

false
Show contents if truth value is true.

<if group></if>

Provided by module: *RXML 2 parser*

Checks if the current user is a member of the group according the groupfile. Group is a *Utils* plugin.

Attributes

group="name"
Choose what group to test.

groupfile="path"
Specify where the groupfile is located.

<if ip></if>

Provided by module: *RXML 2 parser*

Does the users computers IP address match any of the patterns? This plugin replaces the Host plugin of earlier RXML versions. Ip is a *Match* plugin.

Attributes

ip="pattern1[,pattern2,...]"
Choose what IP-adress pattern to test.

<if language></if>

Provided by module: *RXML 2 parser*

Does the client prefer one of the languages listed, as specified by the Accept-Language header? Language is a *Match* plugin.

Attributes

language="language1[,language2,...]"
Choose what language to test.

<if match></if>

Provided by module: *RXML 2 parser*

Evaluates patterns. More information can be found in the *If tags tutorial*. Match is an *Eval* plugin.

Attributes

match="pattern"
Choose what pattern to test. The pattern could be any expression. Note!: The pattern content is treated as strings:

```
<set variable='var.hepp' value='10' />

<if match='var.hepp is 10'>
 true
</if>
<else>
 false
</else>
```

This example shows how the plugin treats "var.hepp" and "10" as strings. Hence when evaluating a variable as part of the pattern, the entity associated with the variable should be used, i.e. var.hepp instead of var.hepp. A correct example would be:

```
<set variable='var.hepp' value='10' />

<if match='&var.hepp; is 10'>
 true
</if>
<else>
 false
</else>
```

Here, var.hepp is treated as an entity and parsed correctly, letting the plugin test the contents of the entity.

<if pragma></if>

Provided by module: *RXML 2 parser*

Compares the HTTP header pragma with a string. Pragma is a *State* plugin.

Attributes

pragma="string"
Choose what pragma to test.

```
<if pragma='no-
cache'>The page has been reloaded!</if>
<else>Reload this page!</else>
```

<if prestate></if>

Provided by module: *RXML 2 parser*

Are all of the specified prestate options present in the URL? Prestate is a *State* plugin.

Attributes

prestate="option1[,option2,...]"
Choose what prestate to test.

<if referrer></if>

Provided by module: *RXML 2 parser*

Does the referrer header match any of the patterns? Referrer is a *Match* plugin.

Attributes

referrer="pattern1[,pattern2,...]"
Choose what pattern to test.

<if sizeof></if>

Provided by module: *RXML 2 parser*

Compares the size of a variable with a number.

```
<set variable="var.x" value="hello"/>
<set variable="var.y" value=""/>
<if sizeof="var.x == 5">Five</if>
<if sizeof="var.y > 0">Nonempty</if>
```

<if supports></if>

Provided by module: *RXML 2 parser*

Does the browser support this feature? Supports is a *State* plugin.

Attributes

supports="feature"
Choose what supports feature to test.

The following features are supported:

<if time></if>

Provided by module: *RXML 2 parser*

Is the time hhmm? The attributes before, after and inclusive modifies the behavior. Time is a *Utils* plugin.

Attributes

time="hhmm"
Choose what time to test.

after
The time after present time.

before
The time before present time.

inclusive
Adds present time to after and before.

```
<if time='1200' before='' inclusive=''>
  ante meridiem
</if>
<else>
  post meridiem
</else>
```

<if true></if>

Provided by module: *RXML 2 parser*

This will always be true if the truth value is set to be true. Equivalent with <*then*>. True is a *State* plugin.

Attributes

true
Show contents if truth value is false.

<if user></if>

Provided by module: *RXML 2 parser*

Has the user been authenticated as one of these users? If any is given as argument, any authenticated user will do. User is a *Utils* plugin.

Attributes

user="{name1[,name2,...], any}"
Specify which users to test.

<if variable></if>

Provided by module: *RXML 2 parser*

Does the variable exist and, optionally, does it's content match the pattern? Variable is an *Eval* plugin.

Attributes

variable="name[is pattern]"
Choose variable to test. Valid operators are '=', '==', 'is', '!=', '<' and '>'.

<then></then>

Provided by module: *RXML 2 parser*

Shows its content if the truth value is true.

<true/>

Provided by module: *RXML 2 parser*

An internal tag used to set the return value of *If Tags*. It will ensure that the next <*else*> tag will not show its contents. It can be useful if you are writing your own <*if*> lookalike tag.

Flow Tags

Flow tags control and guide the flow of RXML code through conditions and error checking. Like the *If Tags* these tags can be used to create advanced dynamic pages. These tags are unlike the If tags as they don't take any attributes that process or find information, only attributes that affect the tag itself.

The tags <*throw*> and <*catch*> for instance, are used to help programmers more easily debug their code.

End of /roxen/2.1/creator/flow/index.xml

<catch></catch>

Provided by module: *RXML 2 tags*

Evaluates the RXML code, and, if nothing goes wrong, returns the parsed contents. If something does go wrong, the error message is returned instead. See also <*throw*>.

<cond></cond>

Provided by module: *RXML 2 parser*

This tag makes a boolean test on a specified list of cases. This tag is almost eqvivalent to the <*if*>/<*else*> combination. The main difference is that the `<default>` tag may be put whereever you want it within the `<cond>` tag. This will of course affect the order the content is parsed. The `<case>` tag is required.

<case></case>

Provided by module: *RXML 2 parser*

This tag takes the argument that is to be tested and if it's true, it's content is executed before exiting the `<cond>`. If the argument is false the content is skipped and the next `<case>` tag is parsed.

Attributes

```
<cond>
 <case variable='form.action = edit'>
   some database edit code
 </case>
 <case variable='form.action = delete'>
   some database delete code
 </case>
 <default>
   view something from the database
 </default>
</cond>
```

<default></default>

Provided by module: *RXML 2 parser*

The `<default>` tag is eqvivalent to the `<else>` tag in an `<if>` statement. The difference between the two is that the `<default>` may be put anywhere in the `<cond>` statement. This affects the parseorder of the statement. If the `<default>` tag is put first in the statement it will allways be executed, then the next `<case>` tag will be executed and perhaps add to the result the `<default>` performed.

<for></for>

Provided by module: *RXML 2 tags*

Makes it possible to create loops in RXML.

Attributes

from="number"
Initial value of the loop variable.

step="number"
How much to increment the variable per loop iteration. By default one.

to="number"
How much the loop variable should be incremented to.

variable="name"
Name of the loop variable.

<throw></throw>

Provided by module: *RXML 2 tags*

Throws a text to be caught by <*catch*>. Throws an exception, with the enclosed text as the error message. This tag has a close relation to <*catch*>. The RXML parsing will stop at the `<throw>` tag.

Graphics Tags

Roxen WebServer features a very advanced graphics engine. Together with output from a database or/and data from the live server the graphics engine can present information that makes a site feel alive. The graphics engine can be set up to automatically generate very nice-looking headers, diagrams, buttons, thumbnails and many other Web graphics :

Images
Through <*cimg*> images can be transformed and converted to any of the almost 20 image-formats available.

Business Graphics
Generate online reports dynamically with <*diagram*>, represented as line, bar, pie charts and many other diagrams.

Graphical Headers
Roxen WebServer supports many font-formats and makes it easy to import them. By combining fonts with images and graphical effects very eyecatching headers can be made. For instance, it is possible to set the ordinary HTML-tag `<h1>` to output a graphical header instead of ordinary boring headers. Graphical headers can be made with <*gtext*> and <*gh*>.

Good looking graphics are an important part of the layout of web pages. Creating graphics can also be very demanding of the site's hardware, especially if it involves transforming, converting and rendering of a lot of images or graphics. It is also important to not overdo the website's layout else people with slow connections will not be able to enjoy it since it will take forever to download it.

File Formats

Some of the tags take images as attributes. For example, to use as a background. The images can be of most formats including GIF, JPEG, PSD and PNG.

Color Attributes

Color attributes can be specified in one of the following ways:

Name
For example *black* or *darkred*.

#RGB value
The color is specified as a hexadecimal-digits, `#RRGGBB`. For example, *#ffdead* or *#00ff00*.

@HSV value
The color is specified with the syntax `@h,s,v` where h is the hue specified as degrees (0 to 359), s is the saturation specified as a percentage and v the value also specified as a percentage. For example, *@150,70,70*.

%CMYK value
The color is specified with the syntax `%c,m,y,k` where all the values are percentages. For example, *%10,20,30,40*.

End of /roxen/2.1/creator/graphics/index.xml

<cimg/>

Provided by module: *Image converter*

Manipulates and converts images between different image formats. Provides the tag `<cimg>` that makes it is possible to convert, resize, crop and in other ways transform images.

Attributes

src="uri"
The path to the indata file.

```
<cimg src='/internal-roxen-testimage'/>
```

data="imagedata"
Insert images from other sources, e.g. databases through entities or variables.

```
<emit source='sql' query='select imagedata from ima
ges where id=37'>
<cimg data='&sql.imagedata;'/>
</emit>
```

format="{gif, jpeg, png, avs, gmp, bd, hrz, ilbm, psx, pnm, ps, pvr, tga, tiff, wbf, wbmp, xbm, xpm}" (gif)
The format to encode the image to. The formats available are:

Acronym	Acronym interpretation
gif	Graphics Interchange Format (might be missing in your roxen)
jpeg	Joint Photography Expert Group image compression
png	Portable Networks Graphics
avs	Advanced Visual Systems Inc. image format
bmp	Windows BitMaP file
gd	Internal format used by libgd
hrz	HRZ is (was?) used for amatuer radio slow-scan TV.
ilbm	Interchangeable File Format: interleaved bitmap
pcx	Zsoft PCX file format (PC / DOS)
pnm	Portable AnyMap

Acronym	Acronym interpretation
ps	Adobe PostScript file
pvr	Pover VR (dreamcast image)
tga	TrueVision Targa (PC / DOS)
tiff	Tag Image File Format
wbf	WAP Bitmap File (WAP 1.0)
wbmp	Wireless Bitmap Format (WAP 1.1-)
xbm	XWindows Bitmap File
xpm	XWindows Pixmap File

```
<cimg src='/internal-roxen-testimage' format='png'/
>
```

```
<cimg src='/internal-roxen-testimage' format='gif'/
>
```

quant="number" (format dependant)
Default for gif is 32(+1 transparent), for most other formats (except black and white) is it unlimited.

```
<cimg src='/internal-roxen-testimage' quant='2'/>
```

dither="{none, random, floyd-steinberg}" (none)
Choose the dithering method.

Method	Meaning
none	No dithering is performed at all.
random	Random scatter dither. Not visually pleasing, but it is useful for very high resolution printing.
floyd-steinberg	Error diffusion dithering. Usually the best dithering method.

```
<cimg src='/internal-roxen-
testimage' dither='random' quant='10'/>
```

```
<cimg src='/internal-roxen-
testimage' dither='floyd-steinberg' quant='10'/>
```

true-alpha

If present, render a real alpha channel instead of on/off alpha. If the file format only supports on/off alpha, the alpha channel is dithered using a floyd-steinberg dither.

```
<cimg src='/internal-roxen-testimage' opaque-
value='20'/>
```

```
<cimg src='/internal-roxen-testimage' opaque-
value='20' true-alpha='1'/>
```

background-color="color" (taken from the page)
The color to render the image against.

```
<cimg src='/internal-roxen-testimage' background-
color='red' opaque-value='50'/>
```

opaque-value="percentage" (100)
The transparency value to use, 100 is fully opaque, and 0 is fully transparent.

cs-rgb-hsv="{0, 1}" (0)
Perform rgb to hsv colorspace conversion.

```
<cimg src='/internal-roxen-testimage' cs-rgb-
hsv='1'/>
```

gamma="number" (1.0)
Perform gamma adjustment.

```
<cimg src='/internal-roxen-testimage' gamma='0.5'/>
```

```
<cimg src='/internal-roxen-testimage' gamma='1.5'/>
```

cs-grey="{0, 1}" (0)
Perform rgb to greyscale colorspace conversion.

```
<cimg src='/internal-roxen-testimage' cs-grey='1'/>
```

cs-invert="{0, 1}" (0)
Invert all colors

```
<cimg src='/internal-roxen-testimage' cs-
invert='1'/>
```

cs-hsv-rgb="{0, 1}" (0)
Perform hsv to rgb colorspace conversion.

```
<cimg src='/internal-roxen-testimage' cs-hsv-
rgb='1'/>
```

rotate-cw="degree" (0)
Rotate the image clock-wise.

```
<cimg src='/internal-roxen-testimage' rotate-
cw='20'/>
```

rotate-ccw="degree" (0)
Rotate the image counter clock-wise.

rotate-unit="{rad, deg, ndeg, part}" (deg)

Select the unit to use while rotating.

Unit	Meaning
rad	Radians
deg	Degrees
ndeg	'New' degrees (400 for each full rotation)
part	0 - 1.0 (1.0 == full rotation)

mirror-x="{0, 1}" (0)
Mirror the image around the X-axis.

mirror-y="{0, 1}" (0)
Mirror the image around the Y-axis.

scale="fact" (1.0)
Scale fact times. (0.5 -> half size, 2.0 -> double size)

```
<cimg src='/internal-roxen-testimage' scale='0.5'/>
```

scale="x,y"
Scale to the exact size x,y. If either of X or Y is zero, the image is scaled to the specified width or hight, and the value that is zero is scaled in proportion to the other value.

```
<cimg src='/internal-roxen-
testimage' scale='20,50'/>
```

max-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect.

max-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect.

span-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect. If width is smaller than 'xsize', extend width to 'xsize' by filling the new space with current background color.

```
<cimg src='/internal-roxen-testimage' span-
width="350" background-color='white'/>
```

span-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect. If height is smaller than 'ysize', extend height to 'ysize' by filling the new space with current background color.

```
<cimg src='/internal-roxen-testimage' span-
height="350" background-color='white'/>
```

x-offset="pixels" (0)
Cut n pixels from the beginning of the X scale.

```
<cimg src='/internal-roxen-testimage' x-
offset='100'/>
```

y-offset="pixels" (0)
Cut n pixels from the beginning of the Y scale.

x-size="pixels" (whole image)
Keep n pixels from the beginning of the X scale.

```
<cimg src='/internal-roxen-testimage' x-size='100'/
>
```

y-size="pixels" (whole image)
Keep n pixels from the beginning of the Y scale.

crop="x0,y0-x1,y1" (whole image)
Crop the image by specifying the pixel coordinates.

```
<cimg src='/internal-roxen-testimage' crop='50,28-
150,92'/>
```

jpeg-quality="percentage" (75)
Set the quality on the output jpeg image.

```
<cimg src='/internal-roxen-
testimage' format='jpeg' jpeg-quality='30'/>
```

```
<cimg src='/internal-roxen-
testimage' format='jpeg' jpeg-quality='1'/>
```

jpeg-optimize="{0, 1}" (1)
If 0, do not generate optimal tables. Somewhat faster, but produces bigger files.

jpeg-progressive=="{0, 1}" (0)
Generate progressive jpeg images.

jpeg-smooth="0-100" (0)
Smooth the image while compressing it. This produces smaller files, but might undo the effects of dithering.

bmp-bpp="1,4,8,24" (24)
Force this number of bits per pixel for bmp images.

bmp-windows="{0, 1}" (1)
Windows or OS/2 mode, default is 1. (windows mode)

bmp-rle="{0, 1}" (0)
RLE 'compress' the BMP image.

gd-alpha_index="color" (0)
Color in the colormap to make transparent for GD-images with alpha channel.

pcx-raw="{1, 0}" (0)
If 1, do not RLE encode the PCX image.

pcx-dpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

pcx-xdpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

<cimg-url/>

Provided by module: *Image converter*

This tag generates an URI to the manipulated picture. `<cimg-url>` takes the same attributes as *<cimg>*, including the image cache attributes. The use for the tag is to insert image-URI's into various places, e.g. a submit-box.

Attributes

src="uri"

The path to the indata file.

```
<cimg-url src='/internal-roxen-testimage'/>
```

data="imagedata"

Insert images from other sources, e.g. databases through entities or variables.

```
<emit source='sql' query='select imagedata from ima
ges where id=37'>
<cimg-url data='&sql.imagedata;'/>
</emit>
```

format="{gif, jpeg, png, avs, gmp, bd, hrz, ilbm, psx, pnm, ps, pvr, tga, tiff, wbf, wbmp, xbm, xpm}" (gif)

The format to encode the image to. The formats available are:

Acronym	Acronym interpretation
gif	Graphics Interchange Format (might be missing in your roxen)
jpeg	Joint Photography Expert Group image compression
png	Portable Networks Graphics
avs	Advanced Visual Systems Inc. image format
bmp	Windows BitMaP file
gd	Internal format used by libgd
hrz	HRZ is (was?) used for amatuer radio slow-scan TV.
ilbm	Interchangeable File Format: interleaved bitmap
pcx	Zsoft PCX file format (PC / DOS)
pnm	Portable AnyMap
ps	Adobe PostScript file
pvr	Pover VR (dreamcast image)
tga	TrueVision Targa (PC / DOS)
tiff	Tag Image File Format
wbf	WAP Bitmap File (WAP 1.0)
wbmp	Wireless Bitmap Format (WAP 1.1-)
xbm	XWindows Bitmap File
xpm	XWindows Pixmap File

```
<cimg-url src='/internal-roxen-
testimage' format='png'/>
```

```
<cimg-url src='/internal-roxen-
testimage' format='gif'/>
```

quant="number" (format dependant)

The number of colors to quantizize the image to.

Default for gif is 32(+1 transparent), for most other formats (except black and white) is it unlimited.

```
<cimg-url src='/internal-roxen-
testimage' quant='2'/>
```

dither="{none, random, floyd-steinberg}" (none)

Choose the dithering method.

Method	Meaning
none	No dithering is performed at all.
random	Random scatter dither. Not visually pleasing, but it is useful for very high resolution printing.
floyd-steinberg	Error diffusion dithering. Usually the best dithering method.

```
<cimg-url src='/internal-roxen-
testimage' dither='random' quant='10'/>
```

```
<cimg-url src='/internal-roxen-
testimage' dither='floyd-steinberg' quant='10'/>
```

true-alpha

If present, render a real alpha channel instead of on/off alpha. If the file format only supports on/off alpha, the alpha channel is dithered using a floyd-steinberg dither.

```
<cimg-url src='/internal-roxen-testimage' opaque-
value='20'/>
```

```
<cimg-url src='/internal-roxen-testimage' opaque-
value='20' true-alpha='1'/>
```

background-color="color" (taken from the page)

The color to render the image against.

```
<cimg-url src='/internal-roxen-
testimage' background-color='red' opaque-
```

```
value='50'/>
```

opaque-value="percentage" (100)
The transparency value to use, 100 is fully opaque, and 0 is fully transparent.

cs-rgb-hsv="{0, 1}" (0)
Perform rgb to hsv colorspace conversion.

```
<cimg-url src='/internal-roxen-testimage' cs-rgb-
hsv='1'/>
```

gamma="number" (1.0)
Perform gamma adjustment.

```
<cimg-url src='/internal-roxen-
testimage' gamma='0.5'/>
```

```
<cimg-url src='/internal-roxen-
testimage' gamma='1.5'/>
```

cs-grey="{0, 1}" (0)
Perform rgb to greyscale colorspace conversion.

```
<cimg-url src='/internal-roxen-testimage' cs-
grey='1'/>
```

cs-invert="{0, 1}" (0)
Invert all colors

```
<cimg-url src='/internal-roxen-testimage' cs-
invert='1'/>
```

cs-hsv-rgb="{0, 1}" (0)
Perform hsv to rgb colorspace conversion.

```
<cimg-url src='/internal-roxen-testimage' cs-hsv-
rgb='1'/>
```

rotate-cw="degree" (0)
Rotate the image clock-wise.

```
<cimg-url src='/internal-roxen-testimage' rotate-
cw='20'/>
```

rotate-ccw="degree" (0)
Rotate the image counter clock-wise.

rotate-unit="{rad, deg, ndeg, part}" (deg)
Select the unit to use while rotating.

Unit	Meaning
rad	Radians
deg	Degrees
ndeg	'New' degrees (400 for each full rotation)
part	0 - 1.0 (1.0 == full rotation)

mirror-x="{0, 1}" (0)
Mirror the image around the X-axis.

mirror-y="{0, 1}" (0)
Mirror the image around the Y-axis.

scale="fact" (1.0)
Scale fact times. (0.5 -> half size, 2.0 -> double size)

```
<cimg-url src='/internal-roxen-
testimage' scale='0.5'/>
```

scale="x,y"
Scale to the exact size x,y. If either of X or Y is zero, the image is scaled to the specified width or hight, and the value that is zero is scaled in proportion to the other value.

```
<cimg-url src='/internal-roxen-
testimage' scale='20,50'/>
```

max-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect.

max-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect.

span-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect. If width is smaller than 'xsize', extend width to 'xsize' by filling the new space with current background color.

```
<cimg-url src='/internal-roxen-testimage' span-
width="350" background-color='white'/>
```

span-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect. If height is smaller than 'ysize', extend height to 'ysize' by filling the new space with current background color.

```
<cimg-url src='/internal-roxen-testimage' span-
height="350" background-color='white'/>
```

x-offset="pixels" (0)
Cut n pixels from the beginning of the X scale.

```
<cimg-url src='/internal-roxen-testimage' x-
offset='100'/>
```

y-offset="pixels" (0)
Cut n pixels from the beginning of the Y scale.

x-size="pixels" (whole image)
Keep n pixels from the beginning of the X scale.

```
<cimg-url src='/internal-roxen-testimage' x-
size='100'/>
```

y-size="pixels" (whole image)
Keep n pixels from the beginning of the Y scale.

crop="x0,y0-x1,y1" (whole image)
Crop the image by specifying the pixel coordinates.

```
<cimg-url src='/internal-roxen-
testimage' crop='50,28-150,92'/>
```

jpeg-quality="percentage" (75)
Set the quality on the output jpeg image.

```
<cimg-url src='/internal-roxen-
testimage' format='jpeg' jpeg-quality='30'/>
```

```
<cimg-url src='/internal-roxen-
testimage' format='jpeg' jpeg-quality='1'/>
```

jpeg-optimize="{0, 1}" (1)
If 0, do not generate optimal tables. Somewhat faster, but produces bigger files.

jpeg-progressive=="{0, 1}" (0)
Generate progressive jpeg images.

jpeg-smooth="0-100" (0)
Smooth the image while compressing it. This produces smaller files, but might undo the effects of dithering.

bmp-bpp="1,4,8,24" (24)
Force this number of bits per pixel for bmp images.

bmp-windows="{0, 1}" (1)
Windows or OS/2 mode, default is 1. (windows mode)

bmp-rle="{0, 1}" (0)
RLE 'compress' the BMP image.

gd-alpha_index="color" (0)
Color in the colormap to make transparent for GD-images with alpha channel.

pcx-raw="{1, 0}" (0)
If 1, do not RLE encode the PCX image.

pcx-dpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

pcx-xdpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

<colorscope></colorscope>

Provided by module: *HTML color wiretap*

Makes it possible to change the autodetected colors within the tag. Useful when out-of-order parsing occurs, e.g.

```
<define tag="hello">
  <colorscope bgcolor="red">
    <gtext>Hello</gtext>
  </colorscope>
</define>

<table><tr>
  <td bgcolor="red">
    <hello/>
  </td>
</tr></table>
```

Attributes

text="color"
Set the text color within the scope.

bgcolor="color"
Set the background color within the scope.

link="color"
Set the link color within the scope.

alink="color"
Set the active link color within the scope.

vlink="color"
Set the visited link color within the scope.

<configimage/>

Provided by module: *RXML 2 tags*

Returns one of the internal Roxen configuration images. The src attribute is required.

Attributes

src="string"
The name of the picture to show.

border="number" (0)
The image border when used as a link.

alt="string" (The src string)
The picture description.

class="string"
This cascading style sheet (CSS) class definition will be applied to the image.
All other attributes will be inherited by the generated img tag.

<diagram></diagram>

Provided by module: *Business graphics*

The `<diagram>` tag is used to draw pie, bar, or line charts as well as graphs. It is quite complex with six internal tags.

Attributes

3d="number"
Draws a pie-chart on top of a cylinder, takes the height in pixels of the cylinder as argument.

background="path"
Use an image as background. Valid types are gif-, jpeg- or pnm-images.

bgcolor="color"
Set the background color to use for anti-aliasing.

center="number"
Centers a pie chart around the *n*th slice.

eng
Write numbers in engineering fashion, i.e like 1.2M.

font="font"
Use this font. Can be overridden in the `<legend>`, `<xaxis>`, `<yaxis>` and `<names>` tags.

fontsize="number"
Height of the text in pixels.

height="number"
Height of the diagram in pixels. Will not have effect below 100.

horgrid
Draw a horizontal grid.

labelcolor="color"
Sets the color for the labels of the axis.

legendfontsize="number"
Height of the legend text. *fontsize* is used if this is undefined.

name="string"
Write a name at the top of the diagram.

namecolor="color" (textcolor)
Set the color of the name, by default set by the *textcolor* attribute.

namefont="font"
Set the font for the diagram name.

namesize="number"
Sets the height of the name, by default *fontsize*.

neng
As eng, but 0.1-1.0 is written as 0.xxx.

notrans
Make bgcolor opaque.

rotate="degree"
Rotate a pie chart this much.

textcolor
Set the color for all text.

tonedbox="color1,color2,color3,color4"
Create a background shading between the colors assigned to each of the four corners.

quant="number"
The number of colors that the result image should have. Default is 128 if tonedbox is used and 32 otherwise.

turn
Turn the diagram 90 degrees. Useful when printing large diagrams.

type="{sumbars, normsum, line, bar, pie, graph}"
The type of diagram. This attribute is required.

vertgrid
Draw vertical grid lines.

voidsep="string"
Change the string that means no such value, by default 'VOID'.

width="number"
Set the width of the diagram in pixels. Values below 100 will not take effect. This attribute is required.

xgridspace="number"
Set the space between two vertical grid lines. The unit is the same as for the data.

ygridspace
Set the space between two horizontal grid lines. The unit is the same as for the data.

Regular `<img>` arguments will be passed on to the generated `<img>` tag.

<colors></colors>

Provided by module: *Business graphics*

This tag sets the colors for different pie slices, bars or lines. The colors are presented to the tag in a tab separated list.

Attributes

separator="string"
Set the separator between colors, by default tab.

<data></data>

Provided by module: *Business graphics*

This tag contains the data the diagram is to visualize It is required that the data is presented to the tag in a tabular or newline separated form.

Attributes

form="{column, row}"
How to interpret the tabular data, by default row.

lineseparator="string"
Use the specified string as lineseparator instead of newline.

noparse
Do not parse the contents by the RXML parser, before data extraction is done.

separator="string"
Set the separator between elements, by default tab.

xnames="number"
If given, treat the first row or column as names for the data to come. If *xnames* is set to a number N, N lines or columns are used. The name will be written along the pie slice or under the bar.

xnamesvert
Write the *xnames* vertically.

<legend></legend>

Provided by module: *Business graphics*

A separate legend with description of the different pie slices, bars or lines.The titles are presented to the tag in a tab separated list.

Attributes

separator="string"
Set the separator between legends, by default tab.

<xaxis/>

Provided by module: *Business graphics*

Used for specifying the quantity and unit of the x-axis, as well as its scale, in a graph. The `<yaxis>` tag uses the same attributes.

Attributes

start="float"
Limit the start of the diagram at this value. If set to *min* the axis starts at the lowest value in the data.

stop="float"
Limit the end of the diagram at this value.

quantity="string"
Set the name of the quantity of this axis.

unit="string"
Set the name of the unit of this axis.

<xnames></xnames>

Provided by module: *Business graphics*

Separate tag that can be used to give names to put along the pie slices or under the bars. The datanames are presented to the tag as a tab separated list. This tag is useful when the diagram is dynamically created. The <ynames> tag uses the same attributes.

Attributes

separator="string" (tab)
Set the separator between names, by default tab.

orient="{vert, horiz}"
How to write names, vertically or horizontally.

<yaxis/>

Provided by module: *Business graphics*

Used for specifying the quantity and unit of the y-axis, as well as its scale, in a graph or line chart.Se the <xaxis> tag for a complete list of attributes.

<ynames></ynames>

Provided by module: *Business graphics*

Separate tag that can be used to give names to put along the pie slices or under the bars. The datanames are presented to the tag as a tab separated list. This tag is useful when the diagram is dynamically created. See the <xnames> tag for a complete list of attributes.

Some examples:

```
  <diagram type='pie' width='200' height='200'  nam
e='Population'
  tonedbox='lightblue,lightblue,white,white'>
    <data separator=','>5305048,5137269,4399993,886
5051</data>
    <legend separator=',>Denmark,Finland,Norway,Sw
eden</legend>
 </diagram>
```

```
 <diagram type='bar' width='200' height='250' name=
'Population'
 horgrid='' tonedbox='lightblue,lightblue,white,whi
te'>
  <data xnamesvert='' xnames='' separator=','>
    Denmark,Finland,Norway,Sweden
    5305048,5137269,4399993,8865051
  </data>
</diagram>
```

```
<diagram type='bar' width='200' height='250'
name='Age structure' horgrid=''
tonedbox='lightblue,lightblue,white,white'>
  <data xnamesvert='' xnames='' form='column'
  separator=','>
    Denmark,951175,3556339,797534
    Finland,966593,3424107,746569
    Norway,857952,2846030,696011
    Sweden,1654180,5660410,1550461
  </data>
  <legend separator=','>
    0-14,15-64,65-
  </legend>
</diagram>
```

```
<diagram type='sumbar' width='200' height='250'
name='Land Use' horgrid=''
tonedbox='lightblue,lightblue,white,white'>
  <data xnamesvert='' xnames='' form='column'
  separator=','>
    Denmark,27300,4200,10500
    Finland,24400,231800,48800
    Norway,9240,83160,215600
    Sweden,32880,279480,102750
  </data>
  <legend separator=','>
    Arable,Forests,Other
  </legend>
  <yaxis quantity='area'/>
  <yaxis unit='km^2'/>
</diagram>
```

```
<diagram type='normsumbar' width='200' height='250
'
name='Land Use' horgrid=''
tonedbox='lightblue,lightblue,white,white'>
  <data xnamesvert='' xnames='' form='column'
  separator=','>
    Denmark,27300,4200,10500
    Finland,24400,231800,48800
    Norway,9240,83160,215600
    Sweden,32880,279480,102750
  </data>
  <legend separator=','>
    Arable,Forests,Other
  </legend>
  <yaxis quantity='%'/>
</diagram>
```

```
<diagram type='line' width='200' height='250'
name='Exchange Rates' horgrid=''
tonedbox='lightblue,lightblue,white,white'>
  <data form='row' separator=','
  xnamesvert='' xnames=''>
    1992,1993,1994,1995,1996
    0.166,0.154,0.157,0.179,0.172
    0.223,0.175,0.191,0.229,0.218
    0.161,0.141,0.142,0.158,0.155
    0.172,0.128,0.130,0.149,0.140</data>
  <yaxis start='0.09' stop='0.25'/>
  <legend separator=','>
    Danish kroner (DKr),
    Markkaa (FMk),
    Norwegian kronor (NKr),
    Swedish kronor (SKr)
```

```
  </legend>
  <xaxis quantity='year'/>
  <yaxis quantity='US$'/>
</diagram>
```

<gbutton></gbutton>

Provided by module: *GButton*

Creates graphical buttons.

Attributes

pagebgcolor="color"

bgcolor="color"
Background color inside and outside button.

```
<gbutton bgcolor='lightblue'>Background</gbutton>
```

textcolor="color"
Button text color.

```
<gbutton textcolor='#ff6600'>Text</gbutton>
```

frame-image="path"
Use this XCF-image as a frame for the button. The image is required to have at least the following layers: background, mask and frame.

alt="string"
Alternative button and alt text.

href="uri"
Button URI.

textstyle="{normal, condensed}"
Set to *normal* or *condensed* to alter text style.

width="number"
Minimum button width.

align="{left, center, right}"
Set text alignment. There are some alignment restrictions: when text alignment is either *left* or *right*, icons must also be aligned *left* or *right*.

state="{enabled, disabled}"
Set to *enabled* or *disabled* to select button state.

icon-src="URI"
Fetch the icon from this URI.

icon-data=""
Inline icon data.

align-icon="{left, center-before, center-after, right}"
Set icon alignment.

left	Place icon on the left side of the text.
center-before	Center the icon before the text.Requires the *align='center'* attribute.
center-after	Center the icon after the text. Requires the *align='center'* attribute.
right	Place icon on the right side of the text.

```
<gbutton width='150' align-icon='center-before'
icon-src='internal-roxen-help'>Roxen 2.0</gbutton>
```

```
<gbutton width='150' align='center' align-
icon='center-after'
icon-src='internal-roxen-help'>Roxen 2.0</gbutton>
```

valign-icon="{above, middle, below}"
Set icon vertical alignment. Requires three horizontal guidelines in the frame image. If set to *above* the icon is placed between the first and second guidelines and the text between the second and third ones. If set to *below* the placement is reversed. Default value is *middle*.

font="fontname"

format="{gif, jpeg, png, avs, gmp, bd, hrz, ilbm, psx, pnm, ps, pvr, tga, tiff, wbf, wbmp, xbm, xpm}" (gif)
The format to encode the image to. The formats available are:

Acronym	Acronym interpretation
gif	Graphics Interchange Format (might be missing in your roxen)
jpeg	Joint Photography Expert Group image compression
png	Portable Networks Graphics
avs	Advanced Visual Systems Inc. image format
bmp	Windows BitMaP file
gd	Internal format used by libgd
hrz	HRZ is (was?) used for amatuer radio slow-scan TV.
ilbm	Interchangeable File Format: interleaved bitmap
pcx	Zsoft PCX file format (PC / DOS)
pnm	Portable AnyMap
ps	Adobe PostScript file
pvr	Pover VR (dreamcast image)

Acronym	Acronym interpretation
tga	TrueVision Targa (PC / DOS)
tiff	Tag Image File Format
wbf	WAP Bitmap File (WAP 1.0)
wbmp	Wireless Bitmap Format (WAP 1.1-)
xbm	XWindows Bitmap File
xpm	XWindows Pixmap File

quant="number" (format dependant)
The number of colors to quantizize the image to.
Default for gif is 32(+1 transparent), for most other formats (except black and white) is it unlimited.

dither="{none, random, floyd-steinberg}" (none)
Choose the dithering method.

Method	Meaning
none	No dithering is performed at all.
random	Random scatter dither. Not visually pleasing, but it is useful for very high resolution printing.
floyd-steinberg	Error diffusion dithering. Usually the best dithering method.

true-alpha
If present, render a real alpha channel instead of on/off alpha. If the file format only supports on/off alpha, the alpha channel is dithered using a floyd-steinberg dither.

background-color="color" (taken from the page)
The color to render the image against.

opaque-value="percentage" (100)
The transparency value to use, 100 is fully opaque, and 0 is fully transparent.

cs-rgb-hsv="{0, 1}" (0)
Perform rgb to hsv colorspace conversion.

gamma="number" (1.0)
Perform gamma adjustment.

cs-grey="{0, 1}" (0)
Perform rgb to greyscale colorspace conversion.

cs-invert="{0, 1}" (0)
Invert all colors

cs-hsv-rgb="{0, 1}" (0)
Perform hsv to rgb colorspace conversion.

rotate-cw="degree" (0)
Rotate the image clock-wise.

rotate-ccw="degree" (0)
Rotate the image counter clock-wise.

rotate-unit="{rad, deg, ndeg, part}" (deg)
Select the unit to use while rotating.

Unit	Meaning
rad	Radians
deg	Degrees
ndeg	'New' degrees (400 for each full rotation)
part	0 - 1.0 (1.0 == full rotation)

mirror-x="{0, 1}" (0)
Mirror the image around the X-axis.

mirror-y="{0, 1}" (0)
Mirror the image around the Y-axis.

scale="fact" (1.0)
Scale fact times. (0.5 -> half size, 2.0 -> double size)

scale="x,y"
Scale to the exact size x,y. If either of X or Y is zero, the image is scaled to the specified width or hight, and the value that is zero is scaled in proportion to the other value.

max-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect.

max-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect.

span-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect. If width is smaller than 'xsize', extend width to 'xsize' by filling the new space with current background color.

span-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect. If height is smaller than 'ysize', extend height to 'ysize' by filling the new space with current background color.

x-offset="pixels" (0)
Cut n pixels from the beginning of the X scale.

y-offset="pixels" (0)
Cut n pixels from the beginning of the Y scale.

x-size="pixels" (whole image)
Keep n pixels from the beginning of the X scale.

y-size="pixels" (whole image)
Keep n pixels from the beginning of the Y scale.

crop="x0,y0-x1,y1" (whole image)
Crop the image by specifying the pixel coordinates.

jpeg-quality="percentage" (75)
Set the quality on the output jpeg image.

jpeg-optimize="{0, 1}" (1)
If 0, do not generate optimal tables. Somewhat faster, but produces bigger files.

jpeg-progressive=="{0, 1}" (0)
Generate progressive jpeg images.

jpeg-smooth="0-100" (0)
Smooth the image while compressing it. This produces smaller files, but might undo the effects of dithering.

bmp-bpp="1,4,8,24" (24)
Force this number of bits per pixel for bmp images.

bmp-windows="{0, 1}" (1)
Windows or OS/2 mode, default is 1. (windows mode)

bmp-rle="{0, 1}" (0)
RLE 'compress' the BMP image.

gd-alpha_index="color" (0)
Color in the colormap to make transparent for GD-images with alpha channel.

pcx-raw="{1, 0}" (0)
If 1, do not RLE encode the PCX image.

pcx-dpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

pcx-xdpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

<gbutton-url></gbutton-url>

Provided by module: *GButton*

Generates an URI to the button. `<gbutton-url>` takes the same attributes as <*gbutton*> including the image cache attributes.

Attributes

pagebgcolor="color"

bgcolor="color"
Background color inside and outside button.

```
<gbutton bgcolor='lightblue'>Background</gbutton>
```

textcolor="color"
Button text color.

```
<gbutton textcolor='#ff6600'>Text</gbutton>
```

frame-image="path"
Use this XCF-image as a frame for the button. The image is required to have at least the following layers: background, mask and frame.

alt="string"
Alternative button and alt text.

href="uri"
Button URI.

textstyle="{normal, condensed}"
Set to *normal* or *condensed* to alter text style.

width="number"
Minimum button width.

align="{left, center, right}"
Set text alignment. There are some alignment restrictions: when text alignment is either *left* or *right*, icons must also be aligned *left* or *right*.

state="{enabled, disabled}"
Set to *enabled* or *disabled* to select button state.

icon-src="URI"
Fetch the icon from this URI.

icon-data=""
Inline icon data.

align-icon="{left, center-before, center-after, right}"
Set icon alignment.

left	Place icon on the left side of the text.
center-before	Center the icon before the text.Requires the *align='center'* attribute.
center-after	Center the icon after the text. Requires the *align='center'* attribute.
right	Place icon on the right side of the text.

```
<gbutton width='150' align-icon='center-before'
icon-src='internal-roxen-help'>Roxen 2.0</gbutton>
```

```
<gbutton width='150' align='center' align-
icon='center-after'
icon-src='internal-roxen-help'>Roxen 2.0</gbutton>
```

valign-icon="{above, middle, below}"
Set icon vertical alignment. Requires three horizontal guidelines in the frame image. If set to *above* the icon is placed between the first and second guidelines and the text between the second and third ones. If set to *below* the placement is reversed. Default value is *middle*.

font="fontname"

format="{gif, jpeg, png, avs, gmp, bd, hrz, ilbm, psx, pnm, ps, pvr, tga, tiff, wbf, wbmp, xbm, xpm}" (gif)
The format to encode the image to. The formats available are:

Acronym	Acronym interpretation
gif	Graphics Interchange Format (might be missing in your roxen)
jpeg	Joint Photography Expert Group image compression
png	Portable Networks Graphics
avs	Advanced Visual Systems Inc. image format
bmp	Windows BitMaP file

Acronym	Acronym interpretation
gd	Internal format used by libgd
hrz	HRZ is (was?) used for amatuer radio slow-scan TV.
ilbm	Interchangeable File Format: interleaved bitmap
pcx	Zsoft PCX file format (PC / DOS)
pnm	Portable AnyMap
ps	Adobe PostScript file
pvr	Pover VR (dreamcast image)
tga	TrueVision Targa (PC / DOS)
tiff	Tag Image File Format
wbf	WAP Bitmap File (WAP 1.0)
wbmp	Wireless Bitmap Format (WAP 1.1-)
xbm	XWindows Bitmap File
xpm	XWindows Pixmap File

quant="number" (format dependant)
The number of colors to quantizize the image to.
Default for gif is 32(+1 transparent), for most other formats (except black and white) is it unlimited.

dither="{none, random, floyd-steinberg}" (none)
Choose the dithering method.

Method	Meaning
none	No dithering is performed at all.
random	Random scatter dither. Not visually pleasing, but it is useful for very high resolution printing.
floyd-steinberg	Error diffusion dithering. Usually the best dithering method.

true-alpha
If present, render a real alpha channel instead of on/off alpha. If the file format only supports on/off alpha, the alpha channel is dithered using a floyd-steinberg dither.

background-color="color" (taken from the page)
The color to render the image against.

opaque-value="percentage" (100)
The transparency value to use, 100 is fully opaque, and 0 is fully transparent.

cs-rgb-hsv="{0, 1}" (0)
Perform rgb to hsv colorspace conversion.

gamma="number" (1.0)
Perform gamma adjustment.

cs-grey="{0, 1}" (0)
Perform rgb to greyscale colorspace conversion.

cs-invert="{0, 1}" (0)
Invert all colors

cs-hsv-rgb="{0, 1}" (0)
Perform hsv to rgb colorspace conversion.

rotate-cw="degree" (0)
Rotate the image clock-wise.

rotate-ccw="degree" (0)
Rotate the image counter clock-wise.

rotate-unit="{rad, deg, ndeg, part}" (deg)
Select the unit to use while rotating.

Unit	Meaning
rad	Radians
deg	Degrees
ndeg	'New' degrees (400 for each full rotation)
part	0 - 1.0 (1.0 == full rotation)

mirror-x="{0, 1}" (0)
Mirror the image around the X-axis.

mirror-y="{0, 1}" (0)
Mirror the image around the Y-axis.

scale="fact" (1.0)
Scale fact times. (0.5 -> half size, 2.0 -> double size)

scale="x,y"
Scale to the exact size x,y. If either of X or Y is zero, the image is scaled to the specified width or hight, and the value that is zero is scaled in proportion to the other value.

max-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect.

max-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect.

span-width="xsize"
If width is larger than 'xsize', scale width to 'xsize' while keeping aspect. If width is smaller than 'xsize', extend width to 'xsize' by filling the new space with current background color.

span-height="ysize"
If height is larger than 'ysize', scale height to 'ysize' while keeping aspect. If height is smaller than 'ysize', extend height to 'ysize' by filling the new space with current background color.

x-offset="pixels" (0)
Cut n pixels from the beginning of the X scale.

y-offset="pixels" (0)
Cut n pixels from the beginning of the Y scale.

x-size="pixels" (whole image)
Keep n pixels from the beginning of the X scale.

y-size="pixels" (whole image)
Keep n pixels from the beginning of the Y scale.

crop="x0,y0-x1,y1" (whole image)
Crop the image by specifying the pixel coordinates.

jpeg-quality="percentage" (75)
Set the quality on the output jpeg image.

jpeg-optimize="{0, 1}" (1)
If 0, do not generate optimal tables. Somewhat faster, but produces bigger files.

jpeg-progressive=="{0, 1}" (0)
Generate progressive jpeg images.

jpeg-smooth="0-100" (0)
Smooth the image while compressing it. This produces smaller files, but might undo the effects of dithering.

bmp-bpp="1,4,8,24" (24)
Force this number of bits per pixel for bmp images.

bmp-windows="{0, 1}" (1)
Windows or OS/2 mode, default is 1. (windows mode)

bmp-rle="{0, 1}" (0)
RLE 'compress' the BMP image.

gd-alpha_index="color" (0)
Color in the colormap to make transparent for GD-images with alpha channel.

pcx-raw="{1, 0}" (0)
If 1, do not RLE encode the PCX image.

pcx-dpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

pcx-xdpy="0-10000000.0" (75.0)
Resolution, in pixels per inch.

<gh></gh>

Provided by module: *Graphic text*

Creates a graphical header. `<gh>` takes the same attributes as <*gtext*>. `<gh>` comes in six flavors, from `<gh1>` through `<gh6>` and are the RXML counterpart to the HTML tags `<h1>` through `<h6>`.

Attributes

alpha="path"
Use the specified image as an alpha channel, together with the background attribute.

background="path"
Specifies the image to use as background.

tile
Tiles the background and foreground images if they are smaller than the actual image.

mirrortile
Tiles the background and foreground images around x-axis and y-axis for odd frames, creating seamless textures.

bevel="width"
Draws a bevel-box around the text.

bgcolor="color"
Sets the background color. Normally taken from the normal HTML tags in your document (Currently: body, table, tr or td).
If you set the background color, it is probably best to add the notrans attribute as well.

bgturbulence="frequency,color;frequency,color..."
Apply a turbulence effect on the background.

bold
Use a bold version of the font, if available. Can not be used together with the black or light attributes.

black
Use a black, or heavy, version of the font, if available. Can not be used together with the bold or light attributes.

light
Use a light version of the font, if available. Can not be used together with the bold or black attributes.

italic
Use an italic version of the font, if available.

bshadow="distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible, but you might have to add 'spacing'.

chisel
Make the text look like it has been cut into the background.

crop
Remove all white-space around the image.

encoding="string"
Choose with which charset the text is encoded with.

fadein="blur,steps,delay,initialdelay"
Generates an animated GIF file of a fade-in effect.

fgcolor="color"
Sets the text color.

font="string"
Selects which font to use. You can get a list of all available fonts by using the list fonts task in the administration interface, or by using the fonts emit plugin.

fontsize="number"
Selects which size of the font that should be used.

format="string"
Set the image format, e.g. "png".

fs
Apply floyd-steinberg dithering to the resulting image. Most of the time it is much better to increase the number of colors, instead of dithering the image, but sometimes when using very complex background images dithering is O.K.

ghost="dist,blur,color"
Apply a ghost effect. Cannot be used together with shadow or magic coloring.

glow="color"
Apply a 'glow' filter to the image. Quite a CPU eater. Looks much better on a dark background, where a real 'glow' effect can be achieved.

maxlen="number"
Sets the maximum length of the text that will be rendered into an image, by default 300.

move="x,y"
Moves the text relative to the upper left corner of the background image. This will not change the size of the image.

narrow
Use a narroe version of the font, if available.

notrans
Do not make the background transparent. Useful when making 'boxes' of color around the text.

nowhitespace
Removes all whitespaces before and after the real text.

opaque="percentage"
Sets the 'opaque' value of the color used to draw the text. Default is 100%. In the example below, notice how the text color mixes with the two background colors.

outline="color,extra-radius"
Draw an outline around the text. Quite useful when combined with textscale.

pressed
Inverts the direction of the bevel box, to make it look like a button that is pressed down. The magic modifier will do this automatically.

quant="number"
Quantifies the image with this number of colors. Using a lower number will decrease the image (file)size, but make the text look more 'edgy', and if you use complex backgrounds or image textures, more colors will be neded. At most 255 colors can be used, and less than 2 is quite useless. It is advisable to use powers of 2 to optimize the palette allocation.

rescale
Rescale the background to fill the whole image.

rotate="angle"
Rotates the image this number of degrees counter-clockwise.

scale="number"
Sets the scale of the image. Larger than 1.0 is enlargement.

scolor="color"
Use this color for the shadow. Used with the shadow attribute.

scroll="width,steps,delay"
Generate an animated GIF image of the text scrolling.

shadow="intensity,distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible,

size="width,height"
Set the size of the image.

spacing="number"
Add space around the text.

talign="{left, right, center}"
Adjust the alignment of the text.

textbelow="color"
Place the text centered in a box of the given color below the image area. Useful together with background to make captions for images.

textbox="opaque,color"
Draw a box with an opaque value below the text of the specified color.

textscale="color,color,color,color"
Apply a color filter to the text. The colors are, respectively, upper left, lower left, upper right and lower right. It is probably a good idea to increase the 'quant' value when using this argument.

texture="path"
Uses the specified images as a field texture.

verbatim
Allows the gtext parser to not be typographically correct.

xsize="number"
Sets the width.

xspacing="number"
Sets the horizontal spacing.

ypad="percentage"
Sets the padding beteen lines.

ysize="number"
Sets the height.

yspacing="number"
Sets the vertical spacing.

<gtext></gtext>

Provided by module: *Graphic text*

Creates an image with the tag content texts. It is possible to pass attributes, such as the target attribute, to the resulting tags by including them in the gtext tag.

Attributes

alt="string"
Sets the alt attribute of the generated `<img>` tag. By default the alt attribute will be set to the contents of the `<gtext>` tag.

```
  <gtext fgcolor="blue" alt="Hello!">Welcome!</
gtext>
```

border="width,color"
Draws a border around the text of the specified width and color.

```
<gtext fgcolor="blue" border="2,red">Red border</
gtext>
```

href="URL"
Link the image to the specified URL. The link color of the document will be used as the default foreground rather than the foreground color.

magic="message"
Used together with the href attribute to generate a JavaScript that will highlight the image when the mouse is moved over it. The message is shown in the browser's status bar.

```
<gtext href="http://
www.roxen.com" magic="Roxen">www.roxen.com</gtext>
```

magic-attribute="value"
Same as for any <gtext> attribute, except for the highlighted image.

```
<gtext fgcolor="blue" magic-
glow="yellow" magic="">Magic attribute</gtext>
```

noxml
Do not terminate the image tag with "/".

split
Make each word into a separate gif image. Useful if you are writing a large text, and word wrap at the edges of the display is desired.

```
 <gtext scale='0.4' split='split'>
 Useful if you are writing a large text, and word w
rap at the edges
 of the display is desired.
 </gtext>
```

This will allow the browser to word-wrap the text, but will disable certain attributes like *magic*. Note that the word wraping functionality of this example cannot be shown as the size of the browser window is determined by the largest example box.

```
<gtext scale="0.4" split="">Make each word..</
gtext>
```

submit
Creates a submit-button for forms. Does not work together with *split* or *magic* attributes.

alpha="path"
Use the specified image as an alpha channel, together with the background attribute.

background="path"
Specifies the image to use as background.

tile
Tiles the background and foreground images if they are smaller than the actual image.

mirrortile
Tiles the background and foreground images around x-axis and y-axis for odd frames, creating seamless textures.

bevel="width"
Draws a bevel-box around the text.

```
<gtext bevel="2">Ok</gtext>
```

bgcolor="color"
Sets the background color. Normally taken from the normal HTML tags in your document (Currently: body, table, tr or td).
If you set the background color, it is probably best to add the notrans attribute as well.

```
<gtext notrans="" bgcolor="pink">Pink</gtext>
<gtext notrans="" bgcolor="#ff0000">Red</gtext>
<gtext notrans="" bgcolor="%50,0,100,0">%50,0,100,0
</gtext>
```

bgturbulence="frequency,color;frequency,color..."
Apply a turbulence effect on the background.

bold
Use a bold version of the font, if available. Can not be used together with the black or light attributes.

```
<gtext font='lucida'>Aa3</gtext><br />
<gtext font='lucida' bold=''>Aa3</gtext><br />
<gtext font='lucida' italic=''>Aa3</gtext><br />
<gtext font='lucida' bold='' italic=''>Aa3</
gtext><br />
```

black
Use a black, or heavy, version of the font, if available. Can not be used together with the bold or light attributes.

light
Use a light version of the font, if available. Can not be used together with the bold or black attributes.

italic
Use an italic version of the font, if available.

bshadow="distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible, but you might have to add 'spacing'.

```
<gtext scale="0.8" fgcolor="#FF6600" bshadow="1">&l
t;gtext
bshadow=1&gt;</gtext><br />

<gtext scale="0.8" fgcolor="#FF6600" bshadow="2">&l
t;gtext
bshadow=2&gt;</gtext>
```

chisel
Make the text look like it has been cut into the background.

```
<gtext font="lucida" bold="" chisel="" talign="cent
er" tile=""
opaque="70" fgcolor="gold" bevel="2"
background="/internal-roxen-
squares"> Chisel opaque="70"</gtext>
```

crop
Remove all white-space around the image.

encoding="string"
Choose with which charset the text is encoded with.

fadein="blur,steps,delay,initialdelay"
Generates an animated GIF file of a fade-in effect.

fgcolor="color"
Sets the text color.

```
<gtext fgcolor="#0080FF">#0080FF</gtext>
```

font="string"
Selects which font to use. You can get a list of all available fonts by using the list fonts task in the administration interface, or by using the fonts emit plugin.

fontsize="number"
Selects which size of the font that should be used.

format="string"
Set the image format, e.g. "png".

fs
Apply floyd-steinberg dithering to the resulting image. Most of the time it is much better to increase the number of colors, instead of dithering the image, but sometimes when using very complex background images dithering is O.K.

ghost="dist,blur,color"
Apply a ghost effect. Cannot be used together with shadow or magic coloring.

```
<gtext spacing="2" crop="" ghost="1,1,red">ghost=1,
1,red</gtext>
<gtext spacing="2" crop="" ghost="1,3,blue">ghost=1
,3,blue</gtext>
<gtext spacing="2" crop="" bshadow="1" opaque="90"
ghost="-1,1,yellow">
ghost=-1,1,yellow opaque=90 bshadow=1</gtext>
```

glow="color"
Apply a 'glow' filter to the image. Quite a CPU eater. Looks much better on a dark background, where a real 'glow' effect can be achieved.

```
<gtext glow="red">&lt;gtext glow=red&gt;</gtext>
```

maxlen="number"
Sets the maximum length of the text that will be rendered into an image, by default 300.

move="x,y"
Moves the text relative to the upper left corner of the background image. This will not change the size of the image.

narrow
Use a narroe version of the font, if available.

notrans
Do not make the background transparent. Useful when making 'boxes' of color around the text.

```
<gtext bgcolor="red">&lt;gtext bgcolor=red&gt;</
gtext><br />
<gtext bgcolor="red" notrans="">&lt;gtext
bgcolor=red notrans&gt;</gtext>
```

nowhitespace
Removes all whitespaces before and after the real text.

opaque="percentage"
Sets the 'opaque' value of the color used to draw the text. Default is 100%. In the example below, notice how the text color mixes with the two background colors.

```
<gtext scale="0.6" textbox="100,pink,-
11" bgcolor="lightblue"
notrans="" opaque="40" fgcolor="black"
>&lt;Demonstration of opaque text&gt;</gtext>
```

outline="color,extra-radius"
Draw an outline around the text. Quite useful when combined with textscale.

```
<gtext xspacing="4" quant="128" textscale="red,red,
yellow,yellow"
outline="black,1"
>black, 2 pixels</gtext>
```

pressed
Inverts the direction of the bevel box, to make it look like a button that is pressed down. The magic modifier will do this automatically.

quant="number"
Quantifies the image with this number of colors. Using a lower number will decrease the image (file)size, but make the text look more 'edgy', and if you use complex backgrounds or image textures, more colors will be neded. At most 255 colors can be used, and less than 2 is quite useless. It is advisable to use powers of 2 to optimize the palette allocation.

```
<gtext quant="2">A</gtext>
<gtext quant="6">A</gtext>
<gtext quant="20">A</gtext>
<gtext quant="200">A</gtext>
```

rescale
Rescale the background to fill the whole image.

rotate="angle"
Rotates the image this number of degrees counter-clockwise.

scale="number"
Sets the scale of the image. Larger than 1.0 is enlargement.

```
<gtext scale="1.0">&lt;gtext scale=1.0&gt;</gtext>
<gtext scale="0.5">&lt;gtext scale=0.5&gt;</gtext>
```

scolor="color"
Use this color for the shadow. Used with the shadow attribute.

scroll="width,steps,delay"
Generate an animated GIF image of the text scrolling.

shadow="intensity,distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible,

```
<gtext scale="0.8" fgcolor="blue" shadow="40,0">&lt
;gtext
shadow=40,0&gt;</gtext><br />

<gtext scale="0.8" fgcolor="blue" shadow="40,2">&lt
;gtext
shadow=40,2&gt;</gtext><br />
```

size="width,height"
Set the size of the image.

spacing="number"
Add space around the text.

talign="{left, right, center}"
Adjust the alignment of the text.

textbelow="color"
Place the text centered in a box of the given color below the image area. Useful together with background to make captions for images.

```
<img src="/internal-roxen-roxen" />  
```

```
<gtext scale="0.5" background="/internal-roxen-
roxen"
textbelow="#c0c0c0">Roxen</gtext>
```

textbox="opaque,color"
Draw a box with an opaque value below the text of the specified color.

textscale="color,color,color,color"
Apply a color filter to the text. The colors are, respectively, upper left, lower left, upper right and lower right. It is probably a good idea to increase the 'quant' value when using this argument.

```
<gtext quant="128" textscale="blue,red,black,darkgr
een"
>Blue, red, black, darkgreen</gtext>
```

texture="path"
Uses the specified images as a field texture.

```
<gtext texture="/internal-roxen-ihfc">Ah</gtext>
```

verbatim
Allows the gtext parser to not be typographically correct.

xsize="number"
Sets the width.

xspacing="number"
Sets the horizontal spacing.

ypad="percentage"
Sets the padding beteen lines.

ysize="number"
Sets the height.

yspacing="number"
Sets the vertical spacing.

<gtext-id/>

Provided by module: *Graphic text*

Returns an internal URL to an image with the specified gtext attributes applied. Rendering a text image with the given arguments is accomplished by concatenating the text of your choice to the end of the URL returned by the tag, as in "`<gtext-id/>`Hello%20World!".

Be aware that this tag could be abused for denial of service attacks by malicious users swarming your server with requests for images of great length that the server would happily try to render for them. Hence this tag should only be used in environments where you trust all your users, e.g. Intranets.

In most cases the tag `<gtext-url>` solves this problem, but if you would like to generate new text images live without reloading some RXML page, you need this tag. An example application:

```
<define variable='var.id' preparse='' trimwhites=''
>
<gtext-id font='FranklinGothicDemi' fgcolor='blue'/
>
</define>

<img src='&var.id;Please type some text here:'
     alt='' name='banner' width='468' height='60'/>

<script language='javascript'>
var image = document.images.banner;
function alter_image(label)
{
  image.src = '&var.id:js;' + escape(label.value);
  label.focus();
  return false;
}
</script>

<form onsubmit='return alter_image(this.label);'>
<input type='text' size='40' name='label' />
</form>
```

Attributes

short
Returns a relative path to the image, i.e. a shorter one.

alpha="path"
Use the specified image as an alpha channel, together with the background attribute.

background="path"
Specifies the image to use as background.

tile
Tiles the background and foreground images if they are smaller than the actual image.

mirrortile
Tiles the background and foreground images around x-axis and y-axis for odd frames, creating seamless textures.

bevel="width"
Draws a bevel-box around the text.

```
<gtext bevel="2">Ok</gtext>
```

bgcolor="color"
Sets the background color. Normally taken from the normal HTML tags in your document (Currently: body, table, tr or td).
If you set the background color, it is probably best to add the notrans attribute as well.

```
<gtext notrans="" bgcolor="pink">Pink</gtext>
<gtext notrans="" bgcolor="#ff0000">Red</gtext>
<gtext notrans="" bgcolor="%50,0,100,0">%50,0,100,0
</gtext>
```

bgturbulence="frequency,color;frequency,color..."
Apply a turbulence effect on the background.

bold
Use a bold version of the font, if available. Can not be used together with the black or light attributes.

```
<gtext font='lucida'>Aa3</gtext><br />
<gtext font='lucida' bold=''>Aa3</gtext><br />
<gtext font='lucida' italic=''>Aa3</gtext><br />
<gtext font='lucida' bold='' italic=''>Aa3</
gtext><br />
```

black
Use a black, or heavy, version of the font, if available. Can not be used together with the bold or light attributes.

light

Use a light version of the font, if available. Can not be used together with the bold or black attributes.

italic
Use an italic version of the font, if available.

bshadow="distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible, but you might have to add 'spacing'.

```
<gtext scale="0.8" fgcolor="#FF6600" bshadow="1">&l
t;gtext
bshadow=1&gt;</gtext><br />

<gtext scale="0.8" fgcolor="#FF6600" bshadow="2">&l
t;gtext
bshadow=2&gt;</gtext>
```

chisel
Make the text look like it has been cut into the background.

```
<gtext font="lucida" bold="" chisel="" talign="cent
er" tile=""
opaque="70" fgcolor="gold" bevel="2"
background="/internal-roxen-
squares"> Chisel opaque="70"</gtext>
```

crop
Remove all white-space around the image.

encoding="string"
Choose with which charset the text is encoded with.

fadein="blur,steps,delay,initialdelay"
Generates an animated GIF file of a fade-in effect.

fgcolor="color"
Sets the text color.

```
<gtext fgcolor="#0080FF">#0080FF</gtext>
```

font="string"
Selects which font to use. You can get a list of all available fonts by using the list fonts task in the administration interface, or by using the fonts emit plugin.

fontsize="number"
Selects which size of the font that should be used.

format="string"
Set the image format, e.g. "png".

fs
Apply floyd-steinberg dithering to the resulting image. Most of the time it is much better to increase the number of colors, instead of dithering the image, but sometimes when using very complex background images dithering is O.K.

ghost="dist,blur,color"
Apply a ghost effect. Cannot be used together with shadow or magic coloring.

```
<gtext spacing="2" crop="" ghost="1,1,red">ghost=1,
1,red</gtext>
<gtext spacing="2" crop="" ghost="1,3,blue">ghost=1
,3,blue</gtext>
<gtext spacing="2" crop="" bshadow="1" opaque="90"
ghost="-1,1,yellow">
ghost=-1,1,yellow opaque=90 bshadow=1</gtext>
```

glow="color"
Apply a 'glow' filter to the image. Quite a CPU eater. Looks much better on a dark background, where a real 'glow' effect can be achieved.

```
<gtext glow="red">&lt;gtext glow=red&gt;</gtext>
```

maxlen="number"
Sets the maximum length of the text that will be rendered into an image, by default 300.

move="x,y"
Moves the text relative to the upper left corner of the background image. This will not change the size of the image.

narrow
Use a narroe version of the font, if available.

notrans
Do not make the background transparent. Useful when making 'boxes' of color around the text.

```
<gtext bgcolor="red">&lt;gtext bgcolor=red&gt;</
gtext><br />
<gtext bgcolor="red" notrans="">&lt;gtext
bgcolor=red notrans&gt;</gtext>
```

nowhitespace
Removes all whitespaces before and after the real text.

opaque="percentage"
Sets the 'opaque' value of the color used to draw the text. Default is 100%. In the example below, notice how the text color mixes with the two background colors.

```
<gtext scale="0.6" textbox="100,pink,-
11" bgcolor="lightblue"
notrans="" opaque="40" fgcolor="black"
>&lt;Demonstration of opaque text&gt;</gtext>
```

outline="color,extra-radius"
Draw an outline around the text. Quite useful when combined with textscale.

```
<gtext xspacing="4" quant="128" textscale="red,red,
yellow,yellow"
outline="black,1"
>black, 2 pixels</gtext>
```

pressed
Inverts the direction of the bevel box, to make it look like a button that is pressed down. The magic modifier will do this automatically.

quant="number"
Quantifies the image with this number of colors. Using a lower number will decrease the image (file)size, but make the text look more 'edgy', and if you use complex backgrounds or image textures, more colors will be neded. At most 255 colors can be used, and less than 2 is quite useless. It is advisable to use powers of 2 to optimize the palette allocation.

```
<gtext quant="2">A</gtext>
<gtext quant="6">A</gtext>
```

```
<gtext quant="20">A</gtext>
<gtext quant="200">A</gtext>
```

rescale
Rescale the background to fill the whole image.

rotate="angle"
Rotates the image this number of degrees counter-clockwise.

scale="number"
Sets the scale of the image. Larger than 1.0 is enlargement.

```
<gtext scale="1.0">&lt;gtext scale=1.0&gt;</gtext>
<gtext scale="0.5">&lt;gtext scale=0.5&gt;</gtext>
```

scolor="color"
Use this color for the shadow. Used with the shadow attribute.

scroll="width,steps,delay"
Generate an animated GIF image of the text scrolling.

shadow="intensity,distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible,

```
<gtext scale="0.8" fgcolor="blue" shadow="40,0">&lt
;gtext
shadow=40,0&gt;</gtext><br />

<gtext scale="0.8" fgcolor="blue" shadow="40,2">&lt
;gtext
shadow=40,2&gt;</gtext><br />
```

size="width,height"
Set the size of the image.

spacing="number"
Add space around the text.

talign="{left, right, center}"
Adjust the alignment of the text.

textbelow="color"
Place the text centered in a box of the given color below the image area. Useful together with background to make captions for images.

```
<img src="/internal-roxen-roxen" />  
<gtext scale="0.5" background="/internal-roxen-
roxen"
textbelow="#c0c0c0">Roxen</gtext>
```

textbox="opaque,color"
Draw a box with an opaque value below the text of the specified color.

textscale="color,color,color,color"
Apply a color filter to the text. The colors are, respectively, upper left, lower left, upper right and lower right. It is probably a good idea to increase the 'quant' value when using this argument.

```
<gtext quant="128" textscale="blue,red,black,darkgr
een"
>Blue, red, black, darkgreen</gtext>
```

texture="path"
Uses the specified images as a field texture.

```
<gtext texture="/internal-roxen-ihfc">Ah</gtext>
```

verbatim
Allows the gtext parser to not be typographically correct.

xsize="number"
Sets the width.

xspacing="number"
Sets the horizontal spacing.

ypad="percentage"
Sets the padding beteen lines.

ysize="number"
Sets the height.

yspacing="number"
Sets the vertical spacing.

<gtext-url></gtext-url>

Provided by module: *Graphic text*

Returns an internal URL to an image with the specified attributes applied.

Attributes

short
Returns a relative path to the image, i.e. a shorter one.

alpha="path"
Use the specified image as an alpha channel, together with the background attribute.

background="path"
Specifies the image to use as background.

tile
Tiles the background and foreground images if they are smaller than the actual image.

mirrortile
Tiles the background and foreground images around x-axis and y-axis for odd frames, creating seamless textures.

bevel="width"
Draws a bevel-box around the text.

bgcolor="color"
Sets the background color. Normally taken from the normal HTML tags in your document (Currently: body, table, tr or td).
If you set the background color, it is probably best to add the notrans attribute as well.

bgturbulence="frequency,color;frequency,color..."
Apply a turbulence effect on the background.

bold
Use a bold version of the font, if available. Can not be used together with the black or light attributes.

black
Use a black, or heavy, version of the font, if available. Can not be used together with the bold or light attributes.

light

Use a light version of the font, if available. Can not be used together with the bold or black attributes.

italic
Use an italic version of the font, if available.

bshadow="distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible, but you might have to add 'spacing'.

chisel
Make the text look like it has been cut into the background.

crop
Remove all white-space around the image.

encoding="string"
Choose with which charset the text is encoded with.

fadein="blur,steps,delay,initialdelay"
Generates an animated GIF file of a fade-in effect.

fgcolor="color"
Sets the text color.

font="string"
Selects which font to use. You can get a list of all available fonts by using the list fonts task in the administration interface, or by using the fonts emit plugin.

fontsize="number"
Selects which size of the font that should be used.

format="string"
Set the image format, e.g. "png".

fs
Apply floyd-steinberg dithering to the resulting image. Most of the time it is much better to increase the number of colors, instead of dithering the image, but sometimes when using very complex background images dithering is O.K.

ghost="dist,blur,color"
Apply a ghost effect. Cannot be used together with shadow or magic coloring.

glow="color"
Apply a 'glow' filter to the image. Quite a CPU eater. Looks much better on a dark background, where a real 'glow' effect can be achieved.

maxlen="number"
Sets the maximum length of the text that will be rendered into an image, by default 300.

move="x,y"
Moves the text relative to the upper left corner of the background image. This will not change the size of the image.

narrow
Use a narroe version of the font, if available.

notrans
Do not make the background transparent. Useful when making 'boxes' of color around the text.

nowhitespace
Removes all whitespaces before and after the real text.

opaque="percentage"
Sets the 'opaque' value of the color used to draw the text. Default is 100%. In the example below, notice how the text color mixes with the two background colors.

outline="color,extra-radius"
Draw an outline around the text. Quite useful when combined with textscale.

pressed
Inverts the direction of the bevel box, to make it look like a button that is pressed down. The magic modifier will do this automatically.

quant="number"
Quantifies the image with this number of colors. Using a lower number will decrease the image (file)size, but make the text look more 'edgy', and if you use complex backgrounds or image textures, more colors will be neded. At most 255 colors can be used, and less than 2 is quite useless. It is advisable to use powers of 2 to optimize the palette allocation.

rescale
Rescale the background to fill the whole image.

rotate="angle"
Rotates the image this number of degrees counter-clockwise.

scale="number"
Sets the scale of the image. Larger than 1.0 is enlargement.

scolor="color"
Use this color for the shadow. Used with the shadow attribute.

scroll="width,steps,delay"
Generate an animated GIF image of the text scrolling.

shadow="intensity,distance"
Draw a blured black drop-shadow behind the text. Using 0 as distance does not currently place the shadow directly below the text. Using negative values for distance is possible,

size="width,height"
Set the size of the image.

spacing="number"
Add space around the text.

talign="{left, right, center}"
Adjust the alignment of the text.

textbelow="color"
Place the text centered in a box of the given color below the image area. Useful together with background to make captions for images.

textbox="opaque,color"
Draw a box with an opaque value below the text of the specified color.

textscale="color,color,color,color"
Apply a color filter to the text. The colors are, respectively, upper left, lower left, upper right and lower right. It is probably a good idea to increase the 'quant' value when using this argument.

texture="path"
Uses the specified images as a field texture.

verbatim
Allows the gtext parser to not be typographically correct.

xsize="number"

Sets the width.

xspacing="number"
Sets the horizontal spacing.

ypad="percentage"
Sets the padding beteen lines.

ysize="number"
Sets the height.

yspacing="number"
Sets the vertical spacing.

<imgs/>

Provided by module: *RXML 2 tags*

Generates a image tag with proper dimensions.

Attributes

src="string"
The name of the file that should be shown.

alt="string"
Description of the image.
All other attributes will be inherited by the generated img tag.

<tablist></tablist>

Provided by module: *Tab list*

`<tablist>` produces graphical navigationtabs. For example, the Administration interface for Roxen WebServer uses tablists for better navigation.

The `<tablist>` tag is by design a wrapper for the *<gbutton>* tag, i.e. it inherits all `<gbutton>` attributes. Also, the `<tab>` tag is in turn a wrapper for `<tablist>` meaning that all attributes which may be given to `<tablist>` may also be used in `<tab>`. Those attributes given to `<tablist>` has a global effect on the tablist, while the same attributes given to a `<tab>` only will have a local effect, thus overriding the globally given attribute.

All contents inside `<tablist>` except for the `<tab>` tags will be discarded. `<taglist>` is used in this way to make it possible for tabs to look different when they are for instance first or last in the tablisting.

The *<define>* tag can be used to globally define the tablist *fgcolor* (foreground color). The define, *<define name="fgcolor">*, declared prior to the `<tablist>` tag, will be sent as an extra argument to `<gbutton>`.

Ett bra exempel h‰r

```
<tablist>
<tab selected='selected'>About</tab>
<tab>Info</tab>
<tab>Browse</tab>
</tablist>
```

Attributes

frame-image="" (/internal-roxen-tabframe)

A layered Photoshop (PSD) or Gimp (XCF) image which portrays the tab's appearance. Descriptions of the different layers follows below. If a `<define name="frame-image">`Image_path`</define>` definition is set that image will be the default value instead of `/internal-roxen-tabframe`.

selcolor="color" (white)
This attribute sets the backgroundcolor for the image. The effect of this attribute is only shown when the attribute "selected" has been set. If a `<define name="selcolor">`color-definition`</define>` definition is set that color will be the default value instead of `white`.

seltextcolor="color" (black)
This attribute sets the textcolor for the image. The effect of this attribute is only shown when the attribute "selected" has been set. If a `<define name="seltextcolor">`colordefinition`</define>` definition is set that color will be the default value instead of `black`. If this definition is not present, the attribute "textcolor", the definition "textcolor" and finally the color "black" will be tested.

dimcolor="color" (#003366)
This attribute sets the backgroundcolor for the image. The effect of this attribute is only shown when the attribute "selected" has *not* been set. If a `<define name="dim-color">`colordefinition`</define>` definition is set that color will be the default value instead of `#003366` .

textcolor="color" (white)
This attribute sets the textcolor for the image. The effect of this attribute is only shown when the attribute "selected" has *not* been set. If a `<define name="textcolor">`colordefinition`</define>` definition is set that color will be the default value instead of `white` .

<tab></tab>

Provided by module: *Tab list*

`<tab>` defines the layout and function for each and one of the tabs in the tablisting. `<tab>` inherits all attributes available to `<tablist>`, hence all attributes available to *<gbutton>* tag may be used with the `<tab>` tag. For instance, the attribute *href* is very useful when using `<tab>` and a part of *<gbutton>*. For more information about `<gbutton>` attributes, see its documentation.

The contents of the `<tab>` is the tabs text.

Below follows a listing of the attributes unique to the `<tab>` tag. Also, a listing of how imagelayers may be used is presented.

Attributes

selected=""
Using this attribute the layer "selected" in the image will be shown in the generated image. If this attribute has not been given the layer "unselected" will be shown in the generated image.

alt="text" (the tags content)
This attribute sets the alt-text for the tab. By default the alt-text is fetched from the content between the `<tab>...</tab>`.

These lists shows the function of the different image layers as well as how one layer from each group may be combined.

Layer Position
Position layers are the layername prefix.

first
A layer with this prefix is only shown for the *first*`<tab>` tag inside the `<tablist>` tag.

last
A layer with this prefix is only shown for the *last*`<tab>` tag inside the `<tablist>` tag.

Layer Focus
Focus layers are the middle part of the layername.

selected
This layer is only shown when the attribute *selected* has been set.

unselected
This layer is only shown when the attribute *selected* has *not* been set.

Layer Type
Type layers are the layername suffix.

[nothing, i.e. ""]
This layer is inserted above all layers in the image, closest to the viewer that is, if lower layers are considered to further in inside the monitor.

mask
This layer should be transparent where the tab is supposed to be transparent. The only thing that is retrieved from this layer is the mask; any graphical content here will be thrown away.

frame
The framelayer contains the various graphical elements fromwhich the frame around the button is built. This layer will always be run in "Multiply" mode, regardless of what mode it was previously set to. "Multiply" adjusts the framelayers brightness, i.e. Value ("V" in HSV), without affecting the color-components, i.e. Hue and Saturation ("HS" in HSV).

background
This layer will be put beneath the *frame* layer and the printed text.

left
This layer is put on the left side of the of the generated image, thus increasing the width of the images left side.

right
This layer is put on the right side of the of the generated image, thus increasing the width of the images right side.

above
This layer will be shown above the other parts of the generated image, thus increasing the height of the images top.

below
This layer will be shown below the other parts of the generated image, thus increasing the height of the images base.

The *Position*- and *Focus*-layers give instructions on *when* the layer is used while the *Type*-layers indicates its *function*.

Position	Focus	Type
""	""	""
first	selected	background
last	unselected	mask
		frame
		left
		right
		above
		below

These three layertypes can be combined into all possible permutations. The order in the name is always *Position Focus Type*, each type separated by a space. If one or two of the three layertypes is left out, the layer will be shown regardless the extra criterias that might be choosen. For instance, "selected frame" will be shown for the "first" and "last" tabs as well as for the ones in between the two, given that the tab has been marked as "selected".

None of these layers are strictly necessary, as long as there exists at least one layer of the type "background" or "frame". If all "mask"-layers are left out, the mask will primary be the framelayer and secondly the backgroundlayer, if the framelayer is not available.

Emit Tags

This chapter discusses the use of *<emit>* and its various sources.

End of /roxen/2.1/creator/output/index.xml

<emit></emit>

Provided by module: *RXML 2 parser*

`<emit>` is a generic tag used to fetch data from a provided source, loop over it and assign it to RXML variables accessible through entities.

Occasionally an `<emit>` operation fails to produce output. This might happen when `<emit>` can't find any matches or if the developer has made an error. When this happens the truthvalue of that page is set to *false*. By using *<else>* afterwards it's possible to detect when an `<emit>` operation fails.

Attributes

source="plugin"
The source from which the data should be fetched.

scope="name" (The emit source)
The name of the scope within the emit tag.

maxrows="number"
Limits the number of rows to this maximum.

skiprows="number"
Makes it possible to skip the first rows of the result. Negative numbers means to skip everything execept the last n rows.

rowinfo="variable"
The number of rows in the result, after it has been limited by maxrows and skiprows, will be put in this variable, if given.

do-once
Indicate that at least one loop should be made. All variables in the emit scope will be empty.

&_.counter; Gives the current number of loops inside the `<emit>` tag.

<emit source="cimg"></emit>

Provided by module: *Image converter*

Entitybased version of *<cimg>*. Takes the same attributes as `<cimg>`.

&_.data; Returns the imagedata given through other sources, like databases through entities.

&_.file-size; Returns the image's file size.

&_.src; Returns the path to the indata file.

&_.type; Returns the image's content-type.

&_.xsize; Returns the width of the image.

&_.ysize; Returns the height of the image.

<emit source="dir"></emit>

Provided by module: *Dir emit source*

This plugin is used to generate directory listings. The directory module must be added to use these entities. This plugin is only available in the directory template.

Attributes

directory="path"
Apply the listing to this directory.

options="(real-file,thumbnail,imagesize)"
Use these options to customize the directory listings. These argument have been made options due to them demanding a lot of raw computing power, since they involve image manipulation and other demanding tasks. These options can be combined.

real-file	Makes it possible to show the absolute location of the file including the filename from an 'outside Roxen' view.
thumbnail	Makes it possible to use image thumbnails in a directory listing. Note: Remember that some imageformats needs heavy computations to generate thumbnails. *tiff* for instance needs to unpack its image to be able to resolve the image's height and width.
imagesize	Makes it able to show the image's height and width in a directory listing. Note: Remember that some imageformats needs heavy computations to generate thumbnails. *tiff* for instance needs to unpack its image to be able to resolve the image's height and width.

thumbnail-size="number"
Sets the size of the thumbnail. Defaultsize is 60 pixels. The size is set in proportion to the image's longest side, e.g. if the height of the image is longer than it's width, then the thumbnail will be 60 pixels high. The shortest side will be shown in pro-

portion to the longest side. This attribute can only be used together with the *option="thumbnail"* attribute.

thumbnail-format="imageformat"
Set the output format for the thumbnail. Default is *png*. All imageformats that <*cimg*> handles can be used to produce thumbnails. This attribute can only be used together with the *option="thumbnail"* attribute.

strftime="strftime string" (%Y-%m-%d)
Format the date according to this string. Default is the isotime format (%Y-%m-%d), which will return (Year(four characters)-month(two characters)-day(two characters)), e.g. 2000-11-22. See the attribute *strftime* in <*date*> for a full listing of available formats.

glob="glob-pattern1[,glob-pattern2,...]"
Only show files matching the glob-pattern.

type="glob-pattern1[,glob-pattern2,...]"
Only show files which content-type matches the glob-pattern.

sort-order="{alpha, dwim, modified, size, type}" (dwim)
Sort the files and directories by this method.

alpha	Sort files and directories alphabetically.
dwim	Sort files and directories by "Do What I (want) Method". In many methods numeriacal sorts fail as the number '10' often appears before '2'. This method sorts numerical characters first then alphabetically, e.g. 1foo.html, 2foo.html, 10foo.html, foo1.html, foo2.html, foo10.html.
modified	Sort files by modification date.
size	Sort files by size.
type	Sort files by content-type.

sort-reversed
Reverse the sort order.

&_.atime; Returns the date when the file was last accessed.

&_.atime-iso; Returns the date when the file was last accessed. Uses isotime (%Y-%m-%d).

&_.atime-unix; Returns the date when the file was last accessed. Uses unixtime.

&_.dirname; Returns the directoryname.

&_.filename; Returns the filename.

&_.filesize; Returns a file's size in bytes. Directories get the size "-2".

&_.mode; Returns file permission rights represented binary, e.g. "r-xr-xr-x".

&_.mode-int; Returns file permission rights represented by integers. When encoded to binary this represents what is shown when using the Unix command "ls -l" or as shown using _.mode, e.g. "16749".

&_.mtime; Returns the date when the file was last modified.

&_.mtime-iso; Returns the date when the file was last modified. Uses isotime (%Y-%m-%d).

&_.mtime-unix; Returns the date when the file was last modified. Uses unixtime.

&_.name; Returns the name of the file or directory.

&_.path; Returns the path to the file or directory.

&_.size; Returns a file's size in kb(kilobytes).

&_.thumbnail; Returns the image associated with the file's content-type or directory. Only available when *option="thumbnail"* is used.

&_.type; Returns the file's content-type.

&_.type-img; Returns the internal Roxen name of the icon representating the directory or the file's content-type, e.g. internal-gopher-menu for a directory-folder or internal-gopher-text for a HTML-file.

&_.x-size; Returns the width of the image. Only available when *option="imagesize"* is used.

&_.y-size; Returns the height of the image. Only available when *option="imagesize"* is used.

<emit source="fonts"></emit>

Provided by module: *RXML 2 tags*

Prints available fonts. This plugin makes it easy to list all available fonts in Roxen WebServer.

Attributes

type="{ttf, all}"
Which font types to list. ttf means all true type fonts, whereas all means all available fonts.

&_.copyright; Font copyright notice. Only available for true type fonts.

&_.expose; The preferred list name. Only available for true type fonts.

&_.family; The font family name. Only available for true type fonts.

&_.format; The format of the font file, e.g. ttf.

&_.full; The full name of the font. Only available for true type fonts.

&_.name; Returns a font identification name.
This example will print all available ttf fonts in gtext-style.

&_.path; The location of the font file.

&_.postscript; The fonts postscript identification. Only available for true type fonts.

&_.style; Font style type. Only available for true type fonts.

&_.trademark; Font trademark notice. Only available for true type fonts.

&_.version; The version of the font. Only available for true type fonts.

<emit source="languages"></emit>

Provided by module: *Preferred Language Analyzer*

Outputs language descriptions.It will output information associated to languages, such as the name of the language in different languages. A list of languages that should be output can be provided with the langs attribute. If no such attribute is used a generated list of present languages will be used. If such a list could not be generated the list provided in the Preferred Language Analyzer module will be used.

Attributes

langs
Should contain comma separated list of language codes. The languages associated with these codes will be emitted in this order.

&_.code; The language code.

&_.confurl; A URL which makes the language the used one by altering the roxen cookie.

&_.en; The language name in english.

&_.local; The language name as written in the language itself.

&_.localized; The language name as written in the currently selected language.

&_.preurl; A URL which makes this language the used one by altering prestates.

<emit source="ldap"></emit>

Provided by module: *LDAP tags*

Use this source to search LDAP directory for information. The result will be available in variables named as the LDAP entries attribute.

Attributes

server="LDAP URL" (Server URL)
Connection LDAP URL. If omitted the *Default server URL* will be used.

password="user password"
User password for connection to the directory server. If omitted the default will be used.

<emit source="path"></emit>

Provided by module: *Directory Listings*

Prints paths. This plugin traverses over all directories in the path from the root up to the current one.

Attributes

trim="string"
Removes all of the remaining path after and including the specified string.

skip="number"
Skips the 'number' of slashes ('/') specified, with beginning from the root.

&_.name; Returns the name of the most recently traversed directory.

&_.path; Returns the path to the most recently traversed directory.

<emit source="sources"></emit>

Provided by module: *RXML 2 parser*

Provides a list of all available emit sources.

&_.source; The name of the source.

<emit source="sql"></emit>

Provided by module: *SQL tags*

Use this source to connect to and query SQL databases for information. The result will be available in variables named as the SQL columns.

Attributes

host="database"
Which database to connect to, usually a symbolic name set in the *SQL Databases* module. If omitted the default database will be used.

query="SQL statement"
The actual SQL-statement.

<emit source="values"></emit>

Provided by module: *RXML 2 parser*

Splits the string provided in the values attribute and outputs the parts in a loop. The value in the values attribute may also be an array or mapping.

Attributes

values="string, mapping or array"
An array, mapping or a string to be splitted into an array.

split="string" (NULL)
The string the values string is splitted with.

from-scope="name"
Create a mapping out of a scope and give it as indata to the emit.

&_.index; The index of this mapping entry, if input was a mapping

&_.value; The value of one part of the splitted string

Database Tags

The database tags interact with SQL databases or LDAP directories. They can input/output information used in creating everything from simple tables or forms, interactive graphical reports to complete web applications.

The LDAP directory tag <*ldap*> interacts with LDAP directories as well as LDAP accessible directories like Novell NDS or Microsoft Active Directory.

The database tags are almost always combined with other RXML tags. Together with for instance <*vform*>, the <*sqlquery*> tag can be used to put data collected from the form into a database.

<*sqltable*> together with <*diagram*> provides real-time diagrams and with <*tablify*> nice looking tables can be dynamically generated.

The <*emit*> plugins <*emit sql*> or <*emit ldap*> can also be used to search and list information from a SQl database or LDAP directory.

Each database tag needs to know which database it should connect to. This is specified by the *host*-attribute which usually is a symbolic name for the database that the administrator has configured in the *SQL Databases* module. It is also possible to specify the database host, user and password directly in the tags, but this is not recommended.

End of /roxen/2.1/creator/database/index.xml

<ldap/>

Provided by module: *LDAP tags*

Executes a LDAP operation, but doesn't do anything with the result.The `<ldap>` tag is mostly used for LDAP operation that change the contents of the directory, for example *add* or *modify*.

Attributes

server="LDAP URL" (Server URL)
Connection LDAP URL. If omitted the *Default server URL* will be used.

password="password"
User password for connection to the directory server. If omitted the default will be used.

dn="distinguished name"
Distinguished name of object.

op="add,delete,modify,replace"
The actual LDAP operation.
Note that *op='modify'* will change only the attributes given by the *attr* attribute.

attr="attribute_name1:[(attribute_value1[,...])][,attribute_name2,...]"
The actual values of attributes.
for example:

```
(sn:'Zappa'),(mail:'hello@nowhere.org','athell@pandemonium.com')
```

parser
If specified, the query will be parsed by the RXML parser. This is useful if the operation is to be built dynamically.

<sqlquery/>

Provided by module: *SQL tags*

Executes an SQL query, but doesn't do anything with the result. This is mostly used for SQL queries that change the contents of the database, for example INSERT or UPDATE.

Attributes

host="database"
Which database to connect to, usually a symbolic name set in the *SQL Databases* module. If omitted the default database will be used.

query="SQL statement"
The actual SQL-statement.

parse
If specified, the query will be parsed by the RXML parser. Useful if you wish to dynamically build the query.

mysql-insert-id="form-variable"
Set form-variable to the insert id used by Mysql for auto-incrementing columns. Note: This is only available with Mysql.

<sqltable/>

Provided by module: *SQL tags*

Creates an HTML or ASCII table from the results of an SQL query.

Attributes

ascii
Create an ASCII table rather than a HTML table. Useful for interacting with <*diagram*> and <*tablify*>.

host="database"
Which database to connect to, usually a symbolic name set in the *SQL Databases* module. If omitted the default database will be used.

query="SQL statement"
The actual SQL-statement.

parse
If specified, the query will be parsed by the RXML parser. Useful if you wish to dynamically build the query.

SSI Tags

SSI, Server Side Includes, are similar to RXML tags and have the advantage of being a standard supported by many web servers. It is thus possible to write pages using SSI that are portable to other web servers.

The downside of SSI is that it is in no way as flexible or powerful as RXML. The tags are placed within HTML comments, which makes it impossible to combine different SSI tags. However, it is possible to combine SSI tags with regular RXML tags.

Roxen WebServer does not presently implement all the SSI functionality that Apache supports.

End of /roxen/2.1/creator/ssi/index.xml

<!--#config -->

Provided by module: *SSI support*

The config command is used to configure how things should be printed.

Attributes

errmsg="string"
Where msg is a message that is sent back to the client if an error occurs while parsing the SSI-tag.

sizefmt="{bytes, abbrev}"
The value sets the format to be used when displaying the size of a file. Bytes gives a count in bytes while abbrev gives a count in KB or MB, as appropriate.

timefmt="value"
The value is a string to be used when displaying SSI date output.

<!--#echo -->

Provided by module: *SSI support*

Prints a variable from the server or request.

Some of the most useful ones are "http referrer" (the page which contained the link to the current page), "last modified" (date of the file for this document), "remote user" (name of the user), and "remote addr" (IP number of the user/client machine).

Note that these variables are SSI-related. You cannot access them as RXML variables, nor use this tag to print RXML variables.

Attributes

var="sizefmt"
Print format for file sizes.

var="document name"
Name of the current document (= page). RXML counterpart: page.self.

```
<!--#echo var="document name" -->
```

var="document uri"
URI (URL) to the current page. RXML counterpart: page.url.

```
<!--#echo var="document uri" -->
```

var="date local"
Time and date, in current time zone. RXML counterpart: <date strftime="%c"/>

var="date gmt"
Time and date, GMT time zone. RXML counterpart: <date timezone="GMT" strftime="%c"/>

var="last modified"
Last time this document was modified. RXML counterpart: `<modified/>`.

var="server software"
The web server software. RXML counterpart: roxen.version.

var="server name"
The web server name. RXML counterpart: roxen.domain.

var="remote host"
Name of client machine. RXML counterpart: client.host.

var="remote addr"
Numeric IP address of client machine. RXML counterpart: client.ip.

var="auth type"
Authentication type (typically Basic).

var="remote user"
Client user name.

var="http referrer"
URL of the referring page. RXML counterpart: client.referrer.

var="gateway interface"
Answears "CGI/1.1".

var="http cookie"
A list of the set cookies.

```
<!--#echo var="http cookie" -->
```

var="cookie"
A list of the set cookies. Same as http cookie.

var="http accept"
A list of the http accept formats.

```
<!--#echo var="http accept" -->
```

var="http user agent"
The user agent string. RXML counterpart: client.Fullname.

var="path translated"
Translated path.

var="query string unescaped"

The query string.

var="request method"
Request method (typically GET).

var="server protocol"
Protocol used for request.

var="server port"
Server's port number.

<!--#exec -->

Provided by module: *SSI support*

Executes a CGI script or shell command. This command has security implications and therefore, might not be available on all web sites.

Attributes

cgi="URL"
Path to the CGI script URL encoded. That is, a character can be quoted by % followed by its hex value. The CGI script is given the PATH_INFO and QUERY_STRING of the original request from the client. The variables available in `<!--#echo>` will be available to the script in addition to the standard CGI environment. If the script returns a Location header, then this will be translated into an HTML anchor.

cmd="path"
The server will execute the command using /bin/sh. The variables available in `<!--#echo>` will be available to the script.

<!--#flastmod -->

Provided by module: *SSI support*

This tag prints the last modification date of the specified file, subject to timefmt format specification used in the *<!--#config>* SSI tag.

Attributes

file="path"
Path to the file.

virtual="URL"
Path to the file URL encoded. That is, a character can be quoted by % followed by its hex value.

<!--#fsize -->

Provided by module: *SSI support*

Prints the size of the specified file, subject to the sizefmt format specification used in the `<!--#config>` SSI tag.

Attributes

file="path"
Path to the file.

virtual="URL"
Path to the file URL encoded. That is, a character can be quoted by % followed by its hex value.

<!--#include -->

Provided by module: *SSI support*

Insert a text from another file into the page.

Attributes

file="path"
The file as a path relative to the directory containing the current page. It cannot contain ../, nor can it be an absolute path.

virtual="URL"
The path to the file, URL encoded. That is, a character can be quoted by % followed by its hex value. The path may contain ../ and may be absolute, i e starting with a /.

<!--#printenv -->

Provided by module: *SSI support*

This tag outputs a listing of all existing variables and their values. Attributes won't be printed.

```
<pre><!--#printenv --></pre>
```

<!--#set -->

Provided by module: *SSI support*

Sets the value of a variable.

Attributes

var="value"
The name of the variable to set. Value sets the variables value.

Programming Tags

Programming tags can be used for advanced RXML such as making web applications. There are also tags of interest to module programmers. For those interested in combining programming with RXML, the *<?pike ?>* and *<?perl ?>* processing instruction tags enables embedded Pike and Perl code in RXML pages.

End of /roxen/2.1/creator/programming/index.xml

<cache></cache>

Provided by module: *RXML 2 tags*

This simple tag RXML parse its contents and cache them using the normal Roxen memory cache. They key used to store the cached contents is the MD5 hash sum of the contents, the accessed file name, the query string, the server URL and the authentication information, if available. This should create an unique key. The time during which the entry should be considered valid can set with one or several time attributes. If not provided the entry will be removed from the cache when it has been untouched for too long.

Attributes

key="string"
Append this value to the hash used to identify the contents for less risk of incorrect caching. This shouldn't really be needed.

nohash
The cached entry will use only the provided key as cache key.

years="number"
Add this number of years to the time this entry is valid.

months="number"
Add this number of months to the time this entry is valid.

weeks="number"
Add this number of weeks to the time this entry is valid.

days="number"
Add this number of days to the time this entry is valid.

hours="number"
Add this number of hours to the time this entry is valid.

beats="number"
Add this number of beats to the time this entry is valid.

minutes="number"
Add this number of minutes to the time this entry is valid.

seconds="number"
Add this number of seconds to the time this entry is valid.

<crypt></crypt>

Provided by module: *RXML 2 tags*

Encrypts the contents as a Unix style password. Useful when combined with services that use such passwords.

Unix style passwords are one-way encrypted, to prevent the actual clear-text password from being stored anywhere. When a login attempt is made, the password supplied is also encrypted and then compared to the stored encrypted password.

Attributes

compare="string"
Compares the encrypted string with the contents of the tag. The tag will behave very much like an *<if>* tag.

```
<crypt compare="LAF2kkMr6BjXw">Roxen</crypt>
<then>Yepp!</then>
<else>Nope!</else>
```

<debug/>

Provided by module: *RXML 2 tags*

Helps debugging RXML-pages as well as modules. When debugging mode is turned on, all error messages will be displayed in the HTML code.

Attributes

on
Turns debug mode on.

off
Turns debug mode off.

toggle
Toggles debug mode.

showid="string"
Shows a part of the id object. E.g. showid="id->request_headers".

werror="string"
When you have access to the server debug log and want your RXML page to write some kind of diagnostics message or similar, the werror attribute is helpful.

This can be used on the error page, for instance, if you'd want such errors to end up in the debug log:

```
<debug werror='File &page.url; not found!
(linked from &client.referrer;)'/>
```

<dice></dice>

Provided by module: *Additional RXML tags*

Simulates a D&D style dice algorithm.

Attributes

type="string default=D6"
Describes the dices. A six sided dice is called 'D6' or '1D6', while two eight sided dices is called '2D8' or 'D8+D8'. Constants may also be used, so that a random number between 10 and 20 could be written as 'D9+10' (excluding 10 and 20, including 10 and 20 would be 'D11+9'). The character 'T' may be used instead of 'D'.

<eval></eval>

Provided by module: *RXML 2 parser*

Postparses its content. Useful when an entity contains RXML-code. `<eval>` is then placed around the entity to get its content parsed.

<gauge></gauge>

Provided by module: *RXML 2 tags*

Measures how much CPU time it takes to run its contents through the RXML parser. Returns the number of seconds it took to parse the contents.

Attributes

define="string"
The result will be put into a variable. E.g. define=var.gauge vill put the result in a variable that can be reached with var.gauge.

silent
Don't print anything.

timeonly
Only print the time.

resultonly
Only the result of the parsing. Useful if you want to put the time in a database or such.

<maketag></maketag>

Provided by module: *RXML 2 tags*

This tag creates tags. The contents of the container will be put into the contents of the produced container.

Attributes

name="string"
The name of the tag.

noxml
Tags should not be terminated with a trailing slash.

type="{tag, container}" (tag)
What kind of tag should be produced.
Inside the maketag container the container attrib is defined. It is used to add attributes to the produced tag. It has the required attribute attrib, which is the name of the attribute. The contents of the attribute container will be the attribute value. E.g.

```
<eval>
<maketag name="replace" type="container">
 <attrib name="from">A</attrib>
 <attrib name="to">U</attrib>
 MAD
</maketag>
</eval>
```

<nooutput></nooutput>

Provided by module: *RXML 2 parser*

The contents will not be sent through to the page. Side effects, for example sending queries to databases, will take effect.

<noparse></noparse>

Provided by module: *RXML 2 parser*

The contents of this container tag won't be RXML parsed.

<?perl ?>

Provided by module: *Perl support*

This processing intruction tag allows for evaluating Perl code directly in the document.
Note: Read the installation and configuration documentation in the Administration manual to set up the Perl support properly. If the correct parameters are not set the Perl code might not work properly or security issues might arise.
There is also a `<perl>...</perl>` container tag available.

<?pike ?>

Provided by module: *Pike tag*

This processing intruction tag allows for evaluating Pike code directly in the document.
Note: With this tag, users are able to run programs with the same right as the server. This is a serious security hasard.
When the pike tag returns, the contents of the output buffer is inserted into the page. It is not reparsed with the RXML parser.
Use entities within the Pike code, scope.variable is handled like scope.variable in RXML.
Note: It is still possible to use the `<pike>...</pike>` container tag, though it behaves exactly as it did in Roxen 2.0 and earlier and the functionality mentioned below does not apply to it. The use of the container tag is deprecated.
Below is a list of special helper functions and constructs which are only available within the `<?pike ?>` tag.

Attributes

write
write(string fmt, mixed ... args) is a helper function. It formats a string in the same way as printf and appends it to the output buffer. If given only one string argument, it's written

directly to the output buffer without being interpreted as a format specifier.

flush
flush() is a helper function. It returns the contents of the output buffer and resets it.

rxml
rxml(string rxmlcode) is a helper function. It parses the string with the RXML parser.

"//O ..." or "/*O ... */"
Pike comment with an 'O' (the letter, not the number) as the very first character treats the rest of the text in the comment as output text that's written directly to the output buffer.

"//X ..." or "/*X ... */"
A Pike comment with an 'X' as the very first character treats the rest of the text in the comment as RXML code that's executed by the RXML parser and then written to the output buffer.

#include "..."
An #include preprocessor directive includes the specified file.

#inherit "..."
An #inherit preprocessor directive puts a corresponding inherit declaration in the class that's generated to contain the Pike code in the tag, i.e. it inherits a specified file from the Roxen filesystem.

```
  <?pike

    //O <pre>
    int first = 1;
    for( var.counter=100; var.counter>1; var.counte
r--,first=0 )
    {
      if( !first )
      {
        //
X &var.counter; bottles of beer on the wall
        //O
      }
      //X &var.counter; bottles of beer on the wall
      //X &var.counter; bottles of beer
      //O take one down, pass it around
    }
    //O one bottle of beer on the wall
    //O one bottle of beer
    //O take it down, pass it around
    //O no bottles of beer on the wall
    //O </pre>

  ?>
```

<set-max-cache/>

Provided by module: *RXML 2 tags*

Sets the maximum time this document can be cached in any ram caches.

Default is to get this time from the other tags in the document (as an example, <*if supports*> sets the time to 0 seconds since the result of the test depends on the client used.

You must do this at the end of the document, since many of the normal tags will override this value.

Attributes

years="number"
Add this number of years to the time this page was last loaded.

months="number"
Add this number of months to the time this page was last loaded.

weeks="number"
Add this number of weeks to the time this page was last loaded.

days="number"
Add this number of days to the time this page was last loaded.

hours="number"
Add this number of hours to the time this page was last loaded.

beats="number"
Add this number of beats to the time this page was last loaded.

minutes="number"
Add this number of minutes to the time this page was last loaded.

seconds="number"
Add this number of seconds to the time this page was last loaded.

<trace></trace>

Provided by module: *RXML 2 parser*

Executes the contained RXML code and makes a trace report about how the contents are parsed by the RXML parser.

Security

Htaccess is a system for handling access control to your pages. It works by placing an *.htaccess* file in a directory, which contains the access control lists for that directory. It is possible to get fine grained security and to configure exactly who can view which pages. The *.htaccess support* module must be enabled for htaccess to work.

It is possible to distinguish users either by the IP address or domain name of their computer or by letting them authenticate with a user name and password. The user name and password is by default compared via an *authentication* module. That usually means that users are authenticated by the operating systems authentication system.

If you want to have password protected pages usable by users that are not handled by the current Authentication module you can create your own database of users by creating *.htpasswd* and *.htgroup* files.

htaccess is a standard supported by many web servers. The access control you have built with htaccess should therefore be portable to other web servers.

End of /roxen/2.1/creator/security/index.xml

.htaccess

Start of /roxen/2.1/creator/security/htaccess.xml

A `.htaccess` file consists of lines containing directives. Apart from the output: Limit; directive, all directives have the form

```
directive argument(s)
```

where *argument(s)* is one or more arguments. The directives supported are:

AuthUserFile
Use this user and password file to authenticate users. Typically, the AuthUserFile is called *.htpasswd*

AuthGroupFile
Use this group file, which contains a database of which groups users are member of. Typically, the AuthGroupFile is called *.htgroup*, if used.

AuthName
Set the authentication realm, which can be any name you choose. The name will be used to tell browsers how to *label* user authentications within a session, so that the browsers can automatically repeat passwords the user has already entered when accessing new pages with the same access requirements.

Redirect
Redirect all accesses for pages in the directory to this URL.

ErrorFile
Show this page in case the requested page could not be found, maybe because the user did not have permission to view it.

Then there is the `<Limit>` container tag. The attributes are the HTTP method(s) that access should be limited to, *GET*, *PUT*, *POST* or *HEAD*. The contents of the tag are access control directives, one directive on each line. Possible directives are:

allow|deny from URL
Allow or deny access to users from a DNS domain or IP number. *www.roxen.com* means the computer while *.roxen.com* means all computers on the domain *roxen.com*. The same way *194.52.202.3* means the computer while *194.52.* means the net starting with *194.52*

require user|group user(s)|group(s)
Allow access only for the named user(s) or group(s).

require valid-user
Allow access to any user present in the AuthUserFile or Authentication module.

satisfy all|any
Decide what happens if both *require* and *allow* rules are present; *all* indicates that the user must satisfy both kinds of requirements, while *any* means that it is enough that the user satisfies either kind.

order deny,allow|allow,deny|mutual-failure
The *order* rules decides how to prioritize deny and allow rules. If the order is set to *deny,allow*, deny rules will be processed before allow rules. With *allow,deny*, allows will be processed before denies, and with *mutual-failure*, hosts allowed by any *allow* rule will be allowed, and other hosts denied. *Deny,allow* is the default.

The rule evaluation does not stop until all rules have been processed, so the earlier a rule is processed, the lower priority is has in determining access. This only matters when different rules contradict each other, for instance when a wide-ranging deny rule forbids access to a certain domain, and an allow grants access to a smaller part of the domain.

Example

A typical *.htaccess* file would look something like this:

```
AuthUserFile /home/frotz/.htpasswd
AuthGroupFile /home/frotz/.htgroup
AuthName MyTestDomain
AuthType Basic

<Limit PUT HOST HEAD>
require user frotz
</Limit>

<Limit GET>
allow from all
</Limit>
```

The *.htaccess* file above would allow everyone to GET documents in the directory, but all other kinds of access would be restricted to the user frotz, and expect this user to login with the password listed for frotz in the *.htpasswd* file in the home directory of the user frotz.

End of /roxen/2.1/creator/security/htaccess.xml

.htgroup

Start of /roxen/2.1/creator/security/htgroup.xml

The format of the group file is straightforward, one line per group, with the line containing the group name, followed by a colon, followed by the users in the group. Users are separated by commas.

In other words, an *.htgroup* file can look like this:

```
all:frotz,gnusto
admins:frotz
```

with one line per group.

End of /roxen/2.1/creator/security/htgroup.xml

.htpasswd

Start of /roxen/2.1/creator/security/htpasswd.xml

The format of the password file is straightforward, one line per user, with the line containing the user name, followed by a colon, followed by the user's password encrypted with the standard Unix password encryption. <*crypt*> can be used to encrypt such a password.

In other words, an *.htpasswd* can look like this:

```
frotz:taeWr6tbTZKO6
gnusto:jKXVnZH6eXR7
```

with one line for each user.

End of /roxen/2.1/creator/security/htpasswd.xml

Browser Support

The Roxen Browser Feature Support database makes it possible to use features that are only supported by a few browsers and still be compatible with all browsers. This is done through a database of features supported by the different browsers. The `<if>` tag together with the if-plugin *supports* is then used on the pages to make them more dynamic and sensitive to which browsers that use different capabilities.

Pages are not customized for a certain browser, but rather for browsers that support different features. When a new browser is released, all that is necessary is to determine what features it supports. Once that has been done, and the database updated, all pages using the supports database will work with it.

Some features might work to a lesser degree on some browsers. Old versions of the Macintosh version of Netscape support JavaScripts, but some JavaScripts make the Netscape browser hang. If you have such JavaScripts, you would probably want the supports database to make sure they are not sent to that version of Netscape. On the other hand, if you have less complicated JavaScripts you will probably want to send them.

To make the supports database work for you, you might need to tweak it yourself. This can be done by the Roxen WebServer administrator by changing the *Client supports regexps* variable. It is available under the *Globals / Settings*.

Since new browsers get released all the time, updated versions of the supports database are by default fetched regularly from our update server at www.roxen.com.

End of /roxen/2.1/creator/browser_support/index.xml

Supports Flags

Start of /roxen/2.1/creator/browser_support/flags.xml

The following is a list of available supports flags.

End of /roxen/2.1/creator/browser_support/flags.xml

File Syntax

Start of /roxen/2.1/creator/browser_support/filesyntax.xml

By default, the supports database is located in the file *server/etc/supports* which is updated automatically from the update server at www.roxen.com.

The *server/etc/supports* file should not be edited directly, since that might interfere with the automatic updates. If you need to tweak the supports database it is better to create your own local supports file, and change the *Client supports regexps* variable.

The syntax used is: *pattern feature, -feature, ...*

If the regular expression *pattern* matches the name of the client, all features will be added to the list of features handled by the client. If '-' is prefixed to the name of the feature, it will be removed instead.

\ can be used to escape newlines.

Values in the *key=value form will have their values copied to the variable named key in the client namespace.

If a line starts with '#', it is skipped, unless it is any of the following

which means include the contents of that file here.

which means replace all occurrences of the word *from* with *to*.

which is used to speed up parsing. If the name of the client matches *pattern* it will go through the section. If the pattern doesn't match the entire section will be skipped.

End of /roxen/2.1/creator/browser_support/filesyntax.xml

Tag Index

End of /roxen/2.1/creator/index/index.xml