VirthosTalk Reference Manual

Version 2.0

Revised: 25.08.2006

Virthos Systems GmbH
Contrescarpe 8c
28203 Bremen
Germany

Email: info@virthos.net
www.virthos.net

Table of Contents

About This Manual

This manual is intended for persons who want to design templates using the Virthos content management system. It is a complete description of the VirthosTalk language, Virthos' proprietary template language, in all its details, and intended primarily to be used as a reference work. In order to understand the descriptions, a general knowledge of HTML and of website design is assumed.

This manual is complemented by several other publications:

- the "Introduction to Virthos," which provides an overview of the installation process, as well as a step-by-step introduction to Virthos' major functions.
- the "Virthos User Manual," which treats all aspects of Virthos, and includes an introduction to working with Virthos templates.
- a variety of other publications, all available for download from *www.virthos.net*

Before using this reference, you should have had some experience with using Virthos, and have read at least Part 3 of the User Manual (“Templates”). It is especially important that you first read the chapter about creating and installing templates, as well as the chapter about template variants.

Virthos Basic / Virthos Pro

Virthos is available in two versions: Virthos Basic and Virthos Pro. This manual can basically be used with both versions; however, some VirthosTalk language elements are available only in the Pro version. You will be able to distinguish these by the designation “available in”, which can be found in the introduction to each language element.

Font Usages

Several different font styles are used in this manual in an effort to make clear how certain expressions are used.

- `Monospace type` is used to render VirthosTalk language elements and HTML source code.
- *Italics* are used for the names of VirthosTalk parameters, files and directories.

Syntax in General

For VirthosTalk language elements to be correctly identified and interpreted, you must strictly follow certain syntax rules when creating a template. The following topics describe these rules in detail.

Placeholders and Directives

VirthosTalk functions can be divided into two groups: placeholders and directives. Placeholders, generally speaking, include all the functions that are replaced with specific values at the time the template is processed. All other functions are classified as directives, as their purpose is simply to initiate particular actions.

Having a clear understanding of the difference between placeholders and directives should increase your ability to follow the descriptions in this manual; however, you can still successfully use VirthosTalk functions without keeping this distinction in mind.

Integration into HTML

VirthosTalk elements can be found in any part in HTML code, even in headers, style definitions, and JavaScript functions. For a VirthosTalk element to be recognized and correctly processed, you must enclose it in double braces.

any HTML code **{{***VirthosTalk element***}}** *any HTML code*

When your code is parsed, the braces around the VirthosTalk element are removed—and in the case of placeholders, the element is replaced by another value. The surrounding HTML code is not changed.

If a VirthosTalk element is directly preceded and followed by HTML comment tags, these comment tags are also removed from the source code when it is parsed.

any HTML code **<!--{{***VirthosTalk-Element***}}-->** *any HTML code*

For the template to function correctly, it is does not matter if the VirthosTalk element appears in a comment or not; the result is the same in either case. The HTML code will be more readable if you follow the convention of writing placeholders without comment tags and directives within comments. That is also the way the code samples in this manual will appear.

Function Names and Parameters

A VirthosTalk element consists of at least a function name, which may sometimes be followed by additional information (parameters). To make it clear where the function name ends and the parameters begin, a colon is required:

`{{`*Functionname*`:` *Parameters* `}}`

If there is more than one parameter, these are separated by commas.

`{{`*Functionname*`:` *Parameter1*`,` *Parameter2*`,` *Parameter3* `}}`

If a function can have many different parameters, these can be given meaningful names:

`{{`*Functionname*`:` *Name1*`="`*Value1*`",` *Name2*`="`*Value2*`",` *Name3*`="`*Value3*`"}}`

The values can optionally be enclosed in single or double quotes. Within one function, you cannot mix single and double quotes.

Placeholders as Parameters

In most cases, the value of a parameter can also be represented by a placeholder. When such nesting takes place, the inner placeholder is written with single braces only:

`{{`*Functionname*`:` `{`*Placeholdername*`}` `}}` or

`{{`*Functionname*`:` *Name1*`="{`*Placeholdername*`}"}}`

If a positional parameter is supplied by a placeholder, as in the first example, it must be followed by a space so that three closing braces do not occur together.

Page References

Many VirthosTalk functions, like `vtUse` and `vtLink`, require that a page be supplied as a parameter. This can be accomplished in various ways in VirthosTalk, each of which has its advantages and disadvantages. This chapter describes the different ways and under what circumstances you should use each one.

Page Numbers

Within Virthos, each page has a unique number, which is assigned to it when it is created and which never changes, even if the page is moved to another location or otherwise changed. The number lets you refer to a page concisely and unambiguously. for example:

```
{{vtLink: 381}}
```

creates a URL pointing to page 381, no matter where that page exists in the Virthos hierarchy.

Instead of using the actual page number, you can use a placeholder, like this:

```
{{vtLink: {CustomerNumber}  }}
```

In this case, a URL is created that points to the page whose page number is stored in the “CustomerNumber” placeholder.

Path References

Pages can also be designated by a path, which gives the page's position within the Virthos hierarchy. Unlike a page number, a path reference does not always have to point to the same page. Moving and renaming of pages can result in a path reference having a new meaning.

Virthos distinguishes between absolute and relative path specifications. An absolute path always starts from the start page or the top level of the hierarchy. A relative path uses the current page as its starting point.

Absolute Paths

Absolute path references take as their starting point the page designated as the start page or the top level of the page hierarchy. For example, the placeholder

```
{{vtLink: /News/Today }}
```

creates a reference to the “Today” page under the “News” page, which in turn is under the current start page. The placeholder

```
{{vtLink: //Internal }}
```

points to the “Internal” page, located at the top level of the Virthos page hierarchy.

Relative Paths

Relative path references always start at the current page, and can point either upward or downward in the hierarchy. Thus,

```
{{vtLink: Comments }}
```

points to the “Comments” page, which is subordinate to the current page.

```
{{vtLink: Comments/important }}
```

creates a reference to the “important” page, which is subordinate to the “Comments” subpage. You can also refer to subordinate pages using position numbers. So

```
{{vtLink: #2 }}
```

becomes a reference to the second subpage of the current page, where pages are numbered according to the sort order specified in the page specifications.

You can refer to the superior page, or container page, in the following way:

```
{{vtLink: .. }}
```

The placeholder

```
{{vtLink: ../.. }}
```

refers to the container page's container page.

You can also refer to a superior page using a reference to the level of the hierarchy. The placeholder

```
{{vtLink: \1 }}
```

points to the highest level page above the current page (in other words, the page that appears first in the path to the current page). Similarly,

```
{{vtLink: \2 }}
```

points to the second page within the path to the current page, and so on.

Type References

The third method of referring to another page is to use the page type. This is always useful when there is a defined relationship between two page types. For example, a page of the "employee" type might always have to occur under a page of the "company" type. In that case, from the employee page, you could refer to the company page using

```
{{vtLink: \company }}
```

regardless of whether the employee page is directly under the company page, or if there are other pages, such as "office" between them. When given a type specification, Virthos always searches backwards through the whole path from the current page, until it finds a page of the specified type.

To distinguish path references from type references, the latter always use a backslash as the first character. This is another way to emphasize that type references always proceed in a backward direction within the path. Unlike path specifications, type specifications cannot contain multiple levels, so that a specification like "\city\company" would not be valid.

Special Pages

Besides the three ways of specifying pages already discussed, there are in the Pro version three special characters that allow references to pages that have a special connection to the current page.

Special Character	Meaning
>	the next page within the current list
<	the previous page within the current list
^	the page where the list originates

You can use the placeholder `{{vtLink: > }}` to create a link to the next page within the current list, for example. For this to work, the current page must have been accessed through a link within a `vtLoop` block using a `-store` parameter. Should this not be the case, a `vtLink` placeholder that uses this special notation is replaced by a number sign (#).

Which Methods for Which Purposes?

Each method of referring to a page has its advantages and disadvantages. The following guidelines should help you find the most suitable method for your purposes.

- In general, it is best to use path references whenever possible. Path references are most likely to function in the way that a web designer would expect.
- If there is a clear contextual relationship between two page types—meaning a clear hierarchical order—you should use a type reference to refer to the superior page from the subordinate one.
- For building dynamic navigation menus that conform to the Virthos page hierarchy, it is best to use relative paths that specify hierarchical levels.
- Page numbers should generally be used only in connection with other placeholders—meaning when a page has saved within it the number of another page that a user has entered.

Summary

Here once again is an overview of all available methods for specifying pages:

Method	Example
Page Number direct	`{{vtLink: 381 }}`
Page Number indirect	`{{vtLink: {CustomerNumber} }}`
Absolute Path, based on start page	`{{vtLink: /News/Today }}`
Absolute Path, starting at the first hierarchical level	`{{vtLink: //Internal/Memos }}`
Relative Path, subordinate page, by name	`{{vtLink: Notices }}`
Relative Path, superior page, by position	`{{vtLink: #2 }}`
Relative Path, superior page	`{{vtLink: .. }}`
Relative Path, same hierarchical level	`{{vtLink: \2 }}`
Type Reference	`{{vtLink: \company }}`
Special Reference	`{{vtLink: > }}`

Function Reference

The following topics describe all the details of each of the VirthosTalk functions that you can use. Each topic is formatted using the following pattern:

Function Name

Brief Explanation of the Function

Usage	*Illustration of the syntax required by the function*
Version	*Virthos versions where this function is available*
Parameters	*Explanation of the various parameters*

Detailed Explanation and Examples

See also
Functions whose descriptions contain further important information

For easy reference, the functions are presented in alphabetical order. If you are not sure of the name of a function, you can look for it in the overview on the next page. There, you can find the most important VirthosTalk placeholders and directives grouped by the context in which they are used.

Overview of the Most Important Placeholders and Functions

Editable Areas

- (User Placeholders)
- vtFile
- vtFilename
- vtMedia

Page Specifications

- vtName, vtID
- vtCreationDate
- vtModificationDate
- vtCreationUser
- vtModificationUser

Users and Access Rights

- vtUser
- vtUsername
- vtCurrentAccessRights

Process Control Structures

- vtIf, vtElse, vtElseIf
- vtLoop, vtNext, vtExit
- vtRepeat
- vtUse
- vtScript

Lists and Menus

- vtLoop, vtNext, vtExit
- vtPortions
- vtPath
- vtLink
- vtID, vtPathIDs
- vtName

Selections

- vtSelect
- vtEval

Variables and Values

- vtCurrentDate
- vtCurrentTime
- vtSet, vtGet
- vtCalc, vtResult
- vtGetValue, vtPostValue
- vtSession, vtGlobal

Miscellaneous

- vtHeader, vtRedirect
- vtInclude
- vtUserAgent

(User Placeholders)

Are replaced by the text that was associated with this placeholder when a page was created.

Usage	`{{`*Placeholdername*`:` *Coding*`}}`
Version	Basic, Pro (from 1.0)
Parameters	*Coding* Coding to be applied to the text

A user placeholder is one whose name is assigned by a user and whose purpose is to create an editable block of text. Examples: `{{Headline}}` or `{{Content}}`.

The following rules apply to naming user placeholders:

- The name may not begin with `vt`.
- The name may not contain any spaces.
- The name may also not contain punctuation marks or special characters. The only exceptions are hyphens (`-`) and underscores (_).
- The name may not contain accented letters.
- The name may not begin with a digit. Digits are allowed inside the name, however.
- Names are not case-sensitive. The names `headline`, `Headline` and `HEADLINE` are considered by Virthos to be the same placeholder.

When a page is created or saved, the textual content of a placeholder is saved as the user entered it. Then when the content is accessed, additional coding may be applied to it:

Coding	Meaning
`html`	Line breaks are replaced by ` ` tags, and accented letters and special characters are replaced by entities, for example: “ä” is replaced by `&auml;`)
`styled`	Like “html”, but additionally, Internet and email addresses enclosed in `<a>` tags and other marked text is converted to specific HTML markup (see below).
`raw`	The text is left unchanged

If nothing else is specified, `html` is the default coding. It is helpful to use raw coding if there is already valid HTML code within a placeholder, as in multiline text fields on HTML forms:

```
<textarea name="Description">{{Description: raw}}</textarea>
```

When you use `raw` coding, you should be aware that there is a certain security risk when users can enter HTML code. The option should be used with caution in areas where anonymous users can enter content (such as in discussion forums and guest books).

By using `styled` coding, authors can apply certain styles to the text. The following types of markup are available in Virthos:

Markup	is converted to:
!*any text*!	*any text*
/*any text*/	*any text*
__*any text*__	<u>*any text*</u>
$*any text*$	<var>*any text*</var>
:*any text*:	<kbd>*any text*</kbd>
#*any text*#	<code>*any text*</code>
one_word ((*URL*))	*one_word*
one_word ((vt:*page*))	*one_word*
one_word ((js:*Javascript*))	*one_word*
- *any text* - *still more text* ...	 *any text* *still more text* ...

URL's and email addresses are automatically (without any special markup by the user) converted to clickable links. For Virthos to recognize a URL, it must either begin with `www.` or be written in full, beginning with `http://`, `https://` or `ftp://`.

If you enclose a link in double parentheses, the preceding word is formatted as the hyperlink. If several words are used, they must then be separated by no-break spaces (Mac: Alt + space, PC: Alt + 0160).

A styled placeholder should always be enclosed in `<p>` tags in order to ensure correct generation of HTML for lists.

vtButton

Is replaced by an editing button if the current user has the required access rights.

Usage	`{{vtButton:` *Method*`}}`
Version	Basic, Pro (from 1.3)
Parameters	*Method* Name of the method to be invoked by the button

You can use `vtButton` to insert small buttons into a page that permit authors to switch directly from Edit mode to Structure mode, create new pages, or remove existing pages. The following methods are available:

`edit`	Access the page in Edit mode
`sort`	Access the page in Structure mode
`trash`	Put the page in the trash
`new`	Create new subordinate pages (with the designated template)
`add`	Create new subordinate pages (with a choice of template)

When using the `new` method, the second parameter must always be the desired template, for example:

```
{{vtButton: new, item}}
```

A button invoking the `add` method works in exactly the same way as the “New Page” button in Virthos Manager's Toolbar. In other words, a list of available templates is displayed, and the author is able to select one of them. When there is only one template that can be used (because of a `vtLoop` directive with a predetermined page type), the template selection step is skipped.

If the current user does not have the access rights needed to edit the current page, no button is displayed. This applies in particular to guest users, who have not (yet) registered with the Virthos system.

See also
`vtLoop`

vtCalc

Allows values to be calculated using PHP functions.

Usage	`<!--{{vtCalc:` *Expression*`}}-->`
Version	Basic, Pro (from 1.3)
Parameters	*Expression* the PHP expression to be evaluated

You can use `vtCalc` to execute mathematical, string, date time, and all other PHP functions In order to produce a result. The formula may contain VirthosTalk placeholders—these are replaced by their respective values before the calculation takes place. The result of the calculation can be accessed through `vtResult`.

To comply with PHP syntax, numeric constants within `vtCalc` must use decimal points rather than commas. Placeholders that are to be interpreted as strings must be enclosed in quotation marks.

You can find complete documentation for all PHP functions at *www.php.net/manual*.

Example 1

```
<!--{{vtCalc: round( {NetPrice}  * 1.16, 2 ) }}-->
<p>The net price is: <b>{{vtResult}} Euros</b></p>
```

Example 2

```
<!--{{vtCalc: substr( '{LastName}', 0, 1 ) }}-->
<p>Your last name starts with "{{vtResult}}"</p>
```

See also

`vtResult`

vtComment

Is replaced by the content of the comment field for the current page.

Usage	`{{vtComment}}`
Version	Basic, Pro (from 1.0)
Parameters	–

Virthos lets you save comments about each web page in its specifications. These are intended primarily for internal memos, such as notes by web designers or as reminders of company styles and usage. However, this placeholder allows these comments to be inserted in user templates, as well.

vtCountPages

Is replaced by the number of pages that are directly or indirectly subordinate to the current page.

Usage	`{{vtCountPages:` *Depth*`}}`
Version	Basic, Pro (from 1.2)
Parameters	*Depth* The number of hierarchical levels to be taken into account (optional)

The *Depth* must be expressed as an integer. The following values are permitted:

`0` Means that all pages subordinate to the current page are counted, regardless of their level.
`1` Only the immediately subordinate pages are counted (default).
`2` Only the immediately subordinate pages and their subpages are counted.
`3` Like `2`, but subpages of subpages are counted, as well.
etc.

See also
`vtCountSelection`

vtCountSelection

In a `vtSelect` block, is replaced by the number of pages in the selection.

Usage	`{{vtCountSelection}}`
Version	Pro (from 1.3)
Parameters	–

Example

```
<!--{{vtSelect: -filter="{vtName}.bw.A" }}-->
   <!--{{vtIf: {vtCountSelection}  .eq. 0 }}-->
      <p>There were no items found beginning with the letter A.</p>
   <!--{{vtElse}}-->
      <p>There were {{vtCountSelection}} items found.</p>
   <!--{{vtEndIf}}-->
<!--{{vtEndSelect}}-->
```

See also

`vtSelect`, `vtCountPages`

vtCreationDate

Is replaced by the date and time of the page's creation.

Usage	`{{vtCreationDate}}`
Version	Pro (from 1.0)
Parameters	–

In contrast to `vtCurrentDate`, `vtCreationDate` does not return a pure date expression, but rather a string in the format “23.05.2006, 12:34 Uhr.” This placeholder can therefore not be used in `vtIf` or `vtSelect` directives to compare dates.

See also

`vtModificationDate`

vtCreationUser

Is replaced by the name of the user who created the current page.

Usage	`{{vtCreationUser}}`
Version	Pro (from 1.3)
Parameters	–

When a page is created, Virthos saves the name of the current user along with the creation date in the page specifications. If the user settings contain a full name, that name is used—otherwise, the user's login name is used. In either case, the name is retained even if the user settings are later changed or deleted.

See also

`vtModificationUser`

vtCurrentAccessRights

Is replaced by a list of the current user's access rights for the current page.

Usage	`{{vtCurrentAccessRights}}`
Version	Basic, Pro (from 1.3)
Parameters	–

The placeholder is intended primarily to formulate conditions within a `vtIf` directive. This allows certain page elements to be removed from display in case the current user does not have the necessary rights.

Available access rights are:

`view`	the user can view the current page
`update`	the user can modify the content of the current page
`add`	the user can create subpages of the current page or remove existing ones

Example

```
<!--{{vtIf: {vtCurrentAccessRights}.cn.add }}-->
   <a href="{{vtLink:-met='vtAdd'}}>create a new entry</a>
<!--{{vtEndIf}}-->
```

See also

`vtIf`

vtCurrentDate

Is replaced by the current system date on the web server.

Usage	`{{vtCurrentDate}}`
Version	Basic, Pro (from 1.0)
Parameters	–

The date is returned using the common German syntax, i.e. in “dd.mm.yyyy” format.

Example

```
<!--{{vtIf: {vtCurrentDate}  .bw. 25.12. }}-->
   Merry Christmas!
<!-{{vtEndIf}}->
```

See also

`vtCurrentTime`

vtCurrentItem

Within a `vtRepeat` block, is replaced by the current list item.

Usage	`{{vtCurrentItem}}`
Version	Basic, Pro (from 1.2)
Parameters	–

Examples of the use of this placeholder can be found in the explanations of the `vtRepeat` directive.

See also

`vtRepeat`

vtCurrentPosition

Within a `vtLoop` block, is replaced by the number of the current iteration.

Usage	`{{vtCurrentPosition}}`
Version	Basic, Pro (from 1.2)
Parameters	–

You can use `vtCurrentPosition` to number items in dynamic lists. Or, for example, together with a `vtIf` directive, `vtCurrentPosition` can be used to highlight every other item in such a list.

Example

```
<table>
    <!--{{vtLoop}}-->
    <tr>
        <td>{{vtCurrentPosition}}</td>
        <td>{{Name}}</td>
        <td>{{Phone}}</td>
    </tr>
    <!--{{vtEndLoop}}-->
</table>
```

See also

`vtLoop`

vtCurrentTime

Is replaced by the current system time on the web server.

Usage	`{{vtCurrentTime}}`
Version	Basic, Pro (from 1.0)
Parameters	–

The system time is expressed in hours and minutes in “hh:mm” format.

Example

```
<!--{{vtIf: {vtCurrentTime} .gt. 18:00 }}-->
   <h1>Good Evening!</h1>
<!--{{vtElse}}-->
   <h1>Hello!</h1>
<!-{{vtEndIf}}->
```

See also

`vtCurrentDate`

vtElse

Introduces the code that is executed if the condition given in a `vtIf` block is not satisfied.

Usage	`<!--{{vtElse}}-->`
Version	Basic, Pro (from 1.0)
Parameters	–

Explanations and examples can be found in the description of `vtIf`.

See also

`vtIf`, `vtElseIf`

vtElseIf

Allows an additional condition to be named within a `vtIf` block.

Usage	`<!--{{vtElseIf:` *Condition*`}}-->`
Version	Basic, Pro (from 1.2)
Parameters	*Condition* condition under which the HTML code that follows is generated

The rules for expressing the condition are the same as for `vtIf` (see that topic).

See also

`vtIf`, `vtElse`

vtEval

Within a `vtSelect` block, is replaced by the result of a statistical evaluation.

Usage	`{{vtEval:` *Method, Placeholder*`}}`
Version	Pro (from 1.3)
Parameters	*Method* the desired evaluation method *Placeholder* the name of the (user) placeholder to be used for the evaluation

There are three possible values that can be specified for the *Method* parameter:

`sum`	adds the values of the placeholder within all selected pages
`min`	returns the smallest value within the selection
`max`	returns the greatest value within the selection

Example

```
<!--{{vtSelect: -type="reportcard"}}-->
   Best grade: {{vtEval: max, grade}}<br>
   Worst grade: {{vtEval: min, grade}}
<!--{{vtEndSelect}}-->
```

See also

`vtSelect`, `vtCountSelection`

vtExit

Halts execution of a `vtLoop` or a `vtRepeat` loop prematurely.

Usage	`<!--{{vtExit}}-->` or `<!--{{vtExit:` *Condition*`}}-->`
Version	Basic, Pro (from 1.2)
Parameters	*Condition* Condition that halts execution of the loop

In its simplest form, without parameters, `vtExit` causes immediate termination of the loop currently being executed. If a *Condition* is given, the loop is exited if the condition is true. The condition is expressed using the same syntax as in `vtIf`.

Example 1

```
<p>Generate only the first item.</p>
<ul>
<!--{{vtLoop}}-->
   <li>{{vtName}}</li>
   <!--{{vtExit}}-->
<!--{{vtEndLoop}}-->
<ul>
```

Example 2

```
<p>Generate the first five items.</p>
<ul>
<!--{{vtLoop}}-->
   <li>{{vtName}}</li>
   <!--{{vtExit: {vtCurrentPosition}.eq.5}}-->
<!--{{vtEndLoop}}-->
<ul>
```

See also

`vtCurrentPosition`, `vtIf`, `vtLoop`, `vtRepeat`

vtFile

Is replaced by the URL that allows downloading of a file from the current page.

Usage	`{{vtFile:` *Name*`}}`
Version	Basic, Pro (from 1.0)
Parameters	*Name:* internal name of the file placeholder

If this placeholder is used in a template, Virthos automatically allows a file to be uploaded in Edit mode. If there are several files to be uploaded, you can use the placeholder as many times as needed in the template, with a different name each time.

Example

```
<!--{{vtIf: {vtFile: File1}  .neq. }}-->
   <a href="{{vtFile: Downloadfile1}}">download now</a>
<!--{{vtElse}}-->
```

See also

`vtFilename`, `vtMedia`

vtFilename

Is replaced by the name of a downloadable file.

Usage	`{{vtFilename:` *Name*`}}`
Version	Basic, Pro (from 1.0)
Parameters	*Name:* internal name of the file placeholder

In contrast to `vtFile`, `vtFilename` does not generate the entire path, but only the file name of the downloadable file, so that it can be used in a hyperlink or a button label.

Example

```
<a href="{{vtFile: Datei2}}">
    download "{{vtFilename:File2}}" now
</a>
```

See also

`vtFile`

vtGet

Is replaced by the value that was assigned to a variable using `vtSet`.

Usage	`{{vtGet:` *Variable*`}}`
Version	Basic, Pro (from 1.3)
Parameters	*Variable* Name of a variable

Example

```
<!--{{vtUse:..}}-->
   <!--{{vtSet: Container="{vtName}"}}-->
<!--{{vtEndUse}}-->

<p>The parent page is named "{{vtGet:Container}}".</p>
```

See also

`vtSet`

vtGetValue

Is replaced by one parameter value from the current URL.

Usage	`{{vtGetValue:` *Element*`}}`
Version	Basic, Pro (from 1.1)
Parameters	*Element* Name of the value to be generated.

You can use `vtGetValue` to fetch selected individual elements of the current URL, including both Virthos-specific elements like `-pg` or `-met` and those you have supplied by adding them to a hyperlink or by submitting a form using the "get" method.

Example

```
<!--{{vtIf: {vtGetValue: -act}  .eq. create }}-->
   A new entry has been created.
<!--{{vtEndIf}}-->
```

See also

`vtPostValue`

vtGlobal

Is replaced by the value of a global variable.

Usage	{{vtGlobal: Name}}
Version	Basic, Pro (from 1.2)
Parameters	Name Name of the global variable whose value is to be generated

You can use `vtGlobal` to derive values that were assigned using either the *config.php* configuration file or a settings page. Information about ways to assign global variables can be found in the Virthos User Manual.

Example

```
<p>Our Hotline Number: {{vtGlobal: HotlineNumber}}.</p>
```

vtHeader

Lets you add a line to the current page's HTTP header.

Usage	`<!--{{vtHeader:` *Line*`}}-->`
Version	Basic, Pro (from 1.0)
Parameters	*Line* Line of text to be added to the HTTP header

The `vtHeader` directive is intended only for advanced users, who can use it for purposes like changing a page's MIME type. It can also be used to redirect visitors to another URL. However, it is easier to use the `vtRedirect` directive to accomplish this.

See also

`vtRedirect`

vtID

Is replaced by the current page's internal identification number.

Usage	`{{vtID}}`
Version	Basic, Pro (from 1.0)
Parameters	–

Every Virthos page is assigned a unique number from an automatic sequence when it is created. This number cannot later be changed. It serves as an unambiguous reference to the page, and is especially useful for highlighting the current page when a dynamic navigation menu is being built.

Example

```
<!--{{vtSet: ThisPage="{vtID}"}}-->
<!--{{vtUse:..}}-->
   <!--{{vtLoop}}-->
      <!--{{vtIf: {vtID}  .eq. {vtGet:ThisPage}  }}-->
         <strong>{{vtName}}<br>
      <!--{{vtElse}}-->
         <a href="{{vtLink}}">{{vtName}}</a><br>
      <!--{{vtEndIf}}-->
   <!--{{vtEndLoop}}-->
<!--{{vtEndUse}}-->
```

This example generates a list of all pages that are subordinate to the current page's container page. The current page is shown in bold in the list, and the other pages are formatted as hyperlinks.

See also

`vtPathIDs`

vtIf

Generates the HTML code that follows only if a certain condition is met.

Usage	`<!--{{vtIf:` *Condition*`}}-->` ... `<!--{{vtEndIf}}-->` *or* `<!--{{vtIf:` *Condition*`}}-->` ... `<!--{{vtElse}}-->` ... `<!--{{vtEndIf}}-->` *or* `<!--{{vtIf:` *Condition*`}}-->` ... `<!--{{vtElseIf:` *Condition*`}}-->` ... `<!--{{vtElseIf:` *Condition*`}}-->` ... `<!--{{vtElse}}-->` ... `<!--{{vtEndIf}}-->`
Version	Basic, Pro (from 1.0)
Parameters	*Condition* Condition determining whether the HTML code that follows will be generated.

The `vtIf` directive, in conjunction with `vtElse` and `vtElseif`, allows a template to be given a high degree of versatility. The HTML code can be tailored to very specific situations that can be precisely described by the way the conditions are expressed.

A condition can be formulated using the pattern

left_comparand `.`*relationaloperator*`.` *right_comparand*

The comparands can be expressed as literals or as placeholders (in single braces). If the right comparand is a placeholder, you must take care to separate the placeholder's right brace from the closing brace of the `vtIf` directive with a space. The spaces between the comparands and the periods surrounding the relational operator are only intended to aid legibility and are not strictly required.

If a date is used as a comparand, it must be written in “dd.mm.yyyy” format, and times must be in “hh:mm” format. If an hour or minute value is less than 10, it must contain a leading zero. Days and months do not require leading zeroes.

When comparing strings, Virthos ignores case differences. A condition like `{Name}.eq.jones` would be true whether the `{Name}` placeholder contained the word “Jones” or “JONES.”

The following table lists the available relational operators:

Operator	*Explanation*
`.eq.`	is equal to
`.neq.`	is not equal to
`.gt.`	(number, date or time) is greater than
`.gte.`	(number, date or time) is greater than or equal to
`.lt.`	(number, date or time) is less than
`.lte.`	(number, date or time) is less than or equal to
`.bw.`	(string) begins with
`.ew.`	(string) ends with
`.cn.`	(string) contains
`.ncn.`	(string) does not contain

Example 1

```
<!--{{vtIf: {vtCountPages}  .eq. 0 }}-->
    <p>No items found</p>
<!--{{vtEndIf}}-->
```

Example 2

```
<!--{{vtIf: {birthday}.eq.{vtCurrentDate}  }}-->
    <p>Happy Birthday, {{Name}}!</p>
<!--{{vtElse}}-->
    <p>Hello, {{Name}}</p>
<!--{{vtEndIf}}-->
```

Example 3

```
<!--
    {{vtIf: {priority}.eq.1 }}
        {{vtSet: Text="extremely urgent"}}
    {{vtElseIf: {priority}.eq.2}}
        {{vtSet: Text="urgent"}}
    {{vtElseIf: {priority}.eq.3}}
        {{vtSet: Text="normal"}}
    {{vtElseIf: {priority}.eq.4}}
        {{vtSet: Text="can wait"}}
    {{vtElse}}
        {{vtSet: Text="unimportant"}}
    {{vtEndIf}}
-->
```

vtInclude

Is replaced by the content of an external text file (library file).

Usage	`{{vtInclude:` *Filepath*`}}`
Version	Basic, Pro (from 1.3)
Parameters	*Filepath* relative path to the file to be included (based on the location of the template file)

You can use `vtInclude` to bring in HTML code elements that are used by several templates (for example, headers, navigation elements, etc.) from separate files. This makes it easier to maintain templates, since changes made to library files are immediately available in all templates. This also simplifies the templates themselves.

A file brought in with `vtInclude` may include other VirthosTalk placeholders and directives, including other `vtInclude` directives. These are interpreted in the context in which the `vtInclude` command is placed.

Example

```
<!--{{vtSet: PageTitle="Legal"}}-->
<!--{{vtInclude: includes/page_header.html }}-->
```

The effect is that the value specified with `vtSet` in the *page_header.html* library file can be retrieved with `{{vt-Get:PageTitle}}`. This can be used to complete the `<title>` element of an HTML header, for example.

vtLink

Is replaced by a URL pointing to a page within Virthos.

Usage	`{{vtLink:` *page*`}}`
	`{{vtLink:` *Page*`,` *Method*`}}`
	`{{vtLink: -pg="`*page*`", -met="`*method*`", -lg="`*language*`", -act="`*action*`"}}`	

Version	Basic, Pro (from 1.2)	
Parameters	*Page*	Page reference
	method	Name of the method to be invoked with the page
	Language	Language in which the page is to be displayed (Pro only)
	Action	Action to be carried out with the supplied values

This placeholder is intended to be used within an HTML anchor element or a form action. The `-lg` parameter is available only starting with Version 2.0. In addition to the above-mentioned parameters, as many other parameters as desired can be specified. These should be attached to the URL using the pattern "parameter1=value1&parameter2=value2," etc.

The *Page* can be expressed as a page number or a path (see the chapter "Page References"). If the page setting the current context is meant, the parameter may be omitted. The *Method* can be omitted if the default method ("vtview") is meant.

The *Language* can be referred to as an abbreviation that was previously defined in the system settings. This specification is needed only if the language needs to be switched. If you omit this parameter, the current language is retained. Specifying an *Action* is required when `vtLink` is used within the `action` attribute of a `<form>` element. More details can be found in the chapter entitled "Processing Forms" in the User Manual.

Example 1

```
<p>Click <a href="{{vtLink: /Contacts}}">here</a> to get in touch with us.</p>
```

Example 2

```
<form action="{{vtLink: -met="feedback", -act="mail"}}" method="post">
```

Example 3

```
<a href="{{vtLink: -lg="de"}}">German Version</a>
```

See also

`vtSelf`, the chapter "Page References"

vtLoop

Repeats the HTML code that follows for all subpages of the current page, or (within a `vtSelect` block), for all pages that are part of the current selection.

Usage	`<!--{{vtLoop}}-->` ... `<!--{{vtEndLoop}}-->`
	or
	`<!--{{vtLoop:` *Pagetype*`}}-->` ... `<!--{{vtEndLoop}}-->`
	or
	`<!--{{vtLoop: -type="`*Pagetype*`",-show="`*Display*`", -store="`*Name*`", -size="`*Iterations*`"}}-->`
	...
	`<!--{{vtEndLoop}}-->`

Version	Basic, Pro (from 1.0)
Parameters	*Pagetype* Limits iterations to pages of a particular type (permitted only when the `vtLoop` block is not with a `vtSelect` block) *Display* Limits iterations to pages that meet certain criteria (see Explanations) *Name*: Name under which the page list is to be saved, in order to permit simple paging back and forth *Iterations* Maximum number of iterations

You can use `vtLoop` to build dynamic lists—lists that automatically adjust to the addition and removal of subpages. All placeholders used within a `vtLoop` block are interpreted in the context of the page just encountered. Nesting of `vtLoop` blocks is also possible.

The order in which the subpages are encountered depends on the sort order that is defined in the page specifications. If the `vtLoop` directive is within a `vtSelect` block, the sort order that is specified by the `vtSelect` directive's `-sortfield` and `-sortorder` parameters is used.

The `vtLoop` directive basically goes through all subpages of the current page—or all pages in the current selection—that the current user has read access to. The optional parameters can be used to further control which subpages should be within the selection.

The `-show` parameter can have one of the following values:

Display	Meaning
`all`	All pages are considered
`visible`	Only “visible” pages are considered. (A page is visible when the “Do not display page in menus and sitemap” option in its specifications is disabled.)
`viewable`	Only “visible” and “displayable” pages are considered. (A page is “displayable” if there is a template for its page type.)

If you omit the `-show` parameter, `vtLoop` acts as if you had specified the value `visible` for `-show`.

Specifying a *Pagetype*—besides limiting the pages to be considered—has an additional effect: It causes template selection to be bypassed when a new subpage is created, and immediately displays the creation form for the specified page type. This happens even if there is no existing template for the specified page type. The creation form then makes available entry fields for all user placeholders found within the `vtLoop` block. When there are several consecutive (not nested) `vtLoop` directives with page types specified, a selection of available page types is displayed when a new page is created.

If the `-store` parameter is present, Virthos stores the active page list under the specified name and thereafter enables paging through the individual pages using the special "<" and ">" page indicators (see the "Page References" chapter).

If you use `-size` to establish an upper limit for the number of iterations, the loop will be exited once this number is reached. You can continue the loop by accessing the current page again and including the `-prt=2` parameter in its URL. If the loop should again be exited, you can then use `-prt=3`, `-prt=4`, etc. to peruse the following "portions." It is easiest to use the `vtPortion` placeholder to do this. This is explained further under that topic.

Example 1

```
<!--{{vtLoop: -show="viewable"}}-->
    <p>{{Headline}}<br>{{Intro}}  <a href="{{vtLink}}">more</a></p>
<!--{{vtEndLoop}}-->
```

Example 2

```
<table>
    <tr>
        <th>LastName</th>
        <th>FirstName</th>
        <th>Phone</th>
    </tr>
    <!--{{vtLoop: -type="contact", -store="ContactList"}}-->
    <tr>
        <td><a href="{{vtLink}}">{{LastName}}</a></td>
        <td>{{FirstName}}</td>
        <td>{{Phone}}</td>
    </tr>
    <!--{{vtEndLoop}}-->
</table>
```

See also

`vtCurrentPosition`, `vtExit`, `vtNext`, `vtSelect`, "Page References" chapter

vtMedia

Is replaced by the URL of an image file that has been uploaded for the current page.

Usage	`{{vtMedia:` *Imagename*`}}`
Version	Basic, Pro (from 1.3)
Parameters	*Imagename* Name of the image placeholder (may have any name)

The `vtMedia` placeholder is intended for use in `<img>` elements. When `vtMedia` is used in a template, Virthos automatically introduces a form element for uploading an image file when in Edit mode. If there are several image files to be uploaded, you can use `vtMedia` as many times as necessary, with a different image name each time.

As long as no image has been uploaded, the placeholder will point to an invisible image ("clear.gif"). This will prevent the appearance of the "broken image" icon in a browser if an image does not exist.

Example 1

```
<img src="{{vtMedia:image1}}" width="220">
```

Example 2

```
<!--{{vtIf: {vtMedia:image2}  .cn. clear.gif}}-->
   <p>No image available</p>
<!--{{vtElse}}-->
   <img src="{{vtMedia:image2}}">
<!--{{vtEndIf}}-->
```

See also

`vtFile`

vtModificationDate

Is replaced by the date and time of the page's last modification.

Usage	`{{vtModificationDate}}`
Version	Basic, Pro (from 1.0)
Parameters	–

In contrast to `vtCurrentDate`, `vtModificationDate` does not return a pure date expression, but rather a string in the format “23.05.2006, 12:34 Uhr.” This placeholder can therefore not be used for date comparison in `vtIf` or `vtSelect` directives.

See also
`vtCreationDate`

vtModificationUser

Is replaced by the name of the user who last modified the current page.

Usage	`{{vtModificationUser}}`
Version	Pro (from 1.3)
Parameters	–

When a page is modified, Virthos saves the name of the current user along with the modification date in the page specifications. If the user settings contain a full name, this name is used—otherwise, the user's login name is used. The name is retained even if the user settings are later changed or deleted.

See also
`vtCreationUser`

vtName

Is replaced by the current page's internal name.

Usage	`{{vtName}}`
Version	Basic, Pro (from 1.0)
Parameters	–

Each Virthos page has a name that is either built from the first characters of its textual content or selected by the user. You can use `vtName` to generate this name, as in dynamic navigation menus, for example.

Example

```
<!--{{vtUse:/}}-->
   <!--{{vtLoop}}-->
          <a href="{{vtLink}}">{{vtName}}</a><br>
   <!--{{vtEndLoop}}-->
<!--{{vtEndUse}}-->
```

vtNext

Within a `vtLoop` block, switches to the next page even before `vtEndLoop` is encountered.

Usage	`<!--{{vtNext}}-->`
Version	Basic, Pro (from 1.2)
Parameters	–

You can use `vtNext` to create dynamic lists that extend over several columns, as in a picture gallery, where you might find several rows, each containing several pictures.

Example

```
<table>
    <!--{{vtLoop}}-->
    <tr>
        <td>{{LastName}}, {{FirstName}}</td>
        <!--{{vtNext}}-->
        <td>{{LastName}}, {{FirstName}}</td>
        <!--{{vtNext}}-->
        <td>{{LastName}}, {{FirstName}}</td>
    </tr>
    <!--{{vtEndLoop}}-->
</table>
```

See also

`vtLoop`

vtPackage

Is replaced by the name of the template package to which the current page's template belongs.

Usage	`{{vtPackage}}`
Version	Basic, Pro (from 1.0)
Parameters	–

The `vtPackage` placeholder is especially useful when certain kinds of pages within a `vtLoop` loop need to be handled in a special way:

Example

```
<ul>
<!--{{vtLoop}}-->
    <!--{{vtIf: {vtPackage}.eq.shop}}-->
        <li><a href="{{vtLink}}">{{vtName}}</a></li>
    <!--{{vtEndIf}}-->
<!--{{vtEndLoop}}-->
</ul>
```

This builds a list containing only pages whose templates belong to the "shop" template package.

See also

`vtPageType`

vtPageType

Is replaced by the page type or the name of the template on which the current page is based.

Usage	`{{vtPageType}}`
Version	Basic, Pro (from 1.0)
Parameters	–

The `vtPageType` placeholder is especially useful when certain kinds of pages within a `vtLoop` loop need to be handled in a special way: The page type carries the same name as the template on which the page is based, but without the ending ".html."

Example

```
<ul>
<!--{{vtLoop}}-->
   <!--{{vtIf: {vtPageType}.eq.ProductGroup}}-->
      <li><a href="{{vtLink}}">Group "{{vtName}}"</a></li>
   <!--{{vtElseIf: {vtPageType}.eq.Item}}-->
      <li><a href="{{vtLink}}">{{ItemNumber}}</a>
            {{Designation}}</li>
   <!--{{vtEndIf}}-->
<!--{{vtEndLoop}}-->
</ul>
```

This builds a list containing only pages of the "ProductGroup" or "Item" types. Depending on the page type, the list item will be constructed in one way or the other.

See also

`vtPackage`

vtPath

Repeats the HTML code that follows for each page in the path to the current page.

Usage	`<!--{{vtPath}}-->` ... `<!--{{vtEndPath}}-->`
Version	Basic, Pro (from 1.0)
Parameters	–

You can use `vtPath` to easily create a visual reference that shows visitors to your website where the current page is located in the hierarchy. In conjunction with the proper `vtLink` placeholders, `vtPath` can help to make navigation easier.

Example

```
<!--{{vtPath}}-->
    <a href="{{vtLink}}">{{vtName}}</a> |
<!--{{vtEndPath}}-->
{{vtName}}
```

vtPathIDs

Is replaced by a list containing all the page numbers in the path to the current page.

Usage	`{{vtPathIDs}}`
Version	Basic, Pro (from 1.3)
Parameters	–

The `vtPathIDs` placeholder is intended mainly for building dynamic navigation menus. It can be used to check whether a page listed in a navigation menu is in the path to the active page and should therefore be highlighted using some special markup. The list that `vtPathIDs` generates contains both the page numbers of the container pages and that of the current page itself.

Example

```
<!--{{vtSet: Path="{vtPathIDs}"}}-->
<!--{{vtUse:/}}-->
   <!--{{vtLoop}}-->
      <!--{{vtIf: {vtGet:Path}  .cn. {vtID}  }}-->
         <strong>{{vtName}}<br>
      <!--{{vtElse}}-->
         <a href="{{vtLink}}">{{vtName}}</a><br>
      <!--{{vtEndIf}}-->
   <!--{{vtEndLoop}}-->
<!--{{vtEndUse}}-->
```

This example creates a navigation menu encompassing all pages subordinate to the start page. The page at the head of the branch to which the current page belongs is shown in bold in the menu, and the other pages are formatted as hyperlinks.

See also

`vtID`

vtPortions

Is replaced by a navigation aid that facilitates paging through portions of lists.

Usage	`{{vtPortions}}`
	or
	`{{vtPortions:` *class*`}}`

Version	Basic, Pro (from 1.3)
Parameters	*class* CSS class to be invoked by each generated anchor element

The `vtPortions` placeholder is intended only for templates in which you use a `vtLoop` directive and you control the number of iterations using the `-size` parameter. In this situation, `vtPortions` generates a navigation aid that allows the reader to page through the individual "portions" or the list. If the total length of the list is less than the portion specified, `vtPortions` does not generate anything.

Please note that `vtPortions` must be placed after the `vtLoop` directive in the template. If you want to format the hyperlinks in a particular way, you can use the additional parameter to assign a CSS class of your choice to them. Each `<a>` element will then contain an additional class attribute whose value will be the name of the specified class.

Example

```
<p><b>List</b></p>
<ul>
<!--{{vtLoop: -size="5"}}-->
   <li><a href="{{vtLink}}">{{vtName}}</a></li>
<!--{{vtEndLoop}}-->
</ul>
<p>{{vtPortions}}</p>
```

This source code will be displayed in the following (or a similar) way in the browser:

Liste

- Article 367228
- Article 367229
- Article 367232
- Article 367236
- Article 367240

<< 1 2 3 4 >>

See also

`vtLoop`

vtPostValue

Is replaced by a value taken from a form submitted using the “post” method.

Usage	`{{vtPostValue:` *fieldname*`}}`
Version	Basic, Pro (from 1.2)
Parameters	*Fieldname* Name of the form field whose value is to be inserted

Form fields are generally processed through the indicated action (create, update, etc.). Using `vtPostValue`, you can also access form fields directly without regard to the action.

Example

```
<!--{{vtIf: {vtPostValue: SearchArgument}  .eq. }}-->
   <p>Please enter a search argument and then click "Go."</p>
<!--{{vtElse}}-->
   <p>Search for "{{vtPostValue:SearchArgument}}"</p>
   <ul>
   <!--{{vtSelect: -origin="/", -type="item",
             -filter="{content}  .cn. {vtPostValue:SearchArgument}"}}-->
      <!--{{vtLoop}}-->
         <li><a href="{{vtLink}}">{{Title}}</a></li>
      <!--{{vtEndLoop}}-->
   <!--{{vtEndSelect}}-->
   </ul>
<!--{{vtEndIf}}-->
```

See also

`vtGetValue`

vtRedirect

Modifies the current page's HTTP header so that the user's browser is redirected to another page.

Usage	`<!--{{vtRedirect:` *RedirectTarget*`}}-->`
Version	Basic, Pro (from 1.0)
Parameters	*RedirectTarget* Indication of the redirection target page.

If the target is an external website, specify a complete URL, starting with `http://`, `ftp://`, etc. If the target is a Virthos page, it can be identified using the same options as in `vtLink`.

Example 1

```
<!--{{vtRedirect: http://www.virthos.net }}-->
```

Example 2

```
<!--{{vtRedirect: /News }}
```

See also

`vtLink, vtHeader`

vtRepeat

Repeats the HTML code that follows for all items in a list of values.

Usage	`<!--{{vtRepeat:` *ListOfValues*`}}-->` ... `<!--{{vtEndRepeat}}-->`
Version	Basic, Pro (from 1.0)
Parameters	*ListOfValues* several values delimited by commas or line breaks

You can use `vtRepeat` to format several values delimited by commas or line breaks in whatever manner you like. The list can be entered directly or can be in the form of a placeholder that points to the actual list. When using a placeholder, be sure to separate its closing brace from the closing braces of the `vtRepeat` directive with a space.

Within the `vtRepeat` block, the `vtCurrentItem` placeholder can be used to reference the current list item.

Example 1

```
<ul>
   <!--{{vtRepeat: {Employee}  }}-->
   <li>{{vtCurrentItem}}</li>
   <!--{{vtEndRepeat}}-->
</ul>
```

Example 2

```
<select name="Weekday">
   <!--{{vtRepeat: Mon, Tue, Wed, Thu, Fri, Sat}}-->
   <option>{{vtCurrentItem}}</option>
   <!--{{vtEndRepeat}}-->
</select>
```

See also

`vtCurrentItem`

vtResult

Is replaced by the result of the last action or calculation.

Usage	`{{vtResult}}`
Version	Basic, Pro (from 1.3)
Parameters	–

After an action is carried out, `vtResult` contains a numeric value that indicates whether or not it was successful. A negative value means that an error occurred. A zero or a positive value means that the action was successful.

Besides the result of an action, `vtResult` can also contain the result of a calculation using the `vtCalc` directive. Each result of the `vtCalc` directive replaces the preceding one. If you need to refer to the result later, you must save it temporarily using the `vtSet` directive.

Example

```
<!--{{vtCalc: {NetPrice}  * 1.16 }}-->
<p>The total price is {{vtResult}}  Euros.</p>
```

See also

`vtCalc`, `vtSet`

vtScript

Removes the VirthosTalk and HTML elements that follow from the source code.

Usage	`<!--{{vtScript}}-->` ... `<!--{{vtEndScript}}-->`
Version	Basic, Pro (from 1.3)
Parameters	–

If a series of VirthosTalk directives is included in a template, the source code of the resulting page may often contain large gaps consisting of empty lines or spaces. To prevent this, the series of directives can be enclosed in a `vtScript` block. Between a `vtScript` directive and the following `vtEndScript`, Virthos suppresses any output to the browser.

Example

```
<!--{{vtScript}}-->

   We will first calculate the net amount:

   {{vtSelect: -origin="{vtGet:ShoppingCartID}" -type="item"}}
       {{vtSet: NetAmount="{vtEval: sum, Price}"}}
   {{vtEndSelect}}

   ... then the after-tax amount:

   {{vtCalc: round( {vtGet:NetAmount} * 1.16, 2 ) }}
   {{vtSet: AfterTax}}

<!--{{vtEndScript}}-->

<p>The total amount is {{vtGet: AfterTax}} EUR.</p>
```

vtSelect

Defines a selection of pages to be used in lists, evaluations, etc.

Usage	`<!--{{vtSelect: -origin="`*StartPage*`", -type="`*PageType*`", -filter="`*Filter*`", -max="`*Maximum*`", -sortfield="`*SortField*`", -sortorder="`*SortOrder*`"}}-->`
	...
	`<!--{{vtEndSelect}}-->`

Version	Pro (from 1.2)
Parameters	*StartPage* — Page where the selection starts *PageType* — Page type to be used in the selection *Filter* — Criteria that a page must meet to be included in the selection *Maximum* — Maximum number of pages to be selected *SortField* — Name of the field to be used for sorting the selection *SortOrder* — Indication of whether the pages are to be sorted in ascending or descending order, or randomly presented

The `vtSelect` directive has no direct effect on the HTML code that follows. It only serves to define the scope of other VirthosTalk directives and placeholders—in particular `vtLoop`, `vtEval` and `vtCountSelection`. There can only be one selection active at a time. This means that `vtSelect` blocks cannot be nested. In order to aggregate the results of several selections, you must first save them using `vtSet` and then retrieve them at the desired location using `vtGet`.

To specify the *StartPage*, the same options are available as for `vtLink` (see the chapter "Page References"). If this parameter is omitted, the page forming the current context is used as the start page. The selection can include only pages that are directly or indirectly subordinate to the start page.

The additional specification of a *PageType* results in the inclusion of only pages of a particular type—even if other filter criteria are satisfied. The page type is the same as the name of the template on which the page is based, without the ending ".html." If the `-filter` parameter contains template-specific placeholders, a page type must be specified.

The *Filter* consists of a comma-separated list of conditions, each of which must be in the following format:

placeholder `.`*relational operator*`.` *comparand*

The *placeholder* may be any placeholder that occurs in the specified page type or one of the type-independent placeholders `{vtName}` or `{vtPageType}`.

The *comparand* may be a literal value (string, number, date or time) or another placeholder. You should be aware that a placeholder on the right—in contrast to the placeholders to the left of the relational operator—is interpreted in the context of the `vtSelect` directive, and not in that of each individual page to be checked. It is therefore not possible to select all the pages where placeholders like {`date_a`} and {`date_b`} have the same content.

If a date is used as a comparand, it must be written in "dd.mm.yyyy" format, and time expressions must be in "hh:mm" format. If an hour or minute value is less than 10, it must contain a leading zero. Days and months do not require leading zeroes.

The available relational operators are described in the following table:

Operator	Explanation
`.eq.`	is equal to
`.neq.`	is not equal to
`.is.`	(string) is like
`.gt.`	(number, date or time) is greater than
`.gte.`	(number, date or time) is greater than or equal to
`.lt.`	(number, date or time) is less than
`.lte.`	(number, date or time) is less than or equal to
`.bw.`	(string) begins with
`.ew.`	(string) ends with
`.cn.`	(string) contains
`.ncn.`	(string) does not contain

The difference between `.eq.` and `.is.` is that `.eq.` used with numbers, dates, and times performs a numeric comparison, while `.is.` performs a character comparison, even when used with numeric values. For example, if a page contained a placeholder named `{Zip}` with the value "01023," the filter expression `{Zip} .eq. 1023` would cause the page to be selected, but the condition `{Zip} .is. 1023` would not select it.

If you provide several filter criteria, all pages that fulfill *all* the conditions would be selected—this is a logical "and" evaluation. It is not possible to combine conditions using "or" logic.

The *SortField* determines in which order the selected pages will be examined in a `vtLoop` directive. If a page type is specified, you may use any placeholder that occurs in the specified page type as a sort field. The predefined placeholders `vtName`, `vtPageType`, `vtCreationDate` and `vtModificationDate` are also available. Since the sort field indicates only the name, and not the content of the placeholder, braces may not be used.

For *SortOrder*, you can choose from three possible values:

`ascending`	sort by ascending value
`descending`	sort by descending value
`random`	display in random order (different each time)

If you do not specify a sort field or a sort order, the selection is sorted using the same criteria as the current page, as set forth in its specifications. If the `random` option is in effect, the sort field is ignored.

By providing a value for *Maximum*, you can limit the size of the selection. In that case, only the first n pages that meet the filter criteria, after applying the sort field and sort order, are selected. In contrast to the `-size` parameter used in a `vtLoop` directive, the maximum specification in `vtSelect` also affects the `vtCountSelection` and `vtEval` placeholders. Using a maximum value helps control the load on the server when you have a large site containing a significant number of pages.

Example 1

```
<!--{{vtSelect: -origin="/", -type="news", -max="3",
              -sortfield="Date", -sortorder="descending"}}-->
   <ul>
   <!--{{vtLoop}}-->
       <li><a href="{{vtLink}}">{{Headline}}</li>
   <!--{{vtEndLoop}}-->
   </ul>
<!--{{vtEndSelect}}-->
```

Creates a list containing the first three pages under the start page that are based on the "news.html" template. The list is sorted in descending order based on the "Date" placeholder.

Example 2

```
<!--{{vtSelect:-origin="//Download",
              -filter="{vtName}.ew.pdf"}}-->
   <p>There are {{vtCountSelection}}  PDF files available for download.</p>
<!--{{vtEndSelect}}-->
```

Returns the count of pages in the branch under "Downloads" (at the top level of the hierarchy) whose names end in ".pdf."

Example 3

```
<!--{{vtSelect: -origin="//Products", -type="item",
-filter="{Category}.eq.SpecialOffers, {Availability}.gt.0"}}-->
   <p>Special offers starting at {{vtEval: min, Price}} Euros</p>
<!--{{vtEndSelect}}-->
```

Selects all pages of type "item" in the branch under the "Products" page (at the top level of the hierarchy) that have a "Category" placeholder value equal to the word "SpecialOffers" and "Availability" equal to a number greater than zero. Then returns the lowest occurring value of the "Price" placeholder within all the selected pages.

See also

`vtCountSelection`, `vtEval`, `vtLoop`

vtSelf

Is replaced by a URL that invokes the URL of the current *virthos.php* file.

Usage	`{{vtSelf}}`
Version	Basic, Pro (from 1.3)
Parameters	–

You can use `vtSelf` to create hyperlinks that (in contrast to hyperlinks using `vtLink`) do not contain any information about the current session. A visitor who follows this kind of hyperlink is treated just like a visitor who comes to your website from outside (as from a search engine, for example). Another use for `vtSelf` is to create URL's that serve only informational purposes (see example).

Example

```
<p>URL of this page: {{vtSelf}}?{{vtID}}</p>
```

See also

`vtLink`

vtSession

Is replaced by the value of a session variable.

Usage	`{{vtSession: Name}}`
Version	Pro (from 1.3)
Parameters	*Name* Name of a session variable

Session variables can be assigned using the “updateSession” action. They retain their value for the entire duration of a work session, or until they are overwritten with another value. You can use a session variable to perform a simple form of password protection, as shown in the following example.

Example

```
<!--{{vtIf: {vtSession:Password}  .neq. ht2y8t}}-->
   Please enter your password first.
   <form action="{{vtLink:-act='updateSession'}}" method="post">
      <p>Password: <input type="password" name="Password"></p>
      <p><input type="submit" value="go"></p>
   </form>
<!--{{vtElse}}-->
   ...
<!--{{vtEndIf}}-->
```

See also

vtLink

vtSet

Stores a value in a variable.

Usage	`<!--{{vtSet:` *Name*`="`*Value*`"}}-->`
	or
	`<!--{{vtSet:` *Name1*`="`*Wert1*`", `*Name2*`="`*Wert2*`", ...}}-->`
	or
	`<!--{{vtSet:` *Name*`}}-->`

Version	Basic, Pro (from 1.3)
Parameters	*Name*: Name of the variable *Value* the value to be stored

You can use `vtSet` to temporarily store values that can be retrieved in another location within the same template (using the `vtGet` placeholder). You cannot save values that persist over several invocations of a page using `vtSet`; to do this, you must use session variables (see `vtSession`).

If you specify a name, but no value for a variable, Virthos assigns the content of the `vtResult` placeholder containing the result of the last form action or the last calculation made using `vtCalc`. Thus, the expression

```
<!--{{vtSet: Result="{vtResult}"}}-->
```

functions in the same way as

```
<!--{{vtSet: Result}}-->
```

Name and *Value* can be expressed either as a placeholder or a literal value. A combination of literals and placeholders is possible, as is the use of directives, as the following examples show:

Examples:

```
<!--{{vtSet: Title="{Headline}", Text="{Content}"}}-->

<!--{{vtSet: Greeting="Hello {Title}  {LastName}!"}}-->

<!--{{vtSet: List="{vtLoop}{vtName}, {vtEndLoop}"}}-->

<!--{{vtSet: Var_{vtGet:Number}="abcde"}}-->
```

See also

`vtCalc`, `vtGet`, `vtResult`, `vtSession`

vtUse

Changes the context in which the placeholders and page references that follow are interpreted.

Usage	`<!--{{vtUse:` *Page*`}}-->` ... `<!--{{vtEndUse}}-->`
	or
	`<!--{{vtUse:` *Page*`,` *Pagetype*`}}-->` ... `<!--{{vtEndUse}}-->`

Version	Basic, Pro (from 1.1)
Parameters	*Page* Indication of the page that will form the new context *Pagetype* Directive calling for the page with the indicated type to be created if it does not exist

In addition to content from the current page, you can use `vtUse` to incorporate content from any other page in the current template. To refer to the page that should form the new context, the same options are available as with `vtLink` (see the chapter "Page References"). Nesting of `vtUse` directives is permitted as well—meaning that relative page references in an "inner" `vtUse` directive are interpreted in the context established by the "outer" `vtUse` directive (see examples).

Specifying a *Pagetype* allows the page being referenced to be automatically created if it does not yet exist. This is possible only if the first parameter is a page name (not a page number or a path). Virthos then checks whether the page forming the current context has a subpage with the specified name. If not, a new page with the specified name and page type is created.

Example 1

```
<!--{{vtUse: //CompanyData}}-->
    <p>Switchboard: {{Phone}}</p>
<!--{{vtEndUse}}-->
```

Example 2

```
<!--{{vtUse: Comments, commentlist}}-->
    <p>Notes:</p>
    <ul>
        <!--{{vtLoop}}-->
        <li><a href="{{vtLink}}">{{Subject}}</a> {{Date}}</li>
        <!--{{vtEndLoop}}-->
    </ul>
<!--{{vtEndUse}}-->
```

See also

the chapter "Page References"

vtUser

Is replaced by information about the current user.

Usage	`{{vtUser:` *Property*`}}`
Version	Pro (from 1.3)
Parameters	*Property* Type of information desired

You can use `vtUser` to access information that is saved as part of the current user's settings. This allows you to design templates that display different content to different users.

Any of the following values can be a *Property*:

Property	Meaning
`id`	the unique user number
`name`	the user's login name
`realname`	the user's full name
`admin`	whether the user has Administrator rights
`startPage`	the user's predetermined start page
`workspace`	whether the user is automatically taken to Virthos Manager after logging in
`groups`	a list of the groups to which the user belongs

For the `admin` and `workspace` properties, the placeholder is replaced by a "1" if the user has the property, or a "0" or a null value if he does not.

Example 1

```
<h1>Welcome, {{vtUser:name}}!</h1>
```

Example 2

```
<!--{{vtIf: {vtUser:groups}.cn.Authors }}-->
   <a href="{{vtLink:-met='vtedit'}}">Edit page</a>
<!--{{vtElse}}-->
   <i>Editing permitted for authors only.</i>
<!--{{vtEndIf}}-->
```

See also

`vtCurrentAccessRights`

vtUserAgent

Is replaced by information about the browser being used.

Usage	`{{vtUserAgent}}`
	or
	`{{vtUserAgent:` *Property*`}}`

Version	Pro (from 1.3)
Parameters	*Property* Type of information desired

You can use `vtUserAgent` to determine what type of browser the current user is using and what operating system. Any of the following values can be used as a Property:

Property	*Meaning*	*Possible Values*
`browser`	Type of browser	opera, ie, omniweb, konqueror, safari, mozilla, other
`version`	Browser Version	e.g. "5.0"
`os`	Operating System	Win, Mac, Linux, Unix, OS/2, other

If you omit *Property*, `vtUserAgent` acts as if you had supplied the `browser` parameter.

Example

```
<!--{{vtIf: {vtUserAgent} .neq. ie}}-->
   <p>This website has been optimized for Internet Explorer.
     Using a different browser could cause
   errors in the way pages are displayed.</p>
<!--{{vtEndIf}}-->
```

vtUsername

Is replaced by the current user's login name.

Usage	`{{vtUsername}}`
Version	Pro (from 1.1)
Parameters	–

This placeholder should no longer be used. The same results can be achieved with the much more versatile placeholder `vtUser`.

See also

`vtUser`