

# **Spam Guard at TYPO3**

*by Bernd Warken at Netcos AG*

<b>Chapter 0: Introduction</b>	<b>3</b>
<i>0.1 Preface</i>	3
<i>0.2 License</i>	3
<b>Chapter 1: Activation and Configuration</b>	<b>4</b>
<i>1.1 config.spamProtectEmailAddresses</i>	4
<i>1.2 config.spamProtectEmailAddresses_atSubst</i>	4
<i>1.3 config.spamProtectEmailAddresses_lastDotSubst</i>	4
<b>Chapter 2: Application</b>	<b>5</b>
<i>2.1 Content</i>	5
<i>2.2 TypoScript Template</i>	5
<b>Chapter 3: Advanced Methods</b>	<b>8</b>
<i>3.1 Inserting of Graphics</i>	8
<i>3.2 JavaScript for Letters</i>	8
<i>3.3 Complete Example</i>	9

# Kapitel 0: Introduction

## 0.1 Preface

TYPO3 has a guard against spam emails. By this, the arising email addresses are changed unrecognizably in the HTML source code and a *JavaScript* program is run when the email address is clicked. The spam guard is activated and configured by two *TypoScript* commands. Email addresses can be used in the content and in *TypoScript*. But then some points have to be covered.

## 0.2 License

This document is stored in the file **spamguard\_en.sxw**. The actual version is **1.1** of 27 February 2008. Author is **Bernd Warken** at [netcos AG](http://netcos.de).

The document runs under the license **GPL (GNU General Public License)** version **3**. The original text of this license is available under <http://www.gnu.org/licenses/gpl.html>.

Copyright © 2007, 2008 Bernd Warken.

# Chapter 1: Activation and Configuration

Activation and configuration of the spam guard is very simple, these are done by two *TypoScript* commands.

## 1.1 *config.spamProtectEmailAddresses*

The activa is done by the *TypoScript* command **config.spamProtectEmailAddresses**. This command is added to the TYPO3 template. As value, either a number from **-5** to **5** can be chosen or the word **ascii**.

```
config.spamProtectEmailAddresses = number
```

If a *number* is used then only **-5** to **5** can be used. By this process, the ASCII code of the characters in the email address are shifted by this region in the HTML source text. For example, the character **z** has the ASCII code **#122**. When the value **1** was chosen as *number* this character is shifted to ASCII code **#123**, the character **{**, the left curly brace.

This shifting is uncritical for the values **-5** to **1**. But when a *number* of **2** to **5** is chosen then the result of ASCII value **#124**, the letter **|**, can occur. But this vertical bar is a special character for TYPO3 and should not be used. Therefore it makes sense to restrict to the uncritical values.

When using a *number* a *JavaScript* program is additionally run. The link of the email address points to this script, such that a usage by spam bots is almost impossible.

```
config.spamProtectEmailAddresses = ascii
```

When the value is set to **ascii** then all characters of the email address will simply replaced by their ASCII code in the HTML source text. But no *JavaScript* program is run when the email address is clicked. It is your choice whether you use this value or prefer a *number*.

## 1.2 *config.spamProtectEmailAddresses\_atSubst*

Additionally the character **@** within the email address can be replaced by a different text. A good choice for the alternative text is **(at)**. When the command

```
config.spamProtectEmailAddresses_atSubst = (at)
```

is set in the template then the example email address [test@test.com](mailto:test@test.com) is replaced by **test(at)test.com**. This replacement is done automatically. But the email address is always written together with the **@** character, without worrying about the replacement. TYPO3 takes the task to apply the replacement.

## 1.3 *config.spamProtectEmailAddresses\_lastDotSubst*

Additionally the last dot-character **.** within the email address can be replaced by a different text. Some choice for the alternative text is **(dot)**. When the command

```
config.spamProtectEmailAddresses_lastDotSubst = (dot)
```

is set in the template then the example email address [test@test.com](mailto:test@test.com) is replaced by **test(at)test(dot)com**. This replacement is done automatically. But the email address is always written together with the **@** character and the dot, without worrying about the replacement. TYPO3 takes the task to apply the replacement.

The real advantage of this field comes with the handling of JavaScript, as explained in a later chapter.

This completes the configuration of the spam guard.

## Chapter 2: Application

To apply the spam guard, some things are to be noted. The *TypoScript reference* explains that the spam guard for email addresses is applied in *Typolink*. In normal content, this is done by inserting links; in *TypoScript* template, this is activated by a corresponding command.

### 2.1 Content

In TYPO3, the text content of a normal page is edited with an *RTE editor*. An email address that is simply added as text is not recognized as such. It is necessary to insert the email address as link, i. e. mark the text of the email address and push the button *Insert Web Link*.

This creates a new window. Therein choose the area *Email*. In the field *Email Address*, insert the right name of the email address without replacements, including the @ character. Push the button *Set Link*. This activates the spam guard for this email address.

### 2.2 TypoScript Template

When an email address is repeated on each page, e. g. in a footer, this must be represented in the template. We assume that a *TemplaVoilà* data structure with **lib.footer** was generated; at classical TYPO3, a marker is used instead.

We declare **lib.footer** as **COA**, such that we can split the code into several parts. We assign a number value to all elements before the email address, the next number value to the email address, and a third number value to all parts after.

We insert the *Typolink* command **typolink.parameter** with the right email address as value into the are with the email address..

A text example looks like

```
lib.footer = COA
lib.footer {
    10 = HTML
    10.value (
        <p>text before email
    )
    20 = TEXT
    20.value = email@testenterprise.com
    20.typolink.parameter = email@testenterprise.com
    30 = HTML
    30.value (
        text after email</p>
    )
}
```

An example with a table would look like

```
lib.footer = COA
lib.footer {
  10 = HTML
  10.value (
    <table>
      <tr>
        <td>
          <a href="http://www.testenterprise.com">test enterprise</a>
        </td>
        <td>email:
      </tr>
    </table>
  )
  20 = TEXT
  20.value = email@testenterprise.com
  20.typolink.parameter = email@testenterprise.com
  30 = HTML
  30.value (
    </td>
  </tr>
</table>
)
}
```

Please that the values for HTML use parentheses.

## Chapter 3: Advanced Methods

The spam-robots can be outsmarted by inserting invisible images and generating some of the letters by JavaScript functions. By that the email address in the HTML code is heavily distorted, without changing the display.

### 3.1 Inserting of Graphics

If you insert an image within the text the copy & paste is suppressed. As image we take the file **clear.gif**, which is part of the actual TYPO3 version. It is situated in the main directory of the TYPO3 code. This image has a width and height of only 1 pixel and has a transparent color, such that it is invisible and needs only little space.

To include it, the following code must be inserted within the text:

```

```

The value of **alt** must be set, in order that no automatic value suitable for spam robots is generated.

It is best to define this code as a TypoScript constant, because the code needs a lot of space and creates very long lines. As constant name, I choose here **img\_invis**.

```
img_invis = 
```

It is easy and quite short, to embed this constant by **{\$img\_invis}** into the TypoScript text. Possible embedding places are **config.spamProtectEmailAddresses\_atSubst** and **config.spamProtectEmailAddresses\_lastDotSubst**. In section 3.3, we show a complete example.

### 3.2 JavaScript for Letters

Email addresses can be identified by spam robots through the **@** symbol and some dot within a word. TypoScript allows to replace these two symbols by something else.

First we decide to replace the **@** symbol by the text **(at)**. Next we let all letters be generated by JavaScript, such that none of the letters can be directly seen in the HTML source code; only the call of the JavaScript function is displayed here. This makes it much harder for spam robots to find an email address.

The definition of the JavaScript functions is done in the header of the HTML file. This can be programmed in TYPO3 by using **page.headerData** in the template setup. This needs additionally a number. It can be chosen arbitrarily, but it should not occur with other values of **page.headerData**. We choose number **250**.

As an example, we represent a single JavaScript function **letter** for generating a single letter.

```
page.headerData.250 = TEXT
page.headerData.250.value (
<script type="text/javascript">
<!--
function letter() {
    document.write('a');
}
// -->
</script>
)
```

This JavaScript function can be used within a TypoScript text by the code

```
<script type="text/javascript"> letter() </script>
```

In section 3.3, we provide a complete example with several **letter** functions.

### 3.3 Complete Example

Our example includes the replacement of **@** to **(at)**, the displaying of this value and the dot within the email address by JavaScript, and the encircling of all letters with an invisible picture.

First we define the JavaScript functions for generating single letters. For each letter that has to be generated, we provide a function of its own. You should not just take over the names of these functions, in order that spam robots cannot zero in on this.

These functions are defined in the template setup.

```
page.headerData.250 = TEXT
page.headerData.250.value (
<script type="text/javascript">
<!--
function letter_at1() {
    document.write('(');
}
function letter_at2() {
    document.write('a');
}
function letter_at3() {
    document.write('t');
}
function letter_at4() {
    document.write(')');
}
function letter_dot() {
    document.write('.');
}
// -->
</script>
)
```

Next we define constants in template constants for the long code:

```
img_invis = 
letter_at1 = <script type="text/javascript"> letter_at1() </script>
letter_at2 = <script type="text/javascript"> letter_at2() </script>
letter_at3 = <script type="text/javascript"> letter_at3() </script>
letter_at4 = <script type="text/javascript"> letter_at4() </script>
letter_dot = <script type="text/javascript"> letter_dot() </script>
```

Now we use these constants to define the TypoScript anti-spam variables in the template setup:

```
config.spamProtectEmailAddresses = 1
config.spamProtectEmailAddresses_atSubst = {$img_invis} {$letter_at1} {$img_invis}
{$letter_at2} {$img_invis} {$letter_at3} {$img_invis} {$letter_at4} {$img_invis}
config.spamProtectEmailAddresses_lastDotSubst = {$img_invis} {$letter_dot} {$img_invis}
```

Please remove the line break at **\_atSubst**.

For testing, insert an email address into some page of your TYPO3 code. Have a look at the HTML source code of this page. Then you will see that it is hard to recognise the email address.