FORMSPAMMERTRAP
THE BEST SPAMBOT-BLOCKING CONTACT FORM

IMPLEMENTATION INSTRUCTIONS AND MORE – VERSION 17

The FormSpammerTrap contact form is built to deny bot access to your comment/contact form. It uses several techniques that are very effective in blocking contact form spam. You can test the techniques on the Contact page at https://www.FormSpammerTrap.com .

LICENSE AND COPYRIGHT INFORMATION

Although the program is free, if you wish to donate any amount for my efforts in this contact form, please see the information at www.formspammertrap.com and/or www.securitydawg.com .

WELCOME!!

This file contains all implementation instructions, including how to customize how the form works, what is displayed, and even more. Carefully read this so that you can implement this effective contact spam blocking solution on your site. The change log is in a separate file. See separate documentation for implementation on a WordPress site.

There may seem to be a lot of information, but basic implementation is pretty simple. *It can take as little as just a few minutes to implement a spam-free contact form on your site.* There are additional features that you may consider if you want even more control of your contact form. You can even add your own fields to the form. Some additional features may require a bit more technical expertise, but we're here to help. (We can even arrange to 'do the geeky' if you want.)

No matter how you implement - the basic/simple way, or advanced features - you will find that your contact form spam will stop.

If you want us to do all of the work, we will provide that service for a nominal fee. See the FST site for details, and contact us. If you have any questions or concerns about anything, or you want to discuss letting us implement the form on your site, please use our Contact page on our site at https://www.FormSpammerTrap.com site.

We use “FST” as an abbreviation for “FormSpammerTrap” in this document. There are references to YOURDOMAIN.COM or EXAMPLE.COM in this document and the PHP files; change those to your actual domain name.

SET UP AND INSTALLATION ON YOUR SITE

Two ways to get the no-spambot contact on your site; both are described below:

1. Use our sample contact page. It won't look like the other pages of your site, but it is quick – you can use it if you want to test FST.
2. Use your own contact page, with your site's 'look', and insert the FST pieces. Still easy, and the contact page will look like the rest of your site. The easiest way to do that is to make a copy of your existing contact page, and modify it as explained later in this document. Even complex forms can be easily changed to use FST bot blocking techniques.

If you have a WordPress site, there are separate instructions for creating a template that will work with your theme. We encourage you to review all of the information here, as it is important to understand how to configure and install FST. Then look at the WordPress-specific document (separate file) for additional information.

NOTE – PHPMAILER IS REQUIRED FOR VERSION 17

Version 17 requires adding the three PHPMailer files to your site. This version uses PHPMailer functions instead of the basic PHP mail() command to send the contact form email. PHPMailer is a much easier way to send mail, especially with advanced form features like attaching files. PHPMailer is a well-established and maintained project hosted on GitHub; there are no risks involved in using it. Note that many sites (including WordPress-based sites) already have PHPMailer installed.

The PHPMailer files are not included in the ZIP file you got from the FST site. You will need to get and install those files onto your site. It's an easy process. If you have a WordPress-based site, PHPMailer is already included, so you can skip this part of the implementation. FST will check to make sure that PHPMailer is properly installed on your site as part of the 'Sanity Check' performed on your form page.

To install PHPMailer on your site, follow these steps:

1. Download these three files from the https://github.com/PHPMailer/PHPMailer site. You'll find these three files in the 'src' folder. These are the only three additional files needed to for PHPMailer; you can look at the other files. There are many articles on using PHPMailer if you want to get into the geeky details:
    a. Exception.php
    b. PHPMailer.php
    c. SMTP.php
2. On your web site, use your hosting's File Manager to create a subfolder called 'phpmailer' (lowercase) off of your web sites 'root' folder. That folder is often called 'public_html'.
3. Copy the above three files to that 'phpmailer' folder.

That's all that is needed. Those three files are needed by FST. If they aren't available in the 'phpmailer' folder, or don't work properly, your form will display the "Sanity Check" error message.

THE VERY EASIEST AND BASIC WAY TO IMPLEMENT

Now that you've installed PHPMailer, the rest of these implementation instructions will give you a very basic form. It won't look like the rest of your site, but it will work just fine. You might use it to test out things. And there's a way to integrate it into the 'look' of your site – those instructions follow this very basic technique.

Here's the very basic implementation. The `formspammertrap-contact-basic.php` is an example of this very basic form.

1. Un-zip the files to a new folder on your local computer.
2. Use your favorite text editor (not word processor; we like NotePad++ - it's free to use) and open the `formspammertrap-contact-basic.php` file. Make one change to the `FST_XEMAIL_ON_DOMAIN` value in the `FST_MORE_FIELDS` function. Enter an email address on your site's domain. That's the person that will get the submitted form, so it must be a valid email address where you get email for your site. Save the file with a name like `contactus.php` (that's so you don't change the original file, although it's also in the ZIP file.)
3. Copy the new `contactus.php` and `formspammertrap-contact-functions.php` to your web site. (If you already have a `contactus.php` file on your site, use another name. We'll assume a file name of `contactus.php` in these instructions.)

That's it! Now go to that new page on your site, with a URL of `https://www.yoursite.com/contactus.php` (using your site's domain name, and the name of the contact page you created above). You'll see the form on your site. Fill out the form, and send it. The email will go to the email account that you specified. Easy-peasy!

Of course, the form doesn't look like the rest of your site. But it will stop spam bots!

To get the FST form to look like the rest of your site, see the next section.

MAKING A CONTACT PAGE THAT LOOKS LIKE YOUR SITE

This takes a bit more effort – and technical knowledge - to do, but it can be done. Basic web programming skills are what you need for this. We can provide technical help for your site; just contact us for details. We'll even do all the work for a nominal charge. Just contact us via the FST site. Of course, email support is always free.

To make your own contact form, follow these steps. This process will allow your contact page to have the visual 'look' of your site. This assumes that your contact form has the basic four fields – name, email, subject, and message. If your form is more complex, there is a way to add FST features to your complex form – see below.

The first step is to make a copy of your current contact page. Then remove everything between the 'form' tags (including the 'form' tag) – that's the old form. You might want to put some text like "contact form goes here" instead of the form so you know where the form will go.

Give the file a PHP extension, as in `CONTACTUS.PHP` . (Don't overwrite any existing file.) The result will be a contact page – minus the form – with your site's visual look. Make sure that the file extension is '`PHP`' – '`HTML`' will not work. Upload the new contact form page to your site and test it out. It should look just like your old contact form – but without the form (and with the text you added). And without the FST features, which you will fix next.

Once you have your basic contact page with your site's look (without the form), follow these steps to add the form and other necessary parts. Edit the `CONTACTUS.PHP` page again with your favorite text editor.

1. In your `CONTACTUS.PHP` file, add a section with the code **below** all of the existing page code.

```
<?php
function FST_MORE_FIELDS() {
	/* this section redefines some of the variables defined
	in the main functions file. This allows you to use the one main
	function file, but have several different forms that
	use the basic FST functions .
	Note that you need to make each the variables GLOBAL before you
set them
	 */

	// GLOBAL all of the customization override variables that you
will use !!
	// change or adjust values as needed
```

```
            // Note: the first setting below is important! It specifies the
email of who will recieve the submitted form. Change it to an email on your
domain (not another domain)

            global $FST_XEMAIL_ON_DOMAIN; // this is who will get the form;
change this to an email address on your domain

            $FST_XEMAIL_ON_DOMAIN = "youremail@yourdomain.com";
```

There is only **one** required customization field to change for your site - `FST_XEMAIL_ON_DOMAIN` . That is the email for who will send receive the submitted form. Set the proper value, and save the file. Make sure the email address exists on your domain. (If your domain is 'example.com', then the email address should be 'someone@example.com'. It cannot be 'someone@anotherdomain.com .)

All other customization fields are optional, and explained in the functions file, along with this documentation. The optional fields allow the experienced programmer to customize a lot of the FST form and processing. But you only need to change that one setting for a basic form to work. (If you don't change that setting to an email on your domain, you'll see an error message when you load the page – that's part of the 'sanity check' feature in FST, which makes sure that basic settings are correct.)

Using the `FST_MORE_FIELDS()` function in your form to set customized values makes it easier to upgrade to the next version of FST (never modify the main FST functions file). And you can have multiple contact forms that use FST, each with its own customized values. We recommend that technique. We don't recommend making changes to settings in the FST functions file – making changes to the FST functions file will make it difficult to upgrade to the next version of FST. We tell you this three times.

Save your new contact page file locally with the changes that you made.

3. Edit your new page again. At the *very beginning* of your contact page code (don't put any white space or blank lines before this part), put this code block:

```
<?php
session_start() ;
include_once ("formspammertrap-contact-functions.php") ; // (required)
?>
```

(Watch out for those tilted quote characters if you copy/paste these code blocks; replace them with the straight-style quotes in your text editor, or the code won't work.)

4. In that same file (your `CONTACTUS.PHP` file), insert the following lines in the HEAD section (just before the closing HEAD tag).

```
<?php
formspammertrap_contact_css() ; // required
formspammertrap_contact_script() ; // required
```

```
?>
```

5. Inside the BODY tag, place this code where you want the contact form to appear:

```
<?php formspammertrap_contact_form() ;?>
```

You can place additional text or HTML code before or after that line, if you wish.

That's all that is needed! Upload your new `contactus.php` file, and the `formspammertrapcontact-functions.php` file to your site (make sure both files are in the same folder – and that you also installed the PHPMailer files as described above). Test the form. Huzzah! It works!

> If there are problems, go back and verify that you entered the commands correctly. Watch out for curly quote characters. You might see a 'Sanity Check' message on the screen that will contain additional issues to correct.

That's all that is needed. If you want to get fancier, keep reading (especially the next section). But the above procedures will result in a basic contact form that the spam bots can't spam.

CUSTOMIZING THE MAIN FST SETTINGS

There's a separate file included in the FST zip file with all of the customization codes. There's a lot of them – FST is built to allow you to change just about everything – adding fields to a form, storing names/emails in a database, change colors, use a different language on your form, and more. If you are interested, take a look at the 'all settings' PDF. All of the codes have explanations and default values.

CUSTOMIZING YOUR CONTACT FORM OVERVIEW

You can further customize your form, like adding fields, an upload button, checkboxes, or whatever. The following instructions are a bit more advanced, but within the ability of most web site programmers. We can do all of the modifications and testing for a nominal fee; just contact us through the FST site (of course, email support is free). All of this is optional, but the customization options allow you to have a form that looks and works like you want.

1. On your contact form page, add additional form fields using the instructions below.
    a. Add fields or settings within the `FST_MORE_FIELDS` function. Don't forget to `GLOBAL` any variable before you set its value.
    b. Process any added fields, including any field verification code, with the `FST_CUSTOM_AFTER_SUBMIT()` function in your custom contact code file.
2. Add any needed additional CSS code at the top of your contact page if you want to adjust the CSS to your own needs. Add your CSS *after* the FST CSS function calls in the previous step 4.
3. Upload your changed contact page to your site root folder. Make sure it has a `php` file extension.
4. Make sure that you upload the `formspammertrapcontact-functions.php` and your `contactus.php` files to your site root folder. Make sure that you made the needed customizations.
5. If you use our files, please include our copyright/license terms in each of the files.
6. Test and enjoy less (or even no) contact form spam.

Note that the FST main functions file sanitizes the `$_GET` and `$_POST` arrays for you.

ABOUT ALL OF THE CUSTOM SETTINGS

There are many custom settings you can use to modify how the form works, and how it processes the data. Each customization setting in the `FORMSPAMMERTRAPCONTACT-FUNCTIONS.PHP` file has details of what the setting does, the default values, and the options for changing it.

The `FORMSPAMMERTRAP-ALL-SETTINGS-V-17.PDF` file contains all of the customization functions that are in the main functions file.

MORE ABOUT ADDING ADDITIONAL FIELDS TO THE BASIC CONTACT FORM

The latest versions (starting with version 7) use a new and we think simpler way to change customization values from the values set in the main FST file. Again, an advanced topic, so watch out for dragons! The basic process still works, though. The advantage to this new process is that you don't have to modify the FST core code files. Version 8/9 added the additional field types, and even more customization is available in version 17. You've already seen this process in the previous section.

This section shows how to add additional fields to the form. For an example on how to use these extended functions, see the `FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP` file. You can upload that file to your site to see the possibilities of adding new fields to the form.

There are three parts to this process of adding additional fields:

- Adding the additional fields to the form.
- Validating the fields.
- Processing the fields.

Adding the additional fields

In your customized contact form, create a `FST_MORE_FIELDS()` function (don't change the name of the function). Adjust the variable values for your needs. Place this new code at the bottom of your contact page source code file. See the sample extended contact form – `FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP` - for details on each of the types of fields that can be added, and the requirements for each type of field that can be added.

Here's the basic syntax of how to use the FST_XCUSTOM_FIELDS array to set up additional fields for your form. Complete examples are in the `FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP` file.

```
/*
FST_MORE_FIELDS() - allows you to change values of customization variables
before they are turned into constants. An easy way of testing or using
variations of those variables with separate form pages.

Also allows you to add additional fields to the form. They are placed above
the message textarea.

PARAMETERS: none
```

```
RETURNS: NULL
The $FST_XCUSTOM_FIELDS array elements
      $FST_XCUSTOM_FIELDS[] = array(
      'ORDER' => '1', // order number for the fields; start at 1 (not used but
      must exist)
      'NAME' => '', // name/id/label of the field
      'TYPE' => '', // type of field - TEXT, CHECKBOX, SELECT, FILE, OPTION or
      any other valid input type
      'MAXCHARS' => '', // character width of field
      'MAXLENGTH' => '', // max number of characters for the field
      'REQUIRED' => true, // true if required, false if not required; must be
      false if checkbox field
      'PLACEHOLDER' => '', // placeholder text for input area
      'LABELMSG' => '', // text to display to the left (label area) of the
      field )
      'AFTERMSG' => '', // text to display after the checkbox (ignored
      otherwise)
      'CODEBLOCK' => '',  // block of code to insert; valid for option/dropdown
      type only.
      'CLASS' => '',      // class name for the field, optional
      'VALUE' => '',      // the default value for the field; can be blank but
must exist
      ;
 */
```

With this function, you can define an array for additional fields. You can also override customization settings as previously discussed. That allows you to have a customized form that uses the basic FST 'core' to display (and block spambots) your contact form.

So, to add some additional fields and override customization values, that function would look something like this: (Note that there are two single-quote characters for empty strings; they just look like a double-quote.) Add the items to the array in the order you want them to appear. This will add an additional text field.

```
function FST_MORE_FIELDS() {          // this is an example
// an example of overriding customization values


// all overrides need to be GLOBALs
global $FST_XFORMTOP_MESSAGE, $FST_XEMAIL_ON_DOMAIN, $FST_XSHOW_SUBMIT;


// here are the new/override values
$FST_XEMAIL_ON_DOMAIN = "admin@formspammertrap.com"; // who sends and gets the
email
$FST_XFORMTOP_MESSAGE = "A Customized Message above the form for this page
only";
$FST_XSHOW_SUBMIT = true; // normally false, will add the POST values to the
email
```

```
        // example of adding fields to the form. This will add a TEXT type field
       global $FST_XCUSTOM_FIELDS; /* required in this function to make the additional
fields array available to the form */
        $FST_XCUSTOM_FIELDS[] = array(
        'ORDER' => '1', // order number for the fields; start at 1
        'NAME' => 'test', // name/id/label of the field
        'TYPE' => 'TEXT', // type of field - TEXT, CHECKBOX, TEL, FILENAME, or other
        valid input types
        'MAXCHARS' => '30', // character width of field
        'MAXLENGTH' => '30', // max number of characters for the field
        'REQUIRED' => true, // true if required, false if not required; must be false
        for checkboxes
        'PLACEHOLDER' => 'test field', // placeholder text for input area
        'LABELMSG' => 'the test', // text to display to the left (label area) of the
        field)
        'AFTERMSG' => '', // text to display after the checkbox (ignored otherwise)
         'CODEBLOCK' => $codeblock,     // the codeblock for the select/option if type
is SELECT (see example extended form for the requirement for this codeblock)
       'CLASS' => '',      // class name for the field, optional
       'VALUE' => '',      // the default value for the field; can be blank but must
       exist
        ;


        return;
        }
```

The FST_XCUSTOM_FIELDS array is used to create the additional fields on the form. You can have multiple additional fields; just define additional form fields with another added FST_XCUSTOM_FIELDS array. Examples for each supported field type are in the FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP file.

Version 17.10 added support for hidden fields (TYPE => 'HIDDEN'). You will probably want to assign a VALUE for a hidden field. It won't be shown on the form, but will be processed when submitted.

The FST code will loop through the arrays to build and display the additional input fields of the form. The additional fields will be displayed above the message text area. You can change the location of these additional fields with the FST_CUSTOM_FIELDS_LOCATION setting.

The array values will be tested for proper values. If any errors are found, a fatal error will happen, and the page will 'die'. Only when the fields are properly configured (you don't see the any form field errors on form submit) should you consider the additional fields (and the form) ready to be used on a production site.

You need to add your array values as part of your custom contact form page. Each array should be defined in the order you want them to appear on the form.

The mailed message will include the additional form contents when the form is submitted. The message will show the additional fields, showing the LABELMSG value and then the contents of that input field.

If you need examples of the possible added fields, check out the `formspammertrap-sample-contact-extended.php` file.

ALL AVAILABLE OPTIONS FOR EXTRA FIELDS

The 'extended' contact form example (`formspammertrap-sample-contact-extended.php`) contains many of the available options. See that file for usage examples. Note that there are special considerations for some additional field types.

OPTION/DROPDOWNS

If you add a dropdown type field, you will also need to include the HTML codeblock for the option dropdown. This is defined separately. Here is an example of how to add the field, and define the codeblock. This example is taken from the 'extended' contact form file.

```
		/* DROPDOWN SELECT   example
		- allows a drop-down list that you define


		- the CODEBLOCK element is the entire input item to output; optional
		- the CODEBLOCK can be a return value of properly formatted OPTION values from a function
		- watch out for single-quotes inside the value of this array variable; escape them if you need single-quote characters
		- entire CODEBLOCK is output in the form
		- no syntax checking; that's up to you
		 */
		// put the select/options codeblock here. You must use htmlspecialchars on the codeblock so that it will be properly rendered on the contact page.
	// syntax is not checked, so verify it works. Be careful about embedded quotes.
		$codeblock = htmlspecialchars('<select name="myOptions" size=6>
			<option value="">Select Your Favorite Vehicle</option>
			<option value="VW Bug">VW Bug</option>
			<option value="Chevy Camaro">Chevy Camaro</option>
			<option value="Chevy Astro Van">Chevy Astro Van</option>
			<option value="Toyota Camry">Toyota Camry</option>
			<option value="Big Giant Honking RV">Big Giant Honking RV</option>
		</select>');
		$FST_XCUSTOM_FIELDS[] = array(
			'ORDER' => '1', // order number for the fields; start at 1
			'NAME' => 'myOptions', // name/id/label of the field
			'TYPE' => 'SELECT', // type of field
			'MAXCHARS' => '30', // character width of field
			'MAXLENGTH' => '30', // max number of characters for the field
			'REQUIRED' => false, // true if required, false if not required
```

```
                    'PLACEHOLDER' => 'test field input', // placeholder text for
input area
                    'LABELMSG' => 'Choose one item', // text to display to the left
(label area) of the field
                    'AFTERMSG' => 'One item only ', // message after the dropdown list
                    'CODEBLOCK' => $codeblock,  // html code for select/option
dropdown (previously defined)
                    'CLASS' => '',      // class name for the field, not used, as your
code should contain any needed CLASS statement
                    'VALUE' => '',      // the default value for the field; can be
            blank but the element must exist
            'READONLY' => true,        // if the field is read-only (not changeable
by visitor)
            );
```

FILE UPLOADS

Adding an optional FILE type will display a browse button for the visitor to select files (one or more) from their local computer. Those files will be attached to the message you receive.

Here's an example from the 'extended' sample file.

```
            /* UPLOAD FILE example

            - this adds a file upload type field; code will be similar to this
            <input type="file" name="myFile">
            - will output a browse button to select the file to upload
            your processing code will need to take care of processing the uploaded
file
            - there is no validation of the file type that is uploaded
            - this sample field does not include a CODEBLOCK array value
            - WARNING - the NAME parameter must be 'fst_uploadfile', or the file will
not be processed !
             */
            $FST_XCUSTOM_FIELDS[] = array(
            'ORDER' => '1', // order number for the fields; start at 1
            'NAME' => 'fst_uploadfile', // DO NOT CHANGE - or file will not upload !!
name/id/label of the field - defined in the main functions file
            'TYPE' => 'FILE', // type of field
            'MAXCHARS' => '30', // character width of field
            'MAXLENGTH' => '30', // max number of characters for the field
            'REQUIRED' => false, // true if required, false if not required
            'PLACEHOLDER' => 'test field input', // placeholder text for  input area
            'LABELMSG' => 'Select file to upload with the browse button', // text to
display to the left (label area) of the field
            'AFTERMSG' => 'Here is the file you selected. ',
            'CLASS' => '',      // class name for the field, optional
            'VALUE' => '',      // the default value for the field; can be blank but
      the element must exist
```

```
            'READONLY' => true,        // if the field is read-only (not changeable
by visitor) (not used on file upload field)
            );
```

Version 17+ will display a list of the files and the file size that were selected with the Browse button. Any selected image file will also show a thumbnail of that image next to the file name. If you hover over the image, it will be enlarged 200%.

If the files are not correct, the user can use the Browse button again to reselect the desired files.

VALIDATING THE ADDITIONAL FIELDS

Since you have added additional form fields, then there needs to be a way to validate those fields when the form is submitted. So, we have added two function names that will be run at the appropriate time (if they exist). To enable, just put these function names in your form page code (with the needed code).

To validate the additional fields, use the `FST_CUSTOM_CHECK()` function.

```
FST_CUSTOM_CHECK() - check additional fields for proper values. Called during
the field validation processes on a form submit. Add whatever code you need to
validate the additional fields. Remember that your code should check against
the $_GET or $_REQUEST array, not the $_POST array (that one will be empty).

Note that all variables need to be set as GLOBAL, so they can be used in FST.
If they don't seem to work, check to make sure that you have the GLOBAL for
that field.

PARAMETERS: none

RETURNS: an array of individual error messages, which will be displayed with
any other validation error messages. Optionally, your function can display any
messages, in which case you would return a NULL.
```

Your code in the function should place error messages in an array, and return that array. If the array contains values (your error messages), those messages will be displayed on screen and the form will be displayed. If your function returns anything other than an array, the form will be processed.

Processing the additional fields

You may have a need to do additional processing on the form. Perhaps you would like to store the form information in a contact database, for example (we do that on the FST contact form that allows you to request the code). So we provide an additional function that will be run at the end of the submit process: FST_CUSTOM_AFTER_SUBMIT().

Add this function to your form (or site). Inside the function, add any additional processing of the form data. Remember that form data is available in the $_POST or $_REQUEST. All of those variables are automatically sanitized.

```
FST_CUSTOM_AFTER_SUBMIT() - additional processing of submitted values after
submit.

You might use this to process form fields into a database, for example.

PARAMETERS: none; you can use the $_GET / $_REQUEST values from the form for
processing.

RETURNS: NULL
```

An example:

```
// additional processing of the submitted data:
function FST_CUSTOM_AFTER_SUBMIT() {
// do something with the $_POST data
// make sure you properly sanitize and of the $_POST data
return;
}
```

Remember that these additional functions are placed in your custom form page. Don't make any code changes to the FST core files (other than customizing the settings). These additional functions allow the FST 'core' code to be easily upgraded when we release a new version.

For an example on how to use these extended functions, see the `formspammertrap-sample-contact-extended.php` file.

SPECIFYING ADDITIONAL REQUIRED FIELDS

Normally, all fields on the form are required. You can modify this by adding the following code to your contact form page. The `$FST_REQUIRED_FIELDS` array contains a list of required fields – the 'name' attribute of the input tag. These are the default required fields:

```
$FST_REQUIRED_FIELDS = array("yourname", "youremail", "subject",
"yourcomments");
```

Specify your required fields in your contact form's code page; do not change them in the main functions file. Don't forget to include the GLOBAL statement in your contact form. Example:

```
GLOBAL $FST_REQUIRED_FIELDS;    // required to 'share' this variable with FST
functions
$FST_REQUIRED_FIELDS[] = "address"; // adds the address field to the required
list
$FST_REQUIRED_FIELDS = array("yourname", "youremail"); // sets only these
fields as required
```

Warning! You need to have at least one required field on your form, or the form will not process correctly. We encourage you to have at least the YOURNAME and YOUREMAIL fields required so that the anti-spambot methods will work properly.

REQUIRED FIELD NAMES AND IDS

The FST processing code **requires** that there be fields with these 'name' and 'id' parameters:

```
yourname – the person's name
youremail – the person's email
yoursubject – the subject line
```

`message – the message area`

If these names are not used, form processing may cause an error, and the email you get may not contain the form's information. You can make those fields hidden (via `$FST_XCUSTOM_FIELDS` array), but their name and id values must be as detailed above.

Note that there should not be duplicate ID values – the ID must be unique for each field.

ADDING FUNCTIONS FOR AFTER THE FORM IS PROCESSED

The `FST_CUSTOM_AFTER_SUBMIT()` function can be placed in your custom contact form for additional processing of the form after it is submitted.

For an example on how to use these extended functions, see the `FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP` file. It contains a code sample for adding an additional 'browse' button to select files to send with the mesge.

GETTING YOUR GOOGLE RECAPTCHA API KEYS

FST has the option of adding reCAPTCHA to your form. We don't think it is needed (in fact, they can often be irritating), but the option is there if you wish. We think that the anti-spambot protections built into FST are enough to block spammers.

You must get your own API keys for the reCAPTCHA to work if you want to use this additional protection against bots. Note that keys are assigned to specific domains, so your key will need to be created for your domain. We don't think you'll need reCAPTCHA with all of the protections of FST, but you can add it if you want.

Start here https://developers.google.com/reCAPTCHA/intro . Sign up for a key pair here: https://www.google.com/reCAPTCHA/admin . Make sure you select the Version 2, and the 'Invisible reCAPTCHA' when you create your key pair. It's free, and will help make your contact form even more effective against spam bots.

Place your key pair in the `FORMSPAMMERTRAPCONTACT-FUNCTIONS.PHP` file as noted there, or as overrides as previously discussed.

If you don't have valid reCAPTCHA keys, or don't want to use reCAPTCHA, then leave those two variables blank (empty strings).

If you enable reCAPTCHA, you will also need to ensure CURL is loaded on your server. Most do have CURL enabled, but that is something to check if you don't think reCAPTCHA is loaded or working.

If CURL is not running on your server, you will see an error via the 'Sanity Check' process on your contact page as reCAPTCHA can't work without it. Check with your hosting support to get CURL enabled on your server.

Note that if you don’t use a reCAPTCHA key that matches your domain, the form will appear not to ‘submit’ - nothing will happen when you click the ‘submit’ button.

STORING CONTACT NAME AND EMAIL INTO A DATABASE (version 9+)

Version 9 introduced the ability to optionally store the name/email into a database on your site. This requires several settings, all of which can be placed in your contact form as overrides. We use this on the FST contact form to optionally store names/emails into our FST contact database. (That's how we notify FST users about new FST stuff. And we don't share that information with anyone else.)

There are two parts to enabling this feature.

Note that the database should only contain the one table (in the example "CONTACT"), although you could have separate tables for other contact forms; just define them correctly in each contact form.

You should ensure that the database credentials are strong (including the password). You should not have any other tables in that database. And the user/pass should be unique to all of your site databases.

To use a contact database:

- Set the `FST_SAVE_CONTACT_INFO` value true, and enter the parameters of the database. An example is shown below; you need to
- Make sure the database and table exist, and the structure is correct (see the example in the `FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP .`
- Assign a user/pass to the database
- Assign the proper privileges to the user's access to the table. User credentials should have Create, Delete, Index, Insert, Select, and Update privileges. Without those credentials, the Sanity Check for the database will fail.

Here's an example; adjust for your database environment, credentials, and table information. As before, you would place this in the `FST_MORE_FIELDS()` function in your form.

```
global $FST_SAVE_CONTACT_INFO;    // remember that all overrides must be
GLOBAL'd

$FST_SAVE_CONTACT_INFO = true; // true if name/email should be saved; will also
Sanity Check database credentials (next item)



$FST_CONTACT_DATABASE = array(

       "DATABASE_LOC" => "", // database location; normally localhost

       "DATABASE_NAME" => "", // the database name on your site. Required, must
exist

       "DATABASE_USER" => "", // the user name for that database. Required. Must
be valid.

       "DATABASE_PASS" => "", // the secure/strong password for the database.
Required. Password strength not checked.

       "DATABASE_TABLE" => "contacts", // the table in the database to store
contact information
```

```
        "FIELD_EMAIL" => "email", // the field name for the email address.
    Required. Nust be C/50 (no other type/value).

        "FIELD_FULLNAME" => "name", // the field name to store the name entered
    into the form. Required. Nust be C/50 (no other type/value).

        "FIELD_DATESTAMP" => "last_update",     // datestamp field for last
    record update datestamp

        );
```

When the FST_SAVE_CONTACT_INFO flag is true, the "Sanity Check" will check the database for proper access and structure. If the Sanity Check fails, an error message will be shown on screen or emailed.

The form will display a checkbox (default unchecked) with the FST_SAVE_CONTACT_MESSAGE displayed after the checkbox. You can see an example of this on the FST contact page. The message should include any privacy information required by your site. You are responsible for maintaining data privacy of the contact database, and proper use of the data.

If the checkbox is selected, the contact message sent will include a note that the name/email info was saved into the database.

USING PHPMAILER OPTIONS

FST Version 17 uses PHPMailer for all mailing functions (except for the "Sanity Check"), as it is just easier to program sending complex form data (especially attaching files).

PHPMailer has the option of using SMTP mail. FST version 17 supports those options by including these configuration settings that you can place in the FST_MORE_FUNCTIONS() function of your form.

See the PHPMailer docs for complete information on using these options. The "Sanity Check" does some basic checking of these settings. Note that using PHPMailer's SMTP settings are not required.

```
        $FST_USE_PHPMAILER = true; // enabled by default; set to false to use your own
mail process (FST_MAIL_ALT() function).
        $FST_SMTP_ENABLE   = false; // enable if using SMTP capabilities in PHPMailer
        $FST_SMTP_HOST     = ""; // SMTP host name
        $FST_SMTP_AUTH     = ""; // SMTP authorization
        $FST_SMTP_USER     = ""; // for SMTP user name; not validated (filtered with
htmlspecialchars, so watch for possible filtered characters)
        $FST_SMTP_PASS     = ""; // for SMTP password; not validated (filtered with
htmlspecialchars, so watch for possible filtered characters)
        $FST_SMTP_PORT     = ""; // for SMTP port number; not validated
        $FST_SMTP_SECURE   = ""; // for SMPT Secure setting; not validated
```

You are responsible to ensure that these settings are proper for your site.

USING ALTERNATIVE MAIL PROCESSES (version 10+)

The FST form by default uses PHPMailer (since version 17) to send out mail. Version 10 adds support for using mail processes other than the PHPMailer functions used as of version 17. (Also, see above for using SMTP with PHPMailer.

This support is done via a FST_MAIL_ALT() function that you define in your contact page code. There is one required array called $message_elements that contains the following message elements in the

array. This array has been expanded in version 13, adding the 'raw' elements that contain the unmodified contents of the form fields.:

```
array(
        'subject' => $xsubject,
        'message' => $mail_message,
        'headers' => $mailheader,
        'types' => $filetypes,
        'recipient' => $recipient,
        'raw_from_name' => $xyourname,
        'raw_from_email' => $xyouremail,
        'raw_subject' => $xsubject,
        'raw_message' => $xyourcomment,
        'raw_more_fields' => $morefields, // will contain an array of your extra
        fields
              );
```

Your `FST_MAIL_ALT()` function should return a success/fail flag for the mail sending process. Your function is responsible for parsing the `$message_elements` array into the needed parameters, and adding additional parameters that may be needed for your alternative mail process.

Your `FST_MAIL_ALT()` function might look like this, with code added as needed for your mail process

```
function FST_MAIL_ALT($message_elements = array()) {
        // parse the $message_elements array according to your mail function's
requirements
        // add additional parameters if any are required
        // send the mail
        // set the $mail_status success/fail flag
return $mail_status;
}
```

The "Sanity Check" process (see below) will check for existence of the PHP `mail()` and PHPMailer functions, or the alternative `FST_MAIL_ALT()` function .

Note that you could use PHPMailer functions in your `FST_MAIL_ALT()` function by enabling PHPMailer and adding needed code.

THE SHORT SIGNUP FORM (Version 10+)

Version 10+ has a new option to just show a name/email signup box. You'd use this for a newsletter signup, for instance, where you don't need a message. This option also allows for double-opt-in, where the person has to verify their intention to sign up. That prevents someone from spoofing a signup with someone else's email.

There are several steps to enabling this option.

1. Make sure that the database you use has the proper structure. The default is the structure shown in the array:

```
GLOBAL $FST_CONTACT_DATABASE;
```

```
            $FST_CONTACT_DATABASE = array(
                    "DATABASE_LOC" => "localhost", // database location; normally
localhost
                    "DATABASE_NAME" => "", // the database name on your site.
Required, must exist
                    "DATABASE_USER" => "", // the user name for that database.
Required. Must be valid.
                    "DATABASE_PASS" => "", // the secure/strong password for the
database. Required. Password strength not checked.
                    "DATABASE_TABLE" => "", // the table in the database to store
contact information
                    "FIELD_EMAIL" => "email", // the field name for the email address.
Required. Must be at least 50 length.
                    "FIELD_FULLNAME" => "name", // the field name to store the name
entered into the form. Required. Must be at least 50 length
                    "FIELD_DATESTAMP" => "last_update", // datestamp field for last
record update datestamp
                    "FIELD_GUID" => "guid", // field name for the GUID field used for
verification emails (if inabled) (v10+)
                    "FIELD_STATUS" => 'status' // field name for status of the record:
10=unverified, 20=verified     (v10+)
            );
```

You can specify the database structure by putting these settings in the FST_MORE_FIELDS function of your form

2. Enable the something option. Again, the best spot for this is in the something function of your form
3. The form page should be built similar to other contact pages, inserting the proper commands as specified above.
4. If you want to enable double-opt-in (recommended), set the something option. You can change the message sent in the double-opt-in message with the something option.
5. If you want additional processing of the submitted data, use the something function in your form.

The process will:

1. Display the name/email form.
2. The submit button won't appear until after the delay after the focus/click on one of the two fields.
3. Submitting the form will save the info in the database you specify.
4. If you have enabled double-opt-in, that message will be sent to the email on the form.
5. A special unique URL will be in that email; the person would click that email to verify.
6. When verified, the status of the entry will be changed to '20', indicating verification completed.

You get to decide what to do with the saved data, other than it being stored in the database you specified.

THE SANITY CHECK

The Sanity Check does various checking to ensure that required parameters are set properly. The following checks are done:

1) The FST_XEMAIL_ON_DOMAIN email address must belong to the current domain.
2) The FST_UPLOAD_EXTENSIONS must be an array.
3) The Upload folder must exist.
4) CURL is available on the server (if reCaptcha enabled)

5) If `FST_SAVE_CONTACT_INFO` enabled, the database is checked
    i) For access
    ii) For proper structure
6) Existence of the PHP `MAIL()` function or the use of the `FST_MAIL_ALT()` function in your form.
7) Some other checking, like PHP version (FST requires at least PHP version 7.2; improved in version 13).
8) Checking for proper installation of the PHPMailer files (since version 17.00).

If any part of the Sanity Check fails, error messages will be displayed on the screen and the form will not be available. If you have set the `FST_SANITY_CHECK_EMAIL` true, there will be a 'polite' message displayed on the screen for the visitor, and the detailed error message will be sent to the `FST_XEMAIL_ON_DOMAIN` email address. We suggest that you set `FST_SANITY_CHECK_EMAIL` false during development, and true when the form is live.

USING FST ON WORDPRESS CONTACT FORMS

FST Versions 6+ now supports the use of a shortcode that can be used with a custom page template on a WordPress site. Version 16 added optional tags to specify the email addresses. The process is detailed in the separate documentation file. Since every WP theme is different, it requires a bit of work on your end to do this. (We can help with the work if needed, or answer questions - just use the contact page on the FST site.)

Just take a look at the `Using-FormSpammerTrap-in-WordPress.pdf` file for complete instructions.

The basics are to:

- Copy a page template from your theme.
- Give it a new name (file name and template name).
- Change the template header as needed.
- Remove any code belonging to 'the Loop' as needed
- Add three function calls, and upload it to your site.
- Then create a new page, use the new template, and insert the `[formspammertrap]` shortcode in the content of the page.

It's quite easy (it might take you just a few minutes), and the result is a WordPress contact page that is a spambot killer.

We may update our WordPress FST Contact Page plugin (it doesn't have all of the advanced features we use in FST here), but the above method is best to use.

MODIFYING A COMPLEX CONTACT FORM

If you have a complex contact form, with lots of fields, you might find it easier to adjust that form rather than creating a new form.

Version 14 introduced that feature. See the `formspammertrap-roll-your-own.txt` file included in the distribution ZIP file. It's a much easier and faster process than building your own form. Of course, all the FST feature options can be added to your form.

Version 17 added better support and reliability to adding a file upload button to your form. This version also supports multiple file uploads. See the File Upload section above.

SUPPORTING NON-ENGLISH LANGUAGES

Version 14 introduced the feature that allows you to have non-English text on your form. This is done with the `FST_LANG_TEXT` array. This only applies to user-facing text displayed on the form.

Full details are in the https://www.formspammertrap/formspammertrap-language-english.php (also a separate file in the distribution ZIP file). Copy this file to your site (in the same folder as the FST functions file) to see the current `FST_LANG_TEXT` array settings.

MORE INFORMATION AND HELP

If you are customizing your form, read all of the documentation (PDFs). Look at the 'all-settings' PDF file to see every customized value you can use. That file contains the latest information – it's taken directly from the functions file. There's a ton of available settings that you can use to customize your form.

Remember that we can implement FST on your site. A basic contact form can be done quickly; a more complex form takes a bit longer (although we might use the 'roll your own' procedure to finish your custom field faster). But the cost is reasonable, especially compared to your time. See the FST site for details. Send a request via the contact form – include the link to your current contact form – and we'll provide an estimate for implementation.

If you have any questions about FST – implementation, suggestions, whatever – just use the contact form on the FST site. We usually respond within 24 hours – sometimes faster.

Thanks for using FST – we think it will severely reduce – if not eliminate – the contact form spam you have been getting.

If you would like to make a donation to help out efforts, including offsetting hosting costs, please see the Donation page on the FST site. Your donation is really appreciated.

WE DO OTHER THINGS

We have other sites that might be of interest. If you are an independent book author, check out or **BKLNK** site for some author tools. Plus, our new **BkSubscribe** site – a mailing list system for book authors. And the **BkLanding** site – easy landing pages for authors. And more!

We also have several WordPress plugins that help secure and enhance your WordPress site. More information here: https://cellarweb.com/wordpress-plugins/ .

And we've written a couple of fiction books ourselves. Check out the **RichardHellewell** site for details.

We're available for all sorts of web geekery. Use the contact pages on any of those sites to contact us.

Some of Our Web Sites

Richard Hellewell author site – Hey – I wrote some books! – https://www.RichardHellewell.com

BKLNK – category research for book authors on Amazon – https://www.bklnk.com

BkSubscribe – Mailing list system for authors, including newsletter signups – https://www.bksubscribe.com

BkLanding – Landing pages for authors, including newsletter signups – https://www.bklanding.com

BkTry – an easy way for authors to securely share book excerpts – without worrying about copying or download of their file. See https://www.bktry.com .

CellarWeb.com (our main business site) – https://www.cellarweb.com We manage WordPress sites and do custom website programming.

We can build your site. Or update it. Or make it more secure. Contact us via the FST site contact form.

Thanks!

Thanks again for using FormSpammerTrap – it will block spam bots from abusing your contact pages. It's been working since 2011 ! Use our contact form for any questions, comments, or just a 'Huzzah!' when it works on your site.

Rick Hellewell – Somewhere opposite Mutiny Bay WA – 3 Feb 2024

CHANGE LOG

We've moved the change log to a separate txt file - it was getting a bit long because of all of the new and improved features we keep adding to FST. Check out all of the new features we added to this version.

ABOUT UPDATED VERSIONS OF FST

Starting with Version 7, FST code allows easy future upgrading to the FST 'core' code on your site. All that is required is to copy the new files (backups are always good), set the customization settings (referencing the settings in the prior version of your files, which is why backups are good), and then add any additional processing functions using the about instructions by adding those functions to your contact page's code.

Version 8+ allows you to have multiple contact forms that use the main customization file, and then override settings as needed in each contact page. An example of this is the FORMSPAMMERTRAP-SAMPLE-CONTACT-EXTENDED.PHP contact form, which adds additional fields to the contact form, plus overriding some customization values. That was described above.

Upgrading from prior versions is best done with these steps:

- Make a backup copy of your existing files
- Make a note of your customization settings.
- Install the new files in your development system.
- Change all required customization values as needed, copying values (not code) from the prior version.
- If desired, add additional customization into your contact form using the above instructions.
- Test the new version on your development system.
- Publish the new code files to your site and test (backups are always good, just in case).

We suggest that you use the `FST_MORE_FIELDS()` process for the customization settings. That will make upgrading to a new FST version easier.

BASIC 'HOW IT WORKS' SPAMBOT BLOCKING INFORMATION (geeky stuff here)

The FormSpammerTrap code has several ways to prevent bots from abusing your contact page. They are described in general on this page https://www.formspammertrap.com/v10/howitworks.php .

Technically:

The FORM tag has an initial ACTION value of the FormSpammerTrap web site. You can change this if you wish in the Customization process.

That ACTION value is used to trap bots that try to 'scrape' your form, using your ACTION page against you. A sample response page (FORMSPAMMERTRAP-CONTACT-RESPONSE.PHP) is available as the 'honeytrap' response page. If you put that page as your FORM ACTION value, the bot that tries to use that FORM ACTION page will get caught in your trap. We don't care if you use our FormSpammerTrap site URL as the 'trap'.

When a person clicks/focuses on the required field to input data, the form's action will be changed to the actual form processing page, allowing the form to be processed normally. That change is also delayed as another protection. The delay in both cases is specified in the Customization section, and defaults to 5 seconds.

Each required input field on the form (there must be at least one) has this code added to the INPUT tag:

```
class = 'fst_required'
```

This class should be added to all required fields. (It's done automatically if you add fields via the FST_MORE_FIELDS() process.) The onclick/onfocus events will be automatically added to those fields when the form is generated. There is a delay in adding those events to required fields; another protection against form spam bots. Those functions are JS code, which is why the spam bot can't normally insert those functions.

In addition, the YOURNAME and YOUREMAIL field names are changed to what they are supposed to be – initially they are a random-type name.

Because of the delays in adding the onfocus/onclick events to required fields, an automated (bot) process will not see the events, or the improperly named fields, and will probably just try to fill out the form with their bot process. Because the bot process can't emulate an onclick/onfocus (normally), the form will be submitted to the bogus site. And the random field names provide another layer of protection against spam bots.

Even if the bot author tries to customize their bot process, the delay in adding the onclick/onfocus will probably not be seen, thereby adding another protection against bots.

Bots can't easily do JavaScript functions (without some highly customized code that has to be unique to your comment form), so it is unlikely that a bot will be able to change your FORM ACTION location or make the other required changes to the form and form processing. The new delays make that even harder to detect the need for customizing the bot code for our JavaScript functions. (I suppose that it could be done, but it would take a lot of customizing effort, and there is lots of 'low-hanging fruit' out there for the spam bot actors to attack instead.)

The actual FORM ACTION location is split in two variables. Each of those variable names has a unique name when the page is loaded. The variable values, when assembled, will be the actual form processing page. The sample Contact page uses itself as the processing page, so those variables are empty. In the customization area, enter the full file name of your form processing page. Note that by default, we use

the form's page as the submit processing page. If you use your own processing page, you will need to properly process the form values.

If you enable reCAPTCHA: The SUBMIT button is actually code used to call the Google Invisible reCAPTCHA. In most cases, that 'are you human' check will not be visible to the visitor. In some cases, based on the mysterious Google reCAPTCHA analysis, the visitor might see a visible reCAPTCHA consisting of images to analyze. That can't be helped; it's all a function of the Google reCAPTCHA API. In our testing, we would see the visible reCAPTCHA for a while, and then we wouldn't. The googles are very mysterious in how they determine whether you see the visible reCAPTCHA.

If you don't enable reCAPTCHA: There is a hidden 'normal' submit button that is used to process the form. It's used if you don't enable reCAPTCHA. In any event, the submit button doesn't appear on the page until after the timeout (which is enabled when you click on a required field).

The bot cannot use a 'resubmit' (F5) of the entered contact form data. If they do, they get sent to the bot-catching action page (by default, the https://www.FormSpammerTrap.com site), or they will just 'see' the form again. In some cases, they might see a 'Go away spammer' message. In any case, the form will not be processed.

The sample contact page has the basic input fields: name, email, subject, and message. If you want to add additional fields, you can add them as described above (and add the form processing code if needed).

The email message sent from the form will include the sender's name and email, plus the subject and the message.

All functions and CSS names start with 'FORMSPAMMERTRAP' or 'FST' to reduce chances of conflict with other code on your site.

The sample Contact page sends HTML mail, with the REPLY-TO value set to the visitor's email address that they entered. That allows you to easily reply to the message. You can set a flag to send a text-format message.

If you use our `FORMSPAMMERTRAP-SAMPLE-CONTACT-PAGE.PHP` file, you can rename it to something simpler. Just make sure that all the other files are in the proper place, and you adjust the INCLUDE statements.

HOW BOTS SPAM YOUR CONTACT FORMS

Bots have special processes that 'scrape' your contact form and then submit data via an external, 'non-human' process. The bots are not 'human', so hiding things visually won't affect them (as well as most other commonly-recommended techniques).

So our protection processes don't let those automated processes work very well, without some detailed customization of their process. And the customization is so site-specific that the spammer will probably just skip your site and go on to a site that doesn't have our protections.

If you want more details on how our process works, check out the information on the FormSpammerTrap site at https://FormSpammerTrap.com .

We also wrote a detailed analysis of how bot spammers work, and how we defeat them, on our "Security Dawg" site at https://securitydawg.com/how-bots-spam-contact-forms/ .

QUESTIONS - BUG REPORTS - FEATURE REQUESTS – IMPLEMENTATION HELP

We've tested this form, and use it on our https://www.FormSpammerTrap.com web site (and other sites). It should work 'out of the box', as long as you have made the required basic customization changes.

We've also tested the effectiveness against CURL-based attacks…they don't work!

We can provide you with a complete solution for your site for a modest implementation fee. Just contact us for details.

All of the features we've added in the various versions are built to

- Block spam bots.
- Allow for easy implementation.
- Allow developers to modify the form and functions for their needs.

If you created your own contact form, make sure that your contact page is similar to FORMSPAMMERTRAP-SAMPLE-CONTACT_PAGE.PHP, including the INCLUDES.

Make sure that you have changed the customized variables in the functions file – only one customization value is required for basic use. This includes your own reCAPTCHA key, which is tied to the domain name of your site (if you enable reCAPTCHA). If you use an incorrect key/secret pair, the submit button won't work.

And make sure that you uploaded all files, including the supporting INCLUDES to the proper location.

NEED MORE HELP OR INFORMATION?

If you need help integrating this on your site, let us know. **If you would like us to do all the work for a modest fee, please contact us.**

If you have any issues with the code, or improvement suggestions, or even a 'Thank you - it works!' comment, use the Contact page on our site: https://formspammertrap.com/contact.php . We usually answer within 24 hours…and we won't use your email address for any other purpose.(We delete your email after we respond.)

And remember that all of this is free. No charge. Nada. But we do take donations if you wish to support our efforts. All the details are on the FST site. Even if you don't donate, we'd be pleased to hear from you via the contact form on the site.

Check back on our site for any updates to the files. We don't maintain a mailing list of FST requestors – unless you check the box allowing us to save your name and email, so checking the site is the way to find out about new versions.

Use the FormSpammerTrap site's Contact form to contact us with questions, comments, or enhancement requests. We'll usually respond within 24 hours.

THANKS!!

Rick Hellewell, https://www.FormSpammerTrap.com

December 2020 - somewhere across from Mutiny Bay, WA.

-30-