Form Processor Pro v5 User Manual

Table of Contents

Part I

1 Introduction

1.1 System Requirements

Form Processor Pro 5.x is being offered in PHP version only, PHP available on most Windows- and Unix- compatible platforms (web-servers). Form Processor Pro has standard requirements for PHP scripts. In other words, normally, you do not have to install any additional software to get Form Processor Pro working.

To run Form Processor Pro v5 you must meet the following requirements:

- PHP 4.3.0 or higher
- Standard mail function (SMTP, Secured SMTP or Sendmail)

In addition, we have created special Form Processor Pro test script that you can upload to your server in order to check general Form Processor Pro compatibility and availability of special features that depend on PHP configuration on your host.

In order to check your host with Form Processor Pro checkup script:

1. Download Form Processor Pro test script here.
2. Unpack downloaded zip file. There is only one file in the package: *fpp-test.php*
3. Upload or copy *fpp-test.php* to your public directory (www).
4. Open this script in your favorite web browser directly from the host by correspondent URL.

For Example:
You have a domain: www.mydomain.com and you uploaded *fpp-test.php* script to your host by FTP so that it can be viewed through http://www.mydomain.com/fpp-test.php. By opening this link in your browser, you'll see results of the checkup script.

Note:
If you are not aware of the details above, contact directly your web hosting Provider Company. You may also find this information in your hosting package details.

1.2 Package Contents

Form Processor Pro Download Package has the following structure:

			File(folder name)	Description
📂			**fpp-v5.zip**	Archive of our Form Processor Pro Download Package
↪	📂		**fpp**	Main Form Processor Pro folder with all required files for proper work. You must upload this folder to your public folder (www) on your web server.
↪	📂		**sample-forms**	Folder with examples of typical forms. Feel free to use and change them for your needs.
	↪	📂	**fpp**	Form Processor Pro configured to serve sample forms. You should upload this folder to your web server with other sample-forms folder content to see how sample forms work.
		↪	**attachments**	It is temporary folder for user uploads and files that you would like to attach in emails.
		↪	**classes**	Classes are core files of Form Processor Pro.

		➯	**Install**	Initial configuration script.
		➯	**lang**	Contains language files (translations) of all available languages for error messages. You can change language in a moment by changing only 1 string in configuration file. Please refer to “Language Settings”.
		➯	**plugins**	Plug-ins directory. Each plug-in refers to certain functionality of Form Processor Pro.
		➯	**tmp**	Form Processor Pro use this folder to store temporary files that may be needed during form processing, like PDF, XLS generated files, for counting unique submissions etc.
		➯	**.htaccess**	.htaccess file prevent unauthorized access to the folder contents from the web.
		➯	**captcha.img.php**	Provides CAPTCHA anti-spam protection.
		➯	**config.php**	One and only configuration file of Form Processor Pro and forms. It has simple structure where you should specify all your forms and their settings, as well as global settings.
		➯	**fpp-test.php**	By opening this file in your browser you will check your system requirements for Form Processor Pro. It shows detailed information about availability of certain functionality and general possibility of usage of Form Processor Pro on your web server's current configuration.
		➯	**index.php**	Main Form Processor Pro executable file. All forms' actions should refer to this file. Like: <form name="my_form"

				action="URL_To_This_File">
↪	🗁		**manual**	This folder holds the HTML manual. In order to read it you should open index.html file from this folder.

In order to use Form Processor Pro you only need to know the location of the config. php, URL to index.php and may be attachments directory location in case you will use attachments functionality in your forms.

Permissions:
Set writeable permissions (0777) to the folders “tmp” and “attachments” inside Form Processor Pro after uploading Form Processor Pro directory to your web server.

1.3 How it works?

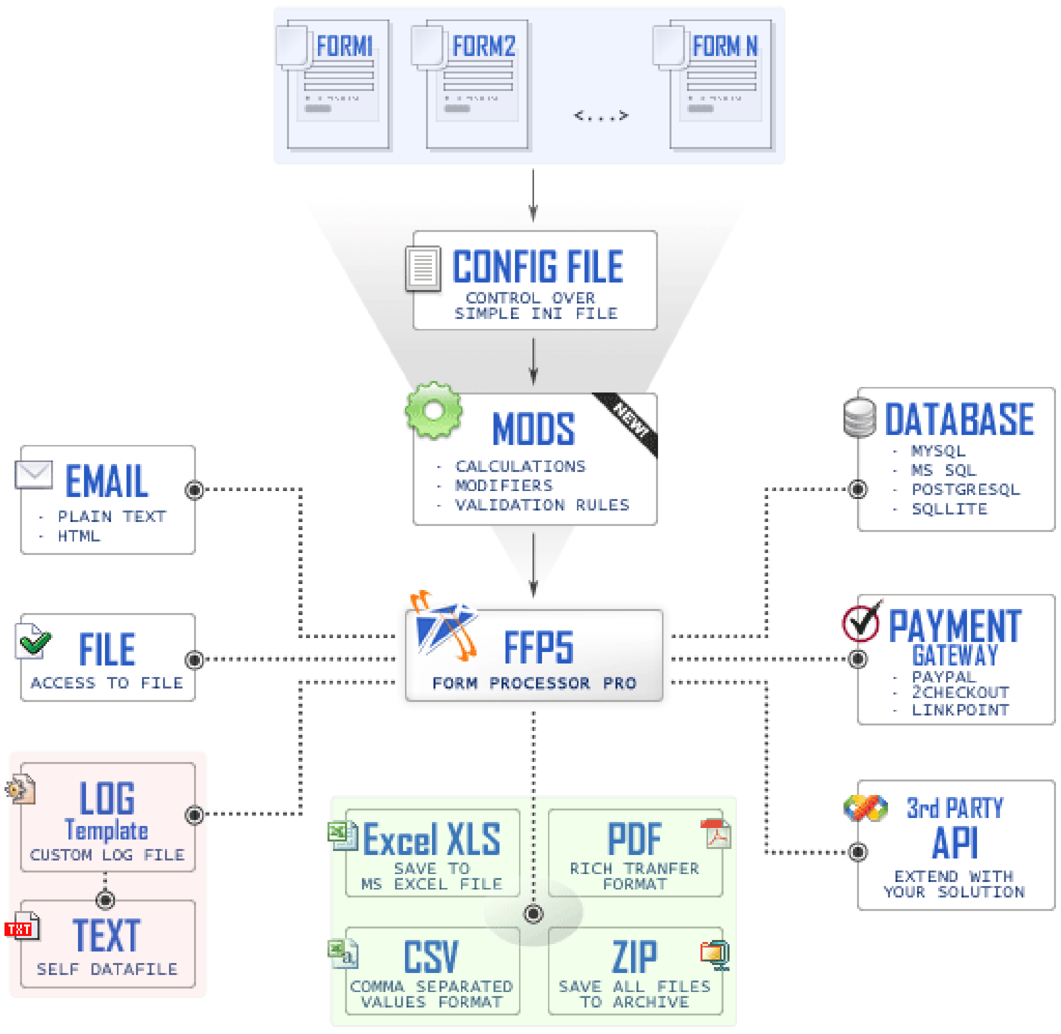
Form Data Input

- When a web site visitor clicks on “Submit” button in your web form, Form Processor Pro starts loading Configuration File. It is a simple INI-file where you can specify all your forms and their settings. This file also sets basic variables to make Form Processor Pro work correctly.
- Submitted data is being checked for compliance with form field validations and modifiers and generates detailed highlighted error messages if check fails or incompliant data is submitted. Modifiers used in order to perform certain transformations with user input prior inserting result into template or generating output.
- Afterward, submitted data is being parsed according to field validations and rules (required fields, email field etc.), processed calculations and function calls. All data gets parsed and processed by Form Processor Pro and now ready

for output.

Form Data Output

- According to variables and their values set in Configuration File, Form Processor Pro sends an output to a web browser, sends data by email and saves data to log files, different types of databases, CSV, XLS, PDF file formats or processes data to the payment gateways. Also Form Processor Pro gives you an opportunity to archive files before sending or redirect all form data to another third party software page via GET request. Combinations of those are possible. You can define the output root in the Configuration file.



1.4 What's new in v5?

Form Processor Pro v5 has completely redesigned core and workflow scheme. Now, there are no separate configuration files for every form created/added. There is only one configuration file responsible for global and local settings.

FPP v5 has several groups of features that help to process any form with very sophisticated requests. These groups are:

- Validators - validation rules, responsible for validating user input
- Actions - actions that have to be done after form submission
- Modifiers - modifiers help to pre-format user inputted data by certain modification rules

Features list

- Multilingual support
- 40+ field validations
- 15+ user input modifiers
- Secured SMTP support
- Security and spam protection
  - Spam Protection with CAPTCHA
  - Spam Injection Prevention
  - Email harvesting protection

  - SSL Support
  - Unique Submissions
  - Block IPs or Hosts
  - Allow only certain IPs and Subnets
- Simplified configuration
  - One configuration file for FPP and forms
  - Minimal configuration to existent form HTML
    Just point it to FPP and give it a name in form's HTML
- User friendly error messages
  Error messages now can be displayed on the same form page with fields highlighting. You have full control over style.
- Back & Edit feature
- Multiple data output
  Processor can perform unlimited combination of tasks per form. Send different emails with PDF, XLS and even ZIP them as well as put data into database.
- Advanced output formats:
  - PDF
  - XLS
  - CSV
  - ZIP
  - TXT and LOG - template based storing
- Storing to databases
  - MySQL

- MS SQL
- SQLite
- PostgreSQL

- Payment gateways
  - PayPal
  - 2CheckOut
  - LinkPoint
- File output
  This feature allows you to output file for download just after form submission
- File uploading validation
- Expire date for the form
- Plug-in architecture
- API for third-party integration
- And much more!

Part II

2 FPP Basics

2.1 How to get started?

In order to start using Form Processor Pro you should follow the following steps in this manual:

1. Read Package Contents for better understanding further chapters of this manual.
2. Perform installation of the package according to Installation instructions.
3. Prepare your installation using configuration script and Initial configuration instructions.
4. Follow the steps in "Setting-up My First Form" chapter and configure your first form in order to understand the process of form configuration.
5. After succeeding these steps you'll be able to configure any form with FPP v5 just in 5 minutes or even less!

2.2 Installation

Form Processor Pro 5 is very easy to install, just follow the following steps:

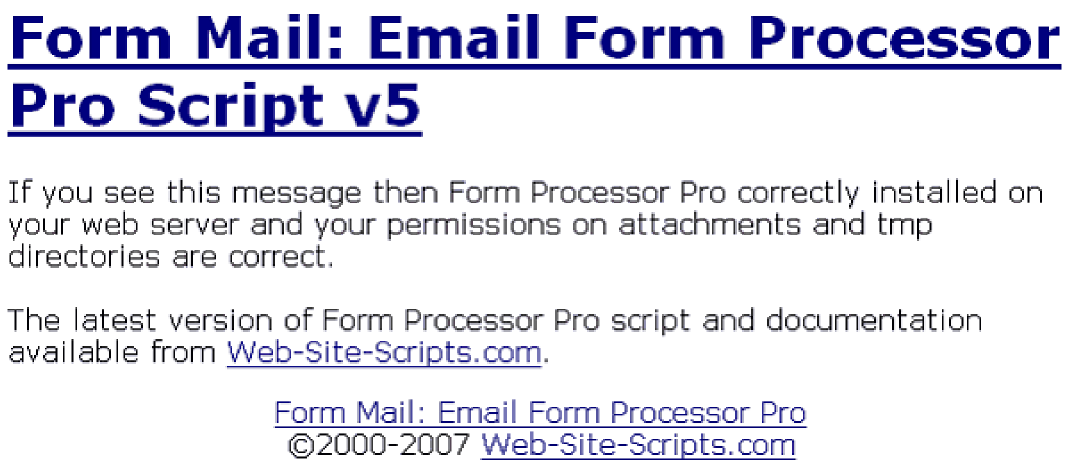
1. Download latest package from the member area.
2. Unpack downloaded package using any ZIP archive extraction program (e. g. WinZIP).
3. Upload complete "fpp" folder from the package to your web server. Main www folder is preferred.
4. If you would like to use or test demo forms you can upload complete "demo-forms" folder as well. Main www folder is preferred.

5. Set writable permissions (0777), using UNIX shell command or any FTP client (e.g. CuteFTP) to the folders “tmp”, “attachments” and config.php file inside Form Processor Pro (fpp) directory.

6. Run Initial Configuration Script and follow the instructions in order to perform initial Form Processor Pro configuration. Initial Configuration Script is located in the fpp/install folder.
   E.g.: If you've uploaded the fpp folder to wwwroot folder of your www.yourdomain.com server, you can access Initial configuration Script via www.yourdomain.com/fpp/install/index.php URL. Default password to Initial Configuration Script is "admin". You will change it during the installation.

7. That's all! Now you can create and configure your existing forms for processing with Form Processor Pro. If you are newbie to Form Processor Pro please read “Setting-up My First Form”.

To test your installation completeness simply open main Form Processor Pro file (index.php) in your browser, like this: http://www.yourdomain.com/fpp/index.php. If you see the message as on the picture below, you've successfully completed installation process and you can move to form configuration. Otherwise you'll see error messages. Thus further actions should be done according to these messages. Most common and known problem is improper permission settings for “attachments”, “tmp” directories and *config.php* file. These have to be writable.


Form Mail: Email Form Processor Pro Script v5

If you see this message then Form Processor Pro correctly installed on your web server and your permissions on attachments and tmp directories are correct.

The latest version of Form Processor Pro script and documentation available from Web-Site-Scripts.com.

Form Mail: Email Form Processor Pro
©2000-2007 Web-Site-Scripts.com

Note:
In order to set writable permissions in UNIX shell command you should run the following command: “*chmod 777 <filedir_name>*”. <filedir_name> is file or folder in question.

2.3 Initial Configuration Script

We have included Initial Configuration Script in the package in order to simplify FPP configuration for further form processing. Using this script you will easily do the following:

1. Check permissions for required files and folders
2. Configure Email Sending feature (set correct SMTP server)
3. Configure Database settings (if you intend to save form submissions to database)
4. Change password for this configuration script

You should know that script stores all parameters in *config.php* file and, actually you can simply edit this file by yourself if you understand what are you doing and know SMTP and DB parameters.

Run Initial Configuration Script and follow the instructions in order to perform initial Form Processor Pro configuration. Initial Configuration Script is located in the fpp/

install folder.
E.g.: If you've uploaded the fpp folder to wwwroot folder of your www.yourdomain.com server, you can access Initial configuration Script via www.yourdomain.com/fpp/install/index.php URL.

When you've completed Initial Configuration Script wizard, you are ready to setup your first form.

2.4 What is actions, validations, etc.?

FPP v5 has several groups of features that help to process any form with very sophisticated requests. These groups are:

- Field Validations - validation rules, responsible for validating user input (e.g.: email validation, ZIP code validation)
- Actions - actions that have to be done after form submission (e.g.: send an email, put data to database, open a web page)
- Modifiers - modifiers help to pre-format user inputted data by certain modification rules (e.g.: capitalize first letters of second name, set all letters of special field to lowercase)

Click on correspondent group for more information and features reference.

2.5 Setting-up My First Form

Here, we'll guide you through the process of creating and configuring your first form with Form Processor Pro. **After learning this sample you'll be able to configure any form in just 5 minutes or less!** For a head start, we'll create a simple contact form with the following in mind:

- We will have such form fields: Name, Phone, Email, Subject and Comment
- We would like to validate only properly formatted email addresses
- Name, Email and Subject fields are required
- We want to receive form results to our email
- We want to send auto-responder confirmation email to form submitter
- We would like to provide user with error messages on the same page and highlight improperly filled fields
- We would like to have custom thank you page with personalized greeting

Hmm, that's pretty much requirements just for Simple form! However, significantly complicated and sophisticated forms are possible with Form Processor Pro too. In this case we just want to have contact form to be processed the right way and accept only correctly formatted submissions, avoiding user typos and mistakes.

To create any form with Form Processor Pro you need to assemble four easy things:

1. **Point your form and give it a name.** You should point ACTION of your form to the Form Processor Pro installation and give it a unique name by inserting HIDDEN field in the form's HTML just after form tag.
2. **Handle with error messages.** Put `<!-- FPP_ERROR -->` tag to your form's page in the where you'd like to display error messages.
3. **Create required Email, HTML or Log templates.** Use {# and #} brackets and hash signs to include any variable (field content) in the template. If you have text field with *name=email* then you can use this variable in any template by putting it in brackets like this: {#email#}. This will be changed with user's entered value after processing.

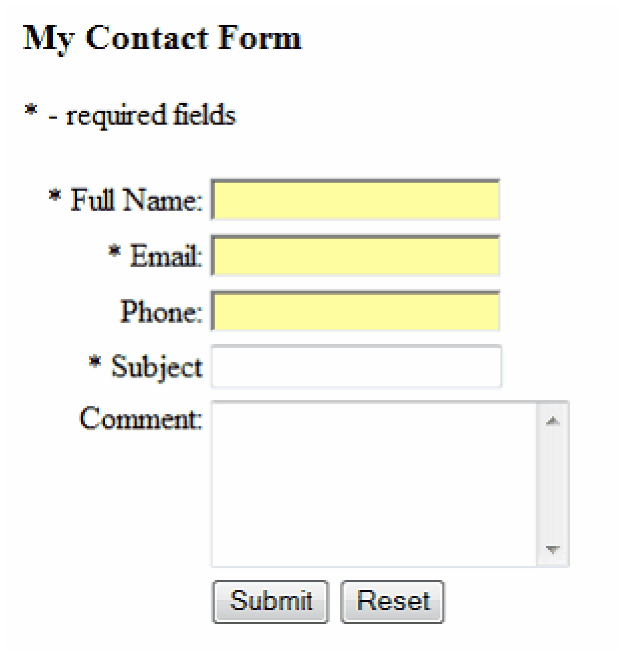
4. **Edit simple config file with required configurations.** Set actions to perform, validation rules etc. All available actions, validations and modifiers can be found in this manual.

In order to follow further steps you need to have Form Processor Pro installed on your web server and initial configuration should be completed. Please refer to Installation section of this manual.

In this example we have a domain: mydomain.com and Form Processor Pro installed in the “fpp” public folder under www root. In other words Form Processor Pro can be accessed through: http://www.mydomain.com/fpp/index.php

2.5.1 1st Step: Creating contact form with HTML editor

We've used Dreamweaver to create the form and we've got the following form and auto-generated HTML code:


My Contact Form

* - required fields

HTML Code of the Form Page:

```
<html>

<head>

<title>My Contact Form</title>

</head>

<body>

<h3>My Contact Form</h3>

<p>* - required fields</p>

<form name="form1" method="post" action="">

  <table width="300" border="0">

    <tr>

      <td><div align="right">* Full Name:</div></td>

      <td><input type="text" name="name"></td>
    </tr>

    <tr>

      <td><div align="right">* Email:</div></td>

      <td><input type="text" name="email"></td>

    </tr>

    <tr>

      <td><div align="right">Phone:</div></td>

      <td><input type="text" name="phone"></td>

    </tr>

    <tr>

      <td><div align="right">* Subject</div></td>

      <td><input type="text" name="subject"></td>

    </tr>

    <tr>

      <td valign="top"><div align="right">Comment:</
div></td>

      <td><textarea name="comment" rows="5"></
textarea></td>

    </tr>

    <tr>

      <td valign="top">&nbsp;</td>

      <td><input type="submit" name="Submit"
value="Submit">

        <input type="reset" name="Reset"
value="Reset"></td>

    </tr>

  </table>
```

```
</form>

</body>

</html>
```

2.5.2 2nd Step: Configuring the form

We put `<!-- FPP_ERROR -->` tag to the form's HTML in a place where we'd like to display error messages.

Now we need to point our form to be processed with Form Processor Pro. We do this by editing FORM tag's ACTION attribute to refer to the URL where Form Processor Pro resides. You can use either relative or absolute path. We need to set METHOD attribute as POST as well. And finally set ENCTYPE as MULTIPART/FORM-DATA.

Form tag before:

```
<form name="form1" method="post" action="">
```

Form tag after required corrections:

```
<form name="form1" method="post" action="http://www.
mydomain.com/fpp/index.php" enctype="multipart/form-data">
```

If you use relative path in the action parameter and do not use "Base href" tag, every page after first one must have action="index.php" parameter:

```
<form name="form1" method="post" action="index.php">
```

That's all for the FORM tag. Now we need to give a unique name to our form, let's name it “contact_form”. We can do this by inserting HIDDEN field just after the FORM tag with the name “fpp_form” and value “contact_form”.

Like this:

```
<input name="fpp_form" type="hidden" value="contact_form">
```

Note:
In form names you can use only letters, numbers, underscore and dash characters.

Corrected HTML Code with highlighted changes:

```
<html>

<head>

<title>My Contact Form</title>

</head>

<body>

<h3>My Contact Form</h3>

<p><!-- FPP_ERROR --></p>

<p>* - required fields</p>

<form name="form1" method="post" action="http://www.
mydomain.com/fpp/index.php" enctype="multipart/form-
data">

<input name="fpp_form" type="hidden"
value="contact_form">

  <table width="300" border="0">
```

```
    <tr>
      <td><div align="right">* Full Name:</div></td>
      <td><input type="text" name="name"></td>
    </tr>
    <tr>
      <td><div align="right">* Email:</div></td>
      <td><input type="text" name="email"></td>
    </tr>
    <tr>
      <td><div align="right">Phone:</div></td>
      <td><input type="text" name="phone"></td>
    </tr>
    <tr>
      <td><div align="right">* Subject</div></td>
      <td><input type="text" name="subject"></td>
    </tr>
    <tr>
      <td valign="top"><div align="right">Comment:</
div></td>
      <td><textarea name="comment" rows="5"></
textarea></td>
    </tr>
    <tr>
      <td valign="top">&nbsp;</td>
      <td><input type="submit" name="Submit"
value="Submit">
        <input type="reset" name="Reset"
```

```
value="Reset"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

Note:
Please, specify absolute paths in all links, images, path of the action parameter, etc. on your form pages. Otherwise, all relative paths would be relative to Form Processor Pro, but NOT to the page of your web-site. Or you may use HTML tag <BASE> to specify base URL:

```
<BASE HREF="URL_TO_PAGE">
```

If you have more than one form page, repeat this step few times. It's not necessary to add hidden field to pages other than first.

Form saved as: contact.html

2.5.3 3rd Step: Configuring - Thank you page

As we would like to have custom thank you page we need to prepare it before dealing with email template and config file.

So, on the thank you page we want to put something like this:

Dear (here we would like to put filler's name),
Thank you for interest in our company. We will reply as soon as possible!
Auto-responder email has been sent to (here we would like to put filler's email).

Initial HTML code for the Preview Page:

```
<html>

<head>

<title>My Contact Form - Thank you!</title>

</head>

<body>

<p>Dear (here we would like to put filler's name),</
p>

<p>Thank you for interest in our company. We will
reply as soon as possible!</p>

<p>Auto-responder email has been sent to (here we
would like to put filler's email).</p>

</body>

</html>
```

In order to insert any variables dynamically we need to use {# and #} brackets. Inside these brackets we need to put appropriate variable name. Variable is the name of the corresponding field in our form. In our case we should use "name" and "email" accordingly. So we will have the following changes in our HTML code for the Thank You Page:

HTML code for the Thank You page with highlighted changes:

```
<html>

<head>

<title>My Contact Form - Thank you!</title>

</head>

<body>

<p>Dear {#name#},</p>

<p>Thank you for interest in our company. We will
reply as soon as possible!</p>

<p>Auto-responder email has been sent to {#email#}.</
p>

</body>

</html>
```

Thank you page saved as: thank-you.html

2.5.4 4th Step: Preparing Email templates

Email templates are just text files with specially formatted content. Generally, email has following required fields (rows in template): To, From, Subject and Message Body. However, other fields like BCC, Attachments etc. are also possible. Please refer to Email Sending Action article for more information. In email template it should look like this:

Email template:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject

Message Body Text in free form (Either plain text or
HTML formatted text)
```

In order to use variables (fields) we need to put them in figure brackets with hash signs (#), just like in the previous step.

We want to have all form fields to be included in our email. So the email template that will be sent to us should look like this:

```
To: info@mydomain.com
From: {#email#}
Subject: Contact Form Filled

Information from the Contact form:
Full Name: {#name#}
Email: {#email#}
Phone: {#phone#}

Subject: {#subject#}

Comments:
{#comment#}
```

Email template "to us" saved as: email.txt

We create email template for the auto generated email to the form's submitter the same way:

```
To: {#email#}
From: info@mydomain.com
Subject: Thank you for contacting Mydomain.com

Dear {#name#},

You have submitted the following information to
mydomain.com
---
Name: {#name#}
Email: {#email#}
Phone: {#phone#}
Subject: {#subject#}

Comments:
{#comment#}
---
```

```
Thank you for contacting us. We will respond as soon
as possible!

Sincerely,
Mydomain.com Team
```

Email template "to client" saved as: autoresponder.txt

2.5.5 5th Step: Configuring config.php - Summarizing configuration

We have configured our form, thank you page and prepared required email templates. Now in order to make final configurations we have to teach FPP how we would like our form to be processed. To do so we have to edit *config.php* located in Form Processor Pro installation.

Files placement on web server

Before that, we should place created files to the web server as we will use paths to these files in our *config.php* file. So, we place contact.html and thank-you.html to the web server's www public folder. Both email template files (email.txt and autoresponder.txt) we place in separate folder under www.

- **www** - public folder of mydomain.com
    - ***fpp*** - folder with installation of Form Processor Pro
    - ***attachments***
    - ***classes***
    - ***install***
    - ***lang***

- ***plugins***
- ***tmp***
- *.htaccess*
- *captcha.img.php*
- *config.php- configuration file*
- *fpp-test.php*
- *index.php*
- *style.css*
- ..
- ***contact*** - folder with email templates
- *.htaccess - protect this folder from outside browsing, see tip-suggestion below*
- *email.txt*
- *autoresponder.txt*
- *contact.html* - our contact form
- *thank-you.html* - our thank you page
- .. - and other web site pages

In our case:
FPP can be accessed through: http://www.mydomain.com/fpp/index.php
Contact form can be accessed through: http://www.mydomain.com/contact.html

Tip:

In order to protect your email template files from undesirable browsing through HTTP we suggest you to place them in separate folder and protect this folder from browsing. You can use .htaccess file for this purpose on any Unix/ Linux compatible host. Just add/create this file(.htaccess) in the folder with email templates with the following content (you can also copy .htaccess file from “tmp” or “attachments” folder of the Form Processor Pro installation):

```
<Files *>
    Order allow, deny
    Deny from all
</Files>
```

You will protect yourself from email harvesting bots by doing this.

On Windows based hosts you can do this by setting correspondent permissions for this folder.

Adding new form to config.php file

Now we have all required files in place and we can edit our *config.php*

Note:
Please, use plain text editor to edit config.php to prevent adding unnecessary lines and symbols.

Sample content for config.php

```
<?php header('Location: index.php'); ?>
/* GLOBAL FPP SETTINGS - BEGIN. THESE SETTINGS ARE
APPLIED TO ALL FORMS! */

mysql_host = localhost
mysql_user = user_name
mysql_password = password
mysql_db = fpp5
```

```
smtp_server = localhost
smtp_port = 25;

date_format = H:m d/m/y

/* GLOBAL FPP SETTINGS - END */

/* FORMs' configurations. [form_name] - Declares form
name and its settings below. */

[some_form]
page = ../form.html
page = ../thank-you.html
required_fields = name (Full Name), email (Contact
Email), subject (Subject)
email_fields = email (Contact Email)
autogen_email = info@mydomain.com
…
```

First, we have to declare new form in our configuration file. We do this by adding form name in [] brackets as a new line at the end of config.php file or just after global settings section. The form's name is the value used for previously added hidden files. In our case it's “contact_form”.
In our case we have:

```
[contact_form]
```

After form's declaration we can put any form settings just after this line.

Defining pages and their order

Ok, now we have to declare all pages that would be used during Form Processor Pro work in a particular order of process. We use “page” setting for this purpose and set full or relative paths to these pages (relative to fpp installation path). If we would like to show thank you page just after successful submission of our one-page contact form we have to put it as second one.
Just like this:

```
page = ../contact.html
page = ../thank-you.html
```

Note:
You have full control over any server-side script execution on form pages.

If you use relative or absolute path to file (e.g.: "../form/file.html" or "/fpp/form/filename.php") Form Processor Pro will open it as a text file. No scripts (e.g.: PHP, ASP) will be executed on the page.

If you use full URL (e.g.: "http://www.yoursite.com/path/file.html") Form Processor Pro will let webserver execute all server-side scripts on the page, and then will parse the form.
PHP option "allow_url_fopen" must be enabled to allow this behaviour.

Setting up validations

In order to set Full Name, Email and Subject fields as required we need to use “required_fields” validation rule and put all required field names here. See below:
required_fields = name (Full Name), email (Contact Email), subject (Subject)

Please note that phrases in (and) brackets are just used in error messages instead of internal field names from HTML. Thus your error messages can be intuitive, user-friendly and you can even use language other than English! As we have already mentioned you can put validation rules as one field per line or write all in one string and separate field names by comma.
Thus:

```
required_fields = name (Full Name), email (Contact Email),
subject (Subject)
```

```
and
required_fields = name (Full Name)
required_fields = email (Contact Email)
required_fields = subject (Subject)
```

Will have the same effect - required form fields: name, email and subject.

In order to restrict only properly formatted emails in email field we will use "email_fields" validation rule, the same way as above:

```
email_fields = email (Contact Email)
```

As you can see you can have more than one validation per one field - any amount, actually. But they have to be reasonable of course.

For more information about possible validation rules refer to Field Validations chapter.

Setting up email sending

We would like to send two emails: one for us with form results and another as auto responder for submitter. We have already created these templates and we use them here with "email" action. Just like this:

```
email = ../contact/email.txt
email = ../contact/autoresponder.txt
```

For more information about possible actions refer to Actions chapter.

That's all configurations for the config.php file according to our initial requirement to this Contact Form.

After adding all of the above to our config.php we will have the following changes in it:

```
<?php header('Location: index.php'); ?>
/* GLOBAL FPP SETTINGS - BEGIN. THESE SETTINGS APPLY
TO ALL FORMS! */

mysql_host = localhost
mysql_user = user_name
mysql_password = password
mysql_db = fpp5

smtp_server = localhost
smtp_port = 25;

date_format = H:m d/m/y

/* GLOBAL FPP SETTINGS - END */

/* FORMs' configurations. [form_name] - Declares form
name and its settings below. */

[contact_form]
page = ../contact.html
page = ../thank-you.html
required_fields = name (Full Name), email (Contact
Email), subject (Subject)
email_fields = email  (Contact Email)
email = ../contact/email.txt
email = ../contact/autoresponder.txt

[some_form]
page = ../form.html
page = ../thank-you.html
required_fields = name (Full Name), email (Contact
Email), subject (Subject)
email_fields = email (Contact Email)
autogen_email = info@mydomain.com
…
```

That's all! We have finished configuring our first form!

2.6 Configuration File Overview

Form Processor Pro is tuned and configured by special configuration file located in Form Processor Pro directory and named as *config.php*. It is a simple INI-file where you can specify all your forms and their global and local settings. This file sets basic variables to make Form Processor Pro work correctly. You can disable certain variables by putting ‘#' in the beginning of the line with variable. All symbols put after ‘#' are not treated as data and are mostly used as comments.

Note:
Please, use plain text editor to edit config.php to prevent adding unnecessary lines and symbols to it.

Each setting may be applied globally as well as only for a custom form.

Form descriptions in configuration file start by declaring a new, INI-file style, section.

Example:

```
[myform]
page = ../index.html
page = ../preview.html
page = ../thank-you.html
email = ../emailtpl.txt
```

In the configuration above we have set up the form named “myform” that has 3 pages and sends one email after filling is over.

Note:
In form names you can use only letters, numbers, underscore and dash characters.

2.7 Templates Syntax

Templates are email template, preview page or any page following the first one.

All template tags are enclosed within delimiters. These delimiters are {# and #}. All content outside of delimiters is displayed as static content, or unchanged. When Form Processor Pro encounters template tags, it attempts to interpret them, and displays the appropriate output in their place. The first variable in {# #} is usually a name of field which value should be displayed.

Examples:

Lets imagine you have a text field named first_name on your form:

```
<input type="text" name="first_name">
```

And you need to show preview. So, in the preview page you should set:

```
First name: {#first_name#}
```

And after user submitted the form and filled the field first_name with John, he will see on preview page:

```
First name: John
```

It is very simple!

2.8 Custom error reporting

If you wish to change the way of error messages showing, you have next settings to implement in configuration file:

error_block_begin - Error block beginning HTML
error_block_end - Error block ending HTML
error_msg_begin - Error message beginning HTML
error_msg_end - Error message ending HTML
error_field_style - Error field CSS style

Example:

```
error_block_begin = <TABLE align="center" border="1">
error_msg_begin = <TR><TD>
error_msg_end = </TD></TR>
error_block_end = </TABLE>
error_field_style = border: solid 1px red; background:
yellow;
```

If error appears, form applicant will see the error message like a table, and error fields will be highlighted by red border and yellow background.

Remember to put in `<!-- FPP_ERROR -->` tag to the page where you'd like to display error messages.

Part III

3 Field Validations

3.1 Field Validations Overview

Form Processor Pro allows you to control, check and verify user inputted data by using a number of bundled validating functions. It means you may set some restrictions or requirements for any field on your form. For example, if you have some text field, named “your_email” and you want to be sure that all applicants will give you correct email address and nothing else; you should use your form configuration and add the following line:

```
[my_form]
…
email_fields = your_email
…
```

By doing this you are forcing form applicants to provide you with correct email address on field named “your_email”

Detailed syntax of major form validators' configuration:

```
validator_name = field_name1 (field title 1), field_name2
(field name 2), …
```

Where:

validator_name - the name of Form Processor Pro validator.
field_name1,field_name2 - the name of the fields on your form. The exact name of HTML element.

Also you may use special “field titles” for your fields, written in round brackets. It will be shown up in error message, if the field contains any errors or misspells, otherwise field_name will be shown.

Examples for your good understanding:

1. email_fields = email (Email address)
2. zip_fields = zip
3. required_fields = name (First name), zip, email (Email address)

By **example #1** we require form applicants to input correct email address in field “email” and in case it wasn't an email address the error message will be generating, like:

Field "Email address" is not correct email address

Example #2: we are checking zip code on field “zip”, as you see it's not necessary to use field title; in case of error the following message will be shown:

Field "zip" is not correct zip code

Moreover, you can set more than one field in one validation set. Please, refer to **example #3**, we set fields name, zip and email as required to be filled

3.2 Fields Validation Reference

3.2.1 australian_phone_fields

Verify Australian phone numbers. Matches all known formats including normal 10-digit landline numbers (valid area code mandatory) 13, 1300, 1800, 1900, 1902 plus mobile 10 and 11-digit formats.

VALID	0732105432 1300333444

	131313
INVALID	32105432 13000456

3.2.2 belgium_postcode_fields

Postcode for Belgium

VALID	1234
INVALID	123 123A

3.2.3 canadian_provincial_fields

Match provincial codes of Canada.

VALID	ON PE NB
INVALID	OB NM

3.2.4 canadian_zip_fields

Accurate Canadian postal code format verification.

VALID	M1R 4B0 L0R 1B1

	L0R1B9
INVALID	W1R4B0 L0R 1D1 LOR1B9

3.2.5 credit_card_fields

Verify major credit card numbers, including: Visa, MasterCard, Diners Club, Carte Blanche, Discover and American Express.

IMPORTANT: It does not do REAL credit card validation; it only checks could inputted data be a credit card number.

VALID	4111-2222-3333-4444 3411 22222 33333 5111222233334444
INVALID	4111-2222-3333-444 3411-2222-3333-4444 Visa

3.2.6 date_fields

Validate date fields

VALID	10 September 2000 12.12.2006 31 Dec 9999

	29 Feb 2004
INVALID	MMVIII 29 Feb 2002 13 Octo 1998 32 May 1913

3.2.7 domain_fields

Validate domain names.

VALID	email-form.com example.co.uk
INVALID	http://email-form.com test@mitridat.com

3.2.8 dutch_postcode_fields

Dutch zip code verification

VALID	1054 WD 1054WD 1054 wd
INVALID	10543

3.2.9 email_fields

Validate an RFC 2822 email addresses

VALID	user@domain.com user.id@domain-name.com
INVALID	userdomain.com user@300.0.0.1 .user@-domain-.com

file_filter_fields

File type validation. Allow user to submit only files with allowed extensions (see *allowed_files*).

3.2.10 float_fields

Validate decimal numbers

VALID	12 0.25 36.678
INVALID	A10 $25.10

3.2.11 france_postcode_fields

Postcode check for France (including colonies)

VALID	12345 F-12345

	F-2B100
INVALID	S123456 F-123456 123456

3.2.12 french_phone_fields

Validate phone numbers in France.

VALID	01 46 70 89 12 01-46-70-89-12 0146708912
INVALID	01-46708912 01 46708912 +33235256677

3.2.13 german_postcode_fields

Validate German postal codes (PLZ or Postleitzahlen)

VALID	12556 01550 80796
INVALID	05234 8973 62980

3.2.14 icd9_code_fields

ICD9 code validation

VALID	E123. 123.0 045.23
INVALID	b123. 3456.0

3.2.15 image_file_fields

Validate image file names

VALID	picture.jpg picture.gif
INVALID	picture.doc picture.zip

3.2.16 int_fields

Validate integer numbers

VALID	0 5 7
INVALID	12.23 ABC

3.2.17 ip_fields

Validation of the format of IP Addresses

VALID	127.0.0.1 255.255.255.0 192.168.0.1
INVALID	1200.5.4.3 abc.def.ghi.jkl 255.foo.bar.1

3.2.18 ipv6_fields

Match full and compressed IPv6 addresses as defined in RFC 2373

VALID	FEDC:BA98:7654:3210:FEDC:BA98:7654:3210 1080::8:800:200C:417A ::FFFF:129.144.52.38
INVALID	FEDC::7654:3210::BA98:7654:3210 FEDC:BA98:7654:3210 ::

3.2.19 isbn_fields

Validate ISBN number

VALID	0 930289 23 4 1-56389-668-0

	1-56389-016-X
INVALID	123456789X 9-87654321-2 123456-789X

3.2.20 italian_codice_fiscale_fields

Validate Italian fiscal code (codice fiscale).

VALID	SPGGRG73A02E625S czzdll74h18f205w
INVALID	SP6FFFF3A02E625S czzdll74h18f205

3.2.21 italian_postcode_fields

Postcode check for Italy (including possible Vatican/Italy indications)

VALID	1234 V-1234
INVALID	12345

3.2.22 mac_address_fields

Designed to verify a MAC address with hex values separated by a colon

VALID	00:00:39:F9:3C:59 00:90:83:6A:B3:B7

	00:00:39:59:30:5C
INVALID	00:0H:39:59:30:5C 00:39:59:30:5C 00:39:59:30:5C:BZ

3.2.23 month_fields

Month name (number) validation

VALID	2 05 11 Jul August
INVALID	0 13 jne

3.2.24 monthday_fields

Field value should be between 1 and 31.

VALID	15 30
INVALID	32 -3

3.2.25 msoffice_file_fields

Microsoft Office filename validation

VALID	document.doc spearsheet.xls
INVALID	document.dll spearsheet.exe

3.2.26 netherlands_postcode_fields

Validation of the postcodes for the Netherlands

VALID	1234AB 1234 AB
INVALID	123BBB 023AB

3.2.27 percentage_fields

Validate percentage numbers

VALID	13% 24.85%
INVALID	13 24.5 .12%

3.2.28 required_fields

Check fields to be required for filling

VALID	Sometext
INVALID	

3.2.29 roman_num_fields

Roman numerals validation

VALID	XXL MCMXCIX III
INVALID	ALPHA I9E

3.2.30 same_fields

Validate that all listed fields have the same value.

3.2.31 spanish_postcode_fields

Match Spanish postcodes

VALID	01234 50000 12345

INVALID	00 999

3.2.32 ssn_fields

U.S. social security numbers (SSN) validation

VALID	145470191 145 47 0191 145-47-0191
INVALID	000470191 145-00-0191 145.47.0191

3.2.33 swedish_phone_fields

Swedish phone numbers according to SIS standard

VALID	+46 8 123 456 78 08-123 456 78 0123-456 78
INVALID	+46 08-123 456 78 08 123 456 78 0123 456 78

3.2.34 swedish_zip_fields

Validate Swedish zip codes (postnr) with or without space between groups. With leading s- or not

VALID	12345 932 68 S-621 46

INVALID	5367
	425611
	31 54

3.2.35 time_fields

Value should be a correct timestamp.

VALID	1:01 AM 23:52:01 11:04 pm 03.24.36 aM
INVALID	24:03 13 pm

3.2.36 uk_bsc_fields

Validate a UK Bank Sort code

VALID	09-01-29

	05-06-25
INVALID	090125

3.2.37 uk_driver_license_fields

Match the UK Drivers License format

VALID	JOHNS711215GG9SY
INVALID	JOHNS731215GG9SY

3.2.38 uk_nin_fields

UK National Insurance Number (NINo) validation

VALID	JG103759A AP019283D ZX047829C
INVALID	DC135798A FQ987654C KL192845T

3.2.39 uk_postcode_fields

UK postcode validation

VALID	W1A 1AA EC2V 1JN GIR 0AA

INVALID	TB12 1AB EC2V 1JM W2A 1AA

3.2.40 url_fields

Value should be a correct full URL address.

VALID	http://www.email-form.com/online-manual/index.html mailto:support@web-site-scripts.com ftp://ftp.mysite.com
INVALID	www.google.com microhard.com

3.2.41 us_phone_fields

US phone number with optional leading 1, optional area code, and optional delimiters (hyphen, space or period)

VALID	18005551212 1.800.555.1212 1-800-555-1212 (800)555-1212 8005551212 800-555-1212 800.555.1212 5551212

	555-1212 555.1212
INVALID	2-800-555-1212 55-5212 551212 15551212

3.2.42 us_state_fields

US state 2-chars long code validation

VALID	CO GA TX
INVALID	A ZZ Florida

3.2.43 vat_num_fields

Validate VAT numbers

VALID	GB464645 AT134645
INVALID	GB1321 D5464D

3.2.44 vin_fields

Test Vehicle Identification Numbers (VINs)

VALID	abcDEFghp3t123456 A1C3E5G6Y98123456 A1C3E5G6FFF123456
INVALID	ABBBC3E5G6F9F12345HHH

3.2.45 weekday_fields

Weekday names (numbers) validation.

VALID	3 05 Tue Sunday
INVALID	8 09 12 Wdn 1st

3.2.46 word_fields

Validate single word

VALID	someword

INVALID	The word

3.2.47 year_fields

Validate number of the year.

VALID	2007 06 98
INVALID	132 24566

3.2.48 zip_fields

US (ZIP and ZIP+4) validation

VALID	55127 55127-1234
INVALID	D11 501-555

3.2.49 zip_files_fields

Zip archive files validation

VALID	secret.zip
INVALID	file.doc

3.3 File extensions allowed for upload

Set allowed file extensions for uploaded user files in file_filter_fields.

Syntax:

```
allowed_files = ext
```
or
```
allowed_files = ext1|ext2|ext3
```

Example:

In your form:

```
…
<input type="file" name="attach">
…
```

Config file:

```
allowed_files = gif|bmp|png
file_filter_fields = attach (Picture)
```

Result:

Only *.gif, *.bmp, *.png files would be allowed for user upload.

Part IV

4 Actions

4.1 Actions Overview

The main purpose of Form Processor Pro is to perform some actions and process submitted data. In form configuration file you can specify the sequence and number of required actions such as sending one or several emails, database queries, redirects, etc. Like a validator, each action has a name and custom parameters. Parameters (arguments) can be static as well as dynamic, that depends on submitted data.

Basic actions:

The very basic action is “page”. It used to set the sequence and names of your form pages.

Example:

```
page = ../index.html
page = ../preview.html
page = ../thank-you.html
```

With such a configuration, after user filled the form on index.html, he/she will see parsed preview.html page and after that - thank-you.html

However, action's argument can be a template tag and depends on submitted data. Like

```
page = ../{#page_from_form#}
```

In this case forms applicant will see the page, which name was submitted in “page_from_form” field.

Action arguments may be very different, depends of action it does. It may be set of field names, SQL query, path to template, etc.

4.2 Email Sending Action

Process email-templates and send emails.

Parameter: Path to the template file.

Example:

```
email = ../somedir/email.txt
```

Email-templates should have the following format:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject

This is an email body.
```

Note:
There is an empty line before email body and no empty lines between headers.

So, first of all you have to set correct headers. The required headers are - “To:”, “From:” and “Subject”. Additionally you can add:

Cc: - carbon copy

Bcc: - blind carbon copy

Format: - sets email format, HTML or plain (plain by default)

Charset: - sets email charset (ISO-8851, UTF-8, etc)

Attachment: {#user-file#} - if you want user to attach file via the form. In this case "user-file" should be an upload field name from your form.

Attachment: - file name to attach with corresponding path. Note, all paths should be provided relatively to Form Processor Pro directory or absolutely.

Examples:

```
Attachment: /path/to/file.zip
```

-or-

```
Attachment: ../myform/test.zip
```

For properly mail contents display, same charsets need to be used both in html pages and in mail templates.

Email templates will be parsed as a usual template and you may use all parser features. Also, you may separate by comma multiple recipients or attachments

Example of the email template:

```
To: {#email#}

From: user@domain.com

Subject: Your data was successfully added to database


Dear {#name#},

Thank you for filling out our form.
```

Optional setting for email action (config.php):

`smtp_server` - SMTP server host name, address

`smtp_port` - SMTP server port number

auth_email

Does the same as email action, but send messages via SMTP server that requires username and password authorization.

Additional settings (config.php):

`smtp_login` `- Username`

`smtp_password` `- Password`

4.3 HTML email Overview

You can also send HTML-emails with Form Processor Pro. HTML-email templates are used to send HTML-emails.

Entry in the config.php for HTML-email templates is the same as for plain text mail template.

Example:

```
email = ../somedir/email.txt
```

Structure of HTML-email template is similar to plain text template. You should only add following header:

Syntax:

```
Format: html
```

So header of the mail template will look like following:

Example:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject
Format: html

This is an email body.
```

Empty line should be inserted after headers. Then the email body is located.

Body of the HTML-mail is similar to HTML-page and most of the html tags can be used there, however some of tags advised not to be used in it. This may cause incorrect display of the email.

Avoid these tags in HTML email-template:

- `DOCTYPE`
- `HTML tag <HTML></HTML>`
- `BODY tag <BODY></BODY>`
- `All Meta tags <META>`
- `Head tag <HEAD></HEAD>`
- `Base tag <BASE>`
- `Link tag <LINK>`
- `Script tag <SCRIPT></SCRIPT>`
- `Title tag <TITLE></TITLE>`
- `Applet tag <APPLET></APPLET>`
- `Frameset tag <FRAMESET></FRAMESET>`
- `Frame tag <FRAME>`
- `IFrame tag <IFRAME></IFRAME>`
- `Comments <!-- comments -->`

Note:
It is advised to create email templates in plain-text editor. If you use some HTML editor (like Dreamweaver® or FrontPage®), make sure that it doesn't add these unwanted tags without notice.

Also there is some general advices for HTML-mail

1. Use tables for layout.

Because of the very limited support of style sheets in webmail clients, the best and safe way to layout your HTML email is to use tables.

2. Use absolute urls for all your images and links.

Example:

```
<img src="http://www.example.com/images/head.gif"
width="20" height="60">
<a href="http://www.example.com/products/shoes/tiger.
html">Tiger</a>
```

3. **Use simple style sheets, that means not to use absolute or relative positioning.**
   This is bad supported by most webmail. Don't use external or embedded style sheets, because email programs are removing or ignoring everything between the <HEAD></HEAD>. That means the <LINK> tag won't work if you want to define external style sheets.

   Use inline stylesheets for fonts, font colors, links, background colors, etc.

Example:

```
<table width="400" cellspacing="0" cellpadding="0">
<tr>
<td style=”font-family: Verdana, Arial, Helvetica, sans-
serif; font-size: 10px; line-height: 150%; color: red”>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing
elit.</p>
</td>
<td style=”font-family: Verdana, Helvetica, sans-serif;
font-size: 10px; text-transform: uppercase; color: black”>
Vivamus ut sem. Fusce aliquam nunc vitae purus.</td>
</tr>
</table>
```

4. **Create your design with a white background color.**
   Most webmail and email clients use a white background color to display messages. Often webmail clients ignore or remove the <BODY> tag, which is why background colors often don't work.

   When you do use a different background color than white you could mess up your design. This could for example happen if you create your design for a red background. Images may have a red background. When viewed in a webmail client, you most likely will see a white background and images with a red background that doesn’t seamlessly integrate with the background color.

The best thing to do is to create your design with a white background color. If you must use a different background color than white, you can use a table with a 100% width and perhaps 100% height to simulate the background color:

Example:

```
<table width="100%" height="100%">
<tr>
<td bgcolor=”#FF0000?>
<table width=”400? align=”center”>
<tr>
<td>Lorem ipsum dolor sit amet consectetuer</td>
</tr>
</table>
</td>
</tr>
</table>
```

5. **You have to specify the font, font color and the font size for all the text and links in every table cell.**
   If you don’t specify a style, then most webmail clients will use their own style sheets. This could result in displaying different fonts, font colors and sizes and could also mess up your design.

   Hotmail and Yahoo display Arial as the default font if you don’t specify a font. Gmail uses Verdana. Hotmail uses a 11px
   as the default font size, Yahoo uses 12px and Gmail uses 16px.

Example:

```
<table width="400" cellspacing="0" cellpadding="0">
<tr>
<td style=”font-family: Verdana, sans-serif; font-size:
11px; color: black”>
<p><a href=”http://www.example.com/link/” style=”font-
family: Verdana, font-size: 11px; color: blue”>Lorem</a>
ipsum dolor sit amet, consectetuer adipiscing elit.</p></
td>
<td style=”font-family: Verdana, sans-serif; font-size:
10px; color: black”>Vivamus ut sem. Fusce aliquam nunc
vitae purus.</td>
</tr>
</table>
```

6. **The <P> tag in Internet Explorer uses more space (margin-top and margin-bottom) than in Gecko-based browsers. This could lead to layout problems.**

   For example, if you're using a table width a fixed width and height and your design is based upon these dimensions. If you have content in the table within <P> tags, than the height of the table could increase in Internet Explorer which could lead to gaps in your design.

   To prevent layout problems use the <BR> tag instead of the <P> tag as it renders the same in all browsers. If you still have troubles with the layout you can also use spacers instead of the <P> and the <BR> tag.

   Another problem with <P> tags is in Windows Live Hotmail. <P> tags are removed. So the safest way is to use the <BR> tag

4.4 Auto generated Email

Provides you with possibility to send auto generated emails without creating any email templates. It takes only one argument - email address.

Example:

```
autogen_email = user@domain.com
```

By adding the line above to form configuration will give you the opportunity to receive auto generated emails that contain all submitted data from the form to user@domain.com email box.

Optional settings are the same as for “email”.

4.5 Log submissions to file

Log template is designed to record all form submissions and to present them in a suitable human-readable form to customer.

Log templates must have the filename to write to as the first line. Everything after the first line will be appended (tacked onto the bottom of) the file specified.

Example (for config.php):

```
log_file = log.txt
```

Example of the content for logging template (DB or CSV file):

```
simple.csv
{#eMail#};{#Name#}
```

Result (simple.csv file will contain such lines):

```
one@email.com;name1
two@email.com;name2
```

You can open CSV file in MS Excel as simple spreadsheet.

Note:
Path to log file in log template is relative to log template file. For example: if they in the same folder, it should be just name of log file. If log template in folder 'public_html/' and log file is in folder 'public_html/log/', then it should be 'log/simple.log'

4.6 Database Storage

You may execute any database query after each form submission to save user input to your database.

4.6.1 MySQL

Use “mysql_query” action to save data to MySQL database. This action works with INSERT and UPDATE queries. If you want to get data from MySQL database, please use Database Select action.

Syntax:

```
mysql_query = <query>
```

“query” may be any valid MySQL query.

Example:

```
mysql_query = INSERT INTO mytable (name, email) VALUES
("{#name#}", "{#emai#}")
```

Entities {#name#} and {#email#} will be changed to user submitted data on “name” and “email” field.

To set MySQL hostname, username, and password use the following **settings** in your config.php file:

mysql_host - MySQL hostname

mysql_user - MySQL username

mysql_password -MySQL password

mysql_db - the name of MySQL database to be used

These settings are common for for all forms and must be set in the first section of config.php before form configuration blocks.

Note:
MySQL database storage support is available in the following licenses: Business, World Wide, ISP only.

4.6.2 PostgreSQL

PostgreSQL query executor.

Syntax:

```
pgsql_query = <query>
```

<query> is any valid PostgreSQL query.

Example:

```
pgsql_query = INSERT INTO test (name, email) VALUES
("{#name#}", "{#email#}");
```

Additional settings:

pgsql_host - PostgreSQL hostname

pgsql_user - PostgreSQL user

pgsql_password - PostgreSQL password

pgsql_db - PostgreSQL database name

Note:
PostgreSQL database storage support is available in the following licenses: Business, World Wide, ISP only.

4.6.3 MS SQL

Microsoft SQL query executor.

Syntax:

```
mssql_query = <query>
```

Where <query> is any valid Microsoft SQL query.

Example:

```
mssql_query = INSERT INTO test (name, email) VALUES
("{#name#}", "{#email#}");
```

Additional settings:

mssql_host - Microsoft SQL hostname

mssql_user - Microsoft SQL user

mssql_password - Microsoft SQL password

mssql_db - Microsoft SQL database name

Note:
MS SQL Database storage support is available in the following licenses:
Business, World Wide, ISP only.

4.6.4 SQLite

SQLite query executor.

Syntax:

```
sqlite_query = <query>
```

Where <query> is any valid SQLite query.

Example:

```
sqlite_query = INSERT INTO test (name, email) VALUES
("{#name#}", "{#email#}");
```

Additional settings:

sqlite_file - SQLite filename

Note:
SQLite database storage support is available in the following licenses:
Business, World Wide, ISP only.

4.6.5 ODBC

ODBC SQL query executor.

Syntax:

```
odbc_query = <query>
```

Where <query> is any valid SQL query.

Example:

```
odbc_query = INSERT INTO test (name, email) VALUES
("{#name#}", "{#email#}");
```

Additional settings:

odbc_dsn - ODBC DSN value

Note:
ODBC Database storage support is available in the following licenses:
Business, World Wide, ISP only.

4.7 Database Select Queries

The "mysql_select" action selects data from database using MySQL query and puts it to specified fields with corresponding names.

This action supports "SELECT" queries only. If you want to update or insert data to MySQL database, use MySQL Database Storage action.

Syntax:

```
mysql_select query[; error message]
```

`query` - mysql SELECT query.

`error message` - optional parameter. If present, will be shown on the form in Error_Message_Block if query returns 0 records. If not present, form processing will be continued even if 0 records were returned.

Example:

mysql_select SELECT `country`, `city`, `street`, `phone` FROM `form_table` WHERE `name` = '{#name#}' AND `surname` = {#surname#}

Result:

This query makes selection from the "form_table" table where 'name' and 'surname' values matches submitted fields, and puts data from the first matched result to "country", "city", "street", "phone" fields. You can use these fields later in templates and form pages as if they where inputted by user (e.g.: {#country#}, {#city#}).

Example showing two forms with add record and edit record functions:

Let's say we have a small form with following fields:

name

surname

country

city

street

phone

This form has following query in configuration file:

mysql_query = INSERT INTO `form_table` (`name`, `surname` , `country`, `city`, `street`, `phone`) VALUES ('{#name#}', '{#surname#}', '{#country#}', '{#city#}', '{#street#}', '{#phone#}',)

User can submit this form and his information will be added to the database by mysql_query action.

Now lets think we have a form where user can edit his record.

This form should have 3 pages: login page, edit page and thank you page

Login page can contain 2 fields (or any other number of fields), these fields will be used to identify the correct record in the database.

Let's think we have login page with fields:

name

surname

So user can restore his data using his name and surname.

Second page has following fields:

name

surname

country

city

street

phone

In config file we have:

mysql_select = SELECT `country`, `city`, `street`, `phone` FROM `form_table` WHERE `name` = '{#name#}' AND `surname` = {#surname#}; Error: no such name/surname in the database

mysql_query = UPDATE `form_table` SET `name` = '{#name#}', `surname` = '{#surname#}', `country` = '{#country#}', `city` = '{#city#}', `street` = '{#street#}', `phone` = '{#phone#}'

First query loads user information (`country`, `city`, `street`, `phone`) from the database record which contains user's name and surname. If record with such name and surname doesn't exist the error message after ';' sign will be shown and form will not go to edit page.

If record was loaded, user will go to edit page with editable fields filled with information from the database.

When user submits edit page, second MySQL query runs and updates record in the database.

Note:
MySQL database queries support is available in the following licenses: Business, World Wide, ISP only.

4.8 Payment Gateways

Payment gateway actions help you to create shopping carts. If your form has this action, user will be redirected to the payment checkout system after submission.

4.8.1 PayPal

Redirects form applicants to PayPal checkout. Require arguments - 3 variables. First sets your PayPal business name, second - sets item to be sold name, third - item's price (in U.S. dollars)

Example:

```
paypal_checkout = user@domain.com, My special product, 30.95
```

Note:
PayPal payment gateway support is only available in the following licenses: Business, World Wide, ISP.

4.8.2 2Checkout

Redirects form applicants to 2Checkout checkout page. Required arguments - first 4 variables.

SID - is your 2CO's vendor id (SID)
total - total amount or purchase. You can use mathematical_calculations and "If"_condition to calculate total amount of purchase. You can also find examples of calculation in the "checkout-page-with-connection-to-payment-gateway" demo form.
unique_number - is a unique number of the shopping cart session, you can use `{#%UNIQUE_REFERENCE#}` function that will generate unique number for each submission.
product1_id, product1_quantity,product1_price - 2CO id of your product, number of items and product price.

You can set as many products as you wish. Product_price and product _quantity are optional parameters, if not specified product price will be set by 2CheckOut system to default value, and product quantity will be set to 1. You can leave the space between commas empty to omit optional parameters.

Syntax:

```
tco_checkout = SID, total, unique_number, product1_id,
product1_quantity, product1_price, product2_id,
product2_quantity, product2_price, ...
```

Example:

```
tco_checkout = 1234567, 139.90, {#%UNIQUE_REFERENCE#},
my_product1, 2, 69.95
```

Note:
2Checkout payment gateway support is only available in the following licenses: Business, World Wide, ISP.

4.8.3 LinkPoint

Charge user credit cards via LinkPoint payment gateway. Require 5 parameters.

1. Your config file number (e.g. 1234567)
2. Credit card number (e.g. 41111111111111111111)
3. Card expire month (e.g. 10)
4. Card expire year (e.g. 07)
5. Amount to charge (e.g. 39.95)

Also, you should put your PEM file to tmp/ directory.

Example:

```
linkpoint_checkout = 1234567, {#card_num#},
{#card_exp_month#}, {#card_exp_year#}, 40.00
```

Note:
LinkPoint payment gateway support is only available in the following licenses: Business, World Wide, ISP.

4.9 File Output

As well as page you may output any file stored on your server. Use action ‘file' instead of ‘page'. This is very useful in case you want to gather some information prior allowing downloading certain file.

Example of config.php:

```
page = ../index.html

page = ../preview.html

file = /usr/home/secured/area/secretfile.zip
```

In the example above Form Processor Pro will output the file secretfile.zip after preview page.

4.10 Simple Redirect

Allows you to redirect form applicants to predefined URL

Example:

```
page = ../index.html

redirect = http://somesite.com/
```

4.11 Referers

"Referrers" is the feature that denies access to you Form Processor Pro from domains that is not listed. This feature is optional. If it is set up, form can be submitted only from specified domains.

Example:

```
referers = www.yourdomain.com, localhost, web-site-scripts.
com

referers = www.yourdomain.com
```

Referrers specifies for each form separately. You can set "localhost" if form is located on the same server as Form Processor Pro is.

4.12 Password Generation

This feature might be helpful, if you want to generate a password for a user who submits the form. You can send generated password to email, save it to a database or use as any other field.

Syntax:

```
gen_pass = field_name(num), dict_file.php
```

Example:

`gen_pass = password(3)` - will generate password with length 3 symbols

`gen_pass = password` - will generate password with default length (6 symbols)

`gen_pass = password, ../forms/sample/dict.php` - will select randomly one of the passwords form dictionary located at `../forms/sample/dict.php` and write it to the "password" field.

Example of dict.php:

```
<?php header('Location: index.html'); ?> #DO NOT
REMOVE THIS LINE!
password0
password1
password2
```

`<?php header('Location: index.html'); ?>` - this line will protect your password dictionary from unauthorized access. You should put the name of actual file instead of "`index.html`" or create an empty "`index.html`" file.

4.13 Time Zone

Time Zone feature help you to set up time zone of your forms. If your server time zone differs from the your time zone, you can set this difference and receive forms with correct timestamp.

Example:

```
time_zone = -5
```

It specified relatively to server time. Time zone sets up for each form separately.

4.14 Expiration Date for the Form

You may set expiration date for any form by using *expire_date* validation.

You should use these settings in config.php file for certain form or global.

Syntax:

```
expire_date = <date>
```

Where <date> is any US English date.

Example:

```
expire_date = 23 September 2007
```

This will deny all users to submit your form after 23 September 2007.

4.15 Unique Submissions

If you want to limit your form submissions by 1 or couple times per some person, use *unique_submits* action in config.php

Syntax:

```
unique_submits = <number>, <period>
```

Where <number> is a number of times the form can be submitted during any <period> of time.

Example:

```
unique_submits = 1, 3M
```

Means that form can be submitted just once per three months.

You can specify period by any integer number followed by a letter exterminating a period:

S - seconds

I - minutes

H - hours

D - days

M- months

Y - years

Examples:

```
6H, 30I, 5D, 1Y
```

Part V

5 Modifiers

5.1 Modifiers Overview

Modifiers used in order to perform certain transformations with user input prior inserting result into template or generating output.

Variable modifiers can be applied to any variable in a template tag. To apply a modifier, specify the value followed by the | (pipe) and the modifier name. A modifier may accept additional parameters that affect its behaviour. These parameters follow the modifier name and are separated by : (colon) and quoted by double-quotes.

Examples:

```
{#title|upper#}

{#title|truncate:40:"..."#}

{#title|lower#}
```

First example uppercases the value of title variable. Second truncates it to 40 characters and add … at the end of string and the third makes the title value lowercased.

5.2 Modifiers Reference

This chapter includes complete reference of Form Processor Pro modifiers.

5.2.1 capitalize

This is used to capitalize the first letter of all words in a variable.

Example:

Input:

john smith

Template:

```
{#name|capitalize#}
```

Output:

John Smith

5.2.2 count_characters

This is used to count the number of characters in a variable.

Example:

Input:

john smith

Template:

```
{#name|count_characters#}
```

Output:

```
9
```

5.2.3 count_paragraphs

This is used to count the number of paragraphs in a variable.

Example:

Input:

```
This is a test paragraph.

The next paragraph
```

Template:

```
{#article|count_paragraphs#}
```

Output:

```
2
```

5.2.4 count_sentences

This is used to count the number of sentences in a variable.

Example:

Input:

His name is John Smith. He is a good man.

Template:

```
{#ext|count_sentences#}
```

Output:

2

5.2.5 count_words

This is used to count the number of words in a variable.

Example:

Input:

His name is John Smith. He is a good man.

Template:

```
{#text|count_words#}
```

Output:

10

5.2.6 date_format

Returns a string formatted according to the given format string. Use variable %TIMESTAMP to get a timestamp

Example:

Template:

```
{#%TIMESTAMP|date_format:"m/d/y"#}
```

Output:

12/15/2006

You can also use characters from following tables to customize the "`date_format`" modifier.

The following characters are recognized in the `date_format` modifier		
***`format`* character**	**Description**	**Example returned values**
Day	---	---
d	Day of the month, 2 digits with leading zeros	*01* to *31*
D	A textual representation of a day, three letters	*Mon* through

The following characters are recognized in the `date_format` modifier		
		Sun
j	Day of the month without leading zeros	*1* to *31*
l (lower case 'L')	A full textual representation of the day of the week	*Sunday* through *Saturday*
N	ISO-8601 numeric representation of the day of the week (allowed in PHP 5.1.0+)	*1* (for Monday) through *7* (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters	*st*, *nd*, *rd* or *th*. Works well with *j*
w	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
z	The day of the year (starting from 0)	*0* through *365*
Week	---	---
W	ISO-8601 week number of year, weeks starting on Monday (allowed in PHP 4.1.0+)	Example: *42* (the 42nd week in the year)
Month	---	---
F	A full textual representation of a month, such as January or March	*January* through *December*

The following characters are recognized in the `date_format` modifier		
m	Numeric representation of a month, with leading zeros	*01* through *12*
M	A short textual representation of a month, three letters	*Jan* through *Dec*
n	Numeric representation of a month, without leading zeros	*1* through *12*
t	Number of days in the given month	*28* through *31*
Year	---	---
L	Whether it's a leap year	*1* if it is a leap year, *0* otherwise.
o	ISO-8601 year number. This has the same value as *Y*, except that if the ISO week number (*W*) belongs to the previous or next year, that year is used instead. (allowed in PHP 5.1.0+)	Examples: *1999* or *2003*
Y	A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*
y	A two digit representation of a year	Examples: *99* or *03*

`format` character	**Description**	**Example returned values**
Time	---	---
a	Lowercase Ante meridiem and Post meridiem	*am* or *pm*
A	Uppercase Ante meridiem and Post meridiem	*AM* or *PM*

`format` character	Description	Example returned values
B	Swatch Internet time	*000* through *999*
g	12-hour format of an hour without leading zeros	*1* through *12*
G	24-hour format of an hour without leading zeros	*0* through *23*
h	12-hour format of an hour with leading zeros	*01* through *12*
H	24-hour format of an hour with leading zeros	*00* through *23*
i	Minutes with leading zeros	*00* to *59*
s	Seconds, with leading zeros	*00* through *59*
u	Microseconds (allowed in PHP 5.2.2+)	Example: *54321*
Timezone	---	---
e	Timezone identifier (allowed in PHP 5.1.0+)	Examples: *UTC, GMT, Atlantic/ Azores*
I (capital i)	Whether or not the date is in daylight saving time	*1* if Daylight Saving Time, *0* otherwise.
O	Difference to Greenwich time (GMT) in hours	Example: *+0200*
P	Difference to Greenwich time (GMT) with colon between hours and minutes (allowed in PHP 5.1.3+)	Example: *+02:00*
T	Timezone abbreviation	Examples: *EST, MDT ...*
Z	Timezone offset in seconds. The offset for timezones	*-43200* through

format character	Description	Example returned values
	west of UTC is always negative, and for those east of UTC is always positive.	*50400*

Note:
Unrecognized characters in the format string will be printed as-is.

5.2.7 default

This is used to set a default value for a variable. If the variable is empty, the given default value is printed instead. Default takes one argument.

Example:

Input:

Template:

```
{#name|default:"-=none=-"#}
```

Output:

```
-=none=-
```

5.2.8 indent

This indents a string at each line, default is 4. As an optional parameter, you can specify the number of characters to indent. As an optional second parameter, you can specify the character to use to indent with. (Use "\t" for tabs.)

Example:

Input:

NJ judge to rule on nude beach.
Sun or rain expected today, dark tonight.
Statistics show that teen pregnancy drops off significantly after 25.

Template:

```
{#article|indent:10#}
```

Output:

```
          NJ judge to rule on nude beach.
          Sun or rain expected today, dark tonight.
          Statistics show that teen pregnancy drops off significantly after 25.
```

5.2.9 lower

This is used to lowercase a variable.

Example:

Input:

John SMITH

Template:

```
{#name|lower#}
```

Output:

john smith

5.2.10 nl2br

All line breaks will be converted to <br> tags in the given variable. It's very usable to output text area fields

Example:

Input:

John

Smith

Template:

```
{#name|nl2br#}
```

Output:

John<br>Smith

5.2.11 regex_replace

A regular expression search and replace on a variable. Use the syntax for preg_replace() from the PHP manual.

Example:

Input:

John Smith

Template:

```
{#name|regex_replace:"/h/":"H"#}
```

Output:

JoHn SmitH

5.2.12 replace

A simple search and replace on a variable.

Example:

Input:

John Smith

Template:

```
{#name|replace:"John":"Samuel"#}
```

Output:

Samuel Smith

5.2.13 spacify

Spacify is a way to insert a space between every character of a variable. You can optionally pass a different character (or string) to insert.

Example:

Input:

John

Template:

```
{#name|spacify#}
```

Output:

```
John
```

5.2.14 string_format

This is a way to format strings, such as decimal numbers and such. Use the syntax for sprintf PHP function for the formatting.

Example:

Input:

```
2.43243252
```

Template:

```
{#num|string_format:"%.2f"#}
```

Output:

```
2.43
```

5.2.15 strip

This replaces all repeated spaces, new lines and tabs with a single space, or with a supplied string

Example:

Input:

John   Smith

Template:

```
{#name|strip#}
```

Output:

John Smith

5.2.16 strip_tags

This strips out markup tags, basically anything between < and >.

Example:

Input:

John <b>Smith</b>

Template:

```
{#name|strip_tags#}
```

Output:

John Smith

5.2.17 truncate

This truncates a variable to a character length, default is 80. As an optional second parameter, you can specify a string of text to display at the end if the variable was truncated. The characters in the string are included with the original truncation length. By default, truncate will attempt to cut off at a word boundary. If you want to cut off at the exact character length, pass the optional third parameter of true.

Example:

Input:

Two Sisters Reunite after Eighteen Years at Checkout Counter

Template:

```
{#article|truncate:30:"...":true#}
```

Output:

Two Sisters Reunite after E...

5.2.18 upper

This is used to uppercase a variable.

Example:

Input:

John Smith

Template:

```
{#name|upper#}
```

Output:

JOHN SMITH

5.2.19 wordwrap

This wraps a string to a column width, default is 80. As an optional second parameter, you can specify a string of text to wrap the text to the next line (default is carriage return \n). By default, wordwrap will attempt to wrap at a word boundary. If you want to cut off at the exact character length, pass the optional third parameter of true.

Example:

Input:

Blind woman gets new kidney from dad she hasn't seen in years.

Template:

```
{#article|wordwrap:"30":"\n":"true"#}
```

Output:

Blind woman gets new kidney

from dad she hasn't seen in

years.

5.3 Combining modifiers

You can apply any number of modifiers to a variable. They will be applied in the order they are combined, from left to right. They must be separated with a | (pipe) character.

Example:

Input:

Smokers are Productive, but Death Cuts Efficiency

Template:

```
{#article#}
{#article|upper|spacify#}
{#article|lower|spacify|truncate#}
{#article|lower|truncate:30|spacify#}
```

```
{#article|lower|spacify|truncate:30:". . ."#}
```

Output:

Smokers are Productive, but Death Cuts Efficiency.

S M O K E R S   A R E   P R O D U C T I V E ,   B U T   D E A T H   C U T S   E F F I C I E N C Y .

s m o k e r s   a r e   p r o d u c t i v e ,   b u t   d e a t h   c u t s...

s m o k e r s   a r e   p r o d u c t i v e ,   b u t . . .

s m o k e r s   a r e   p. . .

Part VI

6 Advanced Features

6.1 Mathematical calculations

Furthermore, you may do some math and calculation by using Form Processor Pro. Just make sure you have separated all variables and signs by a white space. That's the only one rule.

Example:

You have three text input fields:

```
<input type="text" name="field_a">

<input type="text" name="field_b">

<input type="text" name="field_c">
```

You can use following expression on the next form page, in the database template, log file template or email template.

Template:

```
{# field_a + ( field_b * field_c ) #}
```

Let's say you enter following numbers into those fields:

```
Into field_a: 2
Into field_b: 5
Into field_c: 7
```

Result:

```
37
```

6.2 "If" condition

"If" condition allows selecting the data output depending on some condition of user-entered parameters.

Template:

```
{#ifcond (( expr1 ), ( expr2 ), ( expr3 ))#}
```

This expression evaluates and prints result of expr2 if expr1 evaluates to TRUE, and expr3 if expr1 evaluates to FALSE.
You can use "==", "<" and ">" symbols in the "expr1" expression.
Left "{#" and right "#}" delimiters for aren't being used inside "ifcond" operator.
Note that spaces must be placed like in the example.

Example:

```
{#ifcond
(( Number1 > Number2 ), ( Number1 + 10 ), ( Number2 ))#}
```

Let's say we have **Number1 = 20** and **Number2 = 5**. Then **Number1** is greater than **Number2**, expression **Number1 > Number2** is TRUE and we print **Number1 + 10**, this equals **30**.

Result:

```
30
```

6.3 "Switch" function

"Switch" function allows you to output custom message depending on the user input. It simplifies some cases when you can use "If" condition. It should be used if you have radio buttons, dropdown menus, and multiple lists and on one hand want to leave field values short and simple, and on the other hand want to output human-readable text depending on user's choice.

Template:

```
{#sw(field_name, "field_values_separated_by_commas",
"output_strings_separated_by_commas ")#}
```

Example:

```
Which way do you want to travel?

<label><input type="radio" name="travel" value="CAR">Drive a
car</label>

<label><input type="radio" name="travel" value="HORSE"
checked>Ride a horse</label>

<label><input type="radio" name="travel" value="PLANE">Take
a plane</label>
```

We use "Switch" function in one of the templates or another form pages:

```
{#sw(travel, "CAR,HORSE,PLANE", "Drive a car,Ride a horse,
Take a plane")#}
```

Let's say user check the "HORSE" radio button.

Result:

Ride a horse

6.4 CAPTCHA Feature

CAPTCHA is a type of challenge-response test used to determine whether or not the user is a human. CAPTCHA requires that the user type the letters of a distorted image, with the addition of an obscured sequence of letters or digits that appears on the screen. To add it to your page you need to:

- create field to enter CAPTCHA
  ```
  <INPUT TYPE="text" name="mycaptcha"  VALUE="">
  ```
- add CAPTCHA picture using this code:
  ```
  <img src="../fpp/captcha.img.php?">
  ```
- then point in the config.php file:
  ```
  captcha_field = mycaptcha
  ```

Note:
CAPTCHA is only available in the following licenses: Business, World Wide, ISP.

6.5 Fields Counter

Fields Counter helps you to count how many fields from a field set satisfy certain conditions. This may be used in surveys and with repeating field set.

Syntax:

```
count_fields output_field; regular_expression; field1[=
value1][; field2 [=value2][; ... ]]
```

output_field - field where result will be put. You can use it in any page or template as any other field: {#output_field#}

regular_expression - count fields fit this expression. See below for detailed explanation.

fieldN = valueN - field conditions, if value of "fieldN" field equals "valueN", then Form Processor Pro counts this field. If value is not specified, Form Processor Pro checks if field is not empty. If was filled, Form Processor Pro counts this field.

Example:

Let's say we have filled a form with following fields ("" - empty values):

name1 = John

age1 = 18

city1 = New York

name2 = Kim

age2 = 18

city2 = Washington

name3 = John

age3 = 21

city3 = Washington

name4= Mary

age4 = 21

city4 = Sydney

name5 = Adolph

age5 = 70

city5 = Berlin

name6 = ""

age6 = 18

city6 = Paris

Result:

You can count different combinations of these fields using count_fields feature

Example 1:

Count how many names were entered:

```
count_fields name_count; /name(.{0,3})/i
```

This will set variable `{#name_count#}` to **5**

Example 2:

Count how many cities are filled:

```
count_fields city_count; /city(.{0,3})/i
```

This will set variable `{#city_count#}` to **6**

Example 3:

Count how many people are aged 18

```
count_fields count_age18; /age(.{0,3})/i = 18
```

This will set variable `{#count_age18#}` to **3**

Example 4:

Count how many people are aged 18 and filled their name:

```
count_fields count_name18; /name(.{0,3})/i; age = 18
```

This will set variable `{#count_name18#}` to **2**

Example 5:

Count how many people are from washington, are 21 and filled their name:

```
count_fields count_name_washington21; /name(.{0,3})/i; age =
21; city = Washington
```

This will set variable `{#count_name_washington21#}` to **`1`**

As you can see, each example contains the same regular expression structure. Let's see how it works:

```
/name(.{0,3})/i
```

"`/`" and "`/`" are used like brackets to set where regular expression starts and ends.

"`name`" - is a part of field name that remains unchanged. In our example there are three of them: "name" , "age", "city".

`(.{0,3})` - is an expression that captures any string with length up to 3. It allows to capture such fields as "name1", "name6", "nameaaa", "nametjz", "name134" in our example, but ignore such fields as "name_of_person", "name9238", "namefrjhgh". You can set any other number instead of "3", but note that larger numbers slow down the script.

`i` - means that fields names would be insensitive to the register, so Form Processor Pro will not make difference between "nAme3", "Name3" and "name3" field names.

6.6 File Generation

Sometimes it's useful to generate some online files that contain user submitted data and much more useful to send it by emails. Predefined file variables lets you easily include such files as attachments into generated emails.

6.6.1 PDF

Generate PDF file with all form's data represented by rows.

For example, you if need to generate a PDF-file with user submitted data and send it by email, **specify** it in your email template:

Syntax:

```
Attachment: {#%FORMFILE_PDF#}
```

and you will receive email with attached PDFs with data submitted.

Example:

Email template:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject
Attachment: {#%FORMFILE_PDF#}

Message Body Text in free form (Either plain text or
HTML formatted text)
```

Note:
File-generation is only available in the following licenses: Business, World Wide, ISP.

6.6.2 PDF with Table view

Same as above, but data output in a table view.

Syntax:

```
Attachment: {#%FORMFILE_PDF_TABLE#}
```

And you will receive email with attached PDFs with data submitted.

Example:

Email template:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject
Attachment: {#%FORMFILE_PDF_TABLE#}

Message Body Text in free form (Either plain text or
HTML formatted text)
```

Note:
File-generation is only available in the following licenses: Business, World Wide, ISP.

6.6.3 XLS

Returns Excel auto-generated file (all data in one row)

Syntax:

```
Attachment: {#%FORMFILE_XLS#}
```

Example:

Email template:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject
Attachment: {#%FORMFILE_XLS#}

Message Body Text in free form (Either plain text or
HTML formatted text)
```

Note:

File-generation is only available in the following licenses: Business, World Wide, ISP.

6.6.4 XLS with Table view

Returns Excel auto-generated file (all data in a table)

Syntax:

```
Attachment: {#%FORMFILE_XLS_TABLE#}
```

Example:

Email template:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject
Attachment: {#%FORMFILE_XLS_TABLE#}
```

```
Message Body Text in free form (Either plain text or
HTML formatted text)
```

Note:

File-generation is only available in the following licenses: Business, World Wide, ISP.

6.6.5 CSV

Return auto-generated CSV (comma separated values) file.

Syntax:

```
Attachment: {#%FORMFILE_CSV#}
```

Example:

Email template:

```
To: recepient@somehost.com
From: sender@somehost.com
Subject: This is an email subject
Attachment: {#%FORMFILE_CSV#}

Message Body Text in free form (Either plain text or
HTML formatted text)
```

6.7 Error Message Block

You may output error messages block in any position you wish. Use a special parser tag to do it:

```
<!-- FPP_ERROR -->
```

In case of error or improper user input, it will be replaced by error message. To tune its view see custom error reporting section of this manual.

6.8 Allow/Deny access to form by IP or Host

You can allow or deny access to a certain form from certain IP, subnet or host.

You should use these settings in config.php file for certain form or global:

6.8.1 Allow IP

You may allow some users to submit your form only by setting their IP addresses or address ranges.

Examples:

```
allow_ip = 123.45.67.8
```

This will allow all users with IP 123.45.67.8 to submit your form.

```
allow_ip = 123.45.*.*
```

This will allow all users with IP range 123.45.0.0 to 123.45.255.255 to fill your form.

```
allow_ip = 123.45.67.*, 123.55.77.88
```

This will allow all users with IP 123.55.77.88 and on range 123.45.67.0 to 123.45.67.255.

6.8.2 Denied IP

You may deny some user to submit your form by setting their IP addresses or address ranges.

Examples:

```
denied_ip = 123.45.67.8
```

This will deny all users with IP 123.45.67.8 to submit your form.

```
denied_ip = 123.45.*.*
```

This will deny all users with IP range 123.45.0.0 to 123.45.255.255 to fill your form.

```
denied_ip = 123.45.67.*, 123.55.77.88
```

This will deny all users with IP 123.55.77.88 and on range 123.45.67.0 to 123.45.67.255

6.8.3 Denied Address

Works exactly as *denied_ip*, but hosts specifically by their names instead of IP address

Examples:

```
denied_addr = someone.domain.com
```

This will deny all users came from someone.domain.com address.

```
denied_addr = *.edu
```

This will deny all users with domain .edu to fill out your form.

6.9 Zip Feature

You can apply *file2zip* modifier to a file name variable, to receive for example a zipped archive of this file. It's very usable for email attachments.

file2zip - Zip the file

Example:

You can apply it in your email templates, like:

```
Attachment: {#user_file|file2zip#}
```

In result, you will receive an email message with attached zip-archive that contains a user uploaded zipped file.

Note:
Zip-compression is only available in the following licenses: Business, World Wide, ISP.

6.10 Predefined Variables

There are some predefined, dynamically generated variables for your usage in templates.

6.10.1 Date

Returns current date.

Syntax:

`{#%DATE#}`

Returns current date. The format should be specified by setting "`date_format`" in configuration file. You can use following common examples of the "`date_format`" setting and output date value.

Examples (Assuming today is: March 10th, 2001)**:**

```
date_format = F j, Y
```

March 10, 2001

```
date_format = m.d.y
```

03.10.01

```
date_format = j, n, Y
```

10, 3, 2001

```
date_format = Ymd
```

20010310

```
date_format = \i\t \i\s \t\h\e jS \d\a\y.
```

It is the 10th day.

```
date_format = D M j Y
```

Sat Mar 10 2001

Note:
If you want to use different date format for each form, refer to `date_format` validator.

You can also use characters from following table to customize the "`date_format`" setting.

The following characters are recognized in the `date_format` setting		
***`format`* character**	**Description**	**Example returned values**
Day	---	---
d	Day of the month, 2 digits with leading zeros	*01* to *31*
D	A textual representation of a day, three letters	*Mon* through *Sun*
j	Day of the month without leading zeros	*1* to *31*
l (lowercase 'L')	A full textual representation of the day of the week	*Sunday* through *Saturday*
N	ISO-8601 numeric representation of the day of the week (allowed in PHP 5.1.0+)	*1* (for Monday) through *7* (for Sunday)

The following characters are recognized in the `date_format` setting		
S	English ordinal suffix for the day of the month, 2 characters	*st*, *nd*, *rd* or *th*. Works well with *j*
w	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
z	The day of the year (starting from 0)	*0* through *365*
Week	---	---
W	ISO-8601 week number of year, weeks starting on Monday (allowed in PHP 4.1.0+)	Example: *42* (the 42nd week in the year)
Month	---	---
F	A full textual representation of a month, such as January or March	*January* through *December*
m	Numeric representation of a month, with leading zeros	*01* through *12*
M	A short textual representation of a month, three letters	*Jan* through *Dec*
n	Numeric representation of a month, without leading zeros	*1* through *12*
t	Number of days in the given month	*28* through *31*
Year	---	---
L	Whether it's a leap year	*1* if it is a leap

The following characters are recognized in the `date_format` setting		
		year, *0* otherwise.
o	ISO-8601 year number. This has the same value as *Y*, except that if the ISO week number (*W*) belongs to the previous or next year, that year is used instead. (allowed in PHP 5.1.0+)	Examples: *1999* or *2003*
Y	A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*
y	A two digit representation of a year	Examples: *99* or *03*

Note:
Unrecognized characters in the format string will be printed as-is.

6.10.2 Time

Returns current time.

Syntax:

```
{#%TIME#}
```

Returns current time. The format should be specified by setting "`time_format`" in configuration file. You can use following common examples of the "`time_format`" setting and output time value.

Examples (Assuming now is: 5:16:18 pm)**:**

```
date_format = g:i a
```

5:16 pm

```
date_format = G:i:s
```

17:16:08

```
date_format = G-i
```

17-16

You can also use characters from following table to customize the "`time_format`" setting.

The following characters are recognized in the *`time_format`* setting		
***`format`* character**	**Description**	**Example returned values**
Time	---	---
a	Lowercase Ante meridiem and Post meridiem	*am* or *pm*
A	Uppercase Ante meridiem and Post meridiem	*AM* or *PM*
B	Swatch Internet time	*000* through *999*
g	12-hour format of an hour without leading zeros	*1* through *12*
G	24-hour format of an hour without leading zeros	*0* through *23*
h	12-hour format of an hour with leading zeros	*01* through *12*
H	24-hour format of an hour with leading zeros	*00* through *23*
i	Minutes with leading zeros	*00* to *59*

The following characters are recognized in the *`time_format`* setting		
s	Seconds, with leading zeros	*00* through *59*
u	Microseconds (allowed in PHP 5.2.2+)	Example: *54321*
Timezone	---	---
e	Timezone identifier (allowed in PHP 5.1.0+)	Examples: *UTC*, *GMT*, *Atlantic/ Azores*
I (capital i)	Whether or not the date is in daylight saving time	*1* if Daylight Saving Time, *0* otherwise.
O	Difference to Greenwich time (GMT) in hours	Example: *+0200*
P	Difference to Greenwich time (GMT) with colon between hours and minutes (allowed in PHP 5.1.3+)	Example: *+02:00*
T	Timezone abbreviation	Examples: *EST*, *MDT ...*
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	*-43200* through *50400*

Note:
Unrecognized characters in the format string will be printed as-is.

6.10.3 Fields List in HTML

Return all form fields and their values in HTML format.

Example:

```
{#%FIELDS_LIST_HTML#}
```

Result:

<b>Field1_Name:</b> Field1_value<br>

<b>Field2_Name:</b> Field2_value<br>

6.10.4 Fields List in Plain Text

Return all form fields and their values in Plain Text format.

Example:

```
{#%FIELDS_LIST_PLAIN#}
```

Result:

Field1_Name: Field1_Value

Field2_Name: Field2_Value

6.10.5 HTTP Referer

Returns current forms URL

Example:

```
{#%HTTP_REFERER#}
```

6.10.6 HTTP User Agent

Return submitter's user agent: OS Type & Version, Browser Type & Version.

Example:

```
{#%HTTP_USER_AGENT#}
```

Result

Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.2) Gecko/20070219 Firefox/2.0.0.2

6.10.7 Remote IP

Return submitter's IP address.

Example:

```
{#%REMOTE_ADDR#}
```

Result:

192.168.0.1

6.10.8 Timestamp

Returns UNIX-timestamp value. Should be used with `date_format` modifier to set required format.

Example:

```
{#%TIMESTAMP#}
```

6.10.9 Unique Reference Number

Returns a very unique reference number. It may be used for your own records.

Example:

```
{#%UNIQUE_REFERENCE#}
```

6.11 API to third party solutions

Script allows redirecting all form data to another page via GET request. This can be used as API to third party scripts and for further data processing.

Example:

```
http_request = http://example.com/foo.php
```

This will open http://example.com/foo.php with your form data in URL. e.g.: http://

example.com/foo.php? name=Edward&password=abc123

“name=Edward” and “password=abc123” is data received from the form after user entered “Edward” value in the “name” field and “abc123” in the “password” field.

6.12 Back & Edit Feature

This feature allows you to create multi-page forms with back button allowing form applicant return to previous page and edit previously entered values.

As well as SUBMIT, you may add BACK button on your forms. The syntax is very simple. Just add another submit button, but called “fpp_back”.

Example:

```
<input type="submit" name="fpp_back" value="Back">
```

Note:
Please, specify absolute path to Form Processor Pro file in the action parameter of the <form> tag to get "fpp_back" button work correctly.

```
<FORM name="myform" method="post" action="http://www.
mysite.com/fpp/index.php">
```

Or you may use HTML tag <BASE> to specify base URL:

```
<BASE HREF="http://www.mysite.com/fpp/">
```

In this case you can use relative path to Form Processor Pro, as well as path to

	your images, styles, etc. E.g. Your form is located in the root folder on your server and Form Processor Pro is located in the "fpp" folder, so action would be as follows: `<FORM name="myform" method="post" action="fpp/index.php">`

Part VII

7 Language Settings

7.1 Changing Language

You can choose your language by “language” setting.

Example (config file):

```
language = en
```

Result:

Sets English language for all Form Processor Pro messages

7.2 Adding your own language

To add your own language you have to create new language file in fpp/lang.

To do that just copy the previous en.lang.php file and rename to ##.lang.php , where ##- two letter country code, e.g. French- fr.lang.php.

After that you have to translate right part of php file to the language selected.

Syntax:

```
$l ['var_name'] = 'string'
```

Example:

```
$l ['file_permissions'] = 'File Permissions';
```

should be in French:

```
$l ['file_permissions'] = 'File Permis';
```

Note:
You have to leave %s signs and escape special characters (e.g. ' to \' and \ to \\) for the proper work of FPP5.

Example:

Don't=don\'t

‘configure the "MySQL" server‘=‘configure the \"MySQL\" server ‘

Change language value in the config file:

language= fr

Part VIII

8 FAQ

FAQ

8.1 How to restore forgotten password to Initial Configuration Script?

Unfortunately you can't restore your old password in Form Processor Pro because it is not saved in explicit form due to security reasons. Instead you can get a brand new one! To do that you need to follow these steps:

1. Go to the Initial Configuration Script page.
   (e.g.: www.yoursite.com/fpp/install/index.php)

2. Click on “Login” button.

3. Click on “Forgotten Password?” link.
4. Enter e-mail address that you used during installation.
5. Receive an e-mail with your new password.

8.2 I want to allow file download just after form submission. How can I do that?

As well as page you may output any file stored on your server. Use action ‘file' instead of ‘page'. This is very useful in case you want to gather some information prior allowing downloading certain file.

Example of config.php:

```
page = ../index.html

page = ../preview.html

file = /usr/home/secured/area/secretfile.zip
```

In the example above Form Processor Pro will output the file secretfile.zip after preview page.

8.3 How can I allow only certain file extensions for user file uploading?

Set allowed file extensions for uploaded user files in file_filter_fields.

Syntax:

```
allowed_files = ext
```

or

```
allowed_files = ext1|ext2|ext3
```

Example:

In your form:

…

```
<input type="file" name="attach">
```

…

Config file:

```
allowed_files = gif|bmp|png
file_filter_fields = attach (Picture)
```

Result:

Only *.gif, *.bmp, *.png files would be allowed for user upload.

8.4 Does Form Processor Pro have any API? How can I use it?

Script allows redirecting all form data to another page via GET request. This can be used as API to third party scripts and for further data processing.

Example:

```
http_request = http://example.com/foo.php
```

This will open http://example.com/foo.php with your form data in URL. e.g.: http://example.com/foo.php?username=Edward&password=abc123

"username=Edward" and "password=abc123" is data received from the form after user entered "Edward" value in "username" field and "abc123" in the "password" field. This data is being sent to the "foo.php" script by GET request method.

8.5 My second form page/preview has broken images/styles. How can I fix this?

Most likely those relative paths to images/styles are specified in HTML templates.

So, please, specify absolute paths in all links, images etc. on your form pages. Otherwise, all relative paths would be relative to Form Processor Pro, but NOT to the page of your web-site. Or you may use HTML tag <BASE> to specify base URL:

```
<BASE HREF="URL_TO_PAGE">
```

8.6 How to make selection box required to be selected?

The same way as for the other field types, you have to mention the name of this field as required_fields in config.php.

Example:

```
required_fields = department (Department)
```

Also you need to create option with blank value by default.

Example:

```
<select name="department">

     <option value="" selected>Select...</option>

     <option value="sales">Sales Department</option>

     <option value="support">Support Department</
     option>

     <option value="billing">Billing Department</
     option>

</select>
```

8.7 How to make checkbox, choice and dropdown required?

The “required” attribute for these elements is much harder to perform because of their features in html specification. Alike for other fields, you need to specify them in config.

php:

Example:

```
required_fields = interests (Interests)
```

Also, you have to create hidden fields with the same names and blank values:

Example:

```
<input type="hidden" value="" name="interests">

<input type="checkbox" name="interests" value="Finance
News">
```

8.8 How to make group of checkboxes with one line results in templates?

To make a group of checkboxes you need to give them names like **interests[index]** (as it shown in the example), where "index" is field index, starting from “1”

Example:

```
<table>

  <tr>

  <td>

    <label>

       <input type="hidden" value="" name="interests[1]">
```

```
        <input type="checkbox" name="interests[1]" value="
        Finance News" checked>Finance News

      </label>

    </td>

    <td>

      <label>

        <input type="hidden" value="" name="interests[2]">

        <input type="checkbox" name="interests[2]"
        value="Chat" checked>

        Chat

       </label>

    </td>

</tr>

<tr>

    <td>

      <label>

        <input type="hidden" value="" name="interests[3]">

        <input type="checkbox" name="interests[3]"
        value="Weather">Weather

      </label>

    </td>

    <td>

      <label>

        <input type="hidden" value="" name="interests[4]">

        <input type="checkbox" name="interests[4]"
        value="Other" checked>Other

      </label>
```

```
  </td>
</tr>
</table>
```

In future to receive data from this group of checkboxes and to acquire the list of comma-separated values of checked checkboxes we use {#interests#} variable (note that we do not use indexes here).

Result (if we check first, second and forth checkboxes)**:**

Finance News, Chat, Other

8.9 How to allow server-side scripts execution on form pages?

You have full control over any server-side script execution on form pages.

If you use relative or absolute path to file (e.g.: "../form/file.html" or "/fpp/form/filename.php") Form Processor Pro will open it as a text file. No scripts (e.g.: PHP, ASP) will be executed on the page.

If you use full URL (e.g.: "http://www.yoursite.com/path/file.html") Form Processor Pro will let web-server execute all server-side scripts on the page, and then will parse the form.

Note:
PHP option "allow_url_fopen" must be enabled to allow server-side scripts execution.

Part IX

9 Suggestions and Tips

- **Protect email templates**
  Store email templates in separate .htaccess protected directories preventing them from outside browsing.

- **Protecting form from bots and auto-filler scripts**
  In order to protect your form from automated submissions by bots and auto-filler scripts you can use CAPTCHA feature. It will include graphical image with certain symbols that only human can read and type into special verification field.

- **Use absolute paths in HTML templates instead of relative**
  You may store file with the first page of the form separately from other files. You can easily store each HTML template in a different folder. Therefore relative paths to the same files may differ. So, specify absolute paths in all links, images etc. on your form pages. Otherwise, all relative paths would be relative to Form Processor Pro, but NOT to the page of your web-site.

Part X

10 Support Information

- **Have a Pre-Sale question? - Use Live Chat!**

  The quickest possible way to have answer on your presale question is to contact our sales representative via Live Chat. Sales representatives would be glad to assist you in presales questions from 10am to 10pm (GMT+2, EST+7 Time Zone).

- **Help Desk - Ticket System and Knowledgebase.**

  Whether you use Web-Site-Scripts.com products, or are just thinking of ordering any of them, welcome to Help Desk. Here you will receive a quick and informative response to your presales questions, as well as qualified help on technical support issues. A member of our Client Support Center will review, answer and publish the answer to your question to your Help Desk account, and you will be automatically notified by email.

  Registration allows you to submit technical support requests, and review the history of your questions and support responses. In addition, registered users can report bugs, submit suggestions and request product customizations. Registration is short and easy, and takes only 1 minute.