

Table of contents

Installation and configuration	[go] Directory structure [go] File transfer to the server [go] Configuration
Administration panel	[go] Admin menu
Form creation and basic commands	[go] Basic form structure [go] Command fields at a glance [go] Form submission recipients [go] Subject of form submission mails [go] Name and e-mail address of form sender [go] Redirection URL [go] Required fields [go] Sorting of fields [go] Environmental variables report [go] Show command fields [go] Show blank fields
Advanced features in detail	[go] Autoresponders [go] Uploading binary content [go] Save-to-file ability [go] Verification of fields [go] Calculations with fields [go] Auto-numbering and random fields [go] Templates [go] Multi-page forms
Customer services	[go] Need help ? Contact the SunnyScript service department !

Smart tip: This handbook is also designed to be printed... set your printer's page orientation to landscape format, so there is enough space to place hand-written notes on the left and right margin easily.

Installation and configuration

SunnyScript offers an **optional available installation service**, if you don't want to install the software yourself. Please contact our customer service department by e-mail for detailed information and pricing.

Directory structure...

The software can be installed at any place that allows the execution of CGI software: Most server systems provide a special directory called "cgi-bin" where you have to install the application, while others are not limited to a particular location.

In a common environment, we recommend to create a new sub-directory within the existing "cgi-bin" folder to separate the software from other already installed products.

Required "template.en" directory:

Inside the directory you desired for installation, please create a new sub-directory called "template.en". It's used to hold the content of "template.en" directory delivered with your software package.

Required "data" directory:

Inside the directory you desired for installation, please create a new sub-directory called "data". It's used to hold the content of "data" directory delivered with your software package.

Adjust path to Perl:

Open all program files (*.pl and *.cgi) in a text editor and change the first line "#!/usr/bin/perl" that it reflects the correct path to Perl 5 (or higher version) on your system. This is required to let your server know where the Perl interpreter is located.

Usage note: On most systems, /usr/bin/perl is already the default path to Perl and so no modifications are required.

File transfer to the server...

Upload all files to the created directories. If a ftp client is used, ensure to set the "ASCII transfer mode" for file submissions.

Detailed file table: Scheme: [filename | suggested place to upload (short description); file permission]:

Filename...	Description...	Filename...	Description...
*	cgi-bin (program & data files); 755	template.en/*	template.en (templates); 666
		data/*	data (basic data files); 666

Important notes: Please double-check file permissions and transfer mode, otherwise the software may not work as intended. Depending upon the server configuration, different file permissions than the ones stated above may be necessary.

Smart tip: Our installation helper `install.cgi` makes it easier... just call it from your web browsing software (requires correct installation of this script) and it automatically checks your entire installation, adjusts permissions and fixes typical errors.

Configuration...

After you have uploaded all files, you may wish to start the admin panel first time to initialize the system parameters. In order to do so, please call `admin.cgi` with your favorite web browsing software:

Action to perform...	URL to call (location varies on your system, of course)...
Open admin panel	<code>http://www.sunnyscript.com/cgi-bin/pf/admin.cgi</code>

Before you proceed, it is highly recommended to set an administration password. After this is done, access to the admin panel is granted only by authenticating yourself with the chosen password.

Now click on the button "Edit system parameters" and configure the shown parameters according your needs.

Usage note: Since all parameters are automatically filled with correct values, modifications on your side may not be required.

Directories and files:

Name of parameter to set...	Description...
CGI files directory <u>Example:</u> /usr/www/cgi-bin/pf	Absolute path to the location of PowerForm on your server.
"data" directory <u>Example:</u> /usr/www/cgi-bin/pf/data	Absolute path to the location that holds databases and related files.

Platform specific settings:

Name of parameter to set...	Description...
Mailing program (for Unix only) <u>Example:</u> /usr/bin/sendmail	Absolute path to Sendmail (or compatible) application on your server.
SMTP server	SMTP server for handling mails; when left blank, default mail application

<u>Example:</u> smtp.sunnyscript.com	is used. This setting is required for non-Unix systems, otherwise optional.
Don't use flock <u>Example:</u> Unchecked	"flock" is normally supported directly by the operating system; should it be unavailable for any reason (e.g. on Win9x, WinMe), mark this checkbox.

URLs:

Name of parameter to set...	Description...
URL of CGI files directory <u>Example:</u> http://www.sunnyscript.com/cgi-bin/pf	URL to the location of PowerForm on your server.

Autoresponder settings:

Name of parameter to set...	Description...
"From" e-mail address <u>Example:</u> info@sunnyscript.com	E-mail address used in the headers of outgoing autoresponder messages.
Default e-mail subject <u>Example:</u> We received your form submission	Subject used in the headers of outgoing autoresponder messages.

Security settings:

These settings allow you to restrict the access of PowerForm to a given list of domains and recipient e-mail addresses in order to prevent abuse of the software. By default, usage is not restricted.

Name of parameter to set...	Description...
Allowed referring domains <u>Example 1:</u> www.starenterprise.com <u>Example 2:</u> powerform.sunnyscript.com <u>Example 3:</u> https://starenterprise.com/sub1	List of domain names (one per line) allowed to use PowerForm (forms must exist there). Requires "Allow any referring domain" to be disabled. <u>Format:</u> <Protocol><Sub-domain.><Domain>/<String> <Protocol>: "http://", "https://" or skipped to accept both. <Sub-domain.>: Sub-domain followed by a full stop; optional. <Domain>: Main domain name; required. <String>: Any additional path information; optional. <i>Technical note:</i> When trying to access PowerForm from a domain not listed, an error message appears and form submission is not processed.
Allow any referring domain <u>Example:</u> Checked	If enabled, usage of PowerForm is not restricted to known domains.
Allowed recipient e-mail addresses <u>Example:</u> info@sunnyscript.com	List of e-mail addresses (one per line) allowed as recipients for form submissions. Requires "Allow all recipients" to be disabled. <i>Technical note:</i> When trying to use recipients not listed, an error message appears and form submission is not processed.
Allow all recipients <u>Example:</u> Checked	If enabled, any e-mail address can be used as form recipient.
Default recipient e-mail addresses <u>Example:</u> info@sunnyscript.com	List of e-mail addresses (separated by comma) applicable as recipients for form submissions activated by a special keyword. Please refer to the next chapter, part "Form submission recipients" for more information and applications.

After you have verified all settings carefully, click on the button "Save parameters" to take over modifications to the system. It is also possible to bring back previous values with the "Let the script restore defaults" button.

Congratulations - you have successfully installed the software :-)

Administration panel

The administration panel is a convenient way to work interactively with PowerForm. You can start the admin panel by calling `admin.cgi` with your favorite web browsing software:

Action to perform...	URL to call (location varies on your system, of course)...
Open admin panel	<code>http://www.sunnyscript.com/cgi-bin/pf/admin.cgi</code>

Admin menu...

This screen allows the following actions:

Main menu options...	Description...
Edit system parameters	Click on this button to modify system related parameters.
Change password	Change admin password. <i>Technical note:</i> The password protects the admin panel and doesn't restrict access to PowerForm. For this task, check out "Security settings" within "System parameters". <i>Important note:</i> If you forgot the admin password, please remove <code>admin.pass</code> file located in "auth" directory (see software directory of PowerForm). Then you can set a new password next time accessing the admin panel.
Edit templates	Click on this button to edit any kind of templates (e.g. HTML output template, custom mail templates, autoresponder files, multi-page form parts and others).

Usage note: The top menu line provides links to all menu entries regardless on which page you are currently. This will help you to navigate more convenient through the software.

Form creation and basic commands

This chapter contains all information you need to write your own PowerForm powered HTML forms.

Important note: In order to get faster results, it is helpful to be familiar with some basics of HTML programming.

Basic form structure...

An HTML based form may look like this simple example:

1	<code><form action="http://www.sunnyscript.com/cgi-bin/pf/power.cgi" method="POST"></code>
2	<code><input type="hidden" name="recipient" value="info@sunnyscript.com"></code>
3	<code><P>Please enter your name here:</P></code>
4	<code><input name="YourName" size=30 maxlength=100></code>
5	<code><P>Please enter your phone number here:</P></code>
6	<code><input name="YourPhoneNumber" size=30 maxlength=100></code>
7	<code><input type="SUBMIT" value="Send this form now"></code>
8	<code></form></code>

Let's explain it line by line:

Line #...	Description...
1	Initializing form action tag: Starts the form and instructs the web browsing software to send all form content to PowerForm. Has to be a correct URL to access PowerForm (use "https://..." to enable SSL encryption).

	<i>Usage note:</i> The locations allowed to call PowerForm may be limited by "Security settings" (see admin panel).
2	<p>Recipient field: Mandatory for PowerForm; contains e-mail address(es) which receives submitted form content.</p> <p><i>Usage note:</i> Accepted e-mail addresses may be limited by "Security settings" (see admin panel).</p> <p>Generally also other command fields may be located here (e.g. autoresponder, verification rules). They all give instructions to PowerForm how to handle the form content.</p> <p><i>Smart tip:</i> Although PowerForm's command fields (refer to the following part) can be located everywhere in the form, it is suggested to group them here to achieve a clear structure.</p>
3-6	Form body: The form itself; PowerForm supports all form elements.
7	Submit button: Mandatory for each form; if clicked, current form content is sent to PowerForm for processing.
8	End of form tag: Closes the form; required to match HTML language rules.

Technical note 1: Beside of the POST method, you can also use GET to send form content to PowerForm (see line 2).

Technical note 2: Before using SSL encryption, ensure that your server has a certificate installed and correctly configured.

Command fields at a glance...

A "command field" is a (mostly hidden) form field which controls the behavior of PowerForm and enables advanced features like autoresponders or field verifications.

Command field (sorted by relevance)...	Description...
recipient	Recipient(s) of submitted form content; mandatory field.
subject	Subject of the form submission mail (used instead a pre-defined subject).
email	E-mail address of the form sender; mandatory field for autoresponders.
redirect	URL to be called in case of successful form submission.
required	Fields required to be filled in, before form submission is processed.
missing_fields_redirect	URL to be called in case of incomplete form submission (see required).
sort	Sorting of fields according their labels (alphabetically or in defined order).
realname	"Real name" of the user; added to the header of form submission mail.
env_report	Environmental variables delivered by the server.
print_config	Show command fields in form submission mail.
print_blank_fields	Show blank fields in form submission mail.
Uploading binary content *	
Accept uploaded files for form submissions.	
upload_{field}	Upload binary content to a form.
Field verification *	
Verification of submitted field contents.	
{field}_check	Verify field contents before form submission is processed.
failed_verifications_redirect	URL to be called in case of failed field verification.
Field calculation *	
Calculations done basing upon submitted field contents.	
calculation_{field}	Process calculations with fields' values.
Save-to-file ability *	
Store submitted form content to a file on the server.	
storage_filename	Name of the database file which contains the form submission data.
store_fields	List of fields stored in this file.
max_storage	Max. size of file holding form submissions before warning msg. is sent out.

max_storage_warning	E-mail address to which the warning message is sent.
Templates control * Determine mail and HTML output templates.	
mail_template	Template (admin.txt by default) used when sending form content to the recipient(s). Files must be located in "template.en" directory.
templatefile	Template (result.htm by default) used to display system msg. (like form submission results). File must be located in "template.en" directory. <i>Usage note:</i> This setting is ignored, if a redirection command is active.
errortemplatefile	Template (result.htm by default) used to display error msg. (like failed form submissions). File must be located in "template.en" directory. <i>Usage note:</i> This setting is ignored, if a redirection command is active.
Autoresponder * Send autoresponding messages to users.	
autoresp	Text file storing the message sent as autoresponder (must be located in "template.en" directory).
autoresp_encoding	Special content type declaration setting of autoresponding messages.
autoresp_subject	Subject of outgoing messages (used instead of a pre-defined subject).
Auto-numbering * ** Create auto-increased or random numbers as field values.	
auto_number:{start}-{end}	Creates a random number as value for a specified field.
auto_number:autoinc({file.count})	Creates an auto-increased number as value for a specified field.

* These command fields are described in the next chapter more detailed.

** Auto numbering feature requires its commands placed in "value" attribute of a form field (not as "name" attribute).

Form submission recipients (recipient)...

Recipient(s) of the form submission mails; comma-separate multiple entries.

Example:

```
<input type=hidden name="recipient" value="info@sunnyscript.com,max@maxjump.com">
```

Usage note: This field is mandatory. It may be limited to known recipients, see "Security settings" in admin panel.

Hidden form recipients:

It could be a security issue to provide form recipients in clear text. The reason is that these addresses may be spidered (by automatic robots) and then used for receiving UCE (unsolicited commercial e-mail).

This is especially a problem when you've whitelisted the recipient e-mail addresses in your mail filtering software (e.g. to ensure that form submissions pass an existing UCE filter).

PowerForm allows to determine pre-defined form recipients at admin panel (see "Security settings"), which can be activated by the keyword \$default_email provided as value of the recipient command field.

Example:

```
<input type=hidden name="recipient" value="$default_email">
```

It is possible to combine \$default_email and visible e-mail addresses. In this case, use commas to separate each entry.

Example:

```
<input type=hidden name="recipient" value="$default_email,info@sunnyscript.com">
```

Important note: In order to keep hidden form recipients really invisible to UCE senders, it is essential that their e-mail addresses are never used anywhere else (even for sending or receiving e-mails).

Technical note: If "Allowed recipient e-mail addresses" is used to limit form recipients, then ensure that all e-mail addresses listed for \$default_email are also included in this list (otherwise sending a form is not possible).

Subject of form submission mails (**subject**)...

This field allows you to specify a custom subject for the form submission mails; alternatively a pre-defined one applies. It is also possible to use field content within the subject (see example).

Examples:

```
<input type=hidden name="subject" value="My first form submission result">
<input type=hidden name="subject" value="Incoming order from <!--$country-->">
```

Important note: Although possible, it is not recommended to include special characters (ä, ö, ü, &, ...).

Name and e-mail address of form sender (**email, realname; visible**)...

Both fields should be made visible and - except of `email`, which is required for autoresponders, the only purpose is to appear in the header of form submission mails.

Examples:

```
<input type=text name="email">
<input type=text name="realname" maxlength="100">
```

Smart tip 1: It is recommended to use the `email` field, so you have a "reply" address in form submission mails.

Smart tip 2: You should use the validation feature of PowerForm, if these fields are important for your application.

Redirection URL (**redirect**)...

Each time a form was submitted successfully, PowerForm shows a standard page to the user (HTML output template). By including this command field, you can specify a given URL to be displayed instead of the standard page.

Example:

```
<input type=hidden name="redirect" value="http://www.sunnyscript.com/thanks.htm">
```

Usage note: Suited especially for "thank you" / confirmation pages within order forms and surveys.

Required fields (**required, missing_fields_redirect**)...

Determine required fields:

Some fields of a form may be of high importance (e.g. postal address, e-mail). You can force users to fill in selected fields before form submission is processed by marking these fields as required ones:

Example:

```
<input type=hidden name="required" value="field1,field2,field3,...">
```

Usage note: Field names are handled case sensitive.

Smart tip 1: Use the verification feature to check required fields for correct content.

Smart tip 2: All fields involved in calculations should be marked as required to prevent incorrect calculation results.

Required field not filled in - what happens ?

Standard page (default)...

If a required field is not filled in, PowerForm shows a standard page to the user and asks to go back and fill out missing fields.

Missing fields redirection...

Alternatively you can specify a particular URL to be displayed instead of the standard page.

Example:

```
<input type=hidden name="missing_fields_redirect" value="http://sunnyscript.com/error.htm">
```

Technical note: The page used for redirection should state clear that there are some missing form fields to fill in and should also contain a link back to the form (e.g. by using a `javascript:history.back()` tag). A better solution may be to customize the "standard page" (see part "Templates" of chapter "Advanced features in detail").

Sorting of fields (sort)...

All form fields are normally shown as processed in form submission mails. However this command field allows to change the order of appearing fields, either alphabetically or according a defined list.

Examples:

```
<input type=hidden name="sort" value="alphabetic">  
<input type=hidden name="sort" value="order:field1,field2,field3">
```

Technical note: The part "order:" is mandatory within this command field to sort according a defined list.

Usage notes: Fields not included in the sorting list are not being sent with the form submission mails. Binary content fields are sorted by their concerned filenames, not field names (please refer to the following chapter for details).

Smart tip: Choose alphabetical sorting and start each field name with a number (e.g. 01name, 02address, ...). In this way, you don't need to have a long sorting list for a complex form, but still can determine the fields' order as desired.

Environmental variables report (env_report)...

In order to get some technical info about the users filling your forms, PowerForm can retrieve environmental variables.

Example:

```
<input type=hidden name="env_report" value="REMOTE_HOST,HTTP_USER_AGENT">
```

Environmental variable	Description
REMOTE_HOST	Hostname of the remote computer.
REMOTE_ADDR	IP address of the remote computer.
REMOTE_USER	Username (but this is commonly not available).
HTTP_USER_AGENT	Identification string of the browser software.
SERVER_SOFTWARE	Software used on the remote computer.
SERVER_PORT	Port number to which the request was sent.
SERVER_PROTOCOL	Name of protocol used for the request.
REQUEST_METHOD	method used for the request.
GATEWAY_INTERFACE	Gateway interface of the remote computer.
There may be also other environmental variables available, but the ones listed here are the most common.	

Technical note: If a requested environmental variable is not available, it will be ignored.

Smart tip: Let PowerForm include REMOTE_HOST and REMOTE_ADDR to retrieve details about the user's location (useful to prevent fraudulent actions, especially when using PowerForm to process orders).

Show command fields (print_config)...

Normally command fields are not included in form submission mails. However you can determine to show selected command fields (e.g. for referencing purposes) by using this feature.

Example:

```
<input type=hidden name="print_config" value="email,subject">
```

Usage note: Field names are handled case sensitive.

Show blank fields (`print_blank_fields`)...

Empty fields are not shown in form submission mails by default. When setting this command field to value "1", also blank fields will be included in form submission mails.

Example:

```
<input type=hidden name="print_blank_fields" value="1">
```

Advanced features in detail

This chapter contains all details about advanced features of PowerForm. Before you proceed, you should have been read the previous chapters (some matters are described there, we have considered as being known).

Autoresponders...

PowerForm provides personalized autoresponders which allow to answer automatically to incoming form submissions by using customizable messages. You can create an unlimited number of autoresponders for all forms.

Creation of autoresponder files:

First of all, you must create a text message used for the autoresponder. This can be a short note only or a message including form fields which are replaced by the appropriate form content (as submitted).

Save the autoresponder message as ASCII file and upload it to the "template.en" directory of PowerForm.

Example of a simple autoresponder message:

```
Dear Sir or Madam,  
  
thank you very much for your order. Our sales representative will contact you soon.  
  
With kind regards,  
Order processing department.
```

Example of an autoresponder message with personalized content:

```
Dear <!--$Firstname--> <!--$Lastname-->,  
  
thank you very much for your order. It will be processed now.  
  
You have selected the following products:  
  
<!--$Items-->  
-----  
Total amount of this order:           <!--$Total-->  
  
Your desired shipping method is <!--$Ship_method-->.  
  
With kind regards,  
Order processing department.
```

Technical notes: If a field is not available, it will be simply ignored. Field names are handled case sensitive. Binary content fields cannot be submitted in autoresponding messages, while calculated ones are allowed. HTML mails are possible.

Pre-requisite for autoresponders:

In order to allow PowerForm sending autoresponding messages, the form must contain the email command field to ask for a valid e-mail address of the user filling the form, like `<input type=text name="email">`.

How to enable the autoresponder feature:

Structure of the autoresponder command:

```
<input type=hidden name="autoresp" value="{filename}">
```

{filename}: File holding the autoresponder message (must be located in "template.en" directory).

Example:

```
<input type=hidden name="autoresp" value="myanswer.txt">
```

Custom subject of autoresponder messages:

You can specify a custom subject for the autoresponder messages; alternatively a pre-defined one applies. It is also possible to use field content within the subject (see example).

Example:

```
<input type=hidden name="autoresp_subject" value="Hello <!--$name-->, we got your form !">
```

Important note: Although possible, it is not recommended to include special characters (ä, ö, ü, &, ...).

Special message content type declaration:

In some cases, autoresponding messages should be sent in HTML format or a special language encoding, which requires the target mail program to receive a particular content type declaration.

Structure of the encoding command:

```
<input type=hidden name="autoresp_encoding" value="{declaration}">
```

{declaration}: Value html for HTML messages, value text for text messages or a complete content type declaration.

Examples:

```
<input type=hidden name="autoresp_encoding" value="html">
```

```
<input type=hidden name="autoresp_encoding" value="text/html; charset=ISO-8859-5">
```

Usage note: If a special encoding command is missing, the declaration type "text" automatically applies.

Uploading binary content...

PowerForm can handle also file uploads, so people are able to send binary contents like images, programs and data files through a form. Submitted files are attached to the form submission mails (MIME compliant format) and/or stored on the server.

Structure of the file uploading command:

```
<input type=file name="upload_{field}">
```

{field}: Name extension, intended to give each uploaded file a unique name.

Example:

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

Form submission mail content:

The submitted files are attached to form submission mails sent to the recipient(s) by using this scheme:

```
upload_{field}: {path/filename}, {size} bytes, {stored / not stored}
```

upload_{field}: Field name as specified during form creation.

{path/filename}: Internal path and name of the uploaded file.

{size}: Size of the transferred file in bytes.

{stored / not stored}: Flag indicating whether the file is stored on the server or just sent by e-mail.

Pre-requisite to enable file uploads:

In order to allow file uploading, the form action tag must show an enctype attribute for a multipart form.

Example:

```
<form action="cgi-bin/pf/power.cgi" method="post" enctype="multipart/form-data">
```

Technical note: Although not restricted, uploaded files should not be larger than 5 MB due to limitations of some web servers and web browsing software (contact your webhosting provider for details).

File uploads and save-to-file feature:

It is possible to use this feature together with a save-to-file command (see next part). All submitted files are placed in different sub-directories of the "data/uploads" directory and named with their original filenames (as submitted by the user).

However it's recommended to receive uploaded files only via form submission mails, to prevent the risk of storage capacity problems when someone submits many large files without your consent.

Save-to-file ability...

Beside of receiving form submissions by e-mail, PowerForm can also save them to a database file on your server. In order to enable file storage, add this PowerForm command field to the concerned form:

```
<input type=hidden name="storage_filename" value="{filename}">
```

{filename}: File containing the saved form submissions (will be created within the "data" directory automatically).

Example:

```
<input type=hidden name="storage_filename" value="orders.txt">
```

Usage note: For security reasons, having at least one recipient receiving the form submission mails is mandatory.

Smart tip 1: Different forms can write to the same storage file, so you are able to collect submissions from various sources.

Smart tip 2: The save-to-file feature is suited for importing submissions to a database system like [EasyData](#) from Sunnyscript. PowerForm uses a tab delimited storage in ASCII format with a newline character at the end of each record.

Optional... Save only selected fields:

In some cases, you may wish to save only selected fields to the storage file, here is the command field for this application:

```
<input type=hidden name="store_fields" value="{list of fields}">
```

{list of fields}: Comma-separated list of form fields to save (field names are handled case sensitive).

Optional: Enable file size warning:

A warning message can be sent by e-mail when the storage file exceeds a maximum limit. The following two command fields are required to enable the file size warning feature:

```
<input type=hidden name="max_storage" value="{size}">
<input type=hidden name="max_storage_warning" value="{e-mail address}">
```

{size}: Max. allowed size in KB, before the warning message is sent.

{e-mail address}: E-mail address which receives the warning message.

Example:

```
<input type=hidden name="max_storage" value="2048">
<input type=hidden name="max_storage_warning" value="tobias@sunnyscript.com">
```

In this example, a warning message is sent to tobias@sunnyscript.com as soon as the file exceeds 2048 KB (2 MB).

Technical notes: The warning message is hardcoded and cannot be altered; 1 MB is equal to 1024 KB.

Important notes: File size warning covers only the data submitted to the database file, not any binary uploads. New incoming form submissions are still added, even the maximum size is exceeded (to prevent data loss).

Verification of fields...

Form content can be verified before further processing. If a verification procedure fails, PowerForm shows a standard page to the user telling about the occurred error (see below for available redirection command).

Structure of the verification command:

```
<input type=hidden name="{fieldname}_check" value="{conditions}">
```

{fieldname}: Existing field to check against the required conditions.

{conditions}: Verification rules (multiple rules separated by semicolon).

Possible verification conditions:

Condition...	Description...
int	Positive and negative integers.
float	All numbers ("+", "-", "." and digits).
alpha	Characters only (A-Z, a-z).
alphanum	Characters and digits only (A-Z, a-z, 0-9).
positive	Positive numbers only (e.g. number of items).
negative	Negative numbers only.
range:x,y	All numbers from x to y (e.g. age, # of items).
values:val1,val2, ...	Only given values are allowed (e.g. true / false fields, multiple choice answers).
min_size:x	Minimum length of required input (e.g. postal code, telephone & fax numbers).
max_size:x	Maximum length of allowed input (e.g. postal code, telephone & fax numbers).
valid_email	E-mail address checking for correct format.
valid_URL	URL checking for correct format.
accepted	Checkbox is required (checkbox must be marked, e.g. to accept terms of business).
format:x *	Value x must match a Perl regular expression.

* Please refer to available documentation for more information about Perl regular expressions.

Some application examples:

Check the form field `item` to contain a positive integer between 1 and 10.

```
<input type=hidden name="item_check" value="positive;range:1,10">
```

Check the form field `website` to contain a valid URL.

```
<input type=hidden name="website_check" value="valid_URL">
```

Check the form field `gender` to contain either values "female" or "male".

```
<input type=hidden name="gender_check" value="values:female,male">
```

Technical note: Please take care that verification rules are used in a logical way (e.g. "negative;alpha" would not work).

Smart tip: Field verification has priority before field calculation, so you can check fields intended for calculations first.

Redirection in case of failed verification:

By default, a standard page is displayed telling about a failed verification. Instead of, you can also specify a particular URL.

Example:

```
<input type=hidden name="failed_verifications_redirect" value="http://maxjump.com/bad.htm">
```

Technical note: The page used for redirection should state clear that some verifications failed. A better solution may be to customize the "standard page" (see part "Templates" of this chapter).

Calculations with fields...

You can process calculations based upon the form content. Each calculation creates a new form field containing its result. It is possible to perform arithmetic calculations, string manipulations and even execute complex Perl scripts.

Any kind of operator can be used, which is supported by Perl (e.g. if, else, elsif, while, foreach, next and others) as well as all mathematical operators (e.g. +, -, *, / and others).

Important note: Before using Perl scripts for calculations, you should be familiar with the basics of this programming language.

Structure of the calculation command:

```
<input type=hidden name="calculation_{fieldname}" value="{calculation string}">
```

{fieldname}: New field created and intended to contain the result of calculation.

{calculation string}: Calculation to perform.

Smart tip: Calculations are done after verifications, so you can verify fields first for suited contents.

Example of simple arithmetic:

```
<input type=hidden name="calculation_result" value="([amount1]+[amount2])/[age]">
```

A new field `result` is created, containing the result of `[amount1]+[amount2]/[age]`.

Usage note: Fields involved in the calculation string must be enclosed by square brackets, e.g. `[field1]`.

Example of complex calculations (if / then / else):

```
<input type=hidden name="calculation_subtotal" value="
  if ('[article]' eq 'shirt')
    {$price = 20}
  elsif ('[article]' eq '')
    {$price=[subtotal1]+[subtotal2]};
  $price.' [currency]';
">
```

Expert note: Only single quotes are allowed inside the calculations (to avoid conflicts with HTML language specifications).

Expert tip: You can use the calculation feature to build your own tiny shopping solution (e.g. determine subtotals, total amount and taxes; in an enhanced way also shipping fees, required delivery time and more).

Auto-numbering and random fields...

Auto-numbered fields:

This command field lets PowerForm create an auto-increased number as value for a specified field.

Possible applications are to provide an individual ID to each user submitted a form (e.g. for later referencing purposes or as order identification) or to build "intelligent forms" with verification and calculation parts requiring an auto-increasing value.

Structure of an auto-numbered field...

```
<input type=hidden name="{fieldname}" value="<auto-number:autoinc({filename})>">
```

{fieldname}: Name of the field containing the auto-increased number.

{filename}: File holding the counter for this field; if not existing, it will be automatically created in "data" directory.

Usage notes: In order to start an auto-numbered field from 0 again, just go to the "data" directory and remove the concerned counter. You can use an infinite number of fields basing on different counter files and even fields accessing the same

counter.

Examples:

```
<input type=hidden name="Field1" value="<auto-number:autoinc(users.count)>">
<input type=hidden name="Field2" value="anytext<auto-number:autoinc(test.cnt)>">
```

Technical notes: Auto-numbered fields are created before field verifications and calculations, so these fields can be used for calculations. Additional text parts surrounding the generated value are also allowed (see second sample).

Random numbered fields:

This command field lets PowerForm create a random number as value for a specified field.

Possible applications are to provide an individual ID to each user submitted a form (e.g. for later referencing purposes or as order identification) or to build "intelligent forms" with verification and calculation parts requiring a random value.

Structure of a random numbered field...

```
<input type=hidden name="{fieldname}" value="<auto-number:{start}-{end}>">
```

{fieldname}: Name of the field containing the auto-increased number.

{start}: First possible random value.

{end}: Last possible random value.

Technical notes: Leaving the start and end range, a random number between 0 and 1 is generated. Depending upon the internal server configuration, the entire string length may be anything between 3 and 17 digits.

Examples:

```
<input type=hidden name="Field1" value="<auto-number:10-100>">
<input type=hidden name="Field2" value="<auto-number:>">
<input type=hidden name="Field3" value="<auto-number:0-9999>anytext">
```

Technical notes: Randomly numbered fields are created before field verifications and calculations, so these fields can be used for calculations. Additional text parts surrounding the generated value are also allowed (see third sample).

Combination of auto-numbering and random numbers:

It is also possible to combine both features, e.g. to get a random value with an unique auto-increased component.

Example:

```
<input type=hidden name="OrderID" value="<auto-number:autoinc(order.count)>-abc0<auto-
number:1000-99999>">
```

Creates a field OrderID with an auto-increased value, followed by a fix text part and a random number, e.g. "8-abc03790".

Templates...

Form submission mail template:

Whenever PowerForm sends submission mails to form recipients, the template `admin.txt` located within the "template.en" directory is used to generate the message structure.

In order to achieve a custom layout, you can either modify this file directly (don't forget to create a backup copy first) or instruct PowerForm to use a specific submission mail template of your choice:

```
<input type=hidden name="mail_template" value="{filename}">
```

{filename}: Mail template file, which must be located at the "template.en" directory.

System tags used in this template:

Special tag...	Description...
<!--\$FIELDS-->	Body of form submission mails; shows all submitted form content except uploaded

	files. Alternatively usage of specific field names is possible (see sample below).
<!--\$ATTACHMENTS-->	Attachments (binary files) uploaded through forms.
<!--\$ENV-->	Environmental variables (command field env_report).

Sample form submission mail template:

```
Dear PowerForm administrator,

the following form was submitted to you by <!--$Firstname--> <!--$Lastname-->:

Name:           <!--$Firstname--> <!--$Lastname-->
E-mail address: <!--$email-->
Telephone number: <!--$PhoneNumber-->
Telefax number: <!--$FaxNumber-->

Message:        <!--$Message-->
Attachments:    <!--$ATTACHMENTS-->
```

Usage note: In this way, you can also only pass through selected fields (rest may be stored in a file).

HTML output template (the so called "standard page"):

Whenever PowerForm finished a form submission successfully or experienced an error (e.g. failed verification, missing required field), the template `result.htm` located within the "template.en" directory is used to generate the page structure.

In order to achieve a custom layout, you can either modify this file directly (don't forget to create a backup copy first) or instruct PowerForm to use a specific HTML output template of your choice:

```
<input type=hidden name="templatefile" value="{filename}">
```

{filename}: HTML output template file, which must be located at the "template.en" directory.

Technical note: The HTML output template will be ignored, if a redirection command field takes effect.

System tags used in this template:

Special tag...	Description...
<!--\$HEADER-->	Main header, short system message.
<!--\$ERROROUTPUT-->	Error messages (e.g. failed verification).
<!--\$HTMLOUTPUT-->	Form submission data and/or other system output.

Displaying fields in this template:

Furthermore it is possible to use any field from your form also on the HTML output template (format: <!--\$field-->). Hereby field values are passed through in HTML-safe format (e.g. "<" character is replaced by "<" to avoid destroying the layout).

Technical notes: When using fields having the same names as above stated special tags, these tags are processed instead of (they have a higher priority compared to common fields). Non-existing fields or empty field values are ignored.

HTML error output template:

Whenever PowerForm is unable to process a form submission, it either redirects to a set URL or displays error messages by using the standard page (default `result.htm`).

But it is also possible to specify a separate template for this purpose by adding the following system field:

```
<input type=hidden name="errortemplatefile" value="{filename}">
```

{filename}: HTML error output template file, which must be located at the "template.en" directory.

Technical note: The HTML error output template will be ignored, if a redirection command field takes effect.

Displaying system tags and fields in this template:

As for `templatefile`, you may use system tags and field values also here.

Maintaining templates:

You may alter existing templates or create new ones with a text or HTML editor of your choice (depending upon the template).

However a more convenient way is to use the integrated templates editor accessible via admin panel. There you also find further information about existing templates delivered with the software package.

For each template you create, an informational header can be added containing internal comments and notes not appearing at the template itself. Furthermore it is also possible to copy&paste source code from an outside editor.

Multi-page forms...

Another powerful feature is the ability to split long forms (e.g. job applications) or complex processes (e.g. shopping checkouts) into several smaller ones by spreading one form over multiple pages. This way is called "multi-page" forms.

Guidelines for creating multi-page forms:

First page of the form:

1. Just use a common initiating form action tag to start.

Example:

```
<form action="cgi-bin/pf/power.cgi" method="post">
```

2. Add this hidden field to indicate starting a multi-page form:

```
<INPUT type="hidden" name="multipage" value="1">
```

3. Add this hidden field to let PowerForm know the next page of the multi-page form:

```
<INPUT type="hidden" name="templatefile" value="{nextpage}">
```

{nextpage}: Filename of the second form part (see following notes).

Technical note: Please note that `templatefile` has two different meanings here; specifying a custom HTML output template (common intention of this command field) is possible at the last form part (refer below for details).

Further pages of the form:

1. Use the same initiating form action tag as done for the first page.

Example:

```
<form action="cgi-bin/pf/power.cgi" method="post">
```

2. Add this hidden field to indicate that the next page is also part of the multi-page form:

```
<INPUT type="hidden" name="multipage" value="1">
```

3. Add this special tag to let PowerForm hand over all fields from previous pages:

```
<!--$PREVIOUSFORM-->
```

4. Add this hidden field to let PowerForm know the next page in range:

```
<INPUT type="hidden" name="templatefile" value="{nextpage}">
```

{nextpage}: Filename of the next form part.

Last page of the form:

Equal as before, but now set the command field "multipage"s value either to zero or skip this command field entirely.

Examples:

```
<INPUT type="hidden" name="multipage" value="0">
<INPUT type="hidden" name="multipage" value="">
```

Technical note: You can now use the command field "templatefile" to specify a custom HTML output template.

Uploading and testing multi-page forms:

After creating the form pages, please put all files (except the first one, which can reside at its original location) into the "template.en" directory. For security reasons, all follow-up form parts are taken from this directory only.

Helpful hint: You may use the templates editor for creating these pages or just copy&paste forms created with an HTML editor. In this way, you do not need to log in by ftp for transferring files or doing later modifications.

Now give it a try by calling the first form page. If anything is done correctly, the "Submit" button of the first page leads to the second form part located at "template.en" directory and then to the third one, etc..

Implementation notes:

SSL encryption may work also at multi-page forms depending upon the way your server handles it (SSL access must be server-wide available, not limited to a particular directory).

By using tags like `<!--$field-->`, it's possible to take over field values from previous parts for being displayed at the current part; values are passed through in HTML-safe format (e.g. "<" character is replaced by "<" to avoid destroying the layout).

Fields used to upload binary content (`upload_{field}`) have to be placed at the last form part only.

Customer services

Depending upon your license type, this software package includes various support services. Please refer to the license document or visit our website for more information about available services.

Before you contact our customer service department, please read this manual first. In most cases you find the answer here. However if you still experience problems, we will be more than happy to help you...

For all support inquiries, please contact support@sunnyscript.com and provide the following information:

- * The license number of the concerned product.
- * A close description of the problem or question you have (no file attachments, please).
- * For technical inquiries: Used server environment, URL of admin panel & admin password.

Now we wish you a great time with our software products :-)

© **SunnyScript** - A subsidiary of [Star Enterprise](#). Visit the [SunnyScript](#) website.
Please read our terms of business located at... <http://www.sunnyscript.com/terms.htm>