PVB CONTACT FORM 7 CALCULATOR PRO 1.6.12 – DOCUMENTATION

CONTENTS

1. INSTALLATION

Contact Form 7 version 5.0 or later is required.

1. Download the *pvb-contact-form-7-calculator.zip* file to your computer.
2. Unzip the file.
3. Upload the *pvb-contact-form-7-calculator* directory to your */wp-content/plugins/* directory.

2. SETTINGS *(PRO VERSION ONLY)*

In the Pro version, there is a plugin settings page where you can manage various plugin features.

You can access the settings page from the Wordpress dashboard under *Contact → Calculator Settings*.

Some of the configurable options are:

2.1. ACTION AFTER FORM SUBMIT

By default, Contact Form 7 will reset the form after the site visitor submits it via AJAX. PVB Contact Form 7 Calculator Pro allows you to modify this behavior and keep the form values to visible even after submitting the form. You can enable that on the calculator settings page. This is a global setting which will affect all the CF7 forms on your website.

2.2. DECIMAL AND THOUSANDS SEPARATORS

By default, the calculator uses with a decimal point (.) and does not group thousands.

By varying these two settings, you may choose a different number formatting, such as:

- 1,000,000.00
- 1 000 000,00
- 1.000.000,00

This choice affects both the way input numbers are interpreted, and the way calculation results are displayed.

You can also force-format user input fields with these separators (see Input field masking).

2.3. PHP CODE

From the settings page, you may enable the **fn_php**(…) calculator function. This will let you run custom PHP code in your calculation formulas.

Running PHP code is **disabled by default** since it is a significant security risk. Only enable it if you are 100% sure you know what you are doing!

2.4. SPREADSHEET OPTIONS

These settings let you enable the **fn_spreadsheet**(…) calculator function to retrieve information from an external spreadsheet file that resides either locally on the server, or on the Internet.

For security reasons, this option is **disabled by default**. If you intend to use the spreadsheet calculation functions, you can turn it on.

2.5. GOOGLE SERVICES

To use the Google integration services (retrieving data from Google Sheets and distance/travel time calculation with Google Maps), you need to enter a valid Google API key on the plugin settings page. You can create your free API key from the Google Developer Console.

There are two types of keys you need to store in your plugin settings:

- **Server-side API key:** your web server will use this to retrieve information directly from Google, such as Google Sheets cells. This API key is *not* published on your website.
- **Client-side API key:** your visitors' web browsers will use this to retrieve information from Google, such as Google Maps addresses. This API key is published in your website's HTML code, so it should be restricted for protection.

2.6. STRIPE OPTIONS

To accept payments with Stripe, you need to enter the API keys from your Stripe dashboard into the settings page. You also have to choose whether you want to run the Stripe integration in *Test* mode or in *Live* mode.

3. FORM TAGS

The plugin adds support for the following special tags that you can add to any CF7 form:

3.1. CALCULATED VALUE

A calculated value is a read-only input element where a value is calculated according to a predefined formula.

In the free version, you can have one calculated value per form. In the Pro version, you can have as many as you need.

3.1.1. EXAMPLE

```
[calculation total min:0 max:1000 precision:2 "item_price * quantity"]
```

3.1.2. FORMULA

The calculation formula is a mathematical expression that may contain the following elements:

- **Number literals**;
- **Values taken from other fields** – simply use the field name in the equation. Non-numerical values are evaluated as 0;
- **Parentheses** – for grouping operations, e.g. (a+b)*c
- **Operators:**
  - addition (a + b)
  - subtraction (a - b)
  - multiplication (a * b)
  - division (a / b)
  - exponentiation (a ^ b)
  - modulo (a % b)
- **Functions:**
  - Square root: **sqrt**(x)
  - Logarithm: **log**(number, base)
  - Extract day number from date field: **fn_day**(x)
  - Extract month number from date field: **fn_month**(x)
  - Extract year from date field: **fn_year**(x)
  - Extract day of the year from date field: **fn_day_of_year**(x)

- Extract weekday number from date field: **fn_weekday**(x) – Monday is 1, Sunday is 7
- Extract number of business days between two date fields: **fn_business_days**(x, y)
- Convert between two currencies *(Pro version only)*: **fn_currency_convert**(USD_EUR, x)
- Custom PHP code *(Pro version only)*: **fn_php**(`return tan(deg2rad($_POST{'degrees'}));`) – see below for details
- Spreadsheet cell reference *(Pro version only)*: **fn_spreadsheet**(http://www.example.com/pricelist.xls, 1, B, 10) – see below for details
- Road distance *(Pro version only)*: **fn_distance**(address1_formatted, address2_formatted, km) – see below for details
- Travel time *(Pro version only)*: **fn_travel_time**(address1_formatted, address2_formatted) – see below for details

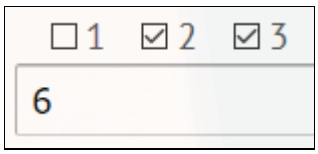
3.1.2.1. NOTE ON CHECKBOX GROUPS

With multiple checkbox groups, it is possible to select more than one value. In that case, if you use a checkbox field name in your formula, all the selections will be automatically summed. For example this form code:

```
[checkbox mycheckbox "1" "2" "3"]

[calculation result "mycheckbox + 1"]
```

could yield a result like this:



3.1.2.2. RUNNING PHP CODE INSIDE THE FORMULA *(PRO VERSION ONLY)*

To run PHP code inside the formula, enclose it within a **fn_php**(…) call.

NOTES ON RUNNING PHP CODE

- Running PHP code is **disabled by default** since it is a significant security risk. You can enable it from the plugin options page, but only if you are 100% sure you know what you are doing!
- Do *not* add any PHP opening/closing tags (*<?php … ?>*).
- Use a **return** statement to set the value that will be used in the calculation formula.
- You can escape double-quotes in your PHP code using the **"** entity.
- You can use the **$_POST** global to access form data.
- Since Contact Form 7 uses square brackets - "[" and "]" - to delimit form tags, square brackets in your PHP code will not be handled properly. If you need to access array elements, **please use curly braces**. In PHP, both square brackets and curly braces can be used interchangeably for accessing array elements (e.g. $array[42] and $array{42} will both do the same thing).

3.1.2.3. REFERENCING AN EXTERNAL SPREADSHEET INSIDE THE FORMULA *(PRO VERSION ONLY)*

You can use the **fn_spreadsheet**(…) function inside your formula to reference a cell in a spreadsheet.

The **fn_spreadsheet** function is **disabled by default**. You can enable it from the plugin options page.

The function takes four parameters:

1. URL or full path to a local file where the spreadsheet document is located;
2. Number of the worksheet in the document (1, 2, 3…)
3. Column of the cell (A, B, C…)
4. Row of the cell (1, 2, 3…)

Supported formats are:

- Google Sheets
- Microsoft Excel 2007+ (XLSX)
- Microsoft Excel 97-2003 (XLS)
- Comma-separated values (CSV)

3.1.2.4. CALCULATING TRAVEL DISTANCE AND TIME *(PRO VERSION ONLY)*

If you have set a Google services API in the plugin options, you can use this function to calculate distance using the Google Maps API.

The **fn_distance**(…) function will take three parameters:

1. Starting point address. This is anything that can be parsed by Google Maps. If you use the input from an address field, you can use the formatted address by appending **_formatted** to the field name (e.g. if the address field name is called "office", use "office_formatted" in this parameter)
2. End point address. Same as above.
3. Distance units. Use "**miles**" for miles, "**km**" for kilometers, or "**m**" for meters (default is meters).

The **fn_travel_time**(…) function works similarly, but returns driving time. Again, it takes three parameters:

1. Starting point address.
2. End point address.
3. Time units. Use "**h**" for miles, or "**m**" for minutes (default is minutes).

3.1.3. OPTIONS

- **Minimum** – by specifying the "*min:X*" attribute, you can set a minimum value for the result of the calculation. For example, if you specify "*min:0*", when the result is negative, it will always be set to zero instead.
- **Maximum** – by specifying the "*max:Y*" attribute, you can set a maximum value for the result of the calculation. For example, if you specify "*max:100*", when the result exceeds 100, it will always be set to 100 instead.
- **Precision** – optionally, you can specify the "precision" attribute, which specifies the maximum number of digits after the decimal point. For example, with "precision:2", 0.786 will be rounded to 0.79. You can specify *"precision:0"* to get only integer results.
- **Roundup –** if the "*roundup*" attribute is present, when rounding to match the "precision" option, the calculator will always round up to the nearest larger integer.
- **Rounddown –** if the "*rounddown*" attribute is present, when rounding to match the "precision" option, the calculator will always round down to the nearest smaller integer.
- **Initial value –** if the "init" attribute is present (e.g. "init:100", the field will be pre-filled with the given number *before* any calculations are performed.

3.2. CALCULATE BUTTON

The calculate button will trigger an AJAX request that will fill in all the calculated values in the form.

3.2.1. EXAMPLES

```
[calculate_button "Click here to calculate"]

[calculate_button my_calculate_button autocalc cf7-hide "No click needed – this is a hidden, automatic Calculate button"]
```

3.2.2. OPTIONS

- **Trigger automatically** ***(Pro version only)*** – by adding the "*autocalc*" option on a calculate button, it will be triggered automatically when any form value changes. The site visitor can then see the calculation result without having to click on the calculate button at all. If you use this option, you may also want to hide the calculate button.

3.3. VARIABLE *(PRO VERSION ONLY)*

The **variable** field is similar to the calculation field, but it maps a list of text values to a list of number values.

The **variable** field takes the input text value from *another form field* (e.g. a drop-down menu) and converts it to a number.

3.3.1. EXAMPLE

```
[select text "One" "Two" "Three" "Four"]
[variable price init:1 "text:" "One 1" "Two 2" "Three 3" "Four 4" "default 0"]
```

3.3.2. OPTIONS AND VALUES

The first value indicates the name of the source field and should end with a colon.

Each subsequent text value is then mapped to a number in the form of "(text) (space) (number)".

You can also specify a **"default"** value to be used if none of the text values match. If a default is not specified, it is assumed to be zero.

If the "**init**" attribute is present (e.g. "init:100", the field will be pre-filled with the given number *before* any calculations are performed.

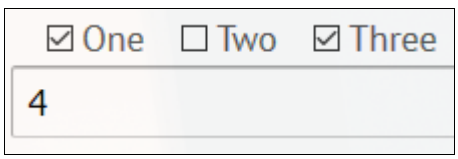
3.3.3. NOTE ON CHECKBOX GROUPS

With multiple checkbox groups, it is possible to select more than one value. In that case, if you use a checkbox field name in your variable, all the selections will be automatically summed. For example this form code:

```
[checkbox mycheckbox "One" "Two" "Three"]

[variable myvariable "mycheckbox:" "One 1" "Two 2" "Three 3"]
```

could yield a result like this:



3.4. SPREADSHEET REFERENCE *(PRO VERSION ONLY)*

This field works similarly to the variable field above, but instead of a hard-coded list, it takes the values from an external spreadsheet.

3.4.1. EXAMPLE

```
[spreadsheet_reference price sheet:1 cells:A1-B10
"http://www.example.com/list.xls"]
```

	A	B
1	One	1
2	Two	2
3	Three	3
4	Four	4
5	Five	5
6	Siz	6
7	Seven	7
8	Eight	8
9	Nine	9
10	Ten	10

3.4.2. OPTIONS AND VALUES

- The "**sheet**" option indicates the number of the sheet in the spreadsheet document (1, 2, 3…)
- The "**cells**" option points to a two-column set of cells that contain the text-to-number mapping (text on the left, numbers on the right – see example above).
- The field value contains the URL or local path to a spreadsheet file (supported formats are: Google Sheets, Microsoft Excel, or comma-separated values).

3.5. SPREADSHEET LIST *(PRO VERSION ONLY)*

The spreadsheet list tag will take a list of options from an external spreadsheet, and display it as either a drop-down menu, a group of checkboxes, or a group of radio buttons.

3.5.1. EXAMPLE

```
[spreadsheet_list product type:dropdown sheet:1 cells:A1-A10
"http://www.example.com/list.xls"]
```

3.5.2. OPTIONS AND VALUES

- The "**type**" option ("*dropdown*", "*checkbox*", or "*radio*") controls how the options should be presented. The default is "dropdown";
- The "**multiple**" option, if present, will allow multiple options to be selected at the same time (not applicable for radio buttons);
- The "**sheet**" option indicates the number of the sheet in the spreadsheet document (1, 2, 3…);
- The "**cells**" option points to a column set of cells that contain the list of options;
- The field value contains the URL or local path to a spreadsheet file (supported formats are: Google Sheets, Microsoft Excel, or comma-separated values).

3.6. ADDRESS INPUT *(PRO VERSION ONLY)*

If you have entered a valid Google Maps API key on the plugin settings page, you can use the **address** tag to generate an address input field with autocomplete features.

This tag will automatically generate three hidden fields alongside the main input field. You can use them in your calculations and email templates.

For example, If your main field name is "**xyz**", the available additional form fields would be as follows:

- **xyz_lat** – location latitude (in decimal degrees)
- **xyz_lng** – location longitude (in decimal degrees)
- **xyz_formatted** – the full formatted address (including street, city, state, zip, etc.)

3.6.1. EXAMPLE

```
[address home]
[address work]
```

3.7. PAYPAL BUTTON *(PRO VERSION ONLY)*

A PayPal button is linked to a calculated value. The button is initially hidden, and when the calculation is ready, if the resulting amount is larger than zero, the PayPal button will be displayed and will allow the site visitor to make a payment for that amount.

3.7.1. EXAMPLE

```
[paypal paypal-btn newtab email:me@privacy.net field:total currency:USD "My Awesome Service"]
```

3.7.2. OPTIONS AND VALUES

- The required "**email**" option indicates your PayPal email. Make sure it's correct – you will receive the payments there!
- The required "**field**" option the name of the calculated field with the amount to pay.
- The optional "**currency**" option indicates the ISO 4217 currency code for the payment. The default is U.S. dollars.
- The optional "**success**" option lets you indicate a page slug (e.g. "*thank-you*") to be displayed after successful payment.
- The optional "**fail**" option lets you indicate a page slug (e.g. "*payment-cancelled*") to be displayed if the payment is cancelled.
- If the "**newtab**" option is present, the PayPal payment site will open in a new browser tab.
- If the "**payonsubmit**" option is present, the user will be automatically sent to PayPal after the contact form is successfully submitted.
- Finally, the field value contains a free-form description of the product or service you are selling.

3.8. STRIPE BUTTON *(PRO VERSION ONLY)*

A Stripe button works very similarly to a PayPal buton (see above). The button is initially hidden, and when the calculation is ready, if the resulting amount is larger than zero, a "Pay with Stripe" button will be displayed and will allow the site visitor to make a payment for that amount.

To use Stripe, you need to enter your Stripe API keys on the plugin settings page.

3.8.1. EXAMPLE

```
[stripe stripe-btn field:total currency:USD "My Awesome Service"]
```

3.8.2. OPTIONS AND VALUES

- The required "**field**" option contains the name of the calculated field with the amount to pay.
- The optional "**emailfield**" option contains the name of a customer email field in your form. If available, that email address will be passed on to Stripe. Otherwise, the Stripe payment pop-up will ask the user to enter an email address.
- The optional "**currency**" option indicates the ISO 4217 currency code for the payment. The default is U.S. dollars.
- The optional "**success**" option lets you indicate a page slug (e.g. "*thank-you*") to be displayed after successful payment.
- The optional "**fail**" option lets you indicate a page slug (e.g. "*payment-cancelled*") to be displayed if the payment is cancelled.
- If the "**payonsubmit**" option is present, the Stripe payment pop-up will be automatically opened after the contact form is successfully submitted.
- Finally, the field value contains a free-form description of the product or service you are selling.

3.9. ORDER ID FIELD *(PRO VERSION ONLY)*

An order ID field is a read-only text field, automatically pre-filled with a short, unique string. If you use PayPal or Stripe, the order ID value is passed along with the item description. In this way, you can match emails from the contact form with payments from your PayPal/Stripe account.

3.9.1. EXAMPLE

```
[order_id my_order_id cf7-hide]
```

4. HIDING FIELDS

You can add the "**cf7-hide**" option to a form field to make it hidden. For example:

```
[calculation total cf7-hide "item_price * quantity"]

[variable price cf7-hide "text:" "One 1" "Two 2" "Three 3" "Four 4"]
```

5. STYLING FIELDS

Each of the three calculation elements supported by this plugin have their own class names, respectively: ***wpcf7-calculation***, ***wpcf7-variable***, and ***wpcf7-calculate_button***. You can apply CSS styles to those classes, for example:

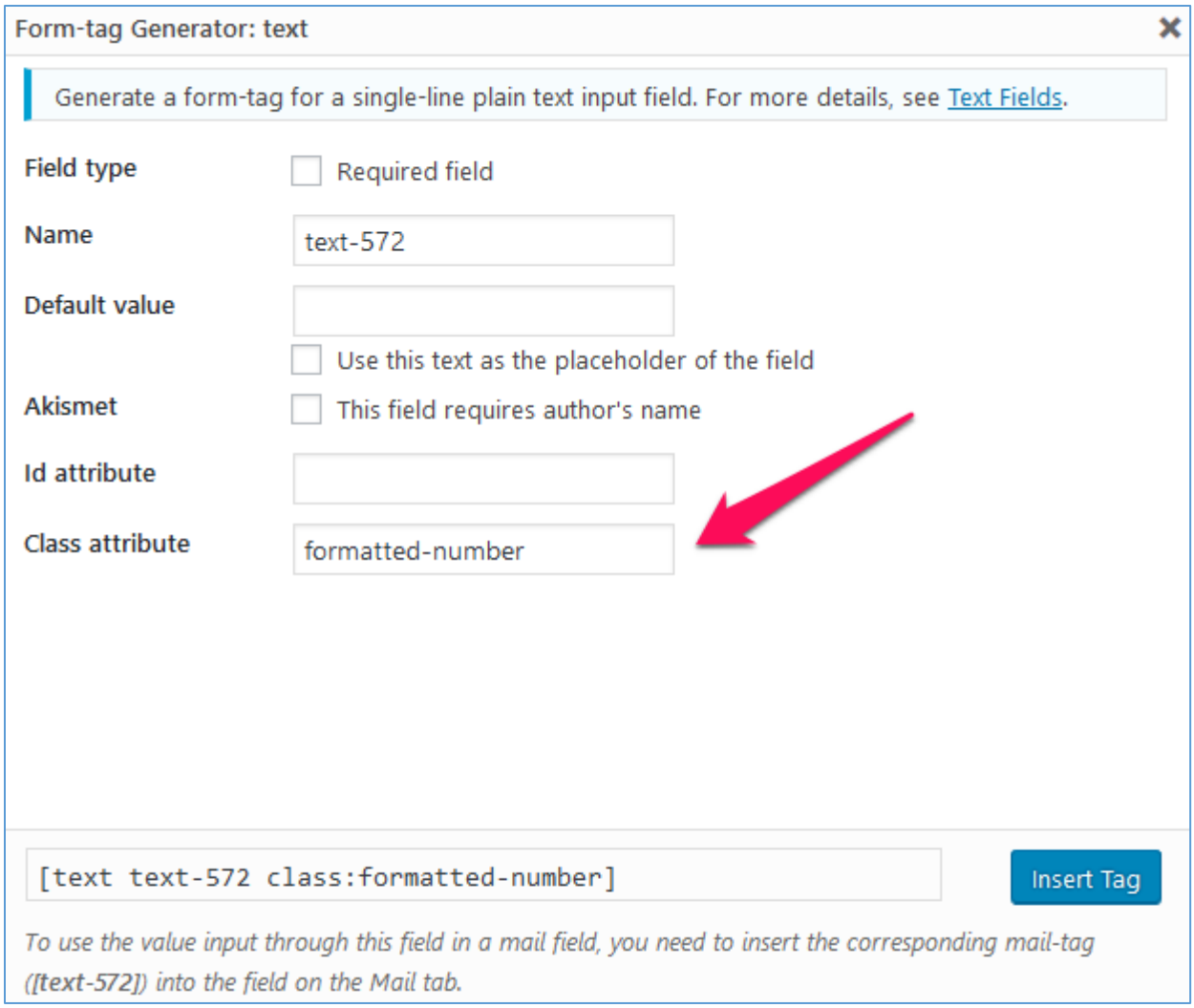
```
.wpcf7-calculation { border: red }
```

6. INPUT FIELD MASKING *(PRO VERSION ONLY)*

If you choose a custom number formatting by modifying the decimal and thousands separators on the settings page, you can also *enforce* this formatting on any user input fields.

To do so, you can create a standard **text** input field and add the "**formatted-number**" class name to it.

Note: For this to work, you need to make the input a **text** field, not **number**!



This will also prevent the user from entering any non-numeric characters in the field.

Before	After
Weight: 1000 kg	Weight: 1,000

7. SCRIPTING

7.1. TRIGGERING CALCULATION PROGRAMMATICALLY

You can trigger calculation programmatically from your own Javascript code by calling the **`cf7Calculate`** function. It takes a single argument a jQuery object constructed from the respective `<form>` element. For example:

```
<script>
jQuery(document).ready(function($){
    var recalc = function() {
        // Find any forms on the current page and activate the calculations
        cf7Calculate(jQuery('form'));
    };
    jQuery('#mybutton').on('click', recalc);
});
</script>
```

7.2. POST-CALCULATION EVENT HANDLING

After the calculation AJAX call is made, the plugin will trigger the ***wpcf7calculate*** jQuery event. If the AJAX call fails, ***wpcf7calculatefail*** will be triggered. You can code your own handlers for these events, for example:

```
<script>
jQuery('form#calculator').on('wpcf7calculate', function() {
  alert('Your calculation is ready!');
}
</script>
```