CONSULTIMATOR

v 1.1.11.01

https://consultimator.com

Technical Reference

Table of Contents

1 Introduction

Welcome to CONSULTIMATOR! With CONSULTIMATOR you can create dialog pages, small expert systems, automatically generated model contracts, calculators and letters.

You can set up a questionnaire as well as the corresponding answers and information. Depending on the complexity of your project you will need no or only little programming skills.

Basic HTML and Javascript skills are sufficient to push CONSULTIMATOR to unexpected power. No complex programming needed, CONSULTIMATOR will do this for you.

The process is easy:

- You set up a project file witch describes the questions to be asked as well as the corresponding content and answers.
- Next, you start CONSULTIMATOR to create a HTML page from this project file.
- You can design this HTML page freely using CSS or you just keep the standard CSS settings used by CONSULTIMATOR.

Demo-Mode

Curious? Please feel free to test CONSULTIMATOR! You can create small dialogs with a maximum of 20 actions for evaluation purposes without a license key. Actions for handling secure data are not available in the demo mode and will be ignored when creating the output pages.

Do you need a license key?

Please contact me, you´ll find my contact data on the CONSULTIMATOR website.

2 The Project File

2.1 General

The project file is the central configuration file for your project. In this project file you define all actions CONSULTIMATOR is supposed to execute, as well as several general information.

Doing this you will normally not be programming, but just setting values. At some points you will be writing program code, but you will probably not even notice it. ;-)

2.2 The Project File XML-Format

You will create the project file in a file format called „XML“. The CONSULTIMATOR Editor will save it with the file extention „.cmp”, but inside, it´s still XML.
XML stands for „Extended Markup Language“, but that´s just nice to know, just mentioned for educational purpose. ;-)

What you should know about XML files, as far as we´re using XML here:

- Every element in your project file has a name. This name is placed between a greater-than- and a lower-than – character, e.g. like this:

  „<elementname>“

- The end of an element is marked similarly, only that there is an additional „/“ between the lower-than-character and the element name, e.g. like this:

  </elementname>

- The actual value assigned to the element stands inbetween these two markers, e.g. like this:

  <elementname>This is the elements value!</elementname>

- When entering the „forbidden“ characters „&“, „<“ und „>“ within values you will have to use a special syntax to avoid problems. They are entered like this:
  - Aus „&“ wird „&“
  - Aus „<“ wird „<“
  - Aus „>“ wird „>“

  e.g..:

  <elementname>I love fish & chips!</elementname>

- There is an alternative way to enter these „forbidden“ characters, which is very useful if they occur very often. You can also use it to enter very long text over multiple lines:

<elementname><![CDATA[Just as well as fish & chips I love steak & kidney pie!]]></elementname>

You will love this when you have to enter very long text over multiple lines, for you can enter any text between <![CDATA[and]]>.

Caution:
Multiline (!) values, those using line feeds, will only be accepted by CONSULTIMATOR **using action type OMESSAGE**! That doesn´t mean, that other actions may not contain line feeds creating HTML-tags like „
“ or „ “, but the project file may not contain line feeds outside of OMESSAGE actions.

- Elements may be placed within each other, e.g. like this:

```
<actions>
    <action1>
        <subaction1></subaction1>
        <subaction1></subaction1>
    </action1>
    <action2></action2>
    <action3></action3>
</actions>
```

But never like this:

```
<action1>
    <action2>
</action1>
    </action2>
```

2.3 Section „GENERAL“

2.3.1 Mixed Settings

2.3.1.1 KEY

This is where you enter your personal license key. Without this key you cannot publish your results to consultimator.com. Nevertheless, you can download them an use them locally, or publish them on your own webserver, within our terms and conditions.

Attention – DELETE-function:

If you add „**-kill**“ at the end of the license key and if RELEASE ist set to „true“, the output file will not be generated, but **deleted**!

Project Signature instead of License Key:

If you would like to pass your project file on to others without handing out your license key, you can alternatively sign your project with your project signature.

You can create your project signature either by using the according function in the CONSULTIMATOR Editor, where you find your KEY settings, or you can call the processor via URL with the additional parameter “signproject=true”.

Just place the project signature into the KEY, just as you would with your personal license key.

Please be aware that this is by purpose not a full alternative to your personal license key. Once signed with the project signature, actions may not be moved, added, deleted, their AID and their ATYPE may not be changed without losing the validity of the project signature.

2.3.1.2 RELEASE

If RELEASE is set to „true“, the output file will be saved in your personal folder, using the file name defined as TEMPLATENAME.
This option will only work, if an individual license key is provided.

2.3.1.3 TEMPLATENAME

This is where you enter the name of the file to be created. Please, don´t add a file extention, this will always be “.html” and added automatically.

2.3.1.4 TEMPLATEHEAD

CONSULTIMATOR will create the complete HTML code necessary. It´s up to you whether you plan to embed it into another Website or use the file “stand alone”.

If you would like to create a complete HTML page, you can add all header data here. This could e.g. look like this:

```
<templatehead><![CDATA[<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0"/>
<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
<link rel="stylesheet" type="text/css" href="cssoptions/standard.css">
<title>This is the title of your page</title>
</head>
<body>
]]></templatehead>
```

As you can see, you can add a CSS-file here, which will handle all design settings for your page.

You could also add individual Javascript files to enhance the functions that can be used within CONSULTIMATOR. For details see below.

If you would like to use a standard HTML-header and foot, simply define TEMPLATEHEAD like as "[html]":

<templatehead>[html]</templatehead>

(After „[html]" you can define further HTML-Code, to show up after the tag „<body>" and before the actual CONSULTIMATOR output.)

In this case you can omit the TEMPLATEFOOT, but you should set the TEMPLATETITLE to your page title, so that it will show up in the browser.

2.3.1.5 JAVASCRIPTBLOCK

Through JAVASCRIPTBLOCK individual JavaScript code can be added, which will be placed at the beginning of the JavaScript output file.

Please enter only the JavaScript code without using „<script>" / „</script>" or similar.

2.3.1.6 LINKSTYLESHEET

Through LINKSTYLESHEET you can define a CSS stylesheet link different to the one used when defining the HTML Header as „[html]".

The link has to be defined this way:

<link rel='stylesheet' type='text/css' href='…….' >

Exemption: Alternative CSS-Files stored in https://consultimator.com/cssoptions/ can be linked simply by their file name.

If this field stays empty, the default value will be used.

2.3.1.7 TEMPLATETITLE

The value assigned to "Templatetitle" will be added to the TEMPLATEHEAD as page title:

<templatetitle>This could be your page title</templatetitle>

This setting will be ignored, if TEMPLATEHEAD is different to „[html]".

2.3.1.8 TEMPLATEFOOT

The value assigned to“Templatefoot” will be inserted at the end of the page. A HTML page would, for example, be finalised like this:

<templatefoot><![CDATA[</body>
</html>]]></templatefoot>

If your TEMPLATEHEAD contains „[html]”, after HTML code set here an additional „</body></html>“ will be set.

2.3.1.9 TEMPLATELANGUAGE

Through “Templatelanguage” you define whether certain dialogs will be shown in English or in German:

English dialogs:
<templatelanguage>en</ templatelanguage >

German dialogs (default, can be omitted):
< templatelanguage >de</ templatelanguage >

2.3.1.10 STORAGETYPE

Through “storagetype” you define whether entered values are saved locally in your browser in LocalStorage or SessionStorage, or not at all.

LocalStorage (data last permanently until deleted in browser):
<storagetype>local</storagetype>

SessionStorage (data last until session is closed, e.g. by closing tab):
<storagetype>session</storagetype >

No saving (default, can be omitted):
<storagetype></storagetype>

2.3.1.11 COMPONENTPATH

If you would like to save output files to a different System, you can change you can change your “components”-path here.

2.3.1.12 SHOWOMESSAGESDEFAULT

CONSULTIMATOR divides all content into the two blocks „INPUTBLOCK“ (for all input actions) and „OUTPUTBLOCK“ (for all output actions).
The OUTPUTBLOCK content can optionally be visible or unvisible on page load.

Here, you can choose between these two alternatives:

- Use „true“ to show OUTPUTBLOCK immediately.
- Use „false“ to keep OUTPUTBLOCK invisible on load. In this case, don´t forget to include the action „SHOWOMESSAGES“, otherwise you will have no button to make the OUTPUTBLOCK visible.

2.3.1.13 PLACEHOLDERUNDERLINE

CONSULTIMATOR regularly will show highlighted action ids, as long as placeholders have not been set within the output area. If you prefer to show „_____“ instead, you can change this here.

Here, you can choose between these two alternatives:

- Use „true“ to simply show „_____“.
- Use „false“to show the action ids.

2.3.1.14 PLACEHOLDERCHAR

If you would like to use a different Character to “_” for the PLACEHOLDERUNDERLINE-function, you can set one or more Characters here. They will be repeated 7 times. Useful, e.g., would be a “ ” to set a fixed space.

2.3.1.15 EDITABLEOMESSAGES

If EDITABLEOMESSAGES is set to “true”, the user can manually edit any generated output inside the web browser.

Attention:

If the user changes any data in the INPUTBLOCK, manual changes to the output will be dismissed.

Attention:

Do not use this function if you would like to be able to reproduce the exact output from saved form data, as manual changes to the output will not be saved as part of the form data.

2.3.1.16 XSSSTRICT

If XSSSTRICT is set to “true”, the automatically generated HTML-Header will add Inline-CSS-Prevention to the Cross Site Scripting Policy. Inline-JavaScript is prohibited by

default. Both JavaScript- as well as CSS-files may only be hosted on the CONSULTIMATOR server (or your own server, if you host the files yourself).

Reminder:
In case you still have inline CSS definitions (style="…"), your webbrowser will not use them and throw an error. To at least suppress these error messages, CONSULTIMATOR will try to find all occurances of **style="…"** and remove them from your actions.
If you would like to set XSSSTRICT to "true" and still like to style your output individually, you can use the action "STYLE" to define CSS classes. These can be assigned to your output via **class="…"**.

2.3.2 Form Action Behaviour

Values entered in the INPUTBLOCK can either be used for creating the OUTPUTBLOCK or to pass them on to another webpage. All data are collected in a form (HTML <form>). You define, how these data shall be used:

2.3.2.1 FORMACTION

A typical form action would be e.g.

<formaction>nextpage.html</formaction>

In this case form data would be passed on to the HTML page „nextpage.html" and can be processed there.

It´s also possible to generate an email with these data::

<formaction>mailto:myemail@email.com</formaction>

Special Case: Transfer via „CONSULTIMATOR Sicheres Kontaktformular" mechanism

To use this form of data transfer, please use the following settings:

secureformtransmit.php|*my@email.com*|*New message*

or

secureformtransmit_toserver.php| *my@email.com*|*New message*

Please replace *„my@email.com"* with your target email address. *„New message"* can be set as you wish, this text will be part of the email subject, next to other information.

If you choose secureformtransmit.php, the complete message will be sent to you as an encrypted email attachment .

Using „secureformtransmit**_toserver**.php", the encrypted message will remain on the server and can be retrieved only through the cryptic URL link sent to you by a notification email.. Retrieval must occur within 365 days after notification and the

message will be deleted 90 days after retrieval or 365 days after notification, whatever date is first.

The encryption itself is described separately. Encryption can only by decrypted by you as the owner of the private key and other people you share the private key with.

To make use of this form of data transfer, please make shure to use the FORMMETHOD „post“, as described below.

Tip: Use of processing instructions

You can also add processing instructions to the "subject" of your message. These serve as supplementary work instructions for subsequent pages or the CONSULTIMATOR Workflow Engine.

The block containing the processing instructions begins with [pi] and ends with [/pi]:

[pi] *List of processing instructions* **[/pi]**

The following processing instructions are currently supported:

Extended Parameters ("[xp]")

You can use the processing instruction [xp] to instruct secureformtransmit(_toserver).php to add further parameters to the link for calling up the output page to be filled with data. For example, the parameter "pkencskip" can be useful:

[xp]pkencskip=*default***[/xp]**

If the parameter "pkencskip" is set, the specified value, here "default", is used to decrypt your RSA private key with the specified value (here: "default") if it is stored encrypted in the browser. This means that the message to be decrypted can be decrypted simply by clicking on the link in the email without any further intermediate steps.

Of course, this option should only be used if the computer with the private key to be decrypted is itself adequately protected against access by unauthorised third parties, as this reveals the password for decrypting the private key.

Also, a value used in the form can also be set via its action name, here, for example, the value of the action "callviamail" with the value "yes":

[xp]*callviamail=yes***[/xp]**

Several parameters are joined together separated by "&":

[xp]pkencskip=default**&**aufrufausmail=yes[/xp]

Alternative return URL ("[altreturnurl]")

Use the "altreturnurl" to instruct secureformtransmit to point the link in the email to open the encrypted data to a different page than the page used to capture the data: [altreturnurl]/customers/customerid/formname.html[/altreturnurl]

For example, data can be collected via a form and processed in another form with a single click.

Customised processors for the Workflow Engine ("[pipnames]")

If you use the CONSULTIMATOR Workflow Engine and would like to instruct it to trigger further processing or workflows in addition to the basic data preparation tasks, depending on the respective form, you can add your own processors to the engine. You can name the processors to be executed in each case using the "pipnames":

[pipnames]*name1*,*name2*,*name3***[/pipnames]**

In this example, the Workflow Engine would execute its processors "name1", "name2" and "name3" in this order after the standard formats that are configured in the engine itself have been processed.

The processors are located in the Workflow Engine in the "settings" subdirectory and follow the following naming convention:

pi_*gewählternam***e_processor.php**

The CONSULTIMATOR Workflow Engine is open source and can be customised as required. To create your own processors, you can use the existing processors in the "settings" folder as a template. However, developing your own processors and customising the Workflow Engine requires knowledge of the PHP programming language. The CONSULTIMATOR Workflow Engine is therefore provided without warranty and without any promise of support, unless expressly agreed otherwise and unless customisations are made exclusively by the provider.

2.3.2.2 FORMMETHOD

Here you can define the method of data transfer. There are two options:

- <formmethod>get</formmethod>

- With „get“ all data are passed on to the next page via the URL. This is not particularly safe, as all data are passed on clearly visible, but this might be ok for you with uncritical data.

  In addition, you will have problems if you try to pass on more than 3000

characters, because most browsers will not be able to process these. Please, take into account that not only the data, but also the datafield-names count. Nevertheless, 3000 characters actually can carry a lot of information if you don´t need extremely long text input fields.

- <formmethod>post</formmethod>

  The „post“ method is much more secure, because data are not handed over via the URL, but directly to the server.

  Please take into account that data handed over directly to the server cannot be processed by a normal “HTML” webpage with JavaScript, because JavaScript is not processed on the server. In this case you could e.g. use PHP, because PHP can read the data handed over to the server.

  If you combine „post“ with the form action „mailto:“ all data are handed over to the local email client.

2.3.2.3 FORMTARGET

This element defines, whether, if used for opening a new webpage, the browser will use the same brower window or use an new one. If you want the browser to stay in the same window, leave this element empty. Otherwise, define it as follows:

<formtarget>_blank</formtarget>

2.3.2.4 FORMENCTYPE

Here you can define in which format data are passed on to the next page.
Options are:

- *application/x-www-form-urlencoded*:
  (default value) All characters are encoded before sending (Spaces to "+", Special Characters to ASCII HEX-values)
- *multipart/form-data*: no encoding
- *text/plain*: Spaces are encoded to "+"; special characters stay unchanged

When using form action „mailto:“, „text/plain“ will make the data show in the email client in plain text. This, however, will not be compatible to the action „PARAMETERS“, which will be explained later.

2.3.2.5 RSAPUBLICKEY

Here you can define your RSA Public Key, which will be used for encrypting user data when usind the actions FORMDOWNLOADSECURE and SUBMITSECURE. There is a separate chapter in this documentation concerning data encryption.

2.4 Section „ACTIONS“

In the section „ACTIONS“ all actions are defined in the order they will be processed.

Actions can roughly be divided in input-actions shown in the INPUTBLOCK, calculation-actions to create necessary additional values, and output-actions shown in the OUTPUTBLOCK.

Every action element consist of several elements which define, how the action will behave.

The most important action elements:

- „AID“: every action has an action-id, „AID“.

  The AID
  - has to be unique,
  - may only contain lowercase characters and numbers,
  - may not contain any special characters,
  - may not begin with a number.

  The „AID“ ist most important for CONSULTIMATOR! Via the AID the value of any action can be queried, z.B. when using the „ACONDITION“, via „COMPUTE“ and also as replacement text during text output.

- The element „ATYPE“ defines, which kind of action is to be performed. The different types are explained below.
- „ACONDITION“ defines, whether the action will be executed or not.
- „ATEXT“ defines, which text will be show with the action.

These already are the most important elements. There are some more elements, which are only necessary for certain action types. They are explained within the context of the particular action below.

3 The Different Action Types

3.1 Data Entry Actions in the „INPUTBLOCK“

3.1.1 Data Entry - Actions

3.1.1.1 Action: TEXT

Action TEXT creates a simple Text-Inputfield.
Text can be entered in a single row.
The input field size can be configured through ATEXTLENGTH oder ATEXTWIDTHPERCENT.

3.1.1.1.1 AID

Unique ID based on the rules explained above: lowercase only, only letters a – z, no special characters, numbers are ok, but not as first character.

3.1.1.1.2 ATYPE

Value: text

3.1.1.1.3 ACONDITION

The content of this element is checked logically, whether its result is “true”.
It is a comparison that has to be entered as you might be used to in Javascript.

You will find an outstandingly good overview here:

https://www.w3schools.com/js/js_comparisons.asp

Important:
You can query the values of all input actions for comparison purposes through their AID! Doing this, all returned values are of the type *String*, except the return value of the action “checkbox”, which will be of the type *Boolean* (containing the value *true* or *false*). In the same way you can query the results of any “compute” action. Their type will be determined by the assigning operation.

Example: You have defined a text action with AID „nameofthedog“ angelegt. You can query this value through ACONDITION e.g. like this:

<acondition>nameofthedog == „Bozo“</acondition>

If your user has entered „Bozo“ as the dogs name, the condition is „true” and the action will be executed.

Furthermore, you can perform practically any JavaScript command that will deliver a result of „true“ or „false“.

3.1.1.1.4 ATEXT

The text entered here will be used for labelling the text input field.

If you would like to adapt the text based on the user input, you can insert these user inputs into the text by placing the according AID into curly braces, „{“ and „}“.

So if your user entered „Bozo“ to the text input field named with AID „nameofthedog“, you could ask for the dogs age like this:

<atext>How old is {nameofthedog}?</atext>

3.1.1.1.5 APLACEHOLDER

If this value is set to „true“, the text entered als ATEXT will not be displayed as a label but as a placeholder text inside the text input field.

3.1.1.1.6 ADEFAULTVALUE

This is the suggested Value for this action.

<adefaultvalue>suggested Value </adefaultvalue>

3.1.1.1.7 ATEXTLENGTH

Here you can define the width of the input field in characters, so e.g. for a 4-character input field:

<atextlength>4</atextlength>

3.1.1.1.8 ATEXTWIDTHPERCENT

Here you can define the width of the input field in percent of the window width. So if you want the input field to span the full window width you would enter:

<atextwidthpercent>100</atextwidthpercent>

3.1.1.1.9 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.2 Action: TEXTAREA

The action TEXTAREA will create a multiline textinput field, which will also allow line feeds using the Enter-Key during text entry.

3.1.1.2.1 AID

see above

3.1.1.2.2 ATYPE

Value: textarea

3.1.1.2.3 ACONDITION

see above

3.1.1.2.4 ATEXT

see above

3.1.1.2.5 APLACEHOLDER

see above

3.1.1.2.6 ADEFAULTVALUE

see above

3.1.1.2.7 ATEXTROWS

Defines the number of textrows the multiline text entry field will show. The text actually entered may be longer.

Example for a multiline text entry field with 5 rows:

<atextrows>5</atextrows>

3.1.1.2.8 ATEXTLENGTH

see above

3.1.1.2.9 ATEXTWIDTHPERCENT

see above

3.1.1.2.10 AWYSIWYG

If AWYSIWIG is set to “true”, the TEXTAREA will use a WYSIWYG HTML editor.

Please note:
This option requires the HTML editor „Trumbowyg“ (https://alex-d.github.io/Trumbowyg/) as well as JQuery to be installed on the webserver.

3.1.1.2.11 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.3 Action: DATE

The action DATE creates a date entry field. Most browsers will support this input type by showing an input mask as well as a date picker tool to select the date from a calendar view.

3.1.1.3.1 AID

see above

3.1.1.3.2 ATYPE

Value: date

3.1.1.3.3 ACONDITION

see above

3.1.1.3.4 ATEXT

see above

3.1.1.3.5 ADEFAULTVALUE

see above, please use the ISO standard “YYYY-MM-DD” for default values.

3.1.1.3.6 Speciality: „ISOFORMAT“

There is a speciality concerning the date input field.

No matter which format the date entry field will show during data entry, the “value” would always be presented in ISO standard “YYYY-MM-DD”. In Germany, where CONSULTIMATOR was developed, dates are usually used formatted as “DD.MM.YYYY”, so the value will be converted in the background into the German standard.

So, you fill a date input field with AID „birthday“ with August 3rd, 1988, the value of „birthday“ will be converted and saved as 03.08.1988.

Nevertheless, for calculation and other purposes, the ISO-format might still be useful. This is why CONSULTIMATOR will automatically create a second value in ISO-format, which is stored to AID + “isoformat”. So if you query for “birthdayisoformat”, you will still receive the original value “1988-08-03”.

3.1.1.3.7 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.4 Action: MONTH

The action MONTH creates a month / year entry field. Most browsers will support this input type by showing an input mask as well as a date picker tool to select the date from a calendar view.

3.1.1.4.1 AID

see above

3.1.1.4.2 ATYPE

Value: date

3.1.1.4.3 ACONDITION

see above

3.1.1.4.4 ATEXT

see above

3.1.1.4.5 ADEFAULTVALUE

see above, please use the ISO standard “YYYY-MM” for default values.

3.1.1.4.6 Speciality: „ISOFORMAT“

There is a speciality concerning the month input field.

No matter which format the date entry field will show during data entry, the “value” would always be presented in ISO standard “YYYY-MM”. In Germany, where CONSULTIMATOR was developed, month / year information are usually used formatted as “*fullmonth* YYYY”, so the value will be converted in the background into the German standard.

So, you fill a month input field with AID „month_of_birth“ with August 1988, the value of „ month_of_birth “ will be converted and saved as “August 1988”.

Nevertheless, for calculation and other purposes, the ISO-format might still be useful. This is why CONSULTIMATOR will automatically create a second value in ISO-format,

which is stored to AID + “isoformat”. So if you query for “month_of_birthisoformat”, you will still receive the original value “1988-08”.

3.1.1.4.7 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.5 Action: TIME

The action TIME creates a time-entry field. Most browsers will support this input type by showing an input mask as well as a Time picker tool to select the time from. The time-entry field uses 24-hour format, “hh:mm”.

3.1.1.5.1 AID

see above

3.1.1.5.2 ATYPE

Value: date

3.1.1.5.3 ACONDITION

see above

3.1.1.5.4 ATEXT

see above

3.1.1.5.5 ADEFAULTVALUE

see above, please use the 24-hour format “hh:mm” for default values.

3.1.1.5.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.6 Action: CHECKBOX

The action type CHECKBOX will create a checkbox.
Important: the value of a checkbox will be „true“, if the checkbox is checked, „false“, if unchecked.

3.1.1.6.1 AID

see above

3.1.1.6.2 ATYPE

Value: checkbox

3.1.1.6.3 ACONDITION

see above

3.1.1.6.4 ATEXT

see above

3.1.1.6.5 ADEFAULTVALUE

see above

3.1.1.6.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.7 Aktion: COLOR

The action COLOR creates a HTML5 color picker dialog.

Attention:
The color picker does not show on all web browsers. Whilst Chrome, Firefox und Edge work fine, Safari and Internet Explorer will only show a text input field. Nevertheless, the color value can still be entered in RGB-Hex style (#xxyyzz).

3.1.1.7.1 AID

see above

3.1.1.7.2 ATYPE

Value: color

3.1.1.7.3 ACONDITION

see above

3.1.1.7.4 ATEXT

see above

3.1.1.7.5 ADEFAULTVALUE

see above

3.1.1.7.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.8 Action: MESSAGE

The action MESSAGE will only create a text output and will not accept any user input. MESSAGE is good for giving advice or interim results to the user during the use of the INPUTBLOCK, because MESSAGE output is shown in the INPUTBLOCK.

Do not mix um MESSAGE and OMESSAGE (Output-Message): OMESSAGE actions are (only) meant for showing text output in the OUTPUTBLOCK.

Tip:
MESSAGE can also be used to do „invisible“ things, e.g. like defining CSS settings. This example will change the color of buttons to blue:

```
  <action>
        <aid>styledeclaration</aid>
        <acondition>false</acondition>
        <atype>message</atype>
        <atext><![CDATA[
<body>
<style>
.con_button {background-color: blue;}
</style>
]]></atext>
  </action>
```

In this case, please remember to set the ACONDITION to „false“ to prevent the output from creating an extra empty row.

3.1.1.8.1 AID

see above

3.1.1.8.2 ATYPE

Value: message

3.1.1.8.3 ACONDITION

see above

3.1.1.8.4 ATEXT

see above

3.1.1.8.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.9 Action: BLINDS

The action „BLINDS“ will create a subtitle-block within the content input area. This block will work as a switch to show or hide the content inside the block. Clicking the subtitle-block will show all input elements inside the block and hide all elements belonging to other blocks.

If the subtitle-block already open and clicked again, it will close.

If the first block is supposed to be closed on page load, this can be achieved via individua style settings:

```
#blindstext1 {
     display:none;
}
```

In the same way, an alternative block can be shown in an open state on page load, e.g. for block 2:

```
#blindstext2 {
     display:block;
}
```

3.1.1.9.1 AID

see above

3.1.1.9.2 ATYPE

Value: blinds

3.1.1.9.3 ACONDITION

see above, but: ACONDITION will not only hide the subtitle on “false”, but the complete block with all input elements.

3.1.1.9.4 ATEXT

see above

3.1.1.9.5 ABLOCKID

ABLOCKID can be set to the AID of an OMESSAGE action. By doing so, the output area will scroll to the selected OMESSAGE when the BLIND is clicked. This, of course, only makes sense, if the input- and output areas are placed next to each other and they can scroll separately.

3.1.1.9.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.10 Action: SELECT

The action SELECT will show a drop-down box with a predefined list of selectable input values. The user can only pick one of these values.

3.1.1.10.1 AID

see above

3.1.1.10.2 ATYPE

Value: select

3.1.1.10.3 ACONDITION

see above

3.1.1.10.4 ATEXT

see above

3.1.1.10.5 ADEFAULTVALUE

see above

3.1.1.10.6 AOPTIONS

Use AOPTIONS to define the list of selectable input values the user can pick from.

3.1.1.10.6.1 AOPTION

This is one of the selectable input values, consisting from the text shown to the user and the actual value set, if the user selects this option.

3.1.1.10.6.1.1 AOPTLABEL

This is the option text shown to the user.

3.1.1.10.6.1.2 AOPTVALUE

This is the value to be set, once the user selects the defined option text.

3.1.1.10.7 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.1.11 Action: OPTION

The action OPTION will show several Option buttons (so called “Radio Buttons”), based on the options defined. Only one of the options can be selected.
Basically, both OPTION and SELECT aim for the same goal and simply look differenty.

3.1.1.11.1 AID

see above

3.1.1.11.2 ATYPE

Value: option

3.1.1.11.3 ACONDITION

see above

3.1.1.11.4 ATEXT

see above

3.1.1.11.5 ADEFAULTVALUE

see above

3.1.1.11.6 AOPTIONS

Use AOPTIONS to define the list of selectable input values the user can pick from.

3.1.1.11.6.1 AOPTION

This is one of the selectable input values, consisting from the text shown to the user and the actual value set, if the user selects this option.

3.1.1.11.6.1.1 AOPTLABEL

This is the option text shown to the user.

3.1.1.11.6.1.2 AOPTVALUE

This is the option text shown to the user.

3.1.1.11.6.1.3 AOPTCHECKED

If this element ist set to „true“, this option is preselected.

3.1.1.11.7 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2 Special Functions

3.1.2.1 Action: SHOWOMESSAGES

SHOWOMESSAGES shows a button, which will toggle the OUTPUTBLOCK visible or unvisible.

You will have to define this button, if you defaulted the visibility of the OUTPUTBLOCK to “false” in the GENERAL section.

3.1.2.1.1 AID

see above

3.1.2.1.2 ATYPE

Value: showomessages

3.1.2.1.3 ACONDITION

see above

3.1.2.1.4 ATEXT

This is the label of this button.

3.1.2.1.5 ATOGGLEINPUTOUTPUT

Default is “false”.

If set to „true“ INPUTBLOCK will be hidden, when OUTPUTBLOCK is made visible.

“true” may not be set, if OUTPUTBLOCK is visible by default. In this case SHOWOMESSAGES would hide both INPUTBLOCK and OUTPUTBLOCK and leave you with an empty page.

3.1.2.1.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.2 Action: SUBMIT

Action SUBMIT will show a button to handle the form input according to the form action settings defined in the GENERAL section. . Data will be encrypted asymetically with your RSA Public Key defined in the project settings.

3.1.2.2.1 AID

see above

3.1.2.2.2 ATYPE

Value: submit

3.1.2.2.3 ACONDITION

see above

3.1.2.2.4 ATEXT

This is the label of this button.

3.1.2.2.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.3 Action: FORMCLIPBOARD

Action FORMCLIPBOARD will show a button which will send all form data to the clipboard in a format compatible to the PARAMETERS action.

3.1.2.3.1 AID

see above

3.1.2.3.2 ATYPE

Value: submit

3.1.2.3.3 ACONDITION

see above

3.1.2.3.4 ATEXT

This is the label of this button.

3.1.2.3.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.4 Action: SUBMITSECURE

Action SUBMIT will show a button to handle the form input according to the form action settings defined in the GENERAL section. Data will be encrypted asymetically with your RSA Public Key defined in the project settings and will be passed along in only one parameter “encdata”.

This action is not available when using the Internet Explorer and will not be displayed there.

3.1.2.4.1 AID

see above

3.1.2.4.2 ATYPE

Value: submit

3.1.2.4.3 ACONDITION

see above

3.1.2.4.4 ATEXT

This is the label of this button.

3.1.2.4.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.5 Action: PARAMETERS

Action PARAMETERS will show a button, which will display a text entry dialog.
If the user copies the according URL parameters into this field, the page will be filled with these values. If the data is encrypted, you will be prompted to enter your RSA Private Key.

These URL parameters are created, when you define the form action to „mailto:“ and click the SUBMIT button, see the description above.

Using these two buttons in combination your user can export and re-import all form data.

3.1.2.5.1 AID

see above

3.1.2.5.2 ATYPE

Value: parameters

3.1.2.5.3 ACONDITION

see above

3.1.2.5.4 ATEXT

This is the label of this button.

3.1.2.5.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.6 Action: RESET

Action RESET shows a button to clear all form data.

3.1.2.6.1 AID

see above

3.1.2.6.2 ATYPE

Value: reset

3.1.2.6.3 ACONDITION

see above

3.1.2.6.4 ATEXT

This is the label of this button.

3.1.2.6.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.7 Action: FORMDOWNLOAD

Action FORMDOWNLOAD shows a button to save all form data to your computer via browser download.

3.1.2.7.1 AID

see above

3.1.2.7.2 ATYPE

Value: formdownload

3.1.2.7.3 ACONDITION

see above

3.1.2.7.4 ATEXT

This is the label of this button.

3.1.2.7.5 AFILENAME

This is the suggested file name. Any added file-extention will be replaced by ".cmc".

3.1.2.7.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.8 Action: FORMDOWNLOADCSV

Action FORMDOWNLOAD shows a button to save all form data as a comma separated file (CSV) to your computer via browser download.

All values in the CSV file are enclosed by double quotation marks and separated by semicolons. The file has UTF-8 character encoding. The first line contains the field names.

Attention:
In general, Microsoft Excel can read CSV files. Nevertheless, Excel doesn´t recognize the UTF-8 encoding automatically. Please, use the option to access external data in text files instead to import the file data.

3.1.2.8.1 AID

see above

3.1.2.8.2 ATYPE

Value: formdownloadcsv

3.1.2.8.3 ACONDITION

see above

3.1.2.8.4 ATEXT

This is the label of this button.

3.1.2.8.5 AFILENAME

This is the suggested file name. Any added file-extention will be replaced by ".cmc".

3.1.2.8.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.9 Action: FORMDOWNLOADSECURE

Action FORMDOWNLOADSECURE shows a button to save all form data to your computer via browser download. Data will be encrypted asymetically with your RSA Public Key defined in the project settings.

This action is not available when using the Internet Explorer and will not be displayed there.

3.1.2.9.1 AID

see above

3.1.2.9.2 ATYPE

Value: formdownload

3.1.2.9.3 ACONDITION

see above

3.1.2.9.4 ATEXT

This is the label of this button.

3.1.2.9.5 AFILENAME

This is the suggested file name. Any added file-extention will be replaced by ".cmc".

3.1.2.9.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.10 Action: PARAMETERUPLOAD

Action PARAMETERUPLOAD shows a button to upload all form data from your computer via file upload. If the data is encrypted, you will be prompted to enter your RSA Private Key.

3.1.2.10.1 AID

see above

3.1.2.10.2 ATYPE

Value: parameterupload

3.1.2.10.3 ACONDITION

see above

3.1.2.10.4 ATEXT

This is the label of this button.

3.1.2.10.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.2.11 Action: ATTACHMENT

Action ATTACHMENT shows a button to upload an attachment to the form data. This attachment will be submitted or saved when using SUBMIT, SECURESUBMIT, FORMDOWNLOAD und FORMDOWNLOADSECURE.

Attention:
This action can only be used once in a project.

3.1.2.11.1 AID

see above

3.1.2.11.2 ATYPE

Value: attachment

3.1.2.11.3 ACONDITION

see above

3.1.2.11.4 ATEXT

This is the label of this button.

3.1.2.11.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.1.3 General Actions

3.1.3.1 Action: STYLE

Action STYLE gives you the opportunity to add Cascading Style Sheet (CSS) definitions to your project. CSS definitions will be placed into a separate file and will not be placed inline into the HTML page.
The action can be used multiple times.

3.1.3.1.1 AID

see above

3.1.3.1.2 ATYPE

Value: style

3.1.3.1.3 ACONDITION

none

3.1.3.1.4 ATEXT

This is where the CSS style definitions are placed. Please do not include them in <style> … </style>, as this could cause errors. CONSULTIMATOR will remove them automatically in most cases, but best not to depend on this.

3.2 Computing Values

3.2.1 Action: COMPUTE

The action COMPUTE is extremely powerful.
COMPUTE is no button or input type, but a data processing function.

Wich COMPUTE you can create new values from existing ones: first, the status of ACONDITION is analyzed, and the value of the COMPUTE action will be set to ATEXT, if ACONDITION is true.

What makes ACONDITON special: if ACONDITION is false, there is a fallback value, defined by ATEXTFALSE.

But there is one more, very important speciality, which is described below at ATEXT.

3.2.1.1 AID

see above

3.2.1.2 ATYPE

Value: compute

3.2.1.3 ACONDITION

see above

3.2.1.4 ATEXT

This is the important difference:

Contrary to the other ATEXT definitions other action types, here ATEXT does not assign text values with optional replacements, but works according to JavaScript standards.

Here, you define in JavaScript format an Operation to be executed. Its value will be assigned to the value of the COMPUTE action.

For example, you can

- Combine several AID values als text:

  <atext>“The dog is called “ + nameofthedog + “ and is “ + ageofthedog + “ years old!“</atext>

- AID values can be computed mathematically, e.g. the age of the dog in 3 years can be computed this way:

  <atext>+ageofthedog + 3</atext>

  Notice:
  In the example above the code doesn´t show

  „ageofthedog + 3“

  but:

  „+ageofthedog + 3“

  The reason is easy to understand: if the „+“ before „ageofthedog“ is missing, the system will think that „ageofthedog“ is a character string and not a numerical value. In that case, e.g., if the dog was 12 years old, attaching the „3“ to the end would result in this value: „123“. This would be fantastic for the dog, but, for what we wanted to achieve, simply incorrect.

- Also, within ATEXT JavaScript specific functions can be executed, in this example here the date of today would be computed in the formate „DD.MM.YYYY”:

  <atext>("0" + new Date(new Date().getTime()).getDate()).slice(-2) + "." + ("0" + (new Date(new Date().getTime()).getMonth() + 1)).slice(-2) + "." + new Date(new Date().getTime()).getFullYear()</atext>

- And you could even define own JavaScript functions you can define or include in the TEMPLATEHEAD:

  <atext>estimatedlifetime (ageofthedog)</atext>

But now it´s getting really complex. Actually, this is what we wanted to avoid. But if you absolutely want to code... ;-)

3.2.1.5 ATEXTFALSE

Like ATEXT, but used, when ACONDITION is “false”.

3.3 Replacements

3.3.1 Standard Replacements

Replacements are extremely useful an have been mentioned before.
Via replacements you can change the produced ATEXT output as well as AOPTLABEL output by inserting AID values by simply writing the AID name in curly braces, „{“ and „}“:

Example: My Dog is called {nameofthedog} and is {ageofthedog} years old!

These placeholders will be replaced in realtime as soon as the according values are entered or calculated via COMPUTE.

Important:

When using OMESSAGE actions, empty values will be show as the placeholders value with a background highlight. This way, you can see at a glance that this value has probably not been set yet.

If, however, you wish to hide an empty value completely, place a “+” at the end of the placeholder:

{nameofthedog+}

3.3.2 Intelligent Replacements

Intelligent Replacements will help you to create even more individualized Output. Intelligent Replacements are used like and in the same places as the Standard Replacements, but have the following structure:

{{*aid***|***Text to show if value of der AID is „true“***|***Text to show if “false”* **}}**

Example:

„Please deliver all goods to {gender|Mr.|Mrs.} {nameofcustomer} by tomorrow.“

If the value of AID „gender“ is set to „true“ (or 1) for male, „Mr.“, if “false”, „Mrs.“ will be shown.

This example is chosen for good reason: this way you can easily create complete documents, correspondence etc. in a gender specific way without using impersonal workarounds.

Tip: How to add more Options

If you like, you can add further output options.

{{aid|Text to show if value of der AID is „true“|Text to show if “false” |Text to show if “-1” |Text to show if “-2” |etc. }}

For any option more than “false” (or “0”) you count down the value of your AID, so you use 1 (or “true”), 0 (or “false”), -1, -2, -3, -4 and so on. There is no limit in the number of options that can be added.

Tip: Check Action-ID for string content

Would you like to check whether aid contains string content or not, just add a question mark behind the aid:

{{aid?|Text to show if string content exists|Text to show if string is empty}}

Tip:

You can also use a Standard Replacement within an Intelligent Replacement.

Attention:

The formerly used syntax using single instead of double brackets still works fine as long as you do not use Standard Replacements within the Intelligent Replacement and you only use the options for “true” and “false”.

3.3.3 Automatic Counters

Automatic Counters help to enumerate Sections and Chapters of your output text dynamically, when OMESSAGE actions are hidden.

You can use

- **[inc]** for the main sections,
- **[incs]** for any enumeration within each main section.

The [inc] counter will increase by 1 with every usage, as long as the OMESSAGE action is set to display.

The [incs] counter will increase by 1 with every usage, until a new [inc] counter is called.

Just place [inc] or [incs] anywhere in your OMESSAGE action and it will be replaced automatically.

Both [inc] and [incs] work across the borders of each OMESSAGE action.

Attention:
Automatic Counters cannot be placed into intelligent replacements.

Referring to the Counters within Output Actions:

If you would like to refer to certain counters within Output Actions, adding the counter value to your text, e.g..

„As mentioned in article 23…“,

you can do this by adding a unique identifier in round brackets to [inc] [incs] or [incstart]:

[inc(*uniqueidentifier*)]

z.B. [inc(*contactconditions*)]. Within the output text you reference it by using “[**link-**inc(*contactconditions*)]” (bzw. link-incs, link-incstart), e.g.

„As mentioned in article **[link-inc(*constractconditions*)]**…“

3.4 Data Output via “Outputblock”

3.4.1 Output - Actions

3.4.1.1 Action: OMESSAGE

OMESSAGE is the standard output action in the OUTPUTBLOCK, the actual output area.

The output area consists of one or more OMESSAGE actions.

- An output should be splitted in several OMESSAGE actions, if these actions shall be shown depending on differen ACONDITION results.

- You can wrap up one or more OMESSAGE actions in a block using the STARTBLOCK and ENDBLOCK action. This is useful e.g. for processing the block with the CLIPBOARD action, described further down.

3.4.1.1.1 AID

see above

3.4.1.1.2 ATYPE

Value: omessage

3.4.1.1.3 ACONDITION

see above

3.4.1.1.4 ATEXT

This is the output text to be presented in the OUTPUTBLOCK.

There is one speciality concerning the ATEXT element within OMESSAGE, which will make life much easier:

Whereas ususally the ATEXT elements may not contain linefeeds, here linefeeds are allowed, as long as they are placed between „<![CDATA“ und „]]>“.

3.4.1.1.5 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.4.1.2 Action: STARTBLOCK

With STARTBLOCK, you can wrap up one or more OMESSAGE actions into one block. This way, you can assign a Caption to this block and you can copy its contents into the clipboard using the CLIPBOARD action.

STARTBLOCK defines the **beginning** oft he block.

Attention: STARTBLOCK has – and needs - no ACONDITION!

3.4.1.2.1 AID

see above

3.4.1.2.2 ATYPE

Value: startblock

3.4.1.2.3 ATEXT

This text is used as the Caption for this block.

3.4.1.2.4 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.4.1.3 Action: ENDBLOCK

The ENDBLOCK action defines the end of the block started with the STARTBLOCK action.

Attention: ENDBLOCK hat has – and needs – neither an ACONDITION nor an ATEXT!

3.4.1.3.1 AID

see above

3.4.1.3.2 ATYPE

Value: endblock

3.4.2 Special Functions

3.4.2.1 Action: CLIPBOARD

The CLIPBOARD action creates a Button to copy a whole block wrapped up by STARTBLOCK and ENDBLOCK into the clipboard.

3.4.2.1.1 AID

see above

3.4.2.1.2 ATYPE

Value: clipboard

3.4.2.1.3 ACONDITION

see above

3.4.2.1.4 ATEXT

This is the label of this button.

3.4.2.1.5 ABLOCKID

Defines, **which block** will be copied to the clipboard.

Use the AID assigned to the STARTBLOCK.

3.4.2.1.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.4.2.2 Action: WORDSAVE

The WORDSAVE action creates a Button to save a whole block wrapped up by STARTBLOCK and ENDBLOCK as a Microsoft Word DOCX-File.

Please be aware that WORDSAVE will only use the font “Calibri” or as subsidiary choice “Arial” and will not work with picture files unless they are directly base64-embedded into the HTML output file.

Compatibility Notice:
WORDSAVE generated DOCX files need a full “desktop” version of Microsoft Word to produce correct results, they are not compatible to the tablet or smartphone editions of Word.

Also, due to system restrictions, Word output files cannot be produced on the iOS Safari web browser and those based on the same browser engine. On iOS devices the WORDSAVE button therefore will be suppressed, if the browser identifies itself correctly.

3.4.2.2.1 AID

see above

3.4.2.2.2 ATYPE

Value: clipboard

3.4.2.2.3 ACONDITION

see above

3.4.2.2.4 ATEXT

This is the label of this button.

3.4.2.2.5 ABLOCKID

Defines, **which block** will be saved.

Use the AID assigned to the STARTBLOCK.

3.4.2.2.6 AFILENAME

Defines, under which filename the block will be saved. Enter the filename without an extension. Any added extension will be ignored.

3.4.2.2.7 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.4.2.3 Action: PRINT

The PRINT action creates a Button to print the whole block wrapped up by STARTBLOCK and ENDBLOCK.

3.4.2.3.1 AID

see above

3.4.2.3.2 ATYPE

Value: print

3.4.2.3.3 ACONDITION

see above

3.4.2.3.4 ATEXT

This is the label of this button.

3.4.2.3.5 ABLOCKID

Defines, **which Block** will be printed.

Use the AID assigned to the STARTBLOCK.

3.4.2.3.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

3.4.2.4 Action SHOWINPUTBLOCK

Action SHOWINPUTBLOCK displays a button which will toggle the INPUTBLOCK between visible and invisible

3.4.2.4.1 AID

See above

3.4.2.4.2 ATYPE

Value: showomessages

3.4.2.4.3 ACONDITION

See above

3.4.2.4.4 ATEXT

This is the label of this button.

3.4.2.4.5 ATOGGLEINPUTOUTPUT

If the value is set to “true”, together with showing the INPUTBLOCK, the OUTPUTBLOCK will be hidden.

3.4.2.4.6 ASTYLE

Here you can give your action its own style-information, using CSS style syntax. So, e.g., if you want your text to be red with a size of 15pt, you would enter:

<astyle>color:red; font-size:15pt</astyle>

4 CONSULTIMATOR Data Encryption

In general, CONSULTIMATOR does not pass on any data. The complete process of creating outputs from user inputs is performed locally inside the users web browser without using an internet connection.

Nevertheless, CONSULTIMATOR enables you to pass on user data for additional processing. If no encryption is used, data will be simply passed on via standard form submit to the target defined in the project settings. For data security “on transit” you can send data via SSL by using the HTTPS protocol. One arrived at the target site, data are no longer encrypted, though.

If you would like to receive completely encrypted data, you can define your personal RSA Public Key within the project settings, value “rsapublickey”.

4.1 Location of encryption process and browser compatibility

CONSULTIMATOR encrypts user data following the schema described below locally inside the web browser.

You can use the web browsers Edge, Safari, Firefox, und Chrome in their up-to-date versions. Internet Explorer is not supported, though.

4.2 How to create a RSA Public Key Private Key Pair

There are plenty of instructions of how to generate RSA key pairs on the internet. In addition, the process is also explained in the CONSULTIMATOR user manual. For optimal data security, please create a 2048-bit key pair or higher. CONSULTIMATOR will support up to 4096-bit key pairs.

4.3 Encryption method

Encryption is asymmetric, meaning data will be encrypted with your public key placed inside your output page, but can only be decrypted with your private key, which should always be at your safe hands.

Because of the amount of data to be encrypted a fully asymmetric encryption is not an option. Therefore, encryption works as follows:

1. A random password of 50 characters length will be created.

2. This will be used to create a key for a symmetric data encryption following the standard SHA-256.

3. Data will now be encrypted using the standard AES-CBC. Encryption uses the web browser-internal, well-established and maintained encryption libraries.

4. Then, the key itself will be encrypted with your RSA Public Key following the standard RSA-256. Encryption will used the well-established library „JSEncrypt“ (https://github.com/travist/jsencrypt), which uses the well-established library „jsbn“ (http://www-cs-students.stanford.edu/~tjw/jsbn/).

5. Even if actually not necessary, the so called “vector” needed for decrypting the data is also encrypted with the RSA Public Key using the RSA-256 standard.

6. The encrypted key, the vector and the data will be concatenated to one string in this way:

CMCENCRAW-V1.0:::::key::::vector:::data

4.4 Decryption

Decryption starts by splitting up the string back into key, vector and data. Then key and vector are decrypted via the RSA Private Key. Both are then used for decrypting the data via AES-CBC.

CONSULTIMATOR output pages already contain all routines necessary for decrypting previously encrypted data, but, of course not, your RSA Private Key. If the output page recognizes encrypted content, you will be prompted to enter your private key. The private key will only be used locally inside the browser to decrypt the data and will not be saved on your computer. As an option the private key can be saved locally as a cookie, if a password is set for encryption an decryption.

4.5 encodeURIComponent / decodeURIComponent

To prevent errors during data encryption and decryption, binary data are encoded and decoded by encodeURIComponent / decodeURIComponent.

4.6 No proprietary changes

The encryption does not use any proprietary changes to the standards described above, so decryption can be ported to other program languages as long as they support these standards.

5 Prefilling CONSULTIMATOR Output Pages automatically

The input fields of CONSULTIMATOR output pages are simple form field and can be prefilled with content on loading the page. As CONSULTIMATOR output pages are static HTML, as long as they are not manually modified, population of content is normally only possible via URL parameters by adding the field names and values to the URL as follows:

….myoutputpage.html?inputvalue=123

This method is limited, as URLs cannot be entered into browsers at any length. Also, this method of data transfer is not very safe.

This problem can be circumvented the following way:

CONSULTIMATOR output pages contain a Javascript-routine that will check on load, whether the browser LocalStorage ore SessionStorage contains content in this variable:

tdc_parameterstring

All content can be deposited there just like with URL transfer, e.g.

Inputvalue1=123&inputvalue2=456&inputvalue3=789

To load data into the variable, there´s a simple trick: just call a routine, that will first load data into the variable and then redirect automatically to the output page.

In PHP this could look as follows, presuming, $content will contain the parameter string mentioned above and $form contains the URL of the output page.

```
<html>
<head>
<script>
 var content = "<?php echo($content);?>";
 var my_form = "<?php echo($form);?>";

 localStorage.removeItem('tdc_parameterstring');
 sessionStorage.removeItem('tdc_parameterstring');

 sessionStorage.setItem("tdc_parameterstring", content);
 window.location = "https://consultimator.com" + my_form;
</script>
```

```
</head>
<body>
</body>
```

Please be aware that this will only work, if both pages , the dynamic page, filling the variable tdc_parameterstring, and the output page exist on the same server.

CONSULTIMATOR offers you a dynamic page to populate your forms via

https://consultimator.com/populate.php

free of chare, but without any liability as a service. Please add your output page to the URL as follows:

?form=*Link to own output page*

Die actual form data are passed on via FORM POST. Please be aware, that this service is only meant for evaluation purposes, as CONSULTIMATOR can read all data transferred this way! For productive purposes please host both the population routine and your output pages on your own server.

Please ask us, if you need the source code for „populate.php“, we will send it to you free of charge for individual use and adaption.

6 CONSULTIMATOR in Action

Once you have created your project file, tell CONSULTIMATOR to process it and create the output file.

> **Important:**
> To use the CONSULTIMATOR you need a personal license key. This key is entered into the project file. Without the license key you can use CONSULTIMATOR only in demo mode and only for evaluation purposes.

You can process your project file by visiting this URL:

https://consultimator.com

There, you can start the CONSULTIMATOR Editor and upload your project file.

Alternatively, if you have uploaded the project file to any webserver and it can be accessed via URL, you can invoke CONSULTIMATOR directly:

https://consultimator.com/process.php?t=https://anyserver.com/anypath/myprojectfile.cmp (or: .xml)

Be aware: if you use this method, your license key will be publicly viewable, if anyone finds the project file xml. Instead, leave the license key value in your project file empty and add another parameter to the URL above: "&s=*yourlicensekey*"

If the project file is faulty, e.g. because of a wrong XML structure, syntax errors in element descriptions or incorrectly defined "forbidden" characters, you will get an error message, usually combined with quite useful hints where to look for the problem.

If the project file has a correct XML structure, CONSULTIMATOR will generate the HTML output.

Important:
CONSULTIMATOR is very tolerant concerning creating the output file, meaning it will create it even if there might be JavaScript or logical errors within ACONDITIONS or COMPUTE actions used, incorrect HTML tags etc. These can prevent the output file to be presented the way you expected. To avoid frustration, it is recommended to proceed as follows:

- **Use, if possible, the CONSULTIMATOR Editor!** The Editor will guide you through the process of building the project file. This significantly reduces possible error sources!
- Build your project file step by step and repeatedly process it through CONSULTIMATOR to get interim results. This way you will see possible errors

immediately and you can remove them before they mix up with other problems, what would make debugging far more difficult.

- If you know how they work, please make use of the developer tools, especially the debug console included in your web browser. They give you very valuable information concerning possible errors. Especially faulty ACONDITIONS and COMPUTE actions will cause JavaScript errors which will show up immediately in the browser debugger tool.
- Often, errors have very human causes. Especially logical errors cause unexpected reactions: mistakes in “bigger than” / “smaller than” conditions happen quite often, or you used the COMPUTE-Action without technical errors, but your dog is now, like in the example above, 123 instead of 15 years old.

  In this case it can be useful for debugging to use the MESSAGE action to let CONSULTIMATOR present you COMPUTE action results. If it doesn´t show the value you had expected, this brings you a big step further.