



CONSULTIMATOR

v 1.1.11.01

<https://consultimator.com>

Technische Referenz



Inhaltsverzeichnis

1	Einleitung	4
2	Die Projektdatei	5
2.1	Grundsätzliches	5
2.2	Das XML-Format der Projektdatei	5
2.3	Bereich „GENERAL“	6
2.3.1	Allgemeine Angaben	6
2.3.1.1	KEY	6
2.3.1.2	RELEASE	7
2.3.1.3	TEMPLATENAME	7
2.3.1.4	TEMPLATEHEAD	7
2.3.1.5	JAVASCRIPTBLOCK	8
2.3.1.6	LINKSTYLESHEET	8
2.3.1.7	TEMPLATETITLE	9
2.3.1.8	TEMPLATEFOOT	9
2.3.1.9	TEMPLATELANGUAGE	9
2.3.1.10	STORAGETYPE	9
2.3.1.11	COMPONENTPATH	10
2.3.1.12	SHOWOMESSAGESDEFAULT	10
2.3.1.13	PLACEHOLDERUNDERLINE	10
2.3.1.14	PLACEHOLDERCHAR	11
2.3.1.15	EDITABLEOMESSAGES	11
2.3.1.16	XSSSTRICT	11
2.3.2	Formular-Aktion	11
2.3.2.1	FORMACTION	12
2.3.2.2	FORMMETHOD	14
2.3.2.3	FORMTARGET	15
2.3.2.4	FORMENCTYPE	15
2.3.2.5	RSAPUBLICKEY	15
2.4	Bereich „ACTIONS“	16
3	Die Aktionen im Einzelnen	17
3.1	Datenerfassung im „Inputblock“	17
3.1.1	Eingabe-Aktionen	17
3.1.1.1	Aktion: TEXT	17
3.1.1.2	Aktion: TEXTAREA	19
3.1.1.3	Aktion: DATE	20
3.1.1.4	Aktion: MONTH	21
3.1.1.5	Aktion: TIME	22
3.1.1.6	Aktion: CHECKBOX	23
3.1.1.7	Aktion: COLOR	24
3.1.1.8	Aktion: MESSAGE	24
3.1.1.9	Aktion: BLINDS	25
3.1.1.10	Aktion: SELECT	27
3.1.1.11	Aktion: OPTION	28
3.1.2	Sonderfunktionen	29
3.1.2.1	Aktion: SHOWOMESSAGES	29



3.1.2.2	Aktion: SUBMIT	30
3.1.2.3	Aktion: FORMCLIPBOARD	30
3.1.2.4	Aktion: SUBMITSECURE	31
3.1.2.5	Aktion: PARAMETERS.....	32
3.1.2.6	Aktion: RESET	33
3.1.2.7	Aktion: FORMDOWNLOAD	33
3.1.2.8	Aktion: FORMDOWNLOADCSV	34
3.1.2.9	Aktion: FORMDOWNLOADSECURE	35
3.1.2.10	Aktion: PARAMETERUPLOAD	36
3.1.2.11	Aktion: ATTACHMENT	36
3.1.3	Generelle Aktionen	37
3.1.3.1	Aktion: STYLE	37
3.2	Berechnungen	37
3.2.1	Aktion: COMPUTE.....	37
3.2.1.1	AID	38
3.2.1.2	ATYPE	38
3.2.1.3	ACONDITION	38
3.2.1.4	ATEXT	38
3.2.1.5	ATEXTFALSE	39
3.3	Ersetzungen	39
3.3.1	Standard-Ersetzungen	39
3.3.2	Intelligente Ersetzungen.....	40
3.3.3	Automatische Zähler	41
3.4	Datenausgabe im "Outputblock"	42
3.4.1	Ausgabe-Aktionen	42
3.4.1.1	Aktion: OMESSAGE	42
3.4.1.2	Aktion: STARTBLOCK	43
3.4.1.3	Aktion: ENDBLOCK	44
3.4.2	Sonderfunktionen.....	44
3.4.2.1	Aktion: CLIPBOARD	44
3.4.2.2	Aktion: WORDSAVE.....	45
3.4.2.3	Aktion: PRINT	46
3.4.2.4	Aktion SHOWINPUTBLOCK.....	46
4	CONSULTIMATOR Datenverschlüsselung	48
4.1	Ort der Verschlüsselung und Browserkompatibilität.....	48
4.2	Erstellung eines RSA Public Key Private Key Schlüsselpaars.....	48
4.3	Verschlüsselungsmethode	48
4.4	Entschlüsselung.....	49
4.5	encodeURIComponent / decodeURIComponent.....	49
4.6	Keine proprietären Ergänzungen.....	49
5	Daten automatisiert in CONSULTIMATOR Ausgabeseiten laden	51
6	Der CONSULTIMATOR in Aktion.....	53



1 Einleitung

Herzliche willkommen zu CONSULTIMATOR! Mit CONSULTIMATOR erstellen Sie Dialogseiten, kleine „Expertensysteme“, automatisch generierte Musterverträge, Berechnungstools und Anschreiben.

Sie erstellen einen Fragenkatalog sowie die in Abhängigkeit von den Antworten anzuzeigenden Informationen. Dabei brauchen Sie – abhängig vom Grad der Komplexität der Aufgabe – keine oder nur sehr geringe Programmierkenntnisse.

Mit grundlegenden HTML- und Javascript-Kenntnissen erwecken Sie CONSULTIMATOR zu ungeahnter Kraft. Komplexen Programmcode brauchen Sie dafür nicht zu erstellen, diese Aufgabe übernimmt CONSULTIMATOR für Sie.

Der Ablauf ist einfach:

- Sie erstellen, so wie auf den nächsten Seiten beschrieben, eine Projektdatei, die die Abfolge der zu stellenden Fragen sowie die davon abhängig anzuzeigenden Inhalte beschreibt.
- Anschließend erteilen Sie CONSULTIMATOR den Auftrag, aus dieser Projektdatei die fertige HTML-Seite zu erzeugen.
- Diese können Sie selbst mittels CSS frei gestalten oder Sie verwenden einfach die Standard-CSS-Einstellungen, die CONSULTIMATOR bereits mitbringt.

Demo-Modus

Neugierig geworden? Dann probieren Sie den CONSULTIMATOR doch einmal aus! Sie können im Demo-Modus kleine Dialoge mit maximal 20 Aktionen auch ohne einen Lizenz-Key zu Evaluationszwecken erstellen. Aktionen zur sicheren Datenverschlüsselung stehen im Demo-Modus nicht zur Verfügung und werden bei der Erstellung der Ausgabeseiten ignoriert.

Benötigen Sie einen Lizenz-Key?

Sprechen Sie mich einfach an, meine Kontaktdaten finden Sie auf der Webseite.



2 Die Projektdatei

2.1 Grundsätzliches

Die Projektdatei ist Ihre zentrale Konfigurationsdatei für Ihr Projekt. In der Projektdatei definieren Sie neben diversen Grundinformationen insbesondere alle Aktionen, die CONSULTIMATOR durchführen soll.

Dabei programmieren Sie regelmäßig nicht, sondern setzen lediglich Werte. Und an den Stellen, an denen Sie programmieren, merken Sie es meist gar nicht. ;-)

2.2 Das XML-Format der Projektdatei

Die Projektdatei wird im Format „XML“ erstellt. Der CONSULTIMATOR Editor wird sie mit der Dateiendung „.cmp“ abspeichern, aber es ist immer noch eine XML-Datei.

XML steht für „Extended Markup Language“, was Ihnen aber grundsätzlich gleich sein kann. Hier nur für die Allgemeinbildung. ;-)

Was Sie aber über XML-Dateien wissen sollten, zumindest soweit wir diese hier verwenden, ist folgendes:

- Die einzelnen Elemente in der Projektdatei haben einen Namen, und der steht zwischen einem Kleiner- und einem Größer-Zeichen. Also z.B. so:

`„<elementname>“`

- Das Ende des Elements wird so ähnlich gekennzeichnet, nur findet sich zwischen dem Kleinerzeichen und dem Namen des Elements noch ein `„/“`, also z.B. so:

`</elementname>`

- Der eigentliche Wert, der dem Element zugewiesen wird, steht dazwischen, also z.B. so:

`<elementname>Dies ist der Wert des Elements!</elementname>`

- Schwierigkeiten machen bei der Eingabe der Werte das die Zeichen `„&“`, `„<“` und `„>“`. Diese werden wie folgt eingegeben:

- Aus `„&“` wird `„&“`
- Aus `„<“` wird `„<“`
- Aus `„>“` wird `„>“`

also z.B.:

`<elementname>Die GmbH & Co. KG wurde gestern gegründet.</elementname>`

- Es gibt noch eine alternative Schreibweise, die gerade bei sehr langen, mehrzeiligen Texten oder auch Texten mit vielen dieser „verbotenen“ Zeichen sehr



praktisch ist:

```
<elementname><![CDATA[Die GmbH & Co. KG wurde gestern gegründet.]]></elementname>
```

Diese Schreibweise werden Sie spätestens dann schätzen, wenn es darum geht, lange, mehrzeilige Textpassagen mit HTML-Auszeichnungen einzugeben, Denn zwischen `<![CDATA[` und `]]>` kann beliebiger Text eingegeben werden.

Aber Vorsicht:

Mehrzeilige (!) Eingaben, also solche, bei denen ein Zeilenumbruch im Text verwendet wird, sind im CONSULTIMATOR **ausschließlich im Aktionstyp OMESSAGE** zulässig! Das heißt nicht, dass andere Aktionen keine Zeilenumbrüche erzeugende HTML-Tags wie „`
`“ oder „`<p></p>`“ enthalten dürfen, nur dürfen Sie in der Projektdatei keine Zeilenschaltungen enthalten.

- Die einzelnen Elemente einer XML-Datei können ineinander geschachtelt werden. Also z.B.

```
<aktionen>
  <aktion1>
    <unteraktion1></unteraktion1>
    <unteraktion1></unteraktion1>
  </aktion1>
  <aktion2></aktion2>
  <aktion3></aktion3>
</aktionen>
```

aber niemals so:

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

2.3 Bereich „GENERAL“

2.3.1 Allgemeine Angaben

2.3.1.1 KEY

Hier setzen Sie Ihren persönlichen Lizenzschlüssel ein. Ohne Lizenzschlüssel können Sie CONSULTIMATOR Ausgabeseiten nicht auf consultimator.com veröffentlichen. Sie können diese allerdings – im Rahmen der Lizenzbedingungen – herunterladen und lokal oder auf Ihrem eigenen Webserver hosten.



Achtung - LösCHFunktion:

Ergänzen Sie den KEY am Ende um „-kill“, so wird, wenn RELEASE auf „true“ gesetzt ist, die Ausgabedatei nicht erstellt, sondern **gelöscht!**

Projektsignatur statt Lizenzschlüssel:

Wenn Sie Ihre Projektdatei an Dritte weitergeben wollen, ohne Ihren persönlichen Lizenzschlüssel aus der Hand zu geben, können Sie statt des Lizenzschlüssels hier auch eine Projektsignatur angeben.

Sie können die Projektsignatur erstellen, indem Sie entweder im CONSULTIMATOR Editor die entsprechende Funktion nutzen, die Sie beim Lizenzschlüssel-Eingabefeld finden, oder Sie rufen den CONSULTIMATOR Processor per URL mit dem zusätzlichen Parameter “signproject=true” auf.

Tragen Sie dann einfach die Projektsignatur als KEY ein, so wie Sie es sonst mit Ihrem persönlichen Lizenzschlüssel machen würden.

Bitte beachten Sie, dass dies bewusst keine vollwertige Alternative zur Nutzung des persönlichen Lizenzschlüssels ist: Sobald das Projekt mittels der Projektsignatur anstelle des Lizenzschlüssels signiert ist, ist eine Änderung der Reihenfolge der Aktionen, das Hinzufügen und Löschen von Aktionen, das Ändern der Aktions-IDs sowie das Ändern des Aktionstyps nicht mehr möglich, ohne dass dabei die Projektsignatur Ihre Gültigkeit verliert.

2.3.1.2 RELEASE

Wenn Sie RELEASE auf „true“ setzen, wird die Ausgabedatei mit dem unter TEMPLATENAME angegebenen Namen in Ihrem individuellen Ausgabeordner abgelegt. Diese Option wird nur berücksichtigt, wenn Sie über einen individuellen Lizenzschlüssel verfügen.

2.3.1.3 TEMPLATENAME

Hier setzen Sie den Namen der zu erzeugenden Datei ein, also z.B. „ergebnis“. Geben Sie keine Dateiendung an. Es wird stets eine Datei mit der Endung „.html“ erzeugt.

2.3.1.4 TEMPLATEHEAD

CONSULTIMATOR erzeugt den kompletten HTML-Code, der zum Betrieb erforderlich ist. Es liegt an Ihnen, wie dieser letztlich eingesetzt wird, also ob er in eine andere Webseite eingebettet wird oder die Seite eigenständig lauffähig sein soll. Möchten Sie eine HTML-Datei erzeugen, so können Sie alle für die HTML-Datei sinnvollen Kopfdaten hier angeben. Das sieht dann z.B. so aus:



```
<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>Der Titel meiner Seite</title>
</head>
<body>
]]></templatehead>
```

Wie Sie sehen, kann hier z.B. auch eine CSS-Datei angegeben werden, über die Sie praktisch alle Aspekte der Darstellung der Seite steuern können.

Sie können hier auch z.B. individuelle Javascript-Dateien mit einbinden, mit denen der Funktionsumfang des CONSULTIMATOR erweitert werden kann. Mehr dazu später.

Wenn Sie einen Standard-HTML-Kopf und -Fuß verwenden möchten, ohne diesen selbst definieren zu müssen, geben Sie bitte „[html]“ als TEMPLATEHEAD an:

```
<templatehead>[html]</templatehead>
```

(Hinter „[html]“ können Sie weiteren HTML-Code angeben, der hinter dem Tag „<body>“ und vor der eigentlichen CONSULTIMATOR-Ausgabe erscheinen soll.)

Den TEMPLATEFOOT können Sie dann leer lassen, er wird automatisch erzeugt. Allerdings sollten Sie dann noch den TEMPLATETITLE definieren, damit Ihre Seite im Browser auch einen Titel anzeigt.

2.3.1.5 JAVASCRIPTBLOCK

Über JAVASCRIPTBLOCK kann individueller JavaScript-Code angegeben werden, der an den Anfang der JavaScript-Ausgabedatei gestellt wird.

Es wird der reine JavaScript-Code ohne „<script>“ / „</script>“ o.ä. angegeben.

2.3.1.6 LINKSTYLESHEET

Über LINKSTYLESHEET kann ein vom Standard Stylesheet-Link, der verwendet wird, wenn als HTML Header „[html]“ angegeben wird, abweichender Link angegeben werden.

Dieser muss in der Form



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

vollständig angegeben werden.

Ausnahme: CSS-Dateien, die unter <https://consultimator.com/cssoptions/> gespeichert sind, können einfach durch Angabe des Dateinamen verlinkt werden.

Bleibt das Feld leer, wird der Defaultwert verwendet.

2.3.1.7 *TEMPLATETITLE*

Der über das Element "Templatetitle" angegebene Wert wird im TEMPLATEHEAD der hier angegebene Wert als Seitentitel eingefügt, also z.B.:

```
<templatetitle>Dies ist der Titel Ihrer Seite</templatetitle>
```

Diese Angabe wird ignoriert, wenn Sie als TEMPLATEHEAD etwas anderes als „[html]“ angegeben haben.

2.3.1.8 *TEMPLATEFOOT*

Der über das Element "Templatefoot" angegebene Wert wird am Ende der Datei eingefügt. Eine HTML-Datei schließen Sie dabei z.B. wie folgt ab:

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

Wenn im TEMPLATEHEAD „[html]“ angegeben wurde, wird hinter dem hier angegebenen HTML-Code noch „</body></html>“ ausgegeben.

2.3.1.9 *TEMPLATELANGUAGE*

Mittels "Templatelanguage" kann angegeben werden, ob bestimmte Dialoge auf Deutsch oder auf Englisch angezeigt werden:

Englische Dialoge:

```
<templatelanguage>en</ templatelanguage >
```

Deutsche Dialoge (Standard, kann auch weggelassen werden):

```
< templatelanguage >de</ templatelanguage >
```

2.3.1.10 *STORAGETYPE*

Über den Wert "storagetype" wird gesteuert, ob eingegebene Daten lokal im Webbrowser im „LocalStorage“, im „SessionStorage“ oder überhaupt nicht gespeichert werden.



LocalStorage (Daten bleiben erhalten, bis sie im Browser gelöscht werden):

```
<storagetype>local</storagetype>
```

SessionStorage (Daten bleiben bis zum Ende der Session erhalten, z.B. bis der Tab geschlossen wird):

```
<storagetype>session</storagetype >
```

Keine Speicherung (Default, kann auch weggelassen werden):

```
<storagetype></storagetype>
```

2.3.1.11 COMPONENTPATH

Wenn Sie die Ausgabedatei nicht an ihrem vom System normalerweise gespeicherten Platz betreiben möchten, können Sie hier den Pfad zum "components"-Verzeichnis angeben.

2.3.1.12 SHOWOMESSAGESDEFAULT

Der CONSULTIMATOR unterteilt die anzuzeigenden Inhalte in den Block "INPUTBLOCK" – für alle Eingabeaktionen – und "OUTPUTBLOCK" – für die Ausgabeaktionen. Dabei werden die Inhalte des OUTPUTBLOCK optional beim Laden der Seite angezeigt oder ausgeblendet.

Welche von beiden Varianten Sie benutzen, können Sie hier wählen:

- Verwenden Sie „true“ (also WAHR), dann werden alle Ausgaben im OUTPUTBLOCK sofort angezeigt.
- Verwenden Sie „false“ (also FALSCH), dann wird der OUTPUTBLOCK zunächst ausgeblendet. In dem Fall sollten Sie nicht vergessen, später die Aktion „SHOWOMESSAGES“ mit einzubinden, sonst fehlt Ihnen der Button, mit dem die Ausgaben per Klick sichtbar gemacht werden können.

2.3.1.13 PLACEHOLDERUNDERLINE

CONSULTIMATOR zeigt normalerweise die im Ausgabebereich einzusetzenden Platzhalter, solange sie nicht mit Werten gefüllt sind, farblich hinterlegt mit ihrer Aktions-ID an. Wenn Sie die Aktions-ID nicht anzeigen, sondern stattdessen lediglich „_____“ anzeigen möchten, können Sie diese Option hier aktivieren.

Welche von beiden Varianten Sie benutzen, können Sie hier wählen:

- Verwenden Sie „true“ (also WAHR), dann wird lediglich „_____“ angezeigt.
- Verwenden Sie „false“ (also FALSCH), dann wird die Aktions-ID angezeigt.



2.3.1.14 PLACEHOLDERCHAR

Wenn Sie ein anderes Zeichen als “_” für die PLACEHOLDERUNDERLINE-Funktion nutzen möchten, können sie dieses (oder mehrere) hier angeben. Sie werden 7 Mal wiederholt. Sinnvoll kann z.B. die Angabe von „ “ sein, wenn Sie einfach nur ein festes Leerzeichen anzeigen wollen.

2.3.1.15 EDITABLEOMESSAGES

Wenn EDITABLEOMESSAGES auf “true” gesetzt wird, kann der Nutzer alle generierten Ausgabeteixe direkt im Webbrowser bearbeiten.

Hinweis:

Wenn der Nutzer auf der Eingabeseite, also im INPUTBLOCK, Änderungen vornimmt, werden die manuellen Anpassungen im Ausgabebereich wieder verworfen.

Achtung:

Nutzen Sie diese Funktion nicht, wenn es Ihnen wichtig ist, später aus den gespeicherten Eingabewerten wieder das gleiche Ausgabeergebnis zu erzeugen. Manuelle Veränderungen am Ausgabeteixt werden nämlich nur auf der Ausgabeseite verarbeitet und nicht als Formulardaten mit abgespeichern.

2.3.1.16 XSSSTRICT

Wenn XSSSTRICT auf “true” gesetzt wird, wird im automatisch generierten HTML-Header die Cross-Site-Scripting Policy so eingerichtet, dass auch Inline-CSS unterbunden wird. Inline-JavaScript wird in jedem Fall unterbunden. Sowohl JavaScript- als auch CSS-Dateien dürfen ausschließlich vom CONSULTIMATOR-Server (oder bei Eigen-Hosting vom eigenen Server) kommen.

Hinweis:

Haben Sie innerhalb Ihrer Aktionen dennoch Inline-CSS-Definitionen getroffen (style=“...“), dann wird der Webbrowser diese als fehlerhaft beanstanden und nicht ausführen. Um zumindest die Fehlermeldungen zu unterbinden, wird CONSULTIMATOR versuchen, möglichst alle Vorkommnisse von **style=“...“** zu erkennen und aus den Aktionen zu beseitigen.

Wenn Sie XSSSTRICT aus Sicherheitsgründen auf „true“ setzen, aber nicht auf die Gestaltung der Ausgaben verzichten wollen, verwenden Sie bitte die Aktion „STYLE“, um zur Formatierung der Ausgaben eigene Klassen zu definieren und diese dann per **class=“...“** zuzuweisen.

2.3.2 Formular-Aktion

Die im INPUTBLOCK eingegebenen Werte können nicht nur für Ausgaben im OUTPUTBLOCK verwendet, sondern auch z.B. an eine weitere Webseite übergeben



werden. Die Daten werden dafür in einem Formular (HTML <form>) gesammelt. Wie sie weiterverarbeitet werden, hängt von den nachfolgenden Angaben ab:

2.3.2.1 FORMACTION

Eine typische Aktion wäre z.B.

```
<formaction>naechsteseite.html</formaction>
```

In dem Fall würde die Daten an die HTML-Seite „naechsteseite.html“ übergeben werden und können dort weiter verarbeitet werden.

Es ist aber z.B. auch möglich, eine E-Mail mit den Daten zu generieren:

```
<formaction>mailto:meineemailadresse@email.de</formaction>
```

Sonderfall: Versendemechanismus des „CONSULTIMATOR Sicheres Kontaktformular“

Nutzen Sie die Versenden-Funktion des sicheren Kontaktformulars, geben Sie hier bitte folgende Daten ein:

```
secureformtransmit.php | meine@emailadresse.de | Neue Nachricht erhalten
```

oder

```
secureformtransmit_toserver.php | meine@emailadresse.de | Neue Nachricht erhalten
```

Tauschen Sie „*meine@emailadresse.de*“ durch Ihre Ziel-E-Mail-Adresse aus. „*Neue Nachricht erhalten*“ können Sie abwandeln, dieser Text wird (nebst anderer Informationen) im Betreff Ihrer E-Mail stehen.

Wenn Sie **secureformtransmit.php** wählen, wird die komplette Nachricht nach der Verschlüsselung an Ihre Benachrichtigungs-E-Mail angehängen.

Wählen Sie „**secureformtransmit_toserver.php**“, so bleibt die verschlüsselte Nachricht auf dem Server und kann nur über den per Benachrichtigungsmail übermittelten, kryptischen Link abgerufen werden. Der Abruf muss binnen 365 Tagen erfolgen. 90 Tage nach Abruf bzw. spätestens nach 365 Tagen wird die verschlüsselte Nachricht vom Server automatisch gelöscht.

Die Verschlüsselung wird an anderer Stelle ausführlich erläutert. Sie kann nur vom Inhaber des persönlichen Schlüssels, also nur von Ihnen und von Ihnen ins Vertrauen gezogenen Personen, entschlüsselt werden.

Bitte wählen Sie bei Verwendung dieser Versendemethode auf jeden Fall als FORMMETHOD „post“, s. folgender Abschnitt.



Tip: Einsatz von Processing Instructions

Sie können den „Betreff“ Ihrer Nachricht zusätzliche um Processing Instructions ergänzen. Diese dienen nachfolgenden Seiten oder auch der CONSULTIMATOR Workflow Engine ergänzende Arbeitsanweisungen.

Der die Processing Instructions enthaltende Block beginnt mit [pi] und endet mit [/pi]:

[pi] Liste der Processing Instructions **[/pi]**

Aktuell werden folgende Processing Instructions unterstützt:

Extended Parameters („[xp]“)

Mit der Processing Instruction [xp] können Sie secureformtransmit(_toserver).php anweisen, dem Link zum Aufruf der mit Daten zu füllenden Ausgabeseite weitere Parameter hinzu zu fügen. Sinnvoll kann z.B. der Parameter „pkencskip“ sein:

[xp]pkencskip=default[/xp]

Ist der Parameter „pkencskip“ gesetzt, wird der angegebene Wert, hier „default“, verwendet, um Ihren RSA Private Key, wenn er im Browser verschlüsselt gespeichert ist, mit dem angegebenen Wert (hier: „default“) zu entschlüsseln. Dadurch kann die zu entschlüsselnde Nachricht einfach per Klick auf den Link in der E-Mail ohne weitere Zwischenschritte entschlüsselt werden.

Von der Option sollte natürlich, da dadurch das Kennwort zur Entschlüsselung des Private Key offen gelegt wird, nur Gebrauch gemacht werden, wenn der Rechner mit dem zu entschlüsselnden Private Key selbst hinreichend gegen den Zugriff unberechtigter Dritter geschützt ist.

Es kann aber auch eine im Formular verwendeter Wert über seinen Aktionsnamen gesetzt werden, hier z.B. der Wert der Aktion „aufrufausmail“ mit dem Wert „ja“:

[xp]aufrufausmail=ja[/xp]

Mehrere Parameter werden mit „&“ getrennt zusammengefügt:

[xp]pkencskip=default&aufrufausmail=ja[/xp]

Alternative Return-URL („[altreturnurl]“)

Über die „altreturnurl“ weisen Sie secureformtransmit an, den Link in der E-Mail zum Öffnen der verschlüsselten Daten auf eine andere Seite weisen zu lassen als die Seite, mit der die Daten erfasst wurden:

[altreturnurl]/customers/kundenid/formularname.html[/altreturnurl]

So können z.B. über ein Formular Daten gesammelt werden und mit einem Klick in einem anderen Formular weiterverarbeitet werden.

Individuelle Prozessoren für die Workflow Engine („[pipnames]“)

Nutzen Sie die CONSULTIMATOR Workflow Engine und möchten sie diese anweisen, abhängig vom jeweiligen Formular über die grundlegenden Datenaufbereitungsaufgaben auch weitere Aufbereitungen oder Workflows auszulösen, können Sie die Engine um eigene Prozessoren erweitern. Die jeweils auszuführenden Prozessoren können Sie über die „pipnames“ benennen:

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

In diesem Beispiel würde die Workflow Engine ihre Prozessoren „name1“, „name2“ und „name3“ in dieser Reihenfolge ausführen, nachdem die Standard-Aufbereitungen, die in der Engine selbst konfiguriert werden, abgearbeitet wurden.

Die Prozessoren befinden sich in der Workflow Engine im Unterverzeichnis „settings“ und folgen folgender Namenskonvention:

pi_gewähltername_processor.php

Die CONSULTIMATOR Workflow Engine ist quelloffen und kann nach Bedarf individuell von Ihnen angepasst werden. Für das Erstellen eigener Prozessoren können Sie die im Ordner „settings“ bereits vorhandenen Prozessoren als Vorlage heranziehen. Die Entwicklung eigener Prozessoren und die Anpassung der Workflow Engine setzt jedoch Kenntnisse der Programmiersprache PHP voraus. Die CONSULTIMATOR Workflow Engine wird daher, sofern nicht anders ausdrücklich vereinbart und Anpassungen ausschließlich von Anbieterseite vorgenommen werden, ohne Gewährleistung und ohne Supportzusage zur Verfügung gestellt.

2.3.2.2 FORMMETHOD

Hier wird die Methode der Datenweitergabe angegeben. Hierfür gibt es zwei Optionen:

- `<formmethod>get</formmethod>`

Dabei werden die Daten bei Weitergabe an eine andere Webseite über die Adresszeile der Webseite (URL) im Klartext weitergegeben. Das ist nicht besonders sicher und nur bei unkritischen Daten halbwegs ok.

Außerdem wird es bei mehr als 3000 zu übergebenden Zeichen irgendwann kritisch, da der Browser dies nicht mehr verarbeiten kann. Bei den 3000 Zeichen sind zudem die Feldnamen mitzuzählen, die mit übergeben werden. Man kann

erfahrungsgemäß mit 3000 Zeichen aber schon ziemlich umfangreiche Eingabedaten sammeln, solange der Nutzer des Formulars keine sehr langen Texte eingeben soll.

- `<formmethod>post</formmethod>`

Die „post“ Methode ist da schon sicherer, denn die Daten werden nicht über die URL, sondern direkt an den Server übergeben.

Gelesen werden können diese dann aber nicht mehr von einer normalen HTML-Seite mit Javascript, da diese nicht an die an den Server übergebenen Daten herankommt. Da muss es schon z.B. PHP sein, denn mit PHP können die so übergebenen Daten ausgewertet werden.

Wird übrigens „post“ mit dem Aktionstyp „mailto:“ verwendet, werden die Werte an das lokal installierte E-Mail-Programm übergeben.

2.3.2.3 FORMTARGET

Formtarget gibt an, ob, wenn eine neue Webseite aufgerufen wird, das gleiche Brower-Fenster weiterverwendet wird, oder ein neues Fenster geöffnet wird. Bleibt es beim gleichen Fenster, bleibt das Element leer. Soll ein neues Fenster geöffnet werden, so wird hier Folgendes angegeben:

```
<formtarget>_blank</formtarget>
```

2.3.2.4 FORMENCTYPE

Hier wird angegeben, in welcher Form die Daten an die nächste Seite übergeben werden. Optionen sind:

- *application/x-www-form-urlencoded*: (Standardwert) Alle Zeichen werden vor dem Senden kodiert (Leerzeichen zu "+", Sonderzeichen zu ASCII HEX-Werten)
- *multipart/form-data*: keine Zeichenkodierung
- *text/plain*: Leerzeichen werden vor dem Senden zu "+" kodiert; Sonderzeichen bleiben so

Wird als Aktionstyp „mailto:“ verwendet, so führt „text/plain“ dazu, dass die Daten im Klartext übergeben. Allerdings wäre das nicht kompatibel zu der weiter unten erläuterten Aktion „PARAMETERS“. Dazu später mehr.

2.3.2.5 RSAPUBLICKEY

Hier definieren Sie Ihren RSA Public Key, der zur Verschlüsselung der Nutzerdaten bei Verwendung der Funktionen FORMDOWNLOADSECURE und SUBMITSECURE verwendet wird. Der Datenverschlüsselung ist weiter unten ein komplettes Kapitel gewidmet.

2.4 Bereich „ACTIONS“

Im Bereich „ACTIONS“ werden nacheinander sämtliche Aktionen definiert, die die Seite ausführen soll.

Aktionen lassen sich grob unterteilen in Eingabe-Aktionen, die sich später im „Inputblock“ wiederfinden, Berechnungen von zusätzlich erforderlichen Werten sowie der Datenanzeige im „Outputblock“.

Jede Aktion besteht aus mehreren Elementen, die ihrerseits definieren, wie sich die Aktion genau verhält.

Wesentliche Elemente der Actions:

- „AID“: Jede Aktion verfügt über das Element „AID“ (Action-ID). Diese muss
 - eindeutig sein,
 - darf nur Kleinbuchstaben und Zahlen enthalten,
 - darf keine Umlaute enthalten und
 - darf nicht mit einer Zahl beginnen.

Die „AID“ spielt für den CONSULTIMATOR eine ganz entscheidende Rolle! Über die AID kann im weiteren Verlauf jederzeit der aktuelle Eingabewert der Aktion z.B. in der „acondition“, per „COMPUTE“ sowie in der Ausgabe als einzusetzender Text abgefragt und weiterverarbeitet werden.

- Das Element „ATYPE“ gibt an, um welchen Aktions-Typ es sich handelt. Zu den einzelnen Typen gleich mehr.
- „ACONDITION“ gibt an, ob die Aktion ausgeführt wird oder nicht.
- „ATEXT“ gibt an, welcher Text bei der jeweiligen Aktion ausgegeben werden soll.

Das sind schon die wichtigsten Elemente. Darüber hinaus gibt es ein paar zusätzliche Elemente, wenn der jeweilige Aktions-Typ dies erforderlich macht. Diese werden bei den einzelnen Aktionen erklärt.



3 Die Aktionen im Einzelnen

3.1 Datenerfassung im „Inputblock“

3.1.1 Eingabe-Aktionen

3.1.1.1 *Aktion: TEXT*

Hier wird über ein einfaches Texteingabe-Feld ein Text entgegengenommen. Der einzugebende Text ist einzeilig. Die Größe des Textfelds kann über ATEXTLENGTH oder ATEXTWIDTHPERCENT beeinflusst werden.

3.1.1.1.1 AID

Eindeutige ID nach den oben geschilderten Regeln, also Kleinschreibung, nur Buchstaben A-Z, keine Umlaute, Zahlen sind ok, aber nicht als erstes Zeichen.

3.1.1.1.2 ATYPE

Einzusetzender Wert: text

3.1.1.1.3 ACONDITION

Der Inhalt dieses Felds wird logisch darauf geprüft, ob als Ergebnis „true“ (also WAHR) herauskommt.

Es handelt sich hierbei um einen Vergleich, der in der Form einzugeben ist, wie dies in Javascript üblich ist. Eine hervorragende Übersicht über die zu verwendende Schreibweise findet sich hier:

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

Wichtig:

Sie können die AIDs aller Eingabeaktionen als Werte abfragen, und in den Vergleich mit einbeziehen! Dabei sind alle zurückgegebenen Werte stets vom Typ einer Zeichenfolge, mit Ausnahme des Rückgabewertes einer Checkbox, die vom Typ „Boolean“ (also *true* oder *false*) ist. Auf die gleiche Weise erhalten Sie die Werte aller Aktionen vom Typ „Berechnung“ zurück, wobei sich der Typ des zurückgegebenen Werts aus der Art der Berechnung ergibt.

Beispiel: Sie haben eine Texteingabe mit der AID „namedeshundes“ angelegt. Sie können dann als ACONDITION z.B. abfragen:

```
<acondition>namedeshundes == „Waldi“</acondition>
```

Wenn der Nutzer als Name „Waldi“ eingegeben hat, ist die Bedingung erfüllt, also WAHR. Dann würde diese Aktion ausgeführt.



Darüber hinaus ist hier praktisch jede JavaScript-Anweisung möglich, die als Ergebnis WAHR oder FALSCH ergibt.

Hinweis:

Verwenden Sie nicht die deutschen Begriffe wahr oder falsch, sondern stets true oder false!

3.1.1.1.4 ATEXT

Der hier eingegebene Text wird als Beschriftung des Eingabefelds verwendet.

Wenn Sie den Text abhängig von den Eingaben des Nutzers individuell anpassen wollen, können Sie in den Text auch Werte aus Nutzereingaben einfügen, indem die AID der Eingabe in geschweiften Klammern „{“ und „}“ eingesetzt wird.

Wenn also der Nutzer zuvor in einem Textfeld mit dem Namen „namedeshundes“ den Namen „Waldi“ eingegeben hat, könnte die Abfrage des Alters wie folgt aussehen:

```
<atext>Wie alt ist {namedeshundes}?</atext>
```

3.1.1.1.5 APLACEHOLDER

Ist dieser Wert auf „true“ gesetzt, dann wird der unter ATEXT definierte Text nicht als Label, sondern als Placeholder-Text im Eingabefeld selbst angezeigt.

3.1.1.1.6 ADEFAULTVALUE

Mit dem hier angegebenen Wert wird die Aktion vorbelegt.

```
<adefaultvalue>vorgegebener Wert</adefaultvalue>
```

3.1.1.1.7 ATEXTLENGTH

Hier können Sie die Breite des Texteingabefelds in Zeichen angeben, also z.B. für ein 4-stelliges Eingabefeld:

```
<atextlength>4</atextlength>
```

3.1.1.1.8 ATEXTWIDTHPERCENT

Hier können Sie die Breite des Texteingabefelds in Prozent der Fensterbreite eingeben. Soll das Eingabefeld also z.B. über die volle Bildschirmbreite gehen, geben Sie an:

```
<atextwidthpercent>100</atextwidthpercent>
```



3.1.1.1.9 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.2 Aktion: TEXTAREA

Die TEXTAREA-Aktion erzeugt ein mehrzeiliges Texteingabe-Feld, welches auch mehrzeilige Texteingaben mit Zeilenschaltungen – also mit Verwendung der Eingabetaste – zulässt.

3.1.1.2.1 AID

s.o.

3.1.1.2.2 ATYPE

Wert: textarea

3.1.1.2.3 ACONDITION

s.o.

3.1.1.2.4 ATEXT

s.o.

3.1.1.2.5 APLACEHOLDER

s.o.

3.1.1.2.6 ADEFAULTVALUE

s.o.

3.1.1.2.7 ATEXTROWS

Gibt die Anzahl der Textzeilen an, die für die Größe des Texteingabefelds angezeigt werden. Der Text selbst darf bei der Eingabe auch länger werden.

Beispiel für ein Texteingabefeld mit 5 Zeilen Höhe:

```
<atextrows>5</atextrows>
```

3.1.1.2.8 ATEXTLENGTH

s.o.



3.1.1.2.9 ATEXTWIDTHPERCENT

s.o.

3.1.1.2.10 AWYSIWYG

Wird AWYSIWYG auf "true" gesetzt, steht bei der Eingabe ein WYSIWYG-HTML-Editor zur Verfügung.

Achtung:

Diese Option setzt voraus, dass der HTML-Editor „Trumbowyg“ (<https://alex-d.github.io/Trumbowyg/>) sowie JQuery auf dem Server installiert ist.

3.1.1.2.11 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.3 Aktion: DATE

Die Aktion DATE erzeugt ein Eingabefeld, in dem Datumseingaben erfasst werden. Die meisten Browser unterstützen dies durch Anzeige einer entsprechenden Eingabemaske und durch einen „Datepicker“, aus dem das Datum über einen Kalender ausgewählt werden kann.

Unrühmliche Ausnahmen sind der „Internet Explorer“, sowie der Browser „Safari“ auf dem Mac. Da müssen Sie von Hand tippen.

3.1.1.3.1 AID

s.o.

3.1.1.3.2 ATYPE

Wert: date

3.1.1.3.3 ACONDITION

s.o.

3.1.1.3.4 ATEXT

s.o.

3.1.1.3.5 ADEFAULTVALUE

s.o., die Eingabe erfolgt im ISO-Format, also „JJJ-MM-TT“

3.1.1.3.6 Besonderheit: „ISOFORMAT“

Beim Datumsformat gibt es eine Besonderheit:

Egal in welchem Format das Datumsformat bei der Eingabe angezeigt wird, in der Variable würde es standardmäßig im ISO-Format, also JJJ-MM-TT stehen. Wird es dann für Textersetzungen etc. genutzt, wird diese im Deutschen untypische Schreibweise verwendet. Das wäre aber regelmäßig nicht so attraktiv.

Aus dem Grund wird der Wert im Hintergrund in das deutsche Format TT.MM.JJJJ umgewandelt.

Wenn also in einem Datumsfeld mit der AID „geburtstag“ der 3. August 1988 eingegeben, wird in dem Wert „geburtstag“ anschließend das Datum gespeichert als 03.08.1988.

Für Rechen- und Vergleichszwecke wird häufig aber doch noch das ISO-Format benötigt. Aus dem Grund legt CONSULTIMATOR automatisch einen zweiten Wert mit der Endung „isoformat“ an. Über den Wert mit dem Namen „geburtstagsisoformat“ erhält man dann zusätzlich das Ergebnis 1988-08-03.

3.1.1.3.7 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.4 Aktion: MONTH

Die Aktion MONTH erzeugt ein Eingabefeld, in dem Monats-/Jahres-Eingaben erfasst werden. Die meisten Browser unterstützen dies durch Anzeige einer entsprechenden Eingabemaske und durch einen „Datepicker“, aus dem das Monat und Jahr über einen Kalender ausgewählt werden kann.

3.1.1.4.1 AID

s.o.

3.1.1.4.2 ATYPE

Wert: date

3.1.1.4.3 ACONDITION

s.o.



3.1.1.4.4 ATEXT

s.o.

3.1.1.4.5 ADEFAULTVALUE

s.o., die Eingabe erfolgt im ISO-Format, also „JJJ-MM“.

3.1.1.4.6 Besonderheit: „ISOFORMAT“

Beim Monats-/Jahres-Format gibt es eine Besonderheit:

Egal in welchem Format das der Wert bei der Eingabe angezeigt wird, in der Variable würde es standardmäßig im ISO-Format, also JJJ-MM stehen. Wird es dann für Textersetzungen etc. genutzt, wird diese im Deutschen untypische Schreibweise verwendet. Das wäre aber regelmäßig nicht so attraktiv.

Aus dem Grund wird der Wert im Hintergrund in das deutsche Format für Monats- und Jahresangaben „*ausgeschriebenerMonat* JJJ“ umgewandelt.

Wenn also in einem Datumsfeld mit der AID „geburtsmonat“ August 1988 eingegeben, wird in dem Wert „geburtsmonat“ anschließend „August 1988“ gespeichert.

Für Rechen- und Vergleichszwecke wird häufig aber doch noch das ISO-Format benötigt. Aus dem Grund legt CONSULTIMATOR automatisch einen zweiten Wert mit der Endung „isoformat“ an. Über den Wert mit dem Namen „geburtsmonatisoformat“ erhält man dann zusätzlich das Ergebnis 1988-08.

3.1.1.4.7 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.5 Aktion: TIME

Die Aktion TIME erzeugt ein Eingabefeld, in dem Uhrzeit-Eingaben im 24-Stunden-Format erfasst werden. Die meisten Browser unterstützen dies durch Anzeige einer entsprechenden Eingabemaske und durch einen „Datepicker“, aus dem Stunden und Minuten ausgewählt werden können.

3.1.1.5.1 AID

s.o.



3.1.1.5.2 ATYPE

Wert: date

3.1.1.5.3 ACONDITION

s.o.

3.1.1.5.4 ATEXT

s.o.

3.1.1.5.5 ADEFAULTVALUE

s.o., die Eingabe erfolgt im 24-Stunden-Format „HH:MM“.

3.1.1.6 Aktion: CHECKBOX

Der Aktionstyp CHECKBOX erzeugt eine Checkbox.

Wichtig: die Checkbox enthält bei gesetztem Haken den Eingabewert „true“, ansonsten den Wert „false“.

3.1.1.6.1 AID

s.o.

3.1.1.6.2 ATYPE

Wert: checkbox

3.1.1.6.3 ACONDITION

s.o.

3.1.1.6.4 ATEXT

s.o.

3.1.1.6.5 ADEFAULTVALUE

s.o.

3.1.1.6.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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



3.1.1.7 Aktion: COLOR

Der Aktionstyp COLOR erzeugt einen HTML5 Farbauswahl-Dialog.

Achtung:

Der Farbauswahl-Dialog wird nicht auf allen Browsern angezeigt. Er funktioniert einwandfrei mit dem Chrome, Firefox und Edge, Safari und Internet Explorer zeigen hingegen lediglich ein Texteingabefeld, in welches der Farbwert dann aber in RGB-Hex-Schreibweise (#xyyyzz) eingegeben werden kann.

3.1.1.7.1 AID

s.o.

3.1.1.7.2 ATYPE

Wert: color

3.1.1.7.3 ACONDITION

s.o.

3.1.1.7.4 ATEXT

s.o.

3.1.1.7.5 ADEFAULTVALUE

s.o.

3.1.1.7.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.8 Aktion: MESSAGE

Die Aktion MESSAGE erzeugt lediglich einen Ausgabertext, nimmt aber keine Eingaben entgegen. MESSAGE eignet sich, da im Bereich des INPUTBLOCK eingesetzt, um den Anwender beim Ausfüllen des Eingabebereichs zu führen, Tipps zu geben oder über Zwischenergebnisse zu informieren.

MESSAGE darf nicht mit der OMESSAGE (Output-Message) verwechselt werden: OMESSAGE Aktionen zeigen Texte im Ausgabebereich (OUTPUTBLOCK) an.

Tipp:



MESSAGE kann auch dafür verwendet werden, „unsichtbare“ Dinge zu erledigen, wie z.B. CSS-Anpassungen vorzunehmen. Ein Beispiel, in dem die Farbe der Buttons auf blau geändert wird:

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

Denken Sie in diesem Fall nur daran, die ACONDITION auf „false“ zu setzen, so verhindern Sie, dass die Ausgabe eine zusätzliche Leerzeile bei der Anzeige in Anspruch nimmt.

3.1.1.8.1 AID

s.o.

3.1.1.8.2 ATYPE

Wert: message

3.1.1.8.3 ACONDITION

s.o.

3.1.1.8.4 ATEXT

s.o.

3.1.1.8.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.9 Aktion: BLINDS

Die Aktion BLINDS erzeugt einen Zwischenüberschrift-Block im Eingabebereich, der gleichzeitig als Blende funktioniert. D.h. nachfolgende Blöcke werden

„zusammengeklappt“ dargestellt und erst beim Anklicken der Zwischenüberschrift eingeblendet. Gleichzeitig werden alle anderen Blöcke wieder ausgeblendet.

Wird die Blende eines bereits geöffneten Blocks angeklickt, wird dieser Block geschlossen.

Soll die erste Blende bei Aufruf der Seite geschlossen sein, ist dies über individuelle Style-Einstellungen möglich:

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

Über die Style-Einstellungen kann auch eine andere Blende beim Aufruf der Seite als erste geöffnet dargestellt werden, hier z.B. Blende 2:

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

3.1.1.9.1 AID

s.o.

3.1.1.9.2 ATYPE

Wert: blinds

3.1.1.9.3 ACONDITION

s.o.

Allerdings blendet ACONDITION nicht nur die Zwischenüberschrift, sondern den kompletten Block bei "false" aus.

3.1.1.9.4 ATEXT

s.o.

3.1.1.9.5 ABLOCKID

ABLOCKID kann die AID einer OMESSAGE Aktion enthalten. In dem Fall scrollt der Ausgabebereich zur OMESSAGE-Ausgabe, wenn die BLIND angeklickt wird. Dies macht natürlich nur Sinn, wenn Ein- und Ausgabebereich nebeneinander platziert sind und separat scrollen können.



3.1.1.9.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.10 Aktion: SELECT

Die Aktion „SELECT“ zeigt eine Aufklapp-Box (Drop-Down-Box), die eine vorgefertigte Liste von Eingabewerten anzeigt. Daraus kann der gewünschte Eingabewert ausgewählt werden.

3.1.1.10.1 AID

s.o.

3.1.1.10.2 ATYPE

Wert: select

3.1.1.10.3 ACONDITION

s.o.

3.1.1.10.4 ATEXT

s.o.

3.1.1.10.5 ADEFAULTVALUE

s.o.

3.1.1.10.6 AOPTIONS

Unter „AOPTIONS“ wird die Liste aller möglichen, auswählbaren Eingabewerte angelegt.

3.1.1.10.6.1 AOPTION

Der einzelne Eingabewert, bestehend aus anzuzeigendem Wert und den im Hintergrund als Eingabewert verwendeten Wert.

3.1.1.10.6.1.1 AOPTLABEL

Dies ist der anzuzeigende Wert.

3.1.1.10.6.1.2 AOPTVALUE

Dies ist der Eingabewert, der bei Auswahl des angezeigten Werts genutzt wird.

3.1.1.10.7 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.1.11 Aktion: OPTION

Die Aktion „OPTION“ zeigt mehrere Option-Buttons (Radio-Buttons) an, entsprechend der definierten Auswahlliste an. Daraus kann der gewünschte Eingabewert ausgewählt werden.

Mit OPTION kann grundsätzlich das Gleiche erreicht werden wie mit SELECT, es sieht primär nur optisch anders aus.

3.1.1.11.1 AID

s.o.

3.1.1.11.2 ATYPE

Wert: option

3.1.1.11.3 ACONDITION

s.o.

3.1.1.11.4 ATEXT

s.o.

3.1.1.11.5 ADEFAULTVALUE

s.o.

3.1.1.11.6 AOPTIONS

Unter „AOPTIONS“ wird die Liste aller möglichen, auswählbaren Eingabewerte angelegt.

3.1.1.11.6.1 AOPTION

Der einzelne Eingabewert, bestehend aus anzuzeigendem Wert und den im Hintergrund als Eingabewert verwendeten Wert.



3.1.1.11.6.1.1 AOPTLABEL

Dies ist der anzuzeigende Wert.

3.1.1.11.6.1.2 AOPTVALUE

Dies ist der Eingabewert, der bei Auswahl des angezeigten Werts genutzt wird.

3.1.1.11.7 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2 Sonderfunktionen

3.1.2.1 *Aktion: SHOWOMESSAGES*

Zeigt einen Button an, der den OUTPUTBLOCK auf Wunsch komplett ein - bzw. ausblendet.

Dieser Button muss auf jeden Fall dann verwendet werden, wenn die Sichtbarkeit des Ausgabeblocks durch die entsprechende Einstellung im Bereich GENERAL ausgeschaltet wurde.

3.1.2.1.1 AID

s.o.

3.1.2.1.2 ATYPE

Wert: showomessages

3.1.2.1.3 ACONDITION

s.o.

3.1.2.1.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.1.5 ATOGGLEINPUTOUTPUT

Standard ist „false“.

Wird der Wert auf „true“ gesetzt, dann wird beim Einblenden des OUTPUTBLOCK gleichzeitig der INPUTBLOCK ausgeblendet.



Der Wert darf nicht auf „true“ gesetzt werden, wenn der OUTPUTBLOCK standardmäßig eingeblendet ist. In dem Fall würde, da SHOWOMESSAGES als Umschalter fungiert, der Ein- und Ausgabeblock ausgeblendet werden und die Seite wäre leer.

3.1.2.1.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.2 Aktion: SUBMIT

Zeigt einen Button an, der die Daten des INPUTBLOCK entsprechend der im Bereich GENERAL definierten Form-Aktion weiterverarbeitet.

3.1.2.2.1 AID

s.o.

3.1.2.2.2 ATYPE

Wert: submit

3.1.2.2.3 ACONDITION

s.o.

3.1.2.2.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.2.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.3 Aktion: FORMCLIPBOARD

Zeigt einen Button an, der die Daten des INPUTBLOCK kompatibel zur Aktion PARAMETERS in die Zwischenablage übernimmt.



3.1.2.3.1 AID

s.o.

3.1.2.3.2 ATYPE

Wert: submit

3.1.2.3.3 ACONDITION

s.o.

3.1.2.3.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.3.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.4 Aktion: *SUBMITSECURE*

Zeigt einen Button an, der die Daten des INPUTBLOCK entsprechend der im Bereich GENERAL definierten Form-Aktion weiterverarbeitet. Die Daten werden dabei mit Hilfe des in den Projekteinstellungen hinterlegten RSA Public Key asymmetrisch verschlüsselt. Und mittels eines einzigen Parameters „encdata“ weitergereicht.

Diese Aktion steht im Internet Explorer nicht zur Verfügung und wird dort ausgeblendet.

3.1.2.4.1 AID

s.o.

3.1.2.4.2 ATYPE

Wert: submitsecure

3.1.2.4.3 ACONDITION

s.o.

3.1.2.4.4 ATEXT

Dies ist die Beschriftung des Buttons.



3.1.2.4.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.5 Aktion: PARAMETERS

Zeigt einen Button an, der beim Anklicken ein Texteingabefeld anzeigt. Wird in dieses Texteingabefeld die Liste aller URL-Parameter eingefügt, wird die Webseite mit den Werten gefüllt. Sind die Daten verschlüsselt, werden Sie automatisch nach Ihrem RSA Private Key gefragt.

Eine solche Liste der URL-Parameter kann z.B. über die Form Aktion „mailto:“ erzeugt werden, siehe dazu die Ausführungen oben.

Im Zusammenspiel dieser beiden Funktionen können so die auf der Seite eingegebenen Daten exportiert und wieder importiert werden.

3.1.2.5.1 AID

s.o.

3.1.2.5.2 ATYPE

Wert: parameters

3.1.2.5.3 ACONDITION

s.o.

3.1.2.5.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.5.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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



3.1.2.6 Aktion: *RESET*

Zeigt einen Button an, der beim Klicken alle im Formular eingegebenen Daten wieder löscht.

3.1.2.6.1 AID

s.o.

3.1.2.6.2 ATYPE

Wert: reset

3.1.2.6.3 ACONDITION

s.o.

3.1.2.6.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.6.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.7 Aktion: *FORMDOWNLOAD*

Zeigt einen Button an, mit dem alle im Formular eingegebenen Daten per Browser-Download auf Ihren Rechner heruntergeladen werden können.

3.1.2.7.1 AID

s.o.

3.1.2.7.2 ATYPE

Wert: formdownload

3.1.2.7.3 ACONDITION

s.o.

3.1.2.7.4 ATEXT

Dies ist die Beschriftung des Buttons.



3.1.2.7.5 AFILENAME

Dies ist der für den Download vorgeschlagene Dateiname. Eine möglicherweise angegebene Dateiendung wird ignoriert und durch „.cmc“ ersetzt.

3.1.2.7.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.8 Aktion: *FORMDOWNLOADCSV*

Zeigt einen Button an, mit dem alle im Formular eingegebenen Daten per Browser-Download auf Ihren Rechner im CSV-Format heruntergeladen werden können.

In der CSV-Datei sind die Werte mit doppelten Anführungszeichen eingeschlossen und mit Semikolon voneinander separiert. Die CSV-Datei ist stets im UTF-8-Zeichensatz formatiert. Die erste Zeile enthält die Feldnamen.

Achtung:

Grundsätzlich kann Microsoft Excel diese CSV-Dateien lesen. Allerdings erkennt Excel die UTF-8-Formatierung nicht automatisch, wenn man auf die Datei über „Datei öffnen“ zugreift. Bitte nutzen Sie daher stattdessen in Excel die Option, Daten aus Textdateien zu importieren.

3.1.2.8.1 AID

s.o.

3.1.2.8.2 ATYPE

Wert: formdownloadcsv

3.1.2.8.3 ACONDITION

s.o.

3.1.2.8.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.8.5 AFILENAME

Dies ist der für den Download vorgeschlagene Dateiname. Eine möglicherweise angegebene Dateiendung wird ignoriert und durch „.csv“ ersetzt.



3.1.2.8.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.9 Aktion: FORMDOWNLOADSECURE

Zeigt einen Button an, mit dem alle im Formular eingegebenen Daten per Browser-Download auf Ihren Rechner heruntergeladen werden können. Die Daten werden dabei mit Hilfe des in den Projekteinstellungen hinterlegten RSA Public Key asymmetrisch verschlüsselt.

Diese Aktion steht im Internet Explorer nicht zur Verfügung und wird dort ausgeblendet.

3.1.2.9.1 AID

s.o.

3.1.2.9.2 ATYPE

Wert: formdownloadsecure

3.1.2.9.3 ACONDITION

s.o.

3.1.2.9.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.9.5 AFILENAME

Dies ist der für den Download vorgeschlagene Dateiname. Eine möglicherweise angegebene Dateiendung wird ignoriert und durch „.cmc“ ersetzt.

3.1.2.9.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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



3.1.2.10 Aktion: *PARAMETERUPLOAD*

Zeigt einen Button an, mit dem alle Formulardaten aus einer Datei hochgeladen werden können. Sind die Daten verschlüsselt, werden Sie automatisch nach Ihrem RSA Private Key gefragt.

3.1.2.10.1 AID

s.o.

3.1.2.10.2 ATYPE

Wert: parameterupload

3.1.2.10.3 ACONDITION

s.o.

3.1.2.10.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.10.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.2.11 Aktion: *ATTACHMENT*

Zeigt einen Button an, mit dem zu den Formular-Eingabedaten noch eine Anlage hochgeladen werden kann.

Diese kann später bei SUBMIT, SECURESUBMIT, FORMDOWNLOAD und FORMDOWNLOADSECURE mit versendet und heruntergeladen werden.

Achtung:

Diese Aktion kann im Projekt nur 1x verwendet werden.

3.1.2.11.1 AID

s.o.

3.1.2.11.2 ATYPE

Wert: attachment



3.1.2.11.3 ACONDITION

s.o.

3.1.2.11.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.1.2.11.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.1.3 Generelle Aktionen

3.1.3.1 *Aktion: STYLE*

Die Aktion STYLE gibt die Möglichkeit, Cascading Style Sheets (CSS) Ihrem Projekt hinzuzufügen. Diese werden erst nach den im HTML-Head definierten Styles ausgeführt. Sie werden in eine separate Datei ausgelagert und nicht direkt in den HTML-Text „inline“ eingefügt.

Die Aktion kann auch mehrfach verwendet werden.

3.1.3.1.1 AID

s.o.

3.1.3.1.2 ATYPE

Wert: style

3.1.3.1.3 ACONDITION

keine

3.1.3.1.4 ATEXT

Hier werden die eigentlichen CSS Definitionen eingefügt. Bitte schließen Sie diese nicht in `<style> ... </style>` ein, da dies Fehler verursachen könnte. CONSULTIMATOR wird diese in den meisten Fällen automatisch beseitigen, aber es ist besser, von vornherein darauf zu verzichten.

3.2 Berechnungen

3.2.1 Aktion: COMPUTE

Der Aktionstyp COMPUTE ist extrem leistungsstark.



COMPUTE ist kein Button oder Eingabe-Typ, sondern eine Datenverarbeitungsfunktion.

Mit COMPUTE können neue Werte definiert werden, und zwar zunächst in Abhängigkeit von der ACONDITION. Ist diese true, wird der Wert auf den Inhalt von ATEXT gesetzt.

Besonders ist hier zunächst, dass es auch ATEXTFALSE gibt. Dieser Inhalt wird verwendet, wenn die Bedingung in ACONDITION nicht zutrifft, also false ist.

Eine weitere, wichtige Besonderheit wird unten bei ATEXT beschrieben.

3.2.1.1 AID

s.o.

3.2.1.2 ATYPE

Wert: compute

3.2.1.3 ACONDITION

s.o.

3.2.1.4 ATEXT

Hier gibt es eine ganz wichtige Besonderheit:

Im Gegensatz zu den anderen ATEXT-Feldern wird hier nicht nur Text zugewiesen, in dem ggf. noch ein anderer Wert per Replace-Funktion ausgetauscht wird.

Hier wird stattdessen im Javascript-Format die Operation angegeben, deren Ergebnis anschließend an den Wert übergeben wird.

So können z.B.

- AID-Werte als Text verknüpft werden:

```
<atext>"Der Hund heißt " + namedeshundes + " und ist " + alterdeshundes + " Jahre alt!"</atext>
```

- AID-Werte können mathematisch weiterverarbeitet werden, so könnte das Alter des Hundes in 3 Jahren wie folgt berechnet werden:

```
<atext>+alterdeshundes + 3</atext>
```

Hinweis:

Im vorstehenden Beispiel steht nicht einfach

„alterdeshundes + 3“

sondern:

„+alterdeshundes + 3“

Der Grund dafür ist einfach: Fehlt das „+“ vor dem „alterdeshundes“, geht das System davon aus, dass „alterdeshundes“ eine Zeichenfolge und keine Zahl ist. Wäre der Hund dann z.B. 12, würde da die „3“ hinten angehängt: „123“. Das wäre dem Hund sicher zu wünschen, aber für das, was wir erreichen wollten, einfach falsch.

- Es können aber auch ebenso Javascript-eigene Funktionen verwendet werden, hier z.B., um das heutige Datum in der Schreibweise DD.MM.YYYY zu erzeugen:

```
<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()/>
```

- Und es können auch eigene Javascript-Funktionen angesprochen werden, die im TEMPLATEHEAD definiert oder aus externen Quellen mit eingebunden wurden:

```
<atext>berechnetelebenserwartung ( alterdeshundes )</atext>
```

Aber jetzt wird es schon richtig komplex. Eigentlich wollten wir ja nicht programmieren. Aber wer unbedingt möchte... ;-)

3.2.1.5 ATEXTFALSE

Wie ATEXT, wird allerdings genutzt, wenn die ACONDITION false ergibt.

3.3 Ersetzungen

3.3.1 Standard-Ersetzungen

Ersetzungen sind extrem praktisch und wurden oben schon einmal angesprochen. Über Ersetzungen können in allen angezeigten ATEXT-Ausgaben sowie in AOPTLABEL die Werte anderer Aktionen durch einfache Angabe der AID in geschweiften Klammern

Beispiel: Mein Hund heißt {namedeshundes} und ist {alterdeshundes} Jahre alt!

eingefügt werden. Diese Platzhalter werden dann in Echtzeit durch den entsprechenden Eingabewert oder per COMPUTE errechneten Wert ersetzt.

Wichtig:



Wenn Sie Ersetzungen in OMESSAGE Aktionen verwenden, wird für den Fall, dass der Wert leer ist, der Name des Platzhalters farblich hinterlegt angezeigt. Dadurch können Sie auf einen Blick sehen, dass der Wert wahrscheinlich noch nicht gesetzt wurde.

Möchten Sie aber, dass ein leerer Wert auch tatsächlich gar nicht angezeigt wird, setzen Sie an das Ende des Platzhalters ein „+“:

{namedeshundes+}

3.3.2 Intelligente Ersetzungen

Mit intelligenten Ersetzungen können Sie Ihre Ausgaben noch stärker individualisieren. Intelligente Ersetzungen verwenden Sie wie Standard-Ersetzungen, ebenfalls für ATEXT und AOPTLABEL, sehen aber folgendermaßen aus:

{{aid|Anzeige bei Wert der AID „true“|Anzeige bei Wert der AID „false“}}

Beispiel:

„Bitte liefern Sie die Ware {geschlecht|meinem Kunden|meiner Kundin} bis morgen.“

Ist nun im Wert der AID „geschlecht“ männlich als „true“ (oder „1“) gespeichert, wird die Formulierung „meinem Kunden“, bei „false“ (oder „0“) wird „meiner Kundin“ angezeigt.

Dieses Beispiel ist nicht ohne Grund gewählt: Sie können so komplette Schreiben, Gutachten, Schriftsätze u.v.m. erstellen und Ihre Kunden geschlechtsspezifisch korrekt bezeichnen, ohne auf neutrale, unpersönliche Formulierungen wie „meine Kundschaft“ oder „meine Mandantschaft“ ausweichen zu müssen.

Tipp: weitere Optionen hinzufügen

Sie können aber auch noch weitere Ausgabeoptionen hinzufügen:

{{aid|Anzeige bei Wert der AID „true“|Anzeige bei Wert der AID „false“|Anzeige bei Wert der AID „-1“|Anzeige bei Wert der AID „-2“|...}}

Für jede über „false“ (oder 0) hinausgehende Option zählen setzen Sie den Wert der AID um jeweils 1 niedriger an, also 1 (oder true), 0 (oder false), -1, -2, -3, -4 u.s.w. Die Anzahl der Optionen ist nicht limitiert.

Tipp: Aktions-ID auf Zeichenfolge prüfen

Möchten Sie überprüfen, ob die aktions_id eine Zeichenfolge enthält oder nicht, geben Sie nach der Aktions-ID ein Fragezeichen (?) ein:

`{{aid?|Ausgabetext falls ja|Ausgabetext falls nein}}`

Tipp:

Innerhalb einer intelligenten Ersetzung können Sie auch eine Standard-Ersetzung verwenden.

Hinweis zur „alten“ Schreibweise mit einfachen geschwungenen Klammern:

Die frühere Schreibweise für intelligente Ersetzungen mit einfachen statt doppelten geschwungenen Klammern funktioniert weiterhin, solange innerhalb der intelligenten Ersetzung keine Standard-Ersetzung erfolgt und solange nur die Optionen für „true“ und „false“ verwendet werden.

3.3.3 Automatische Zähler

Automatische Zähler helfen bei der Darstellung nummerierter Abschnitte und Kapitel, indem sie innerhalb der OMESSAGE Aktionen automatisch hochzählen, sofern diese nicht verborgen sind.

Sie können die Automatischen Zähler wie folgt verwenden:

- `[inc]` für Hauptabschnitte,
- `[incs]` für Unterabschnitte.

Der `[inc]` Zähler wird sich bei jeder Verwendung um „1“ erhöhen, solange die OMESSAGE Aktion angezeigt wird.

Der `[incs]` Zähler wird sich ebenfalls bei jeder Verwendung um „1“ erhöhen, solange die OMESSAGE Aktion angezeigt wird, allerdings wird der `[incs]` Zähler wieder auf „1“ zurückgesetzt, sobald ein neuer `[inc]` Zähler verwendet wird.

Platzieren Sie `[inc]` oder `[incs]` irgendwo in Ihrer OMESSAGE Aktion und sie wird automatisch ersetzt.

Sowohl `[inc]` als auch `[incs]` funktionieren über die Grenzen der einzelnen OMESSAGE Aktion hinweg.

Achtung:

Automatische Zähler können nicht in intelligenten Ersetzungen genutzt werden.

Bezugnahme auf den Zähler innerhalb des Ausgabertextes



Wenn Sie in den Ausgabe-Aktionen auf die gesetzten Zähler Bezug nehmen wollen, er also in den Text eingesetzt werden soll, z.B.

„Wie in § 23 Absatz 2 beschrieben...“,

dann ist das ebenfalls möglich, indem Sie [inc] [incs] oder [incstart] in Klammern eine eindeutige Kennung hinzufügen,

[inc(eindeutigeerkennung)]

z.B. [inc(vertragschluss)]. Im Text selbst referenzieren Sie diesen Zähler, indem Sie **[link-inc(vertragsschluss)]** (bzw. link-incs, link-incstart) verwenden, also z.B.

„Wie in § **[link-inc(vertragschluss)]** Absatz 2 beschrieben...“

3.4 Datenausgabe im “Outputblock”

3.4.1 Ausgabe-Aktionen

3.4.1.1 *Aktion: OMESSAGE*

Die Aktion OMESSAGE ist die Standard-Ausgabeform zur Verwendung im OUTPUTBLOCK, also im Ausgabebereich.

Der Ausgabebereich enthält eine Abfolge von einem oder mehreren OMESSAGE-Aktionen.

- Eine Ausgabe wird dann sinnvoller Weise in mehrere OMESSAGE-Aktionen aufgeteilt, wenn diese Aktionen abhängig von unterschiedlichen ACONDITION-Ergebnissen angezeigt werden sollen.
- Damit mehrere OMESSAGE-Aktionen später zusammengefasst weiterverarbeitet werden können, können sie zu Blöcken, beginnend mit STARTBLOCK, endend mit ENDBLOCK zusammengefasst werden. Dazu unten mehr.

3.4.1.1.1 AID

s.o.

3.4.1.1.2 ATYPE

Wert: omessage

3.4.1.1.3 ACONDITION

s.o.



3.4.1.1.4 ATEXT

Hier kommt der Ausgabe-Text, der im OUTPUTBLOCK angezeigt werden soll.

Für die OMESSAGE-Aktion gilt für ATEXT eine wichtige Besonderheit, die das Leben sehr praktisch macht:

Während die anderen ATEXT Werte keine Zeilenumbrüche enthalten dürfen, kann hier zwischen „<![CDATA“ und „]]>“ auch mehrzeiliger Text verwendet werden.

3.4.1.1.5 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.4.1.2 Aktion: STARTBLOCK

Mit der Aktion STARTBLOCK werden mehrere OMESSAGE-Aktionen zu einem Block zusammengefasst. Dadurch kann der Block eine Überschrift bekommen und über die CLIPBOARD-Aktion als Ganzes in die Zwischenablage übertragen werden.

STARTBLOCK definiert den **Anfang** des Blocks.

Achtung: STARTBLOCK hat keine ACONDITION!

3.4.1.2.1 AID

s.o.

3.4.1.2.2 ATYPE

Wert: startblock

3.4.1.2.3 ATEXT

Dieser Text wird als Überschrift für den Block verwendet.

3.4.1.2.4 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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



3.4.1.3 Aktion: ENDBLOCK

ENDBLOCK schließt den über STARTBLOCK begonnenen Block.

Achtung: ENDBLOCK hat keine ACONDITION und keinen ATEXT!

3.4.1.3.1 AID

s.o.

3.4.1.3.2 ATYPE

Wert: endblock

3.4.2 Sonderfunktionen

3.4.2.1 Aktion: CLIPBOARD

Die Aktion CLIPBOARD erstellt einen Button, mit dem die gesamte zu einem Block zusammengefasste Ausgabe in die Zwischenablage übernommen werden kann.

3.4.2.1.1 AID

s.o.

3.4.2.1.2 ATYPE

Wert: clipboard

3.4.2.1.3 ACONDITION

s.o.

3.4.2.1.4 ATEXT

Beschriftung des Buttons.

3.4.2.1.5 ABLOCKID

Gibt an, **welcher Block** in die Zwischenablage kopiert werden soll. Angegeben wird die für den jeweiligen STARTBLOCK verwendete AID.

3.4.2.1.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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



3.4.2.2 Aktion: WORDSAVE

Die Aktion WORDSAVE erstellt einen Button, mit dem die gesamte zu einem Block zusammengefasste Ausgabe als Microsoft Word Datei im DOCX Format gespeichert werden kann.

Bitte beachten Sie, dass WORDSAVE stets den Schrifttyp „Calibri“ bzw. als Ausweich-Schriftart „Arial“ verwendet und Grafiken nur dann berücksichtigt, wenn sie direkt in die HTML-Ausgabeseite per Base64 eingebettet sind.

Kompatibilitäts-Hinweis:

Die von WORDSAVE generierten DOCX-Dateien benötigen zur korrekten Anzeige eine vollwertige „Desktop“-Version von Microsoft Word. Die im Funktionsumfang beschnittenen Versionen für Tablet PC und Smartphone reichen nicht aus.

Außerdem können aufgrund von Systembeschränkungen auf dem Browser „Safari“ für iOS sowie auf diesen aufbauenden Browsern mit WORDSAVE keine Ausgabedateien erzeugt werden. Auf diesen Browsern wird der WORDSAVE Button automatisch unterdrückt, sofern diese sich korrekt zu erkennen geben.

3.4.2.2.1 AID

s.o.

3.4.2.2.2 ATYPE

Wert: wordsave

3.4.2.2.3 ACONDITION

s.o.

3.4.2.2.4 ATEXT

Beschriftung des Buttons.

3.4.2.2.5 ABLOCKID

Gibt an, **welcher Block** in die gespeichert werden soll. Angegeben wird die für den jeweiligen STARTBLOCK verwendete AID.

3.4.2.2.6 AFILENAME

Gibt an, unter welchem Dateinamen die Datei gespeichert werden soll. Die Angabe erfolgt ohne Dateiendung, jede angegebene Dateiendung wird ignoriert.



3.4.2.2.7 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.4.2.3 Aktion: PRINT

Die Aktion PRINT erstellt einen Button, mit dem die gesamte zu einem Block zusammengefasste Ausgabe gedruckt werden kann.

3.4.2.3.1 AID

s.o.

3.4.2.3.2 ATYPE

Wert: print

3.4.2.3.3 ACONDITION

s.o.

3.4.2.3.4 ATEXT

Beschriftung des Buttons.

3.4.2.3.5 ABLOCKID

Gibt an, **welcher Block** gedruckt kopiert werden soll.
Angegeben wird die für den jeweiligen STARTBLOCK verwendete AID.

3.4.2.3.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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

3.4.2.4 Aktion SHOWINPUTBLOCK

Zeigt einen Button an, der den INPUTBLOCK auf Wunsch komplett ein - bzw. ausblendet.



3.4.2.4.1 AID

s.o.

3.4.2.4.2 ATYPE

Wert: showomessages

3.4.2.4.3 ACONDITION

s.o.

3.4.2.4.4 ATEXT

Dies ist die Beschriftung des Buttons.

3.4.2.4.5 ATOGGLEINPUTOUTPUT

Wird der Wert auf „true“ gesetzt, dann wird beim Einblenden des INPUTBLOCK gleichzeitig der OUTPUTBLOCK ausgeblendet.

3.4.2.4.6 ASTYLE

Hier können Sie der Aktion eine eigene Style-Angabe unter Verwendung der CSS-Style-Syntax mitgeben. Soll z.B. der Text der Aktion in rot, in 15 Punkt Größe angegeben werden, sieht die Angabe folgendermaßen aus:

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



4 CONSULTIMATOR Datenverschlüsselung

Grundsätzlich gibt CONSULTIMATOR keine Daten weiter. Der komplette Prozess der Erstellung der Ausgaben aus den Nutzereingaben erfolgt lokal im Webbrowser des Anwenders ohne Verbindung zum Server.

CONSULTIMATOR ermöglicht Ihnen jedoch, optional Nutzereingaben zur Weiterverarbeitung entgegen zu nehmen. Verwenden Sie keine Verschlüsselung, so werden die Daten einfach per Form Submit mit den in den Projekteinstellungen vorgenommenen Vorgaben an das dort definierte Ziel gesandt. Für die Sicherheit „on transit“, d.h. beim Transport, können Sie natürlich eine SSL Verschlüsselung per HTTPS-Protokoll wählen. Damit sind die Daten jedoch an der Empfangsstelle immer noch unverschlüsselt.

Möchten Sie die Daten vollständig verschlüsselt empfangen, so definieren Sie in den Projekteinstellungen im Wert „rsapublickey“ Ihren persönlichen RSA Public Key.

4.1 Ort der Verschlüsselung und Browserkompatibilität

CONSULTIMATOR verschlüsselt die Nutzerdaten nach dem folgenden Schema vollständig browserseitig.

Dabei können Sie die Webbrowser Edge, Safari, Firefox, und Chrome jeweils in ihren aktuellen Versionen verwenden. Der Internet Explorer wird nicht unterstützt.

4.2 Erstellung eines RSA Public Key Private Key Schlüsselpaars

Für die Erstellung eines RSA Public Key & Private Key Schlüsselpaars folgen Sie den üblichen Anleitungen im Internet. Der Vorgang wird auch in der CONSULTIMATOR Bedienungsanleitung beschrieben. Zur optimalen Sicherheit Ihrer Daten, erzeugen Sie bitte ein 2048-bit Schlüsselpaar oder stärker. CONSULTIMATOR unterstützt eine Schlüsselstärke bis 4096-bit.

4.3 Verschlüsselungsmethode

Die Verschlüsselung erfolgt asymmetrisch, d.h. die Daten werden mit dem auf der Ausgabeseite hinterlegten Public Key verschlüsselt, können aber nur mit Hilfe des dort nicht hinterlegten Private Key wieder entschlüsselt werden, der sich stets zu Ihren sicheren Händen befinden sollte.

Aufgrund der Größe der zu verschlüsselnden Datenmenge kommt eine rein asymmetrische Verschlüsselung nicht in Betracht. Es wird daher wie folgt vorgegangen:

1. Es wird ein zufälliges, 50 Zeichen langes Passwort generiert.



2. Daraus wird ein Schlüssel für eine symmetrische Verschlüsselung der Daten nach dem Standard SHA-256 erstellt.
3. Die Daten werden mittels dieses Schlüssels nach dem Standard AES-CBC verschlüsselt. Dafür werden die in Ihrem Browser herstellerseitig vorhandenen, etablierten Verschlüsselungsbibliotheken verwendet.
4. Anschließend wird der Schlüssel selbst mittels Ihres RSA Public Key nach dem Standard RSA-256 verschlüsselt. Hierzu wird die verbreitete Bibliothek „JSEncrypt“ (<https://github.com/travist/jseencrypt>) verwendet, die ihrerseits die Bibliothek „jsbn“ (<http://www-cs-students.stanford.edu/~tjw/jsbn/>) nutzt.
5. Auch wenn dies eigentlich nicht erforderlich ist, wird der sog. „Vektor“, der für die Entschlüsselung der Daten erforderlich ist und im Verschlüsselungsprozess erstellt wurde, ebenfalls mittels des RSA Public Key nach RSA-256 verschlüsselt.
6. Der verschlüsselte Schlüssel, Vektor sowie die Daten werden wie folgt zu folgender Gesamt-Zeichenfolge zusammengefügt:

```
CMCENCRAW-V1.0::::::schlüssel::::vektor:::daten
```

4.4 Entschlüsselung

Die Entschlüsselung erfolgt durch Zerlegen der Gesamt-Zeichenfolge in ihre drei Bestandteile, anschließende Entschlüsselung von Schlüssel und Vektor mittels des RSA Private Key und schließlich Entschlüsselung der Daten über den entschlüsselten Key via AES-CBC.

Die Ausgabeseiten enthalten bereits die für die browserseitige Entschlüsselung erforderlichen Routinen, allerdings – natürlich – nicht den RSA Private Key. Erkennt die Ausgabeseite, dass einzulesende Daten verschlüsselt sind, werden Sie zur Eingabe des RSA Private Key aufgefordert. Dieser wird ausschließlich lokal im Browser zur Entschlüsselung genutzt und nicht auf dem Rechner gespeichert. Optional kann der RSA Private Key lokal in einem Cookie abgelegt werden, wenn ein Passwort für seine Ver- und Entschlüsselung gesetzt wird.

4.5 encodeURIComponent / decodeURIComponent

Zur fehlerfreien Ver- und Entschlüsselung werden die Daten mit encodeURIComponent / decodeURIComponent codiert bzw. Anschließend decodiert.

4.6 Keine proprietären Ergänzungen



Der Verschlüsselung wurden keine von den o.g. Standards abweichenden, proprietären Änderungen hinzugefügt, sodass eine Portierung in andere Programmiersprachen möglich ist, sofern diese die verwendeten Standards ebenfalls unterstützen.

5 Daten automatisiert in CONSULTIMATOR Ausgabeseiten laden

Die Eingabefelder einer CONSULTIMATOR Ausgabeseite sind normale Form-Felder und können als solche mit Inhalten beim Laden der Seite vorbelegt werden. Da die CONSULTIMATOR Ausgabeseiten, zumindest, wenn diese nicht manuell modifiziert werden, jedoch statisches HTML sind, können sie die Eingabedaten auf herkömmlichem Weg nur per URL Parameter empfangen, indem die URL folgendermaßen ergänzt wird:

```
...meineausgabeseite.html?eingabewert=123
```

Dabei stößt das System an Grenzen. Einmal, weil URLs von Browsern nicht in beliebiger Länge verarbeitet werden, und dann, weil diese Übertragungsmethode nur beschränkt sicher ist.

Dieses Problem kann folgendermaßen umgangen werden:

CONSULTIMATOR Ausgabeseiten verfügen über eine Javascript-basierte Routine, die beim Aufruf der Seite prüft, ob in der wahlweise im Local Storage oder im Session Storage des Browsers abgelegten Variable

```
tdc_parameterstring
```

zu verarbeitende Eingaben abgelegt sind. Die dort abgelegten Eingaben werden in der gleichen Syntax hinterlegt, wie bei der Übergabe per URL, also:

```
eingabewert1=123&eingabewert2=456&eingabewert3=789
```

Um die Daten in die Variable zu laden, kann zu einem Trick gegriffen werden. Rufen Sie eine Routine auf, die die Daten zuerst über ein Javascript-Skript in die Variable lädt, und dann per Redirect die Ausgabeseite öffnet.

Dies kann z.B. in PHP folgendermaßen aussehen, davon ausgehend, dass in \$content der o.g. Parameterstring abgelegt wurde und \$form die URL der Ausgabeseite enthält.

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

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



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

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

Dabei ist zu beachten, dass dies aus Sicherheitsgründen nur dann funktioniert, wenn beide Seiten, also die dynamische Seite, die die Variable `tdc_parameterstring` vorbelegt, und die Ausgabeseite selbst auf dem gleichen Server liegen.

CONSULTIMATOR stellt in der eigenen Umgebung unter

<https://consultimator.com/populate.php>

eine entsprechende Routine kostenfrei und ohne Anspruch auf dauerhafte Erreichbarkeit als Service zur Verfügung. Die URL wird um

?form=Link zur eigenen Ausgabeseite

ergänzt. Die eigentlichen Formulardaten werden per FORM POST übergeben. Bitte beachten Sie, dass Sie diese Routine nur für Evaluationszwecke nutzen sollten, da CONSULTIMATOR alle hierüber übertragenen Daten einsehen kann. Wenn Sie produktiv sensible Daten übertragen wollen, richten Sie bitte (a) eine entsprechende Routine auf Ihrem eigenen Server ein, und hosten Sie (b) ebenfalls die Ausgabeseiten auf Ihrem eigenen Server.

Den Quellcode der Routine `populate.php` stellen wir Ihnen bei Bedarf gerne ohne gesonderte Berechnung zur eigenen Verwendung zur Verfügung.



6 Der CONSULTIMATOR in Aktion

Haben Sie die Projektdatei fertig erstellt, geben Sie dem CONSULTIMATOR den Auftrag, daraus nun die gewünschte Ausgabedatei zu erzeugen.

Wichtig:

Für die Verwendung des CONSULTIMATOR benötigen Sie einen persönlichen Lizenzkey. Dieser wird in der Projektdatei eingetragen. Ohne Lizenzkey können Sie CONSULTIMATOR nur im Demo-Modus und nur zu Evaluationszwecken nutzen.

Dies geschieht, indem Sie folgende Seite aufrufen:

<https://consultimator.com>

Dort können Sie den CONSULTIMATOR Editor starten und die Projektdatei hochladen.

Alternativ können Sie, wenn sich die Projektdatei auf einem Server im Internet befindet und per http aufgerufen werden kann, diese auch direkt aufrufen:

<https://consultimator.com/process.php?t=https://meinserver.de/irgendeinverzeichnis/datei.cmp> (oder .xml)

Achtung: Wenn Sie diese Methode nutzen, ist ihr Lizenzschlüssel öffentlich sichtbar, wenn jemand Ihre Projektdatei findet und aufruft. Stattdessen lassen Sie bitte den Lizenzschlüssel in der Projektdatei leer und fügen einen weiteren Parameter zur obigen URL hinzu: *"&s=IhrLizenzschlüssel"*

Ist die Projektdatei fehlerhaft, d.h. verfügt sie über keine „saubere“ XML-Struktur, weil z.B. Elemente nicht sauber wieder geschlossen wurden, Elemente falsch geschachtelt wurden, die o.g. „verbotenen“ Zeichen verwendet wurden, ohne sie XML-verträglich umzuwandeln etc., erhalten Sie eine Fehlermeldung, die Ihnen regelmäßig ziemlich präzise angibt, wo der Fehler sich in der Projektdatei befindet.

Ist sie aus XML-Sicht fehlerfrei, wird sie anschließend „compiliert“, d.h. aus ihr wird die eigentliche HTML-Ausgabeseite erzeugt.

Wichtig:

Der Compiler selbst ist sehr „fehlertolerant“, d.h. wenn keine „groben“ Fehler gemacht wurden, wird die Ausgabeseite erstellt. Es kann dann aber immer noch sein, dass z.B. fehlerhafte ACONDITION-Angaben oder auch unsaubere HTML-Strukturen in den ATEXT-Angaben eine korrekte Ausgabe verhindern. Damit kein Frust aufkommt, ist folgendes Vorgehen zu empfehlen:



- **Verwenden Sie den CONSULTIMATOR Editor!** Mit dem Editor erstellen Sie die Projektdatei nicht von Hand, sondern komfortabel geführt mit deutlich weniger potenziellen Fehlerquellen!
- Bauen Sie Ihre Projektdatei Schritt für Schritt auf und kompilieren Sie diese immer wieder, um sich das Zwischenergebnis anzuschauen. Damit fallen Fehler sofort auf und können auch sofort beseitigt werden, bevor mehrere Fehlerquellen zusammenkommen und die Suche erschweren.
- Verwenden Sie im Webbrowser auch, wenn Sie sich damit auskennen, den „Entwicklermodus“, mit dem Sie den erzeugten Code untersuchen und so wertvolle Hinweise auf den möglichen Fehler finden können. Gerade fehlerhafte ACONDITION-Angaben und fehlerhafte COMPUTE-Aktionen fallen dort leicht auf, da sie häufig JavaScript-Fehler erzeugen, die dort sofort angezeigt werden.
- Manches Fehlverhalten gar kein technischer Fehler, sondern lediglich ein logisches Problem: wenn eine Aktion einfach das nicht macht, was sie soll, kann es sein, dass dies auf einem Denkfehler beruht. Häufig werden z.B. Größer-Kleiner-Vergleiche einfach „verkehrtherum“ definiert. Oder Sie haben durch eine grundsätzlich fehlerfreie COMPUTE-Aktion Ihren Hund dann doch wie in dem obigen Beispiel 123 Jahre statt 15 Jahre alt gemacht.

Hier kann es nützlich sein, sich solche berechneten Werte zwischendurch einmal über eine MESSAGE-Aktion anzeigen zu lassen. Steht in dem Wert nicht das drin, was Sie erwartet haben, sind Sie schon einen großen Schritt weiter.