

ALLES GEHABELT. ALLES GEREGLT.

# Belegleser Zusatz

Benutzerhandbuch





## Überblick

Belegleser Zusatz ist als Ergänzung zu Belegleser Basis zu sehen und richtet sich gezielt an Administratoren des Beleglesers. Ihnen werden zum besseren Verständnis notwendige Hintergrundinformationen und Definitionen bereitgestellt. Darüber hinaus besteht zur weiteren Unterstützung ein breitgefächertes Angebot an Schulungen im Hause HABEL oder bei Ihnen vor Ort.

## Schreibweisen in diesem Handbuch

Darstellung	Bedeutung
<b><u>HABEL-ANMERKUNGEN</u></b>	Hinweis, dass es sich bei der hier beschriebenen Funktion um ein zusätzliches Modul handelt, das eventuell nicht im Systemumfang enthalten ist.
	(Warn-)Hinweise bzw. zu beachtende Informationen werden mit diesem Symbol gekennzeichnet
	Hintergründe und Tipps werden mit diesem Symbol gekennzeichnet.

## Abweichungen

Die Abbildungen können im Detail von Ihrem HABEL<sup>®</sup> Dokumentenmanagement abweichen, da Funktionen enthalten sein könnten, die für Ihr System nicht erworben bzw. aktiviert sind. Generell ist es problemlos möglich, Ihr System zu erweitern. Sprechen Sie hierzu bitte Ihren Betreuer an.

### Hinweis

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung von HABEL reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Der Inhalt dieses Handbuchs kann Änderungen unterliegen, ohne dass dadurch eine Mitteilungspflicht seitens von HABEL abgeleitet werden kann.

### Haftung und Garantie

Das Handbuch wurde mit der größtmöglichen Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. HABEL übernimmt für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung oder Garantie.

HABEL GmbH & Co. KG  
Untere Hauptstraße 1  
D-78604 Rietheim-Weilheim  
Fon: +49 (0) 74 61 / 93 53 – 0  
Fax: +49 (0) 74 61 / 93 53 – 99  
www.habel.de  
Copyright © 2008 – 2014 by HABEL / Stand 14.10.2014 / Version Z 2

HABEL Dokumentenmanagement GmbH  
Niederlassung Schweiz  
Rheinstrasse 36  
CH-8212 Neuhausen am Rheinflall  
Fon: +41 (0) 52 / 6 74 81 – 51  
Fax: +41 (0) 52 / 6 74 81 – 50  
www.habel.ch



# Inhalt

Anlegen von Templates .....	4
1. Allgemein .....	4
2. Begriffserklärungen .....	4
3. Kurzüberblick .....	5
1. Scannen des Beleges (1. Schritt) .....	6
2. Speichern des Beleges (2. Schritt) .....	6
3. Anlegen der Bezeichnung des Templates (3. Schritt) .....	7
4. Zuordnung des Beleges als erste Seite des Templates (4. Schritt) .....	9
5. Identifizierung des Lieferanten (5. Schritt) .....	10
6. Definition der auszulesenden Felder/Spalten (6. Schritt) .....	14
7. Weitere Seiten hinterlegen (7. Schritt) .....	24
8. Sonderfall: MehrfachAnkerFeld .....	24
9. Sonderfall: MehrfachAnkerFixFeld .....	26
10. Sonderfall: Dummy .....	28
11. Sonderfall: Negieren von Werten .....	28
12. Ordner importieren/exportieren .....	28
Scripting .....	30
1. Allgemein .....	30
2. Eingriffspunkte .....	30
3. Programmiersprache .....	31
4. Aufbau eines Templatescriptes .....	36
5. Die wichtigsten Scripting Funktionen im Überblick .....	38
6. Auslesen von Positionen .....	40
7. Programmbedienung .....	42

# Anlegen von Templates

## 1. Allgemein

---

Templates (auch Schablonen genannt) werden angelegt, um das Auslesen von Beleginformationen zu optimieren. Viele Geschäftsvorgänge werden auf Belegen aufgedruckt, die vom Aufbau her identisch sind. Beleginformationen wie Belegdatum, Belegnummer, Artikeldaten und Preise sind immer an derselben Stelle zu finden. Diese Stellen können in einem Template definiert werden und dem Belegleser zur schnelleren und noch besseren Erkennung der Daten bereitgestellt werden.

Für das Anlegen der Templates steht das Programm hddhab404 zur Verfügung.



Belegleser - Template  
Verwaltung

## 2. Begriffserklärungen

---

In den nachfolgenden Beschreibungen werden im Zusammenhang mit dem Belegleser übliche Begriffe verwendet, die zunächst kurz erläutert werden:

### Template

Ist der englische Begriff für Schablone und gilt als Vorlage, die mit Inhalt gefüllt werden kann. In unserem Fall ist es ein Gerüst, um zu definieren, an welchen Stellen eines Dokuments Werte ausgelesen werden sollen.

### Freiform

Bei der Freiformerkennung wird der Rohtext ausgelesen und mit Begrifflichkeiten, die zur Identifizierung von Werten (z. B. Kreditoren, Rechnungsnummer etc.) definiert sind, verglichen. Wird der „gesuchte“ Begriff gefunden, erfolgt in unmittelbarer Nähe dazu (im Uhrzeigersinn: rechts, unten, links, oben) eine Suche nach dem Wert, der dann als ausgelesene Information verwendet wird.

### Identifizierung

Über die Identifizierung erkennt der Belegleser den zugehörigen Kreditoren/Debitoren. Hierfür wird ein Kennzeichen mit einer festen Koordinate hinterlegt, über die festgestellt wird, um welchen Kreditoren/Debitoren es sich handelt.

### Anker setzen

Ein Anker wird an der Stelle gesetzt, an der eine bestimmte Begrifflichkeit im Dokument zu finden ist und dieses somit identifiziert werden kann. Von diesem Anker aus werden weitere Informationen über das gesamte Dokument hinweg angesteuert und ausgelesen. An welcher Stelle ein Anker sitzt und wo sich die weiteren Daten auf dem Dokument befinden, wird über Koordinaten definiert.

### 3. Kurzübersicht

---

Nachfolgend wird in kurzen Erklärungen zusammengefasst, wie ein Template angelegt wird.

#### 1. Schritt SCANNEN DES BELEGES

Dies erfolgt per Scanerfassung KOFAX (300 dpi) über die Belegart für die Verarbeitung mittels Belegleser.

#### 2. Schritt SPEICHERN DES BELEGES

Der Beleg wird über das Programm *Belege erkennen* (hphab400) im TIFF-Format gespeichert. Hinweis: Erste, Folge und Letzte Seiten werden als einzelne Dateien gespeichert. Übrigens: Die Belege können auch direkt beim *Vorgänge bilden* (hphab401) als Vorlage gespeichert werden.

#### 3. Schritt ANLEGEN DER BEZEICHNUNG DES TEMPLATES

In der Templateverwaltung (hphab404) wird der Button *Neu* betätigt und durch Eingabe des Kreditoren- oder Debitorennamens im Feld Name ein Stammzugriff ausgelöst. Mit *Speichern* wird ein neuer Datensatz angelegt.

#### 4. Schritt ZUORDNUNG DES BELEGES ALS ERSTE SEITE DES TEMPLATES

Durch Auswahl des neuen Datensatzes und des Begriffes Identifizierung, kann der Button *Neu* betätigt werden. Es wird dann für die erste Seite des Templates die zugehörige TIFF-Datei ausgewählt und per Button *Speichern* hinterlegt.

#### 5. Schritt IDENTIFIZIERUNG DES LIEFERANTEN

Mit Drücken des Buttons *Neu* bei Auswahl des Begriffes *Erste Seite* unterhalb der Identifizierung wird festgelegt, mit welchem Kriterium bzw. welchen Kriterien der Kreditor/Debitor zur Verwendung dieses Templates erkannt wird. Hierbei sind wichtige Punkte zu beachten, die unter 8. in diesem Kapitel ausführlich beschrieben werden.

#### 6. Schritt DEFINITION DER AUSZULESENDEN FELDER/SPALTEN

Nach Auswahl des Begriffes *Beleg* wird über den Button *Neu* zuerst das Formular zur ersten Seite zugeordnet, der Anker hinterlegt und anschließend die Positionen definiert, an denen Werte ausgelesen werden. Hierbei sind wichtige Punkte zu beachten, die unter 9. in diesem Kapitel ausführlich beschrieben werden.



Werden Änderungen am Anker durchgeführt, muss stets über das Prüfsymbol geprüft werden, ob der Anker weiterhin richtig erkannt wird!

#### 7. Schritt WEITERE SEITEN HINTERLEGEN

Die Schritte 4 bis 6 werden für das Hinterlegen von Folgeseiten zum Template wiederholt.

## 1. Scannen des Beleges (1. Schritt)

---

Im Scanprogramm KOFAX wird der Beleg unter der Belegart gescannt, mit der die Verarbeitung im Belegleser erfolgt. Vorab ist zu prüfen:

- Handelt es sich um ein einseitiges Dokument?
- Wie sehen die zweite oder auch dritte und weitere Folgeseite des Dokuments aus?
- Gibt es fremdsprachige Versionen zu diesem Dokument?

Um eine optimale Erkennung erreichen zu können, ist beim Scannen darauf zu achten, dass die Scanoptimierungssoftware KOFAX im Einsatz ist und mit 300 dpi gescannt wird.

## 2. Speichern des Beleges (2. Schritt)

---

Im Programm *Belege erkennen* (hphab400) oder *Vorgänge bilden* (hphab401) wird jede Seite des Templates als eigene TIFF-Datei gespeichert.

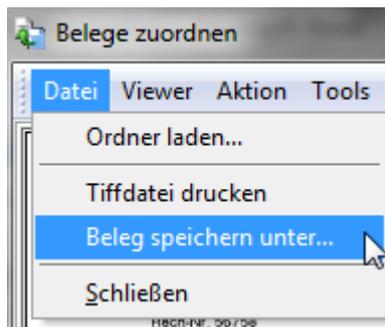


Abbildung 2: Menüauswahl

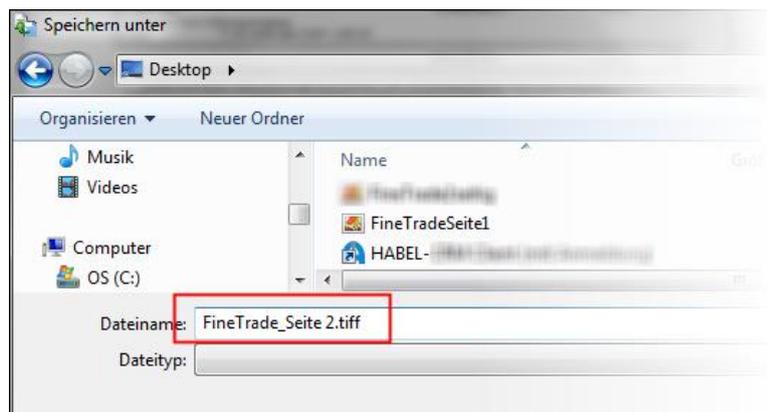


Abbildung 1: Speichervorgang

Für die nächsten Schritte ist das Programm hhab404 zu verwenden.



Belegleser - Template  
Verwaltung

### 3. Anlegen der Bezeichnung des Templates (3. Schritt)

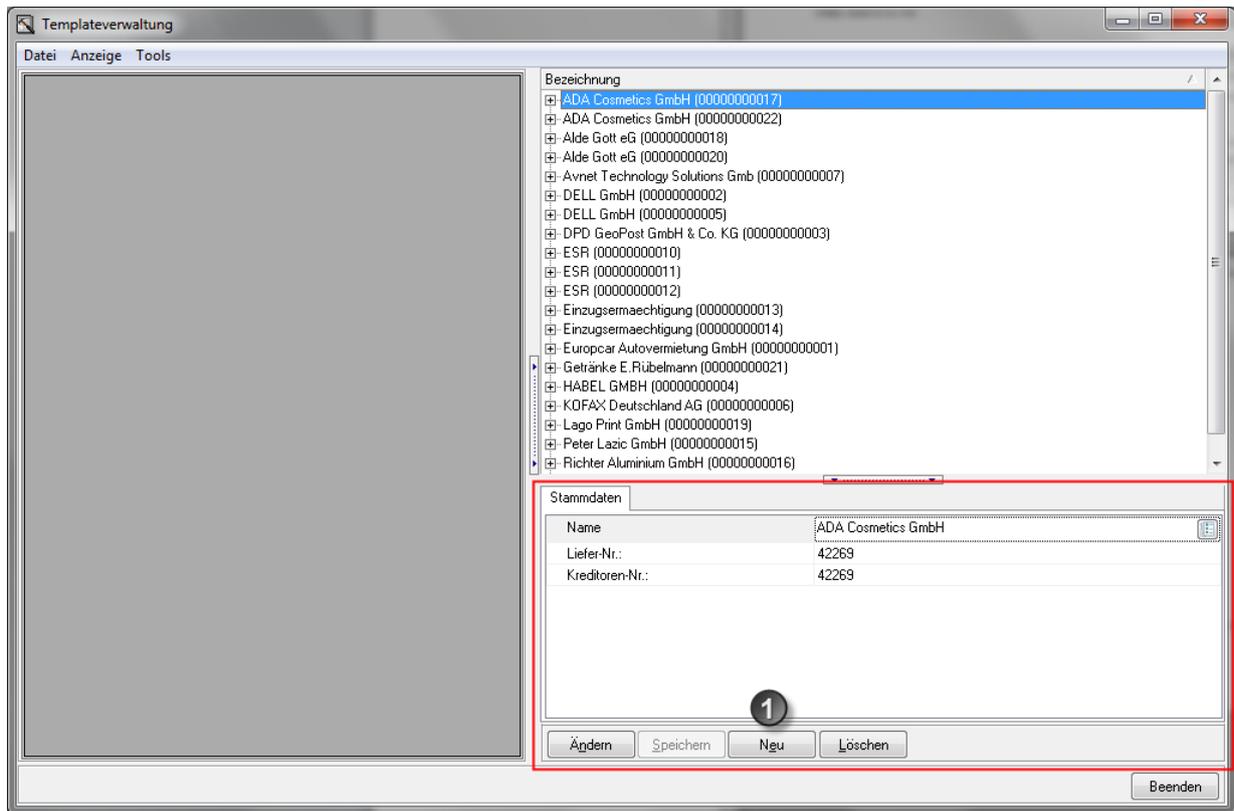
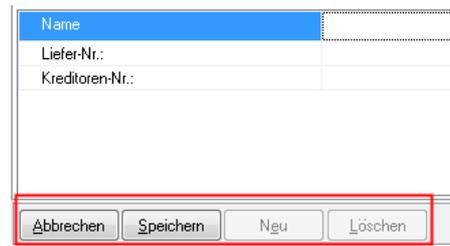


Abbildung 3: Startmaske Templateverwaltung

Betätigen Sie in der Templateverwaltung den Button **Neu (1)**. Der untere Bereich der Maske verändert sich dahingehend, dass die Felder geleert werden und die Buttons *Abbrechen*, *Speichern* und *Löschen* aktiv sind.



Geben Sie im Feld Name den Kreditoren-/Debitorennamen ein und klicken Sie auf das Symbol am Feldende. Sie erhalten eine Auswahlbox der angelegten Adressdaten. Nachdem Sie sich für die passenden Daten entschieden haben, übernehmen Sie diese durch Betätigen des OK-Buttons. Sie können den Namen (also die Bezeichnung des Templates) später auch ändern. Haben Sie beispielsweise ein Template für einen Lieferanten hinterlegt, das auch für andere Lieferanten verwendet kann, bezeichnen Sie dieses Template nach der Art z. B. SAP-Template.



Abbildung 4: Zugriff auf Stammdaten

Die Lieferanten-/Kreditoren-Nr. wird ergänzt. Speichern Sie nun die Daten über den Button *Speichern* und bestätigen das Anlegen des Datensatzes mit Ja. In der Baumstruktur sind die Daten zu sehen.

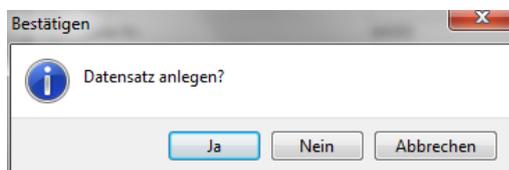


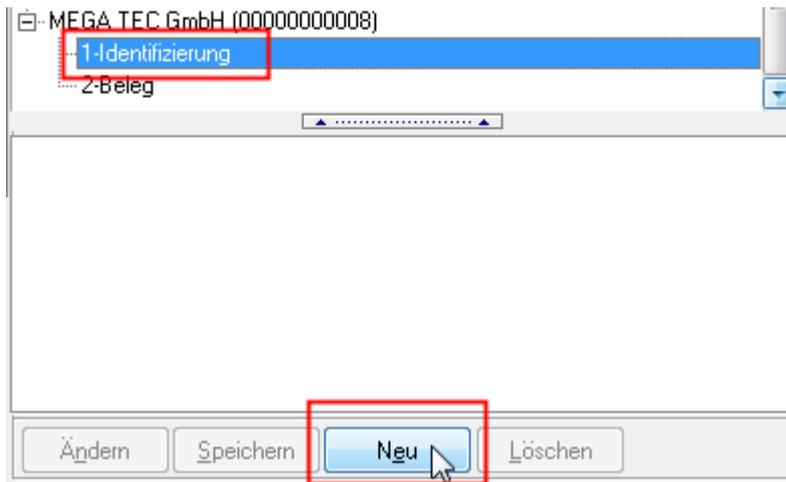
Abbildung 5: Bestätigungsabfrage



Abbildung 6: Zuordnen des Lieferanten

#### 4. Zuordnung des Beleges als erste Seite des Templates (4. Schritt)

Erweitern Sie den Eintrag des neuen Templates und markieren Sie den Begriff Identifizierung. Dadurch verändert sich die Maske dahingehend, dass der untere Bereich geleert und keine Stammdaten mehr angezeigt werden. Der Button *Neu* ist aktiv und wird angeklickt.



Innerhalb der Maske Identifizierung wird zuerst definiert, welche Seite hinterlegt werden soll **(1)** und das zugehörige Formular hochgeladen **(2)**. Über den Button *Speichern* wird der Datensatz angelegt.

Abbildung 7: Belege identifizieren

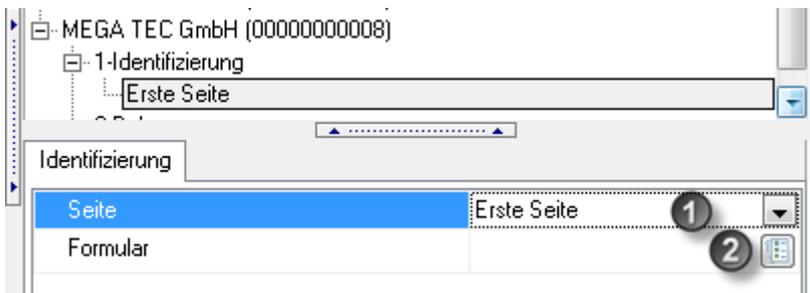


Abbildung 8: Definieren *Erste Seite*

## 5. Identifizierung des Lieferanten (5. Schritt)

Für die Identifizierung der ersten Seite des Templates wird festgelegt, an welcher Stelle welcher Begriff ausgelesen werden muss. Klicken Sie hierfür nach Auswahl der Begrifflichkeit Erste Seite unterhalb der Identifizierung auf den Button **Neu**.

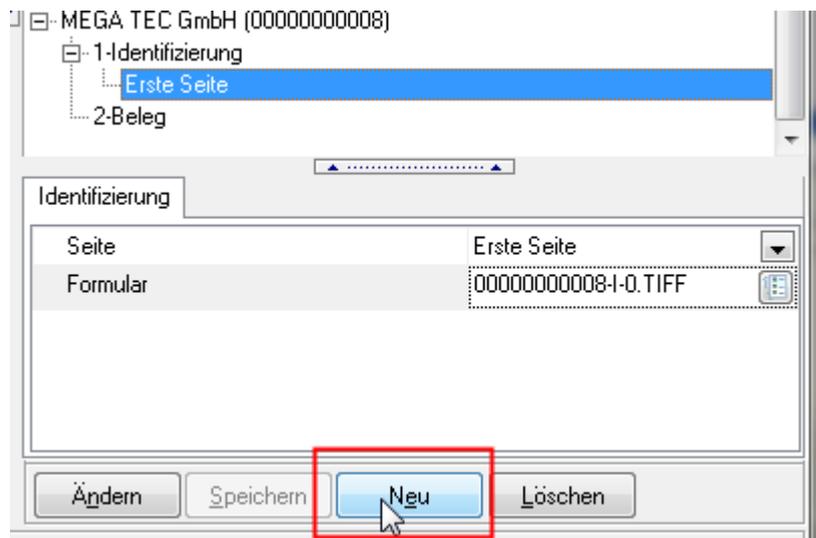


Abbildung 9: Lieferant identifizieren

Es erscheint die Lasche Identifizieren mit den beiden Kriterien *Übereinstimmung zu* und *Attribut*. Über *Übereinstimmung zu* wird eingestellt, wie viel % des erkannten Wertes mit dem vorgegebenen *Attribut* übereinstimmen muss. Beispielsweise ist eine Übereinstimmung von 70% bei einem Begriff von 10 Buchstaben definiert, wodurch 7 Buchstaben übereinstimmen müssen.

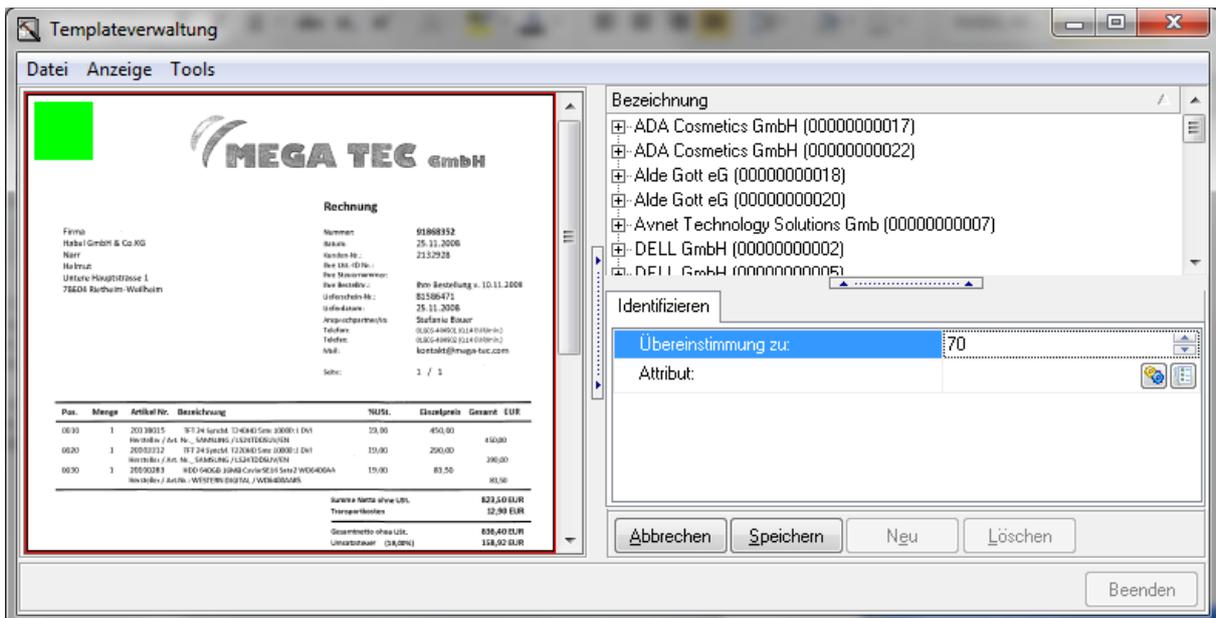


Abbildung 10: Übereinstimmung zu festlegen



Das zur Templatehinterlegung verwendete Dokument ist ein Muster. Bei der Definition der Kriterien/Attribute für die Identifizierung ist daher zu beachten, dass die zukünftig eingehenden Dokumente evtl. abweichen (schiefer Druck, schlechtere Qualität, etc.). Mit der grünen Fläche wird der Bereich festgelegt, innerhalb dessen sich das Attribut zur Identifizierung des Lieferanten befindet. Die Fläche sollte daher nicht zu eng um den Bereich liegen, innerhalb dessen sich das Attribut befindet.

Um ein Attribut zu hinterlegen, muss ein Bereich und der zu findende Text definiert werden. Hierzu wird das Symbol angeklickt und der Text, der sich innerhalb der grünen Fläche befindet, eingegeben.

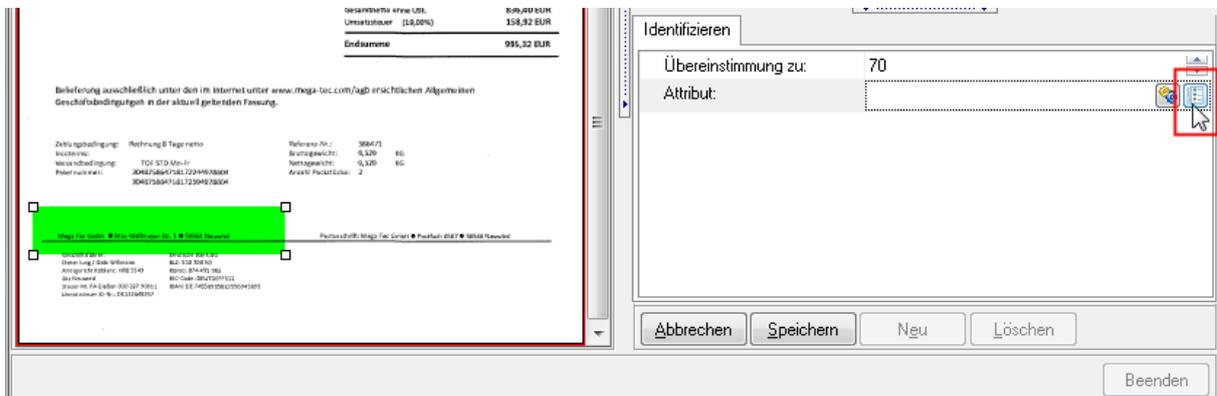


Abbildung 11: Textbereich für das Attribut definieren

In der Attributliste können mehrere Texte hinterlegt werden, die als Attribut zur Identifizierung dienen. Sie werden im Feld Attribut mit | getrennt dargestellt.



Als Attribut eignen sich Begriffe, die sich auf dem Formular des Lieferanten immer an dieser Stelle befinden (Bereich wird mit der grünen Fläche definiert) und ihn somit eindeutig identifizieren. Es können Bestandteile des Briefbogens sein oder auch ein Begriff, der aufgedruckt ist. Wichtig ist, dass dieser immer an derselben Stelle steht, aus einem Wort besteht, gut erkennbar und eindeutig ist. Es können auch mehrere Begriffe, die in Kombination erscheinen, für die Identifizierung herangezogen werden.

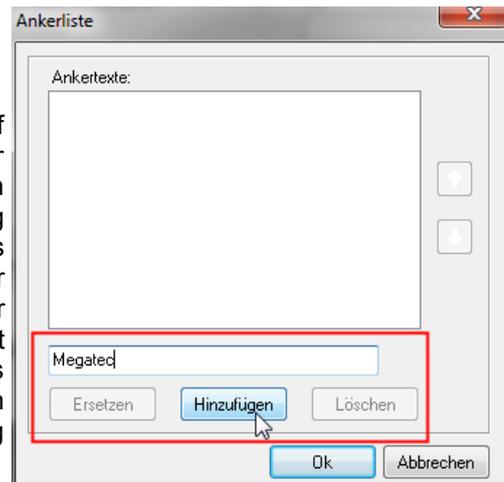


Abbildung 12: Attributliste

Durch Klick auf das Prüfsymbol wird geprüft, inwieweit der Belegleser das Attribut innerhalb des grünen Bereiches finden kann.

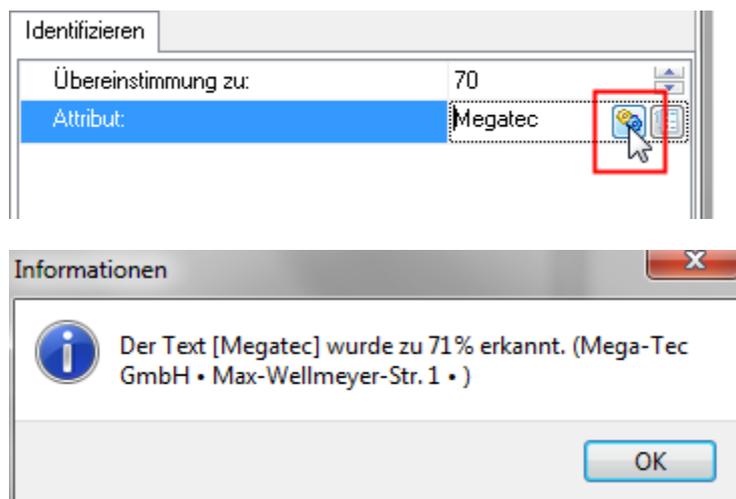


Abbildung 13: Prüfung Erkennbarkeit des Attributtextes

Bitte beachten Sie: diese Attribute dürfen nur für diesen Lieferanten gültig/passend sein, um zu vermeiden, dass falsche Templates herangezogen werden.



Bei der Hinterlegung von Begriffen ist unbedingt auf Groß- und Kleinschreibung zu achten. Sind zu Lieferanten mehrere Templates hinterlegt, muss auf spezielle Unterscheidungsmerkmale in der Definition der Identifizierungsattribute eingegangen werden.

Nachfolgend ein paar Beispiele, was als Identifizierungsmerkmale verwendet werden kann

- Firmenbezeichnung aus dem Briefkopf
- Firmenanschrift in Kombination (Straße und Ort)
- Homepage-Adresse aus der Kopf- oder Fußzeile

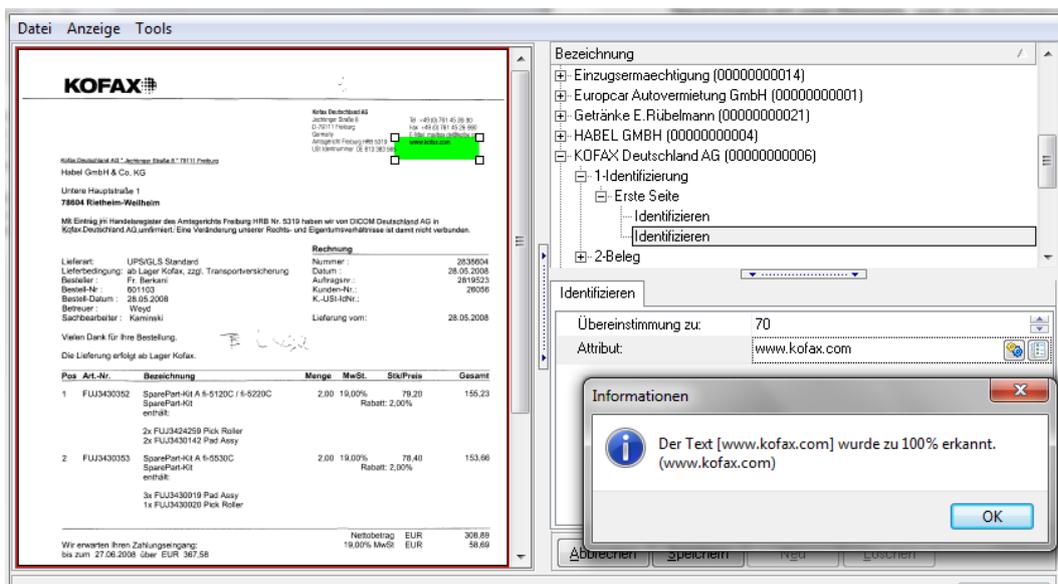


Abbildung 14: Identifizierung anhand Webadresse

Werden mehrere Attribute hinterlegt, müssen alle Attribute für die Identifizierung erkannt werden (in Abhängigkeit der jeweils hinterlegten *Übereinstimmung zu*).



Nicht als Identifizierungsmerkmal verwendet werden sollten

- Logos, die keinen eindeutigen Text sondern Symbole beinhalten
- Adresszeilen, die recht klein und daher schlecht erkannt werden

## 6. Definition der auszulesenden Felder/Spalten (6. Schritt)

Um Felder/Spalten zu definieren, anhand denen Daten ausgelesen werden sollen, muss zunächst 2-Beleg unterhalb der Templatebezeichnung ausgewählt werden.

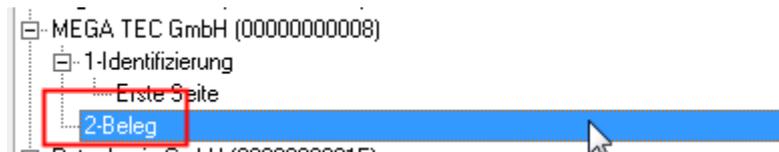
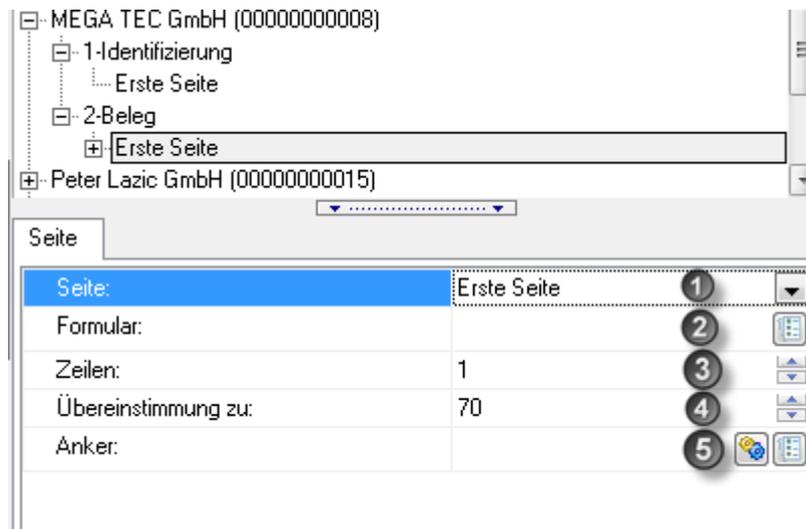


Abbildung 15: Definition Felder/Spalten

Über den Button *Neu* in der unteren Maskenhälfte wird die Eingabemaske für den Beleg aktiviert.



Es wird ausgewählt, um welche Seite es sich handelt **(1)** und das zugehörige Formular hochgeladen **(2)**. Hat das Formular Positionen, die immer aus derselben Anzahl an Zeilen bestehen, kann unter dem Kriterium **(3)** die Anzahl dieser hinterlegt werden. Sind es unterschiedliche Zeilenanzahlen, wird hier 1 eingetragen. Es wird außerdem ein Anker **(5)** gesetzt, um die Erste Seite des Beleges zu identifizieren und die Übereinstimmung zu **(4)** dafür hinterlegt.

Abbildung 16: Beleg - Erste Seite

Der Anker ist das Attribut, an dem erkannt wird, um welchen Beleg es sich handelt. Von dieser Stelle aus werden die weiteren Informationen angesteuert und ausgelesen. Hierzu wird das türkisfarbene Feld an die Stelle gesetzt, an der immer derselbe Begriff steht und den Beleg identifiziert (in unserem Fall der Begriff Rechnung). Im Feld Anker wird der zu erkennende Text über die Ankertext-Schaltfläche definiert und per Prüfsymbol geprüft, inwieweit der Begriff erkannt wird (bitte beachten Sie Groß- und Kleinschreibung). Bei der Wahl des Ankers sollte beachtet werden, dass es sich um einen Begriff handelt der angedruckt wird, denn wenn der Druck schief auf dem Briefformular erfolgt, werden die weiteren auszulesenden Passagen trotzdem vom Anker aus angesteuert und erkannt. Es kann auch immer nur ein Anker für den Beleg gesetzt werden.

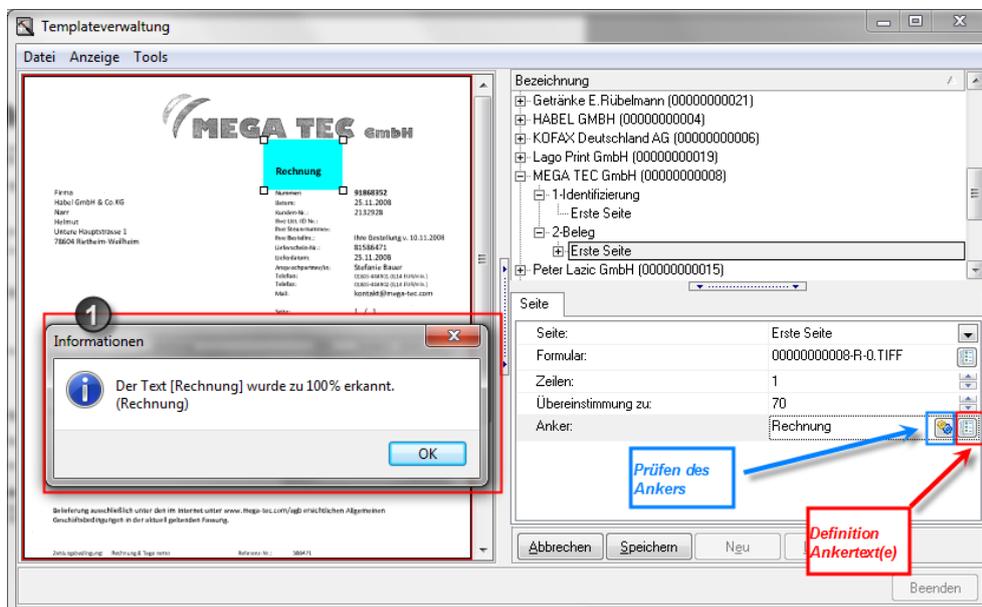


Abbildung 17: Erkennen des Ankertextes

 Das Erkennen dieses Ankers ist sehr wichtig, da in Abhängigkeit dessen die Platzierung der weiteren zu erkennenden Werte errechnet wird. Werden Änderungen an diesem Anker durchgeführt ist stets neu zu erkennen (Prüfsymbol).

Nachfolgend ein paar Beispiele, was als Anker verwendet werden kann

- Bezeichnungen, die bei diesem Kreditor/Debitor **fix** an dieser Stelle angebracht sind

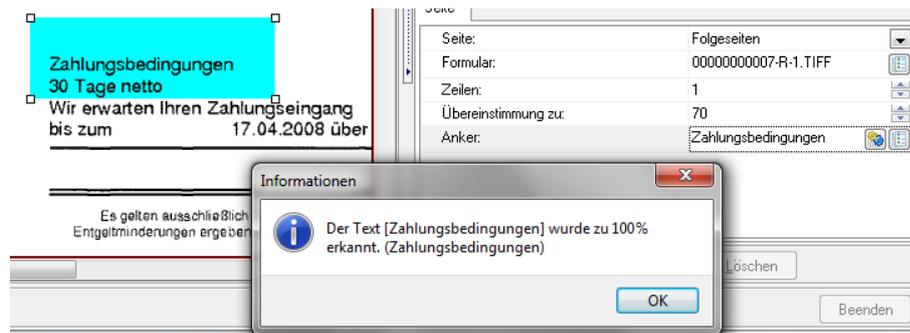


Abbildung 18: Beispiel für Ankertexte: Zahlungsbedingung

- Adressfelder

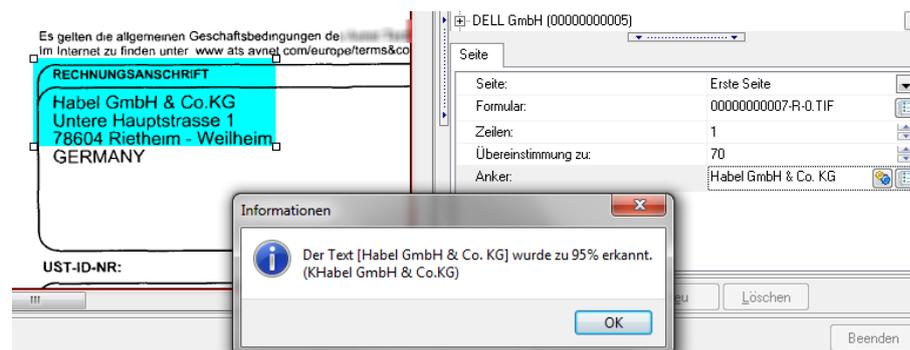


Abbildung 19: Beispiel für Ankertexte: Rechnungsanschrift

- Begriff Rechnungsdatum oder Belegdatum, das diesem Kreditor zugeordnet werden kann

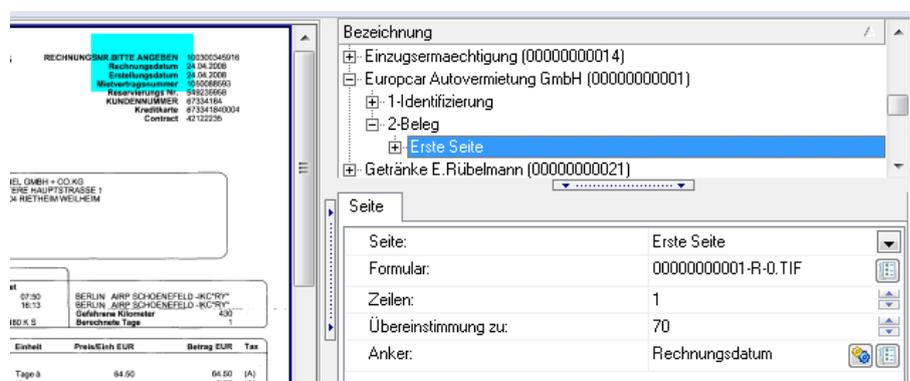


Abbildung 20: Beispiel für Ankertexte: Datum

- Begriff Rechnung oder wie hier Rechnungsnummer (Beispiel abweichender Folgeseite)



Abbildung 21: Beispiel für Ankertexte: Begriff Rechnung

Nicht als Anker verwendet werden sollten

- Adressfeld, sofern mehrere Mandanten hinterlegt sind
- Belegpassagen, die Formularbestandteil sind, da sonst bei verschobenem Druck die weiteren Beleginformationen nicht mehr angesteuert werden können

Sollen alternative Begriffe definiert werden, die für dasselbe Template mit derselben Ausrichtung stehen, kann dies über die Ankerliste erfolgen. In unserem Fall könnte an der Stelle des Begriffes RECHNUNG auch GUTSCHRIFT erkannt werden, der Belegleser müsste sich in beiden Fällen gleich verhalten (d. b. ausgehend vom Begriff RECHNUNG bzw. GUTSCHRIFT werden die weiteren Beleginformationen angesteuert und ausgelesen). Im Feld Anker werden die Begrifflichkeiten durch | voneinander getrennt.



Abbildung 22: Mehrere Ankertexte

Nachdem der Anker gesetzt wurde, werden die Felder und Spalten definiert, wo die Werte ausgelesen werden sollen. Klicken Sie hierzu nach Auswahl des Begriffes Felder unter 2-Beleg | Erste Seite auf den Button **Neu**.

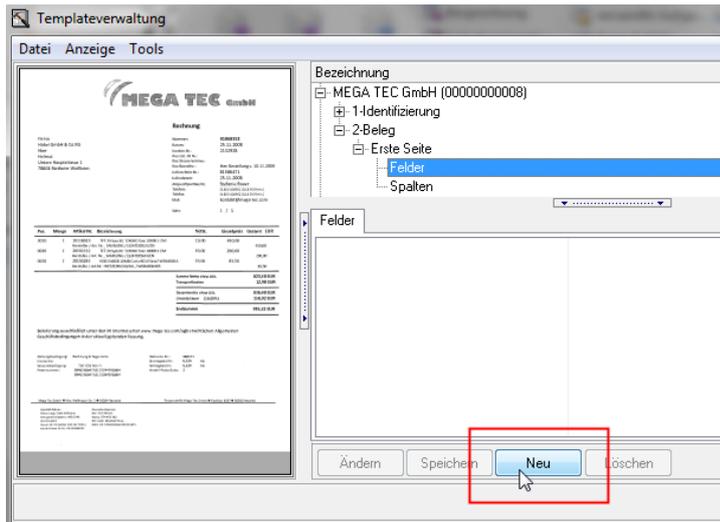


Abbildung 23: Definition der Felder

Die Maske ändert sich dahingehend, dass Informationen zum auszulesenden Feld eingegeben werden können. Mit der gelben Schattierung in der Beleganzeige wird der Bereich markiert, innerhalb dem die Informationen zum zugeordneten Datenfeld stehen.

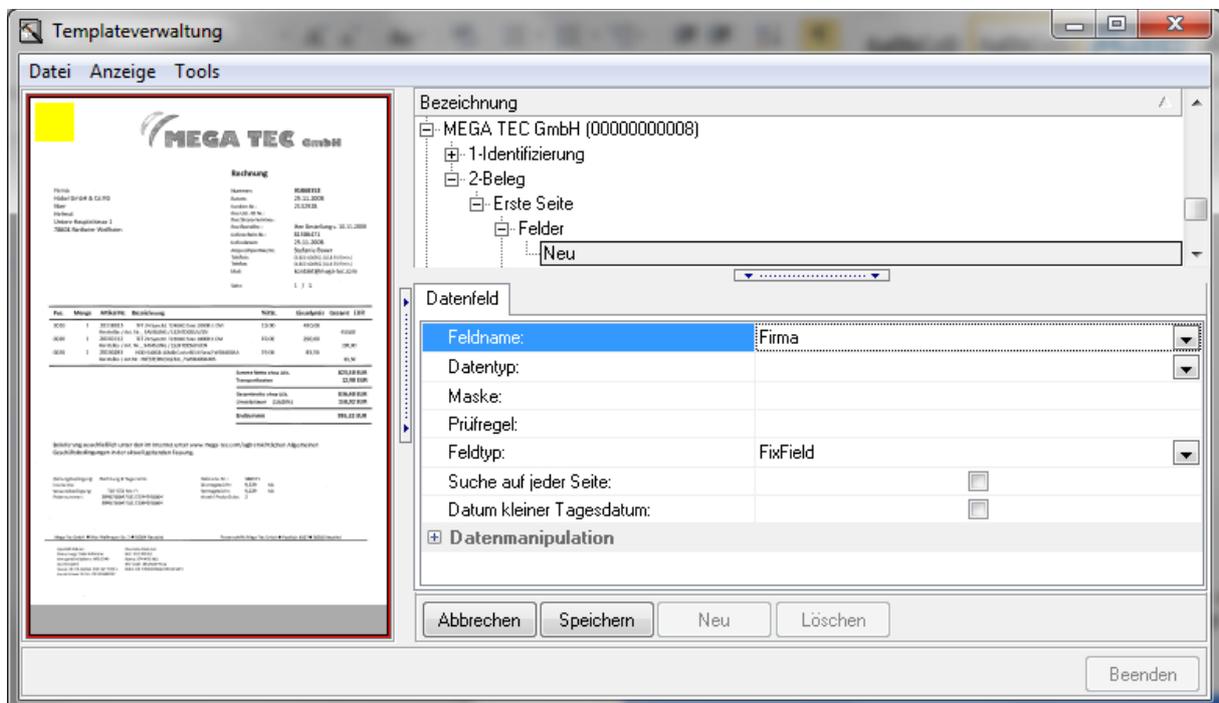


Abbildung 24: Definition Felder

Hier ist zu beachten, dass der Rahmen **eng** um die Werte gelegt wird **(1)**. Wird der Bereich zu großzügig definiert, kann es leicht geschehen, dass Zeichen oder Werte in unmittelbarer Nähe mit ausgelesen werden und somit zu Falscherkennungen führt. Hinweis: könnte der auszulesende Wert jedoch länger als im Muster sein, so ist der Bereich entsprechend der maximalen Möglichkeit zu gestalten **(2)**.

Nummer :  
Datum :  
Auftragsnr.:

**1** 2838604  
28.05.2008  
2819523

Abbildung 26: Definition Rahmen

Nummer :  
Datum :  
Auftragsnr.:

**2** 2838604  
28.05.2008  
2819523

Abbildung 25: Definition Rahmen

**Rechnung**

Nummer: **91868352**  
 Datum: 25.11.2008  
 Kunden-Nr.: 2132928  
 Ihre Ust.-ID Nr.:  
 Ihre Steuernummer:  
 Ihre Bestellnr.: Ihre Bestellung v. 10.11.2008  
 Uferschein-Nr.: 81586471  
 Lieferdatum: 25.11.2008  
 Ansprechpartner/In: Stefanie Bauer  
 Telefon: 01805-404901 (0,14 EUR/min.)  
 Telefax: 01805-404902 (0,14 EUR/min.)  
 Mail: kontakt@mega-tec.com

Seite: 1 / 1

%USt.	Einzelpreis	Gesamt	EUR

DELL GmbH (000000000000)  
 DPD GeoPost GmbH & Co. KG (000000000003)  
 ESR (00000000010)  
 ESR (00000000011)

Datenfeld

Feldname:	<b>1</b> FBeNr
Datentyp:	<b>2</b> Text
Prüfregel:	<b>3</b> Numerisch
Feldtyp:	Betrag
Suche auf jeder Seite:	<input type="checkbox"/>
Datum kleiner Tagesdatum:	<input type="checkbox"/>

Datenmanipulation

Abbildung 27: Definition Datentyp

Wählen Sie den Feldnamen **(1)** aus der Liste aus und ordnen einen Datentyp **(2)** zu. Hier sind Werte vordefiniert, die für Ihr System zutreffen. Die Prüfregel **(3)** wird automatisch gefüllt, kann aber manuell geändert werden. Auch wenn der Datentyp geändert wird, bleibt die bisher hinterlegte Prüfregel bestehen.

	Um diese zu aktualisieren, wählen Sie die Tastenkombination STRG + F5. Die Prüfregeln basieren auf regulären Ausdrücken.
---	--

Nachfolgend eine Übersicht der regulären Ausdrücke:

Zeichen	Bedeutung
.	( <i>Punkt</i> ) Ein Punkt steht für ein beliebiges Zeichen.
[x-y]	Gibt einen Zeichen Bereich an, auf den geprüft wird. Bsp.: [1-3] gültig wäre nun 1,2,3
[^x-y]	Negiert den Zeichen Bereich, d.h. alle Zeichen bis auf x-y sind gültig
\d	Dezimalziffern
\D	Alle Zeichen außer Dezimalziffern
\s	Whitespace (Leerzeichen, Tabulatorzeichen etc.)
\S	Alle Zeichen außer Whitespaces
\w	alle „Wort“-Zeichen (Buchstaben, Ziffern, Unterstrich)
\W	Alle „Nicht-Wort“ Zeichen
\n	Zeilenumbruch
\r	Wagenrücklauf
\\	Ein „\“
\t	Tabulator
?	Prüft ob ein Zeichen kein-/einmal vorkommt
+	Prüft ob ein Zeichen mind. Einmal vorkommt
*	Prüft ob ein Zeichen keinmal oder beliebig oft vorkommt
{x}	Prüft ob ein Zeichen genau x mal vorkommt
{x,}	Prüft ob ein Zeichen min x mal vorkommt
{,x}	Prüft ob ein Zeichen maximal x mal vorkommt
{x,y}	Prüft ob ein Zeichen min xmal aber höchstens y mal vorkommt
^	Prüft auf den Zeilenanfang
\$	Prüft auf das Zeilenende
\.	( <i>Backslash Punkt</i> ) Prüft auf einen Punkt
(xxx)	Beschreibt einen Block (Term) von regulären Ausdrücken. Nehmen wir mal an, Sie wollen prüfen, ob in einer Zeichenkette genau einmal zwei dezimale Zahlen vorkommen: $\d{2}$ um zu prüfen, ob diese Bedingung einmal oder keinmal zutrifft, müssen wir diesen Block in Klammern schreiben ( $\d{2}$ )?
	„Oder“ - Damit kann man mehrere Ausdrücke untersuchen Bsp.: $(M m)?$ überprüft ob M oder m ein oder keinmal vorkommt.



Abbildung 28: Definition Feldtyp

Als Feldtyp gibt es drei hinterlegbare Arten

FixField	Der Wert steht immer genau an dieser Stelle.
Anchor Field	Der Wert steht unmittelbar in der Nähe (rechts, links, darüber oder darunter) diesen Begriffs, dessen Platzierung über einen Bereich definiert wird. Die Eingabedaten erweitern sich um den Ankertext, die <i>Übereinstimmung zu</i> und die Position in Abhängigkeit des gefundenen Ankertextes (links, rechts, oben unten)
ICR-FixFeld	Der auszulesende Wert steht genau an dieser Stelle, es handelt sich jedoch um einen handschriftlichen Wert. Detailsinstellungen sind durch die Erweiterung nach Auswahl des ICR-FixFeld in Bezug auf Art der handschriftlichen Darstellung und ggfs. Anzahl der vorhandenen Zellen möglich. Hinweis: Für die ICR Erkennung ist eine separate ICR-Lizenz (ABBYY) erforderlich.

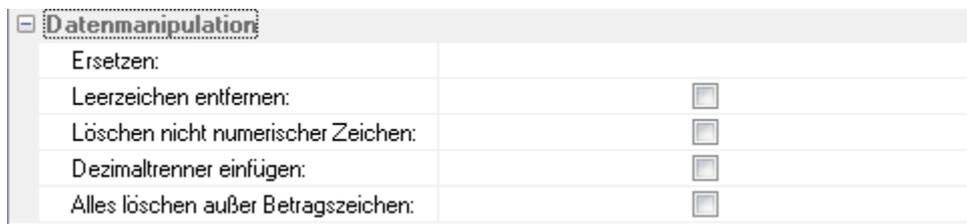


Abbildung 29: Datenmanipulation

Über die Datenmanipulation können die erkannten Werte beeinflusst werden

Ersetzen	<p>Werden Zeichen wie beispielsweise O und 0 oder B und 8 falsch erkannt, können diese automatisiert ersetzt werden. Ersetzregeln können für beliebige Zeichen herangezogen (z. B. Punkt . und leer). Die Ersetzregel wird in einer eckigen Klammer und getrennt durch , hinterlegt (alternativ können auch ASCII Werte verwendet werden – siehe hierzu unter im Menü <i>Tools – ASCII Chart</i>)</p> <p>Bsp: [O,0] [B,8] [.,]</p>
Leerzeichen entfernen	Entfernt alle Leerzeichen im Feldinhalt
Löschen nicht numerischer Zeichen	Entfernt alle nicht numerischen Zeichen außer Punkt, Minus und Komma (.,)
Dezimaltrenner einfügen	<p>Fügt ein Komma zwei Stellen von rechts ein</p> <p>Bsp. Erkannt wurde 2500 – daraus wird 25,00</p>
Alles löschen außer Betragszeichen	Entfernt alle nicht numerischen Zeichen auch Punkt, Minus und Komma (.,)

In ähnlicher Weise werden Spalten angelegt, innerhalb derer Werte auszulesen sind. Spalten werden hauptsächlich für das Erkennen von Positionsdaten wie z. B. Artikelbezeichnungen, Artikelpreise und –mengen verwendet.

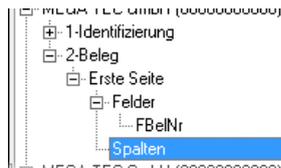


Abbildung 30: Definition Spalten

Nachdem unter dem 2-Beleg | Erste Seite der Begriff Spalten ausgewählt wurde, kann die Eingabemaske mit dem Button Neu aktiviert werden.

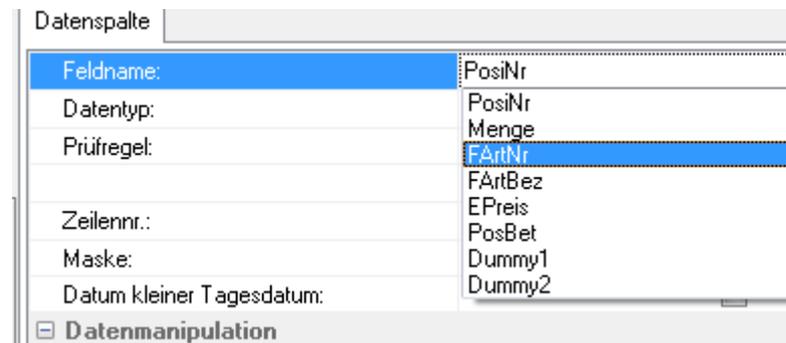


Abbildung 31: Definition Spalten

Unter *Feldname* wird ausgewählt in welche Datenbankfelder die erkannten Werte geschrieben werden sollen. Die gelbe Markierung wird über den Bereich gelegt, innerhalb dem die Werte platziert sind. Bitte beachten Sie, dass Sie für die Templateeinrichtung ein Muster verwenden, das von den tatsächlich eingehenden Dokumenten abweichen kann. Unter diesen Gesichtspunkten ist der Bereich zu gestalten. In diesem Fall könnte die Rechnung noch weitere Artikelnummern beinhalten, die dann unterhalb der aufgeführten Nummer stehen würden. Die Spalten müssen daher den gesamten Bereich abdecken, innerhalb dem die Werte stehen bzw. stehen könnten. Auch für die Spalten können *Datentypen*, *Prüfregeln* und *Datenmanipulationen* definiert werden. Wenn im Template eine mehrzeilige Position definiert wurde, kann über *Zeilennr.* festgelegt werden, in welcher Zeile der Wert auszulesen ist.

## 7. Weitere Seiten hinterlegen (7. Schritt)

---

Die genannten Schritte müssen für jede abweichende Folgeseite ebenfalls durchgeführt werden.

## 8. Sonderfall: MehrfachAnkerFeld

---

Für das Auslesen von Werten, die an einer bestimmten Stelle stehen, werden Fixfelder angelegt, für das Auslesen von Werten, die in unmittelbarer Nähe eines bestimmten Begriffes stehen, Ankerfelder. Es besteht nun auch die Möglichkeit, dass Werte ausgelesen werden, die auf derselben Zeile eines Ankers stehen, aber in größerem Abstand dazu. Hierzu können MehrfachAnkerFelder definiert werden (innerhalb der Definition *Felder* (1) wird der Feldname (2) MehrfachAnkerFeld (3) ausgewählt).

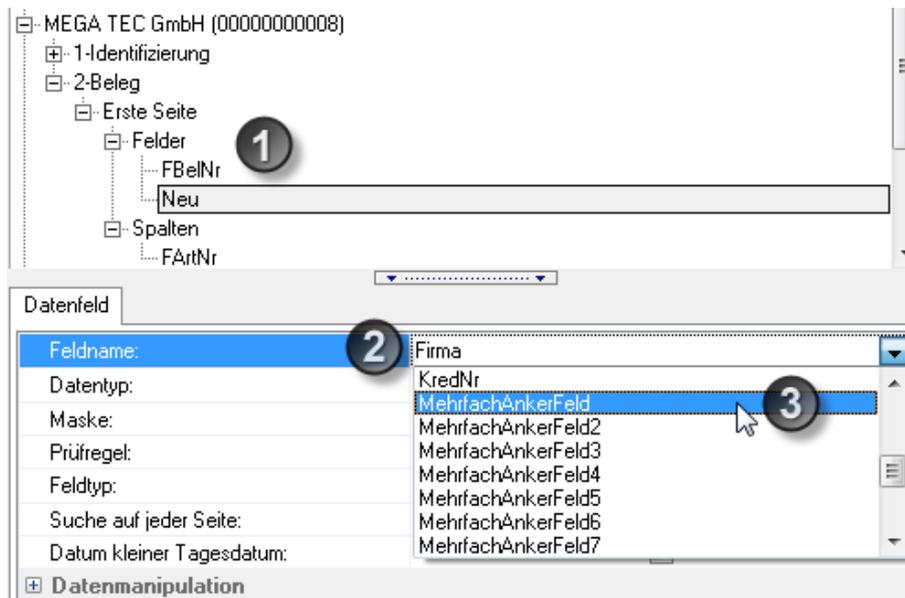


Abbildung 32: MehrfachAnkerFeld



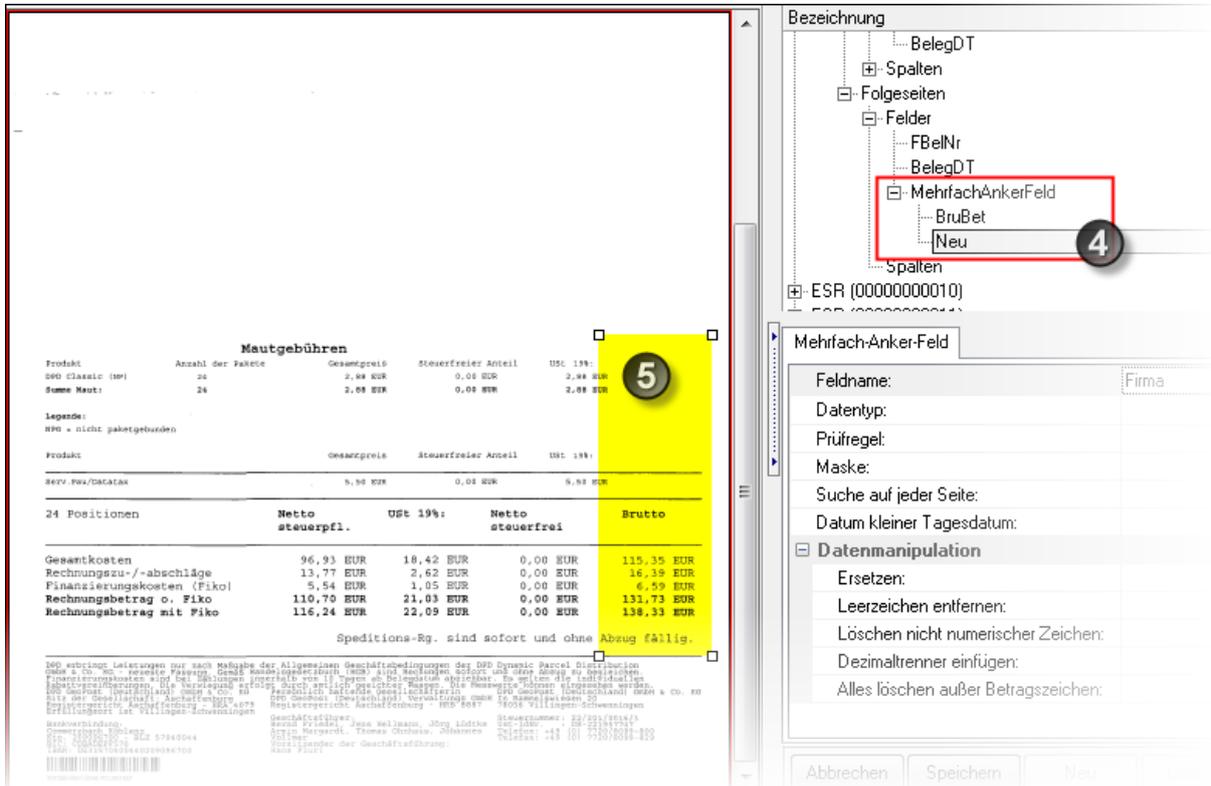


Abbildung 35: Bereich definieren

Der gelbe Bereich (5) hat in etwa denselben Umfang wie vorher der Bereich für das Setzen des Ankers. Die Spalte signalisiert dem Belegleser, dass sich in dieser Spalte der auszulesende Wert befindet und zwar **auf derselben Ebene/Zeile** wie der Anker. Es müssen hier lediglich noch die bekannten Definitionen für Feldname, Datentyp etc. durchgeführt werden.

## 9. Sonderfall: MehrfachAnkerFixFeld

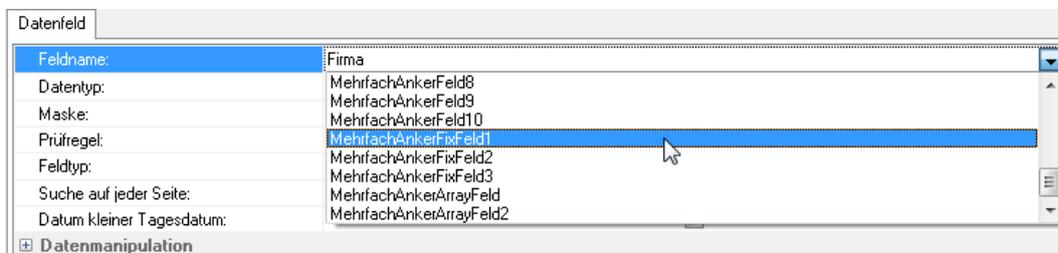


Abbildung 36: MehrfachAnkerFixFeld

Kann das MehrfachAnkerFeld nicht eingesetzt werden, weil das auszulesende Feld nicht in derselben Zeile steht wie der Anker, sondern an fixer Stelle in Abhängigkeit des Ankers, kann das MehrfachAnkerFixFeld eingesetzt werden. Beim MehrfachAnkerFixFeld (innerhalb der Definition *Felder*) wird eine Spalte mit Anker definiert, sowie der auszulesende Wert.

In unserem Fall soll der zu zahlende Betrag in EURO **(1)** ausgelesen werden, wobei der Begriff Zahlungsbedingung **(2)** der Anker ist. Zunächst wird festgelegt, in welcher Spalte der Ankertext zu finden ist (gelbe Markierung). Der Begriff kann sich innerhalb dieser Spalte an beliebiger Stelle befinden (siehe rote Markierung). Von dort aus wird der Betrag „angesteuert“, dessen Wert ausgelesen werden soll (siehe blaue Markierung).

Abbildung 37: Definition der Bereiche

Zusätzlich ist der Ankertext mit dem türkisfarbenen Bereich zu definieren. Bitte legen Sie diesen Bereich eng um den Begriff, damit Falscherkennungen vermieden werden.

Ausgangssituation: Es gelten ausschließlich unsere umsieg abgedruckt Ergebnis: Es gelten ausschließlich unsere umsieg abgedruckt

Anschließend kann über den Button *Neu* das zugehörige FixFeld definiert werden. Das FixFeld ist jedoch abweichend zum MehrfachAnkerFeld nicht als Spalte, sondern als konkreter Bereich zu markieren (auch hier ist darauf zu achten, dass der Rahmen eng gelegt wird; ggf. ist auf Platz für längere Zeichenauswahl zu achten). Wie bei den anderen Feldern ist Feldname, Datentyp und ggf. Datenmanipulation zu ergänzen.

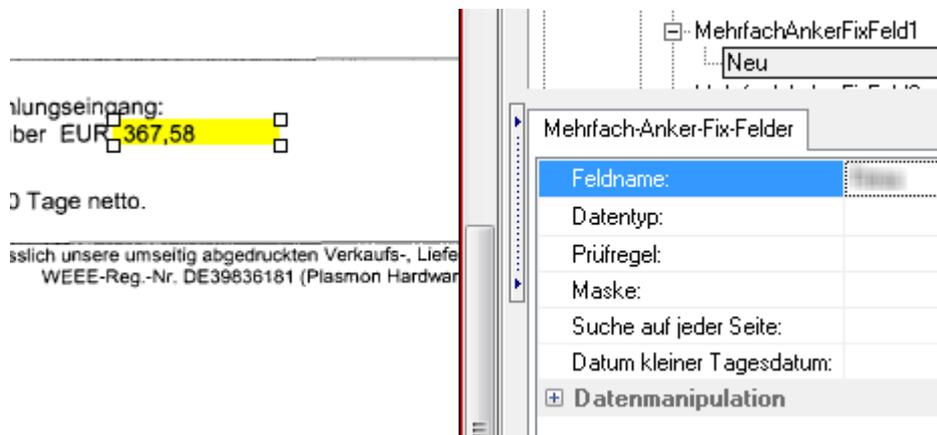


Abbildung 38: Definition des Fixfeldes

## 10. Sonderfall: Dummy

Dummy (innerhalb der Definition *Spalten*) ist als Feldname ausgewählt, wenn für den auszulesenden Wert kein Datenbankfeld gefüllt werden soll, aber dieser für das Bewerten von hinterlegten Regeln (Scripten) bedeutsam ist. Beispielsweise wird ein Dummy eingesetzt, wenn die Regel (das Script) hinterlegt wurde, dass Artikelpositionen nicht gültig sind, wenn das Feld Menge leer ist (Menge wird in unserem Beispiel nicht als Datenbankfeld geführt und daher als Dummy ausgelesen).

## 11. Sonderfall: Negieren von Werten

Bei der Identifizierung kann für Bereiche im Template hinterlegt werden, dass wenn an der markierten Stelle (in diesem Bereich) der definierte Wert nicht steht, bestimmte Regeln eintreffen (z. B. es sich um einen bestimmten Lieferanten oder ein Template eines Lieferanten handelt). Dazu wird der Bereich ausgewählt und in der Prüfregel der Begriff mit vorangestelltem Dollarzeichen \$ hinterlegt.

## 12. Ordner importieren/exportieren

Um HABEL den gesamten Ordner mit den gescannten Stapeln und erkannten Werte zur Verfügung stellen zu können (z. B. zur Lösungsfindung bei Problemen), kann dieser exportiert werden.

Hierfür steht ein separates Programm (hphab405) bereit, das die Daten für den Transfer im HABEL-Verzeichnis Ordner Transfer/Export als zip-Datei mit aktuellem Tagesdatum exportiert. Dieser Ordner ist an HABEL zu leiten.



Die Funktion *Importieren* ist nur nach Rücksprache mit HABEL oder für Testzwecke im Testsystem vorgesehen. Durch einen Import überschreiben die Daten, die zum Zeitpunkt des Exports bestanden haben, die aktuell vorhandenen Daten.

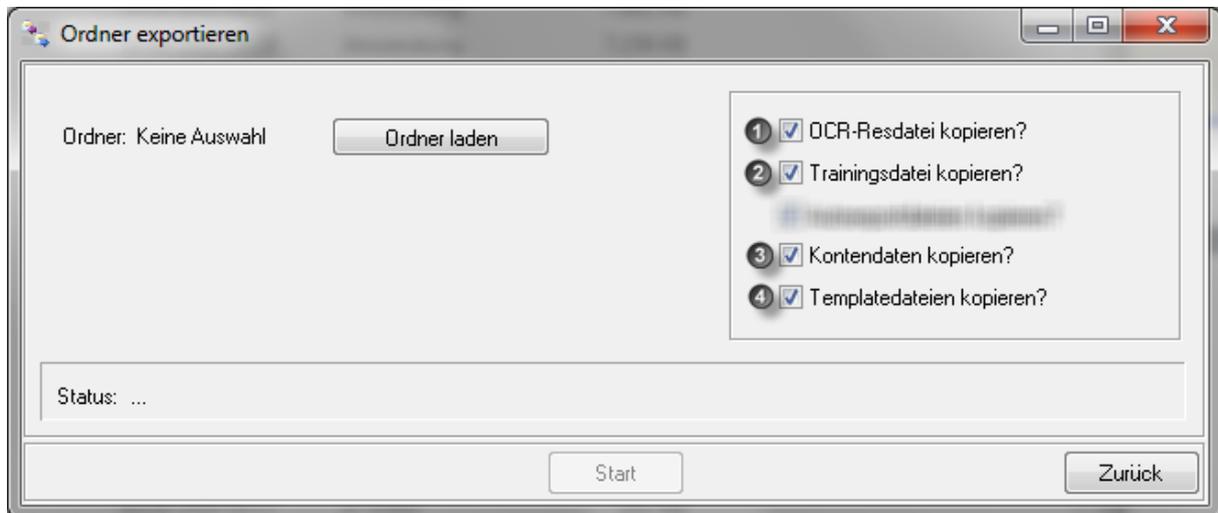


Abbildung 39: Exportieren von Ordnern

Sie haben die Möglichkeit verschiedene Teile des Beleglesers zu exportieren:

<b>(1)</b> OCR-Resdatei kopieren	Rohtext/OCR Ergebnis im Originalzustand wird exportiert
<b>(2)</b> Trainingsdatei kopieren	Alle Bezeichner, Merkmale die für den Belegleser trainiert wurden, werden exportiert.
<b>(3)</b> Kontendaten kopieren	Stammdaten, die dem Belegleser vorliegen, werden exportiert.
<b>(4)</b> Templatedateien kopieren	Hinterlegte Templates werden exportiert.

Über die Schaltfläche Ordner laden starten Sie die Ordnerauswahl und entscheiden sich für den Ordner, der exportiert werden soll. Mit Start werden die angehakten Dateien in das angegebene Verzeichnis exportiert. Sie können anschließend die Zip-Datei an HABEL versenden. Durch die Datei kann HABEL das bei Ihnen aktuelle System inkl. Daten rekonstruieren und Sie bei der Lösungsfindung unterstützen.

# Scripting

## 1. Allgemein

---

Scripte werden im Allgemeinen vor allem für kleine, überschaubare Programmieraufgaben geschrieben. Mit den Scripten wird ein einfaches Bearbeiten und Anpassen von weiteren Programmen ermöglicht.

Mit Scripten für den HABEL-BELEGLESER wird an verschiedenen Stellen der Belegerkennung eingegriffen, um Daten zu lesen, zu verändern oder gar zu löschen. Mithilfe von Scripten kann man umfangreiche Werte eines Beleges auslesen, sowie auch verschiedene Berechnungen an diesen vornehmen wie z. B. Skontobetrag, Mehrwertsteuerbetrag.

An welchen Stellen eingegriffen wird, legt der Eingriffspunkt fest. Eingriffspunkte können in den Kopf- oder Positionsdaten vorkommen. Ein Script in Verbindung mit einem Template liegt generell im Ordner „GEDOSOD\OCR\OCRSCRIPT“ und hat folgende Namenskonvention: BEST[*Templatenummer*].scp. Die jeweilige Templatenummer kann im Programm Templateverwaltung (hphab404) nachgeschlagen werden. Darüber hinaus gibt es noch sogenannte Exportscripte, bei denen man Einfluss auf die Daten nehmen kann, die beim Export aus dem Programm Vorgänge bilden in die Datenbank geschrieben werden. Diese Scripte werden hier nicht beschrieben.

## 2. Eingriffspunkte

---

### **cspModifyHeaderData**

An dieser Stelle werden die sogenannten Kopfdaten modifiziert werden (z.B.: Belegdatum, Belegnummer, Bestellnummer, Skontosatz oder auch die verschiedenen Beträge).

Die hier zur Verfügung stehenden Daten wurden bereits durch das Programm Templateverwaltung (hphab404) verarbeitet und daher schon Prüfregele, Ersatzregeln etc. unterzogen.

### **cspModifyColumnData**

Über diesen Eingriffspunkt werden die Positionsdaten des Beleges und die zugehörigen Regeln gesteuert. Auch hier ist durch die Templateverwaltung bereits eine Verarbeitung erfolgt (Prüfregele, Ersetzungsregeln ...).

### **cspBeforeModifyColumnData**

Dieser Eingriffspunkt wird aufgerufen bevor die Positionen durch die Funktion cspModifyColumnData verarbeitet werden. Hier kann dem HABEL-BELEGLESER mitgeteilt werden, dass die Positionen nicht von oben nach unten, sondern von unten nach oben gelesen werden sollen. Das Setzen dieser Option erfolgt durch den Aufruf der Funktion `bmcRunBackward(True)`.

### 3. Programmiersprache

Für die Scripterstellung wird die Programmiersprache Delphi verwendet. Innerhalb dieser gibt es wichtige Punkte/Abweichung von anderen Programmiersprachen, die für die Scripterstellung zu berücksichtigen sind. Nachfolgend gehen wir auf die wichtigsten davon ein.

#### Variablentypen

Variablentyp	Reservierter Speicher	Werte-Art / Werte-Bereich
Integer	4 Byte	Nummerische Werte – Ganze Zahlen -2147483648 bis 2147483647
String	bis zu 2 GByte	Text bzw. Zeichenketten, z.B. „Hallo Welt“ 2 <sup>31</sup> Zeichen lang
Double	8 Byte	Nummerische Werte – Gleitkomma Zahlen 5,0·10 <sup>-324</sup> bis 1,7·10 <sup>308</sup>
Boolean	1 Byte	Zwei Zustände (richtig oder falsch) True, False
Currency	8 Byte	Meist verwendet zum Handling eines Betrags -922.337.203.685.477,5808 bis 922.337.203.685.477,5807

#### Wie werden Variablen im Script benutzt?

Je nach Script (Recherche, Erfassung, Belegleser) stehen verschiedene Punkte zur Verfügung, mit denen in die Verarbeitung eingegriffen werden kann. Zum Beispiel in den Erfassungsprogrammen der Eingriffspunkt, um beim Verlassen eines beliebigen Feldes entsprechend aktiv werden zu können. Egal um welchen Eingriffspunkt es sich handelt, der Aufbau ist immer derselbe:

```
procedure BeliebigerEingriffspunkt (Übergabe-Parameter)
begin

end;
```

Ein Eingriffspunkt beginnt immer mit dem Wort „procedure“, gefolgt von dem Namen des Eingriffspunktes und Parametern, die an diesen Eingriffspunkt übergeben werden. Wie die Prozedur heißt und welche Argumente übergeben werden, wurde durch HABEL definiert. Danach wird mit „begin“ der Anfang bzw. mit „end“ das Ende des Eingriffspunktes im Script definiert. Alle Anweisungen (Abfragen, Manipulationen etc.) werden innerhalb begin und end wie z. B. auch das Arbeiten mit Variablen (Werte zuweisen, Werte lesen...) definiert. Um aber auf eine Variable zugreifen zu können, muss dem Script zunächst mitgeteilt werden, welche Variablen im Laufe des Eingriffspunktes verwendet werden und um welchen Typ es sich handelt. Dies erfolgt zwischen den Begriffen „procedure“ und „begin“ durch Einleitung mit dem Wort „var“. Im „Var-Block“ müssen alle Variablen angegeben werden, die innerhalb des Eingriffspunktes verwendet werden sollen.

Folgendes Format ist vorgegeben: „VariablenName : VariablenTyp;“

**Beispiel:**

```
procedure BeliebigerEingriffspunkt (Übergabe-Parameter)
var
  iVariable : Integer;
  sVariable : String;
begin

end;
```

Werden mehrere Variablen vom selben Typ benötigt, sind diese entweder einzeln aufzulisten oder im Namen durch ein Komma zu separieren.

**Beispiel:**

```
procedure BeliebigerEingriffspunkt(Übergabe-Parameter)
var
  iVariable : Integer;
  sVariable : String;
  iVariable2, iVariable3, iVariable4 : Integer;
begin

end;
```

Um einer Variablen einen Wert zuzuweisen benutzt Delphi den Operator “:=”

**Beispiel:**

```
procedure BeliebigerEingriffspunkt (Übergabe-Parameter)
var
  iVariable : Integer;
  sVariable : String;
begin
  iVariable := 37000;    // Zuweisen des Wertes 37000 als Zahl
                       // an die Variable iVariable
  sVariable := '37000'; // Zuweisen des Wertes 37000 als Text
                       // an die Variable sVariable
end;
```

## Bedingungen

### IF

Die IF-Bedingung, oder „IF-Abfrage“ ist die einfachste Bedingung innerhalb Delphi und hat generell den folgenden Aufbau:

```
if (Bedingung) then
begin
  //Anweisungen was zu tun ist, wenn die Bedingung zutrifft
end;
```

Operator	Die Bedingung ist erfüllt wenn eine Variable...	Beispiel
=	... genau einen bestimmten Wert hat, z.B. Barcode genau ,999999'	IF ( Barcode = ,999999') then...
<>	... genau einen bestimmten Wert <b>NICHT</b> hat, z.B. „Wenn nicht Belegart VK_Rechnung, dann....“	IF ( BelArt <> ,VK_Rechnung') then...
<	... kleiner als ein bestimmter Wert ist, z.B. „Wenn Betrag kleiner 50, dann...“	IF ( Betrag < 50 ) then ...
>	... größer als ein bestimmter Wert ist, z.B. „Wenn Betrag größer 50, dann...“	IF ( Betrag > 50 ) then ...
<=	... kleiner oder gleich einem bestimmten Wert ist, z.B. „Wenn Betrag kleiner oder gleich 50	IF ( Betrag <= 50 ) then ...
>=	... größer oder gleich einem bestimmten Wert ist, z.B. „Wenn Betrag größer oder gleich 50, dann ....“	IF ( Betrag >= 50 ) then ...

```
ELSE
IF (Bedingung) then
Begin
  //Anweisungen was zu tun ist, wenn die Bedingung zutrifft
End //Hier jetzt kein Semikolon mehr, da unsere IF-Bedingung durch das Else verlängert wird.
Else
Begin
  //Anweisungen was zu tun ist, wenn die Bedingung NICHT zutrifft
End;
```

### *Else IF*

Nach dem IF-Zweig und vor dem Else-Zweig haben Sie die Möglichkeit eine weitere Bedingung zu prüfen. Ein Else-IF Zweig wäre folgendermaßen definiert:

```
IF (Bedingung) then
Begin
  //Anweisungen was zu tun ist, wenn die Bedingung zutrifft
End
Else IF (Bedingung 2) then
Begin
  //Anweisungen was zu tun ist, wenn Bedingung 2 zutrifft
End
Else
Begin
  //Anweisungen was zu tun ist, wenn die Bedingung NICHT zutrifft
End;
```

### **Logische Operatoren**

Es stehen zwei logische Operatoren zur Verfügung: „AND“ und „OR“. Mit dem AND Operator können beliebig viele Bedingungen logisch „UND“ verknüpft werden wie z.B. „Wenn Belegart VK-Rechnung und Betrag kleiner oder gleich 0“.

Mit dem OR Operator können beliebig viele Bedingungen logisch „ODER“ verknüpft werden wie z.B. „Wenn Belegart VK-Rechnung oder Belegart VK-Gutschrift“.

Jede Bedingung zwischen einem logischen Operator sollte in „( )“ (Klammern) gesetzt werden, die Definition ist wie folgt:

```
if ((Bedingung1) LogischerOperator (Bedingung2) LogischerOperator (BedingungX))
then
```

### *Hinweis:*

Die Formatierung trägt viel zur Lesbarkeit bei:

```
if (
  (((sBelArt='VK_Rechnung') AND (Betrag>=0)) OR
   ((sBelArt='VK_Gutschrift') AND (Betrag<=0))) OR
  (sScriptKz='AK')
) then ...
```

### **Prozedur in Delphi**

Eine Prozedur in Delphi hat folgenden Aufbau

```
procedure <Name>(<Übergabe-Parameter>);  
var  
    <Variablen>  
begin  
    <Anweisungen>  
end;
```

### **Funktion in Delphi**

Eine Funktion in Delphi hat folgenden Aufbau

```
function <Name>(<Übergabe-Parameter>):<Rückgabety>;  
var  
    <Variablen>  
begin  
    <Anweisungen>  
end;
```

### **Unterschied zwischen Prozeduren – Funktionen**

Eine Prozedur führt eine Aufgabe aus und liefert keinen Rückgabewert. Eine Funktion dagegen gibt nach Ausführen der Aufgabe ein Ergebnis zurück.

#### 4. Aufbau eines Templatescriptes

---

Im Folgenden wird der Aufbau einer Scriptdatei beschrieben. Dabei handelt es sich um ein „leeres“ Script, das keine Aufgaben beinhaltet, die ausgeführt werden müssten.

```
program BEST000000000001;

uses Windows,
    Receipt,
    BestVar,
    ReceiptPage;
//-----

begin
  case GetCurrentScriptLocation of
    cspModifyHeaderData ;;
    cspModifyColumnData;;
    cspBeforeModifyColumnData;;
  else
    OutputDebugString('ScriptLocation: ' + IntToStr(GetCurrentScriptLocation));
  end;
end.
```

Im unteren Bereich des Scriptes werden die verschiedenen Eingriffspunkte geprüft und festgestellt, ob dafür Funktionen hinterlegt wurden. Wenn ja, werden diese ausgeführt.

Beispiel eines einfachen Scripts, bei dem die Belegnummer des zu erkennenden Beleges ausgegeben werden soll:

```
program BEST000000000001;

uses Windows,
    Receipt,
    BestVar,
    ReceiptPage;

procedure ModifyHeaderData;
var
    sBelNr:String;
begin
    sBelNr := mhdGetHeaderValue(,FBeINr');
    OutputDebugString(,Belegnummer lautet: '+sBelNr);
end;

begin
    case GetCurrentScriptLocation of
        cspModifyHeaderData : ModifyHeaderData;
        cspModifyColumnData;;
        cspBeforeModifyColumnData;;
        else
            OutputDebugString('ScriptLocation: ' + IntToStr(GetCurrentScriptLocation));
    end;
end.
```

Mit „GetCurrentScriptLocation“ wird dem Script mitgeteilt, dass der Eingriffspunkt cspModifyHeaderData an die Prozedur „ModifyHeaderData“ weitergeleitet werden soll, d.h. dass in der Prozedur „ModifyHeaderData“ die Kopfdaten des Beleges modifiziert werden können.

Die Prozedur „ModifyHeaderData“ hat folgenden Aufbau:

Bereich „var“

Hier werden alle Variablen deklariert, die im Verlauf der Prozedur benötigt werden. Wir deklarieren in unserem Beispiel die Variable „sBelNr“ vom Typ String (Zeichenkette).

Bereich „begin“ bis „end“

Hier wird der eigentliche Code der Prozedur hinterlegt und durch begin und end eingeschlossen. In unserem Beispiel benutzen wir die Funktion „mhdGetHeaderValue(,FBeINr)“, um den Inhalt des Feldes „FBeINr“ in der Variablen „sBelNr“ zu speichern. Über die Prozedur „OutputDebugString“ kann die ausgelesene Belegnummer in dem Debug-Fenster des Programms hphab404 ausgegeben werden.

## 5. Die wichtigsten Scripting Funktionen im Überblick

---

Es gibt allgemeine Funktionen und eingriffspunktbezogene Funktionen. Allgemeine Funktionen können überall im Script eingesetzt werden, eingriffspunktbezogene nur bei den dafür vorgesehenen Eingriffspunkten. Nachfolgend wird auf die eingriffspunktbezogenen Funktionen, sowie die wichtigsten allgemeinen Funktionen eingegangen werden. Weitere Informationen finden Sie in der Hilfedatei „BestInfo.chm“.

Folgende Tabelle zeigt die wichtigsten eingriffspunktbezogenen Funktionen.

Eingriffspunkt	Funktion	Beschreibung
cspModifyHeaderData	mhdGetHeaderValue	Auslesen eines Kopffeldes wie z.B. Belegnummer: mhdGetHeaderValue('BelNr'); → Bsp. um vorangestellte Teile der Belegnummer zu entfernen
	mhdSetHeaderValue	Beschreiben eines Kopffeldes z.B. mhdSetHeaderValue('BruBet', '0.00'); → Bsp. um aus Währungskürzeln einheitliche Bezeichnungen ins Feld einzutragen
cspModifyColumnData	mcdGetRowValue	Auslesen eines Positionfelds, z.B. mcdGetRowValue('XPosWer');
	mcdSetRowValue	Schreiben eines Positionsfeldes, z.B. mcdSetRowValue('XPosWert', '0.00');
	mcdDeleteRow	Löscht die aktuelle Position
cspBeforeModifyColumnData	mcdRunBackward	Aktiviert/Deaktiviert die Option, dass die Positionen von Unten nach Oben gelesen werden sollen.

Die wichtigsten allgemeinen Funktionen zeigt nachfolgende Tabelle:

Funktion	Beschreibung
UpperCase	Wandelt einen String in Großbuchstaben um, z.B. <code>Text_Groß := UpperCase(Text_Klein);</code>
LowerCase	Wandelt einen String in Kleinbuchstaben um, z.B. <code>Text_Klein := LowerCase(Text_Groß);</code>
Trim	Trim entfernt alle am Anfang und Ende eines Strings vorhandenen Leer- und Steuerzeichen, z.B. <code>Text_Ohne_Leerzeichen := Trim(' Text ');</code>
OutputDebugString	Gibt einen String auf der Debug Konsole aus, um das Script zu analysieren.
Length	Gibt die Länge eines Strings zurück.
PerlMatch	Überprüft, ob eine Perl Prüfregele auf einen String zutrifft, z.B. <code>If PerlMatch('^d+\$', sMenge) then ...</code>
Similar	Gibt die prozentuale Übereinstimmung zweier Strings zurück, z.B. <code>if Similar('Text', Vergleichs_Variable) then ...</code>
VarStoreValue	Mit dieser Funktion können Werte global in einer Variablen gespeichert werden. Dieser Wert kann beim nächsten Aufruf des Eingriffspunkts mit Hilfe von VarGetValue wieder ausgelesen werden, z.B. <code>VarStoreValue('meineGlobaleVariable', wertAusLokalerVariable);</code>
VarGetValue	Hiermit kann der durch VarStoreValue gespeicherte Wert wieder gelesen werden, z.B. <code>lokaleVariable := VarGetValue('meineGlobaleVariable');</code>

## 6. Auslesen von Positionen

Am häufigsten werden Scripte für das Auslesen von Positionsdaten eines Beleges verwendet. Es soll geprüft werden, ob es sich bei der aktuellen Position, die gelesen wird, um eine gültige Position handelt oder eine ungültige, die gelöscht werden soll. Als ungültige Positionen gelten Positionszeilen, in denen keine relevanten Informationen für eine Position zu finden sind, wie z.B. leere Zeilen oder Zeilen in denen sonstige Informationen stehen. Diese Informationen können zwar den Artikel genauer beschreiben, sind jedoch für die Archivierung oder Gegenprüfung gegen weitere Daten nicht relevant sind.

Beispiel:

Für das Script wird ein Regelwerk definiert, das in unserem Fall besagt, dass wenn der Wert, der in der Spalte Position, Artikel-Nr., Menge, Einzelpreis, Gesamtpreis keine bestimmte Mindestlänge hat, es sich um eine ungültige Position handelt, die gelöscht werden muss.



**Firma**  
Habel GmbH & Co.KG  
Narr  
Helmut  
Untere Hauptstrasse 1  
78604 Riethem-Weilheim

**Rechnung**  
Nummer: **91868352**  
Datum: **25.11.2008**  
Kunden-Nr.: **2132928**  
Ihre USt-ID Nr.:  
Ihre Steuernummer:  
Ihre Bestellnr.: **Ihre Bestellung v. 10.11.2008**  
Lieferschein-Nr.: **81586471**  
Lieferdatum: **25.11.2008**  
Ansprechpartner/in: **Stefanie Bauer**  
Telefon: **01805-404901 (0,14 EUR/Min.)**  
Telefax: **01805-404902 (0,14 EUR/Min.)**  
Mail: **kontakt@mega-tec.com**

Seite: 1 / 1

Pos.	Menge	Artikel Nr.	Bezeichnung	%USt.	Einzelpreis	Gesamt	EUR
0010	1	20118015	TFT 24 SyncM. T240HD 5ms 10000:1 DVI Hersteller / Art. Nr._ SAMSUNG / LS24TDDSUJ/EN	19,00	450,00		450,00
0020	1	20103312	TFT 24 SyncM. T220HD 5ms 10000:1 DVI Hersteller / Art. Nr._ SAMSUNG / LS24TDDSUJ/EN	19,00	290,00		290,00
0030	1	20100283	HDD 640GB 16MB Caviar SE16 SATA2 WD6400AA Hersteller / Art.Nr.: WESTERN DIGITAL / WD6400AAKS	19,00	83,50		83,50
<b>Summe Netto ohne USt.</b>							<b>823,50 EUR</b>
<b>Transportkosten</b>							<b>12,90 EUR</b>
<b>Gesamtnetto ohne USt.</b>							<b>836,40 EUR</b>
<b>Umsatzsteuer (19,00%)</b>							<b>158,92 EUR</b>
<b>Endsumme</b>							<b>995,32 EUR</b>

Belieferung ausschließlich unter den im Internet unter [www.mega-tec.com/agb](http://www.mega-tec.com/agb) ersichtlichen Allgemeinen Geschäftsbedingungen in der aktuell geltenden Fassung.

Zahlungsbedingung: Rechnung 8 Tage netto  
Incoterms: TOF STD Mo-Fr  
Versandbedingung: 304873864718172244978604  
Paketnummer: 304873864718172244978604

Referenz-Nr.: S86471  
Bruttogewicht: 9,529 KG  
Nettogewicht: 9,529 KG  
Anzahl Packstücke: 2

Mega-Tec GmbH • Max-Wallmeyer-Str. 1 • 56566 Neuwied  
Geschäftsführer: Dieter Jung / Gabi Wilmann  
Amtsgericht Koblenz: HRB 5549  
Stz Neuwied  
Steuer-Nr. FA Gießen 030 227 9 0911  
Umsatzsteuer ID-Nr.: DE133648957

Deutsche Bank AG  
BLZ: 510 700 910  
Konto: 074 455 982  
BIC-Code: DEUTDE33  
IBAN: DE 749306082355645885

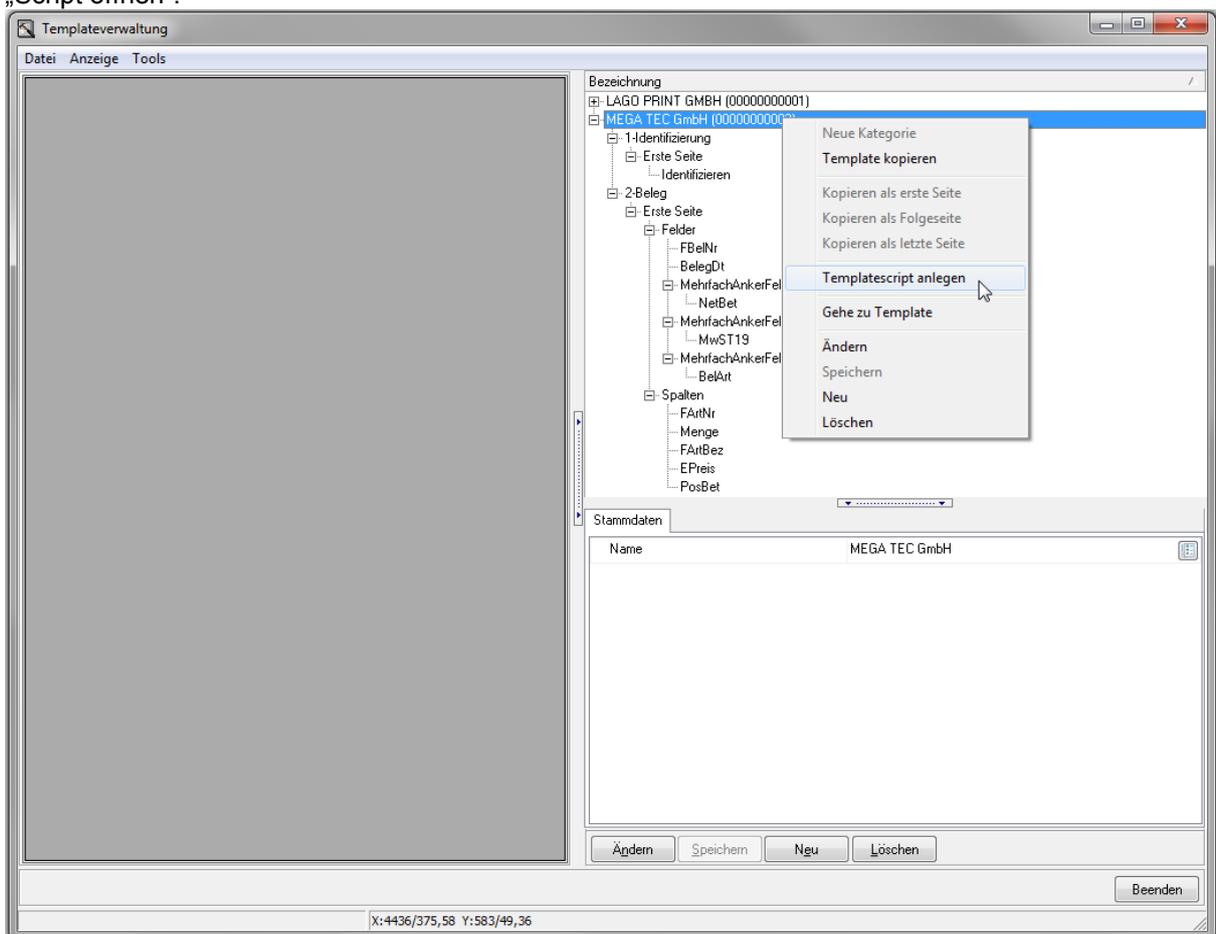
```
procedure ModifyColumnData;
var
  //Auslesen der aktuellen Position und speichern im Block var
  sPos, sMenge, sArtNr, sArtBez, sEPreis:String;
begin
  //Auslesen der Positionsdaten und speichern in lokalen Variablen
  sPos      := mcdGetRowValue(,XPos');
  sMenge    := mcdGetRowValue(,XMenge');
  sArtNr    := mcdGetRowValue(,XArtNr');
  sEPreis   := mcdGetRowValue(,XEPreis');
  sGPreis   := mcdGetRowValue(,XGPreis');

  //Die Variablen auf ihre Länge überprüfen
  if not ((Length(sPos) >=1) and
          (Length(sMenge)>=1) and
          (Length(sArtNr)>=3) and
          (Length(sGPreis)>=3) and
          (Length(sEPreis)>=3))
  then
    mcdDeleteRow; //Die Zeile löschen
end
```

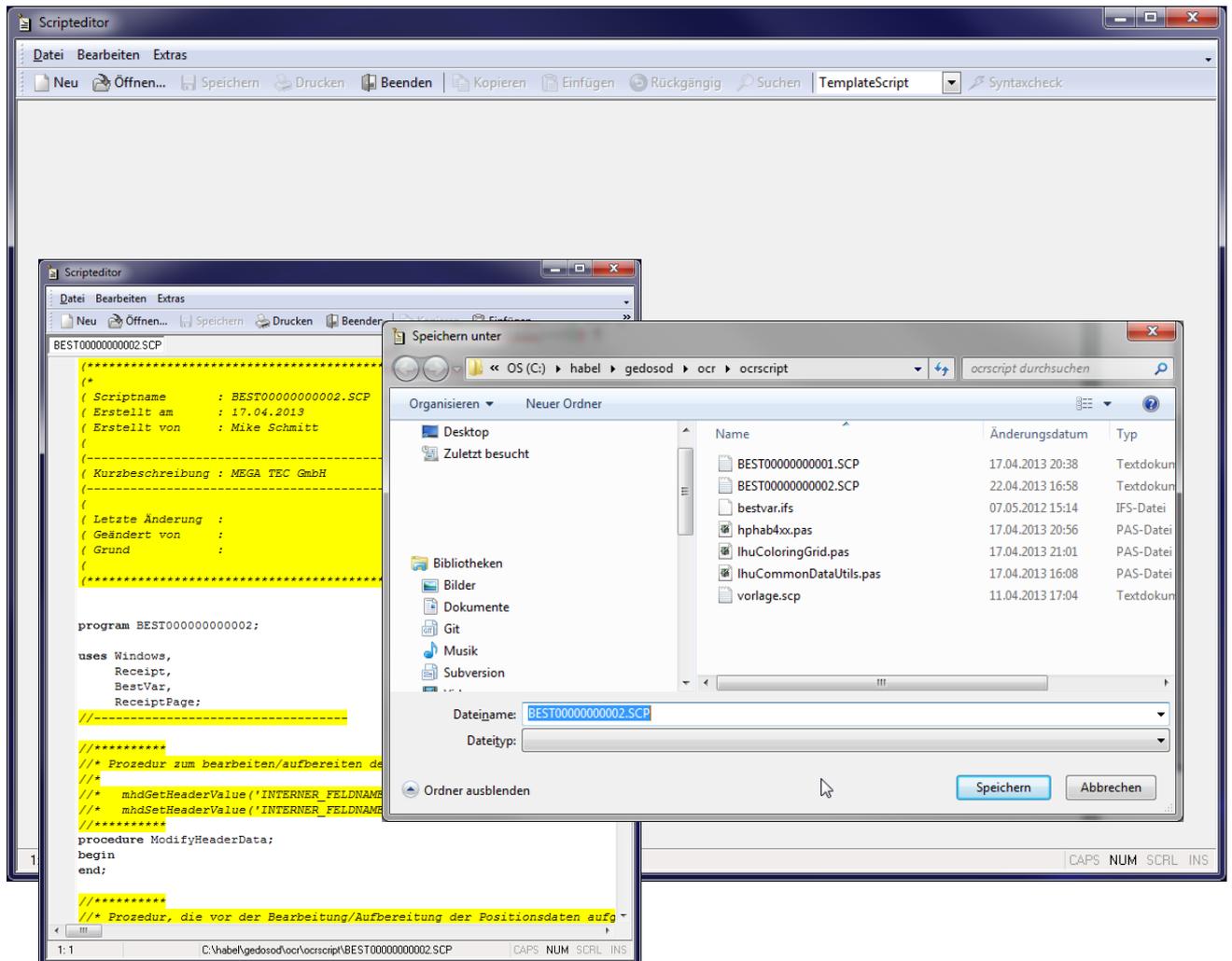
## 7. Programmbedienung

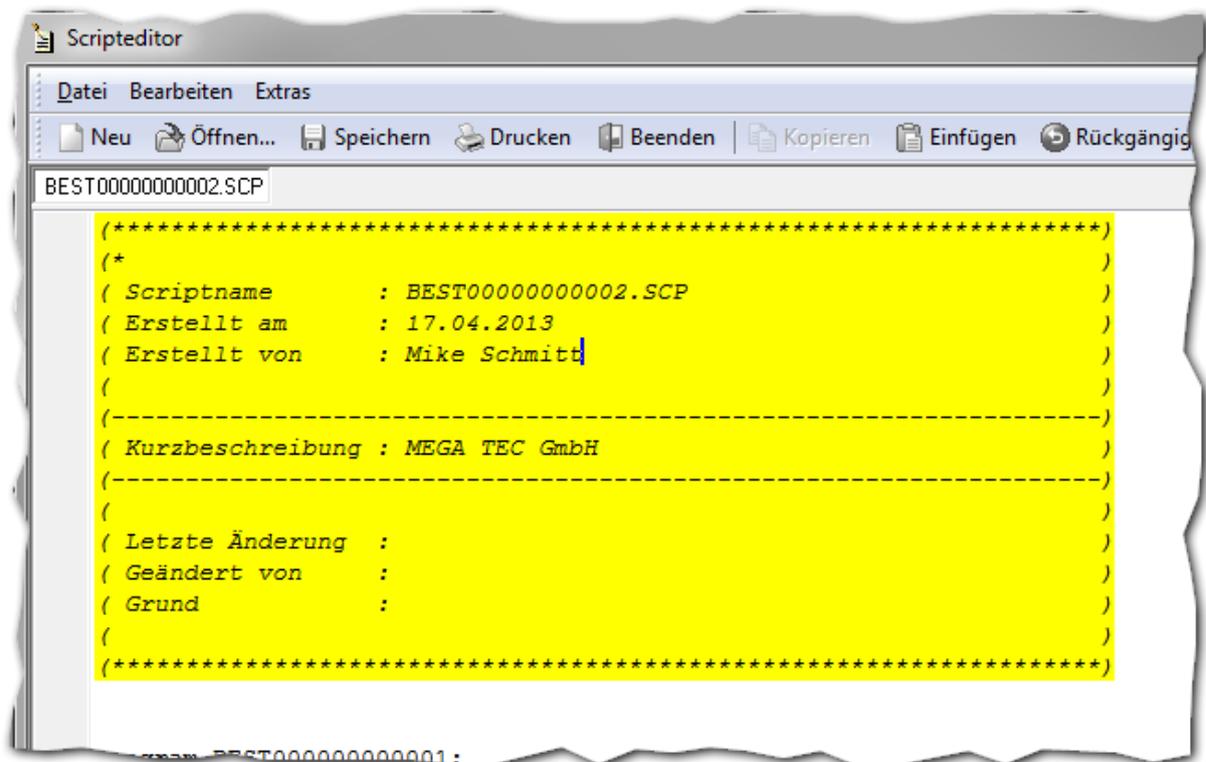
---

Um ein Script zu erstellen, gehen Sie in die Templateverwaltung und klicken Sie per Rechtsklick auf das entsprechende Template. Wählen Sie anschließend „Templatescript anlegen“ bzw. „Script öffnen“:



Alternativ steht Ihnen die Scriptverwaltung (hphab407) auch eigenständig zur Verfügung. Sie können Scripte neu erstellen oder vorhandene (z. B. Musterscripte) öffnen und kopieren. Hierbei müssen Sie selbst Sorge tragen, das Script unter dem korrekten Scriptnamen abzuspeichern (analog Templatenummer).





Im oberen Bereich des Scriptes werden Informationen eingetragen zu Dateiname, Erstellungsdatum, Ersteller und Kurzbeschreibung des Inhaltes. Diese Informationen werden in Bezug auf die Scriptausführung komplett ignoriert, sollten jedoch aufgrund der Transparenz gefüllt werden.

```
program BEST000000000002;  
  
uses Windows,  
    Receipt,  
    BestVar,  
    ReceiptPage;  
  
//-----
```

Um aus der Vorlage ein gültiges Script zu generieren, hinterlegen Sie unter dem Eintrag „program“ den gültigen Dateinamen des Scriptes (in unserem Fall Ergänzung der Ziffer 2).





## DEUTSCHLAND

HABEL GmbH & Co. KG  
Untere Hauptstraße 1-5  
D-78604 Rietheim-Weilheim  
Fon +49 7461 9353-0  
Fax +49 7461 9353-99  
[www.habel.de](http://www.habel.de) | [info@habel.de](mailto:info@habel.de)

Niederlassung Leipzig  
Messe-Allee 2  
D-04356 Leipzig  
Fon +49 341 678-27322  
Fax +49 341 678-28322  
[www.habel.de](http://www.habel.de) | [info@habel.de](mailto:info@habel.de)

## SCHWEIZ

HABEL Dokumentenmanagement GmbH  
Rheinstrasse 36  
CH-8212 Neuhausen am Rheinfall  
Fon +41 52 674-8151  
Fax +41 52 674-8150  
[www.habel.ch](http://www.habel.ch) | [info@habel.ch](mailto:info@habel.ch)