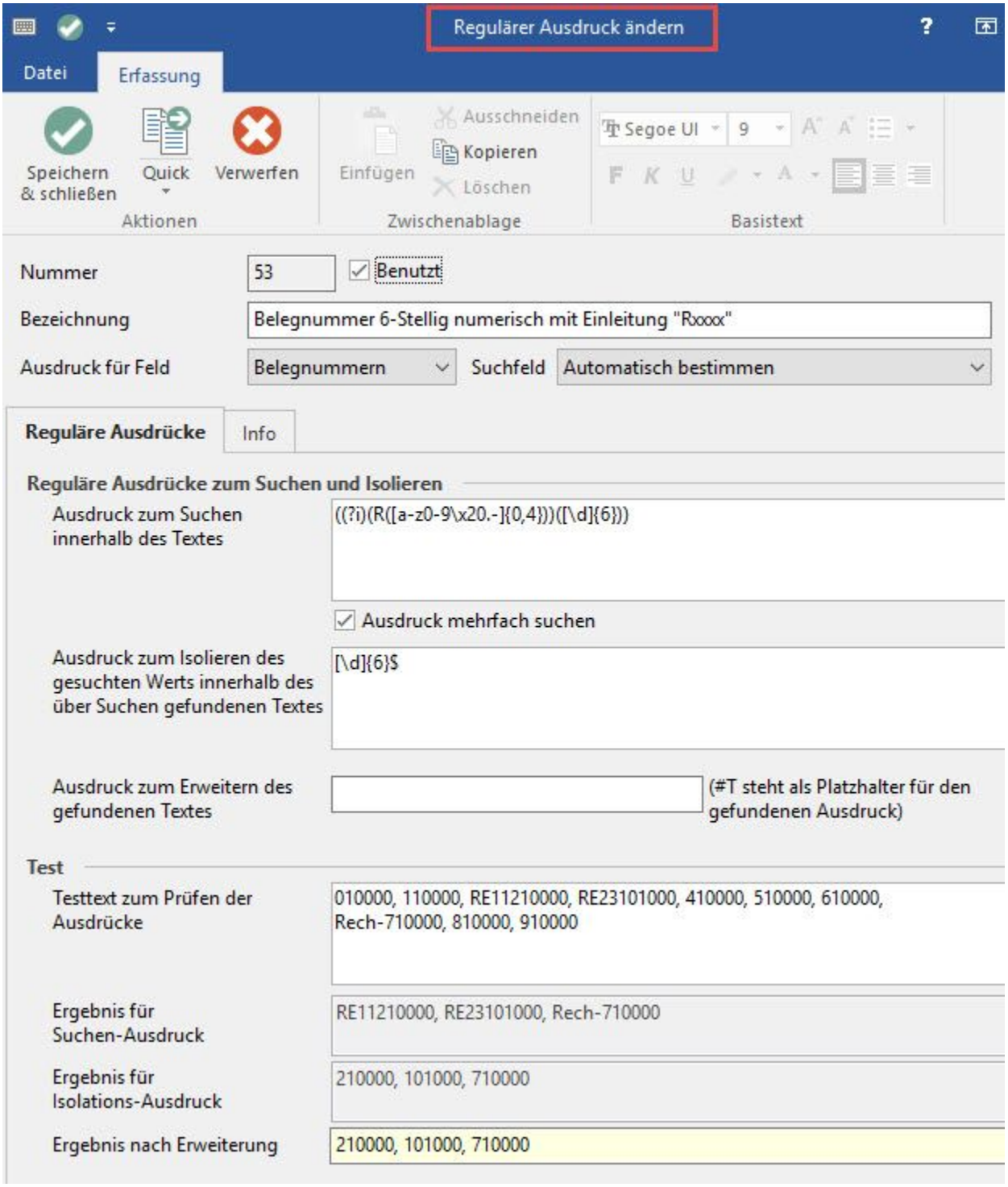
Reguläre Ausdrücke

Um im Verwendungszweck die Adressnummer oder die Belegnummer zu finden, können "Reguläre Ausdrücke" definiert werden (Schaltfläche: PARAMETER - SONSTIGE – REGULÄRE AUSDRÜCKE).



Folgende Felder können Sie hinterlegen:

Nummer

Die fortlaufende Nummer wird für das Speichern und Zuordnen des Regulären Ausdrucks beim Suchen benötigt. Diese Nummer sollte, falls der Ausdruck gelöscht wird, nicht nochmals vergeben werden!

Bezeichnung

Hier hinterlegen Sie eine Bezeichnung, die kurz die Funktionsweise des Ausdrucks bestimmt. Diese Bezeichnung wird später im Assistenten, in welchem Sie die zu verwendenden 'Regulären Ausdrücke' auswählen, angezeigt.

Ausdruck gilt für Feld

- Adressnummern
- Belegnummern
- Auftragsnummern

Reguläre Ausdrücke zum Suchen und Isolieren

Ausdruck zum Suchen innerhalb des Textes

Hier hinterlegen Sie den Suchausdruck, der zum Auffinden eines Textabschnitts verwendet wird, in dem sich wahrscheinlich die zu suchende Zeichenfolge befindet.

Kennzeichen **Ausdruck mehrfach suchen**

Erlaubt dem Programm den Ausdruck mehrfach zu suchen. Dies ist z. B. für eine Belegnummernsuche sinnvoll, wenn im Text mehrere Belegnummern vorkommen können.

Ausdruck zum Isolieren des gesuchten Werts innerhalb des über Suchen gefundenen Textes

Aus dem zuvor gefundenen Textabschnitt soll nun eine bestimmte Zeichenfolge bestimmt werden. Hier hinterlegen Sie den Ausdruck zum Extrahieren dieser Zeichenfolge.

Ausdruck zum Erweitern des gefundenen Textes (#T steht als Platzhalter für den gefundenen Ausdruck)

Mit dieser Funktion kann erreicht werden, dass der gefundene Ausdruck um eine Zeichenkette erweitert wird.

Beispiel:

Das Ergebnis soll die Belegnummer RG123456 sein. Die Zeichenkette 123456 wurde gefunden. Im Feld: "Ausdruck zum Erweitern des gefundenen Textes" tragen Sie nun RG#T ein. Dadurch erhalten Sie als Ergebnis: "RG123456".

Soll die Erweiterung nach dem gefundenen Ausdruck ausgewiesen werden, so tragen Sie in dieses Feld zuerst den Platzhalter "#T" ein, und dann die Ergänzung "RG". Das Ergebnis wäre dann 123456RG.

Test

Hier haben Sie die Möglichkeit, Ihre Eingaben zu prüfen.

Testtext zum Prüfen der Ausdrücke

Hier können Sie Zeichenketten hinterlegen, die sehr häufig im Verwendungszweck eingetragen sind. Dadurch können Sie testen, ob der Such-Ausdruck richtig gewählt wurde, und tatsächlich die von Ihnen gewünschte Zeichenkette gefunden wird.

Ergebnis für Suchen-Ausdruck

Es wird angezeigt, welche Einträge aufgrund des "Ausdruckes zum Suchen" gefunden wurden.

Ergebnis für Isolations-Ausdruck

Hier wird angezeigt, wie das ursprüngliche "Such-Ergebnis" aufgrund des Eintrages im Feld: "Ausdruck zum Isolieren des gesuchten Werts ..." gekürzt wird.

Ergebnis nach Erweiterung

In diesem Feld wird das endgültige Ergebnis unter Berücksichtigung des Eintrages im Feld: "Ausdruck zum Erweitern des gefundenen Textes" angezeigt.

Funktionsweise:

Als erstes wird im Text mit dem "Ausdruck zum Suchen" gesucht. Wird eine entsprechende Textstelle gefunden, wird der "Ausdruck zum Isolieren" angewendet, um die entsprechende Nummer zu extrahieren.

Falls kein "Ausdruck zum Isolieren" angegeben wurde, wird als Ergebnis die Textstelle des "Ausdruck zum Suchen" zurückgegeben. Falls das Kennzeichen: "Ausdruck mehrfach suchen" gesetzt ist, wird solange der Text weiter durchsucht, bis kein Ergebnis mehr gefunden wird.

> **Hinweis**
>
> Beachten Sie: Die Groß-/Kleinschreibung wird bei Regulären Ausdrücken beachtet!

Escapesequenzen

Escape Zeichen sind notwendig, wenn Sie Zeichen suchen, die normalerweise eine Funktion innerhalb des Ausdruckes haben.

\xnn	Zeichen mit einem Hexadezimalwert (nn)
\t	ein Tabulator
\n	Zeilenende
\r	Wagenrücklauf
\e	Escape (ESC)
\x20	Leerzeichen

Beispiel:

test\x20Leerzeichen findet 'Test Leerzeichen'

(\x20 steht für ein Leerzeichen)

Zeichenmengen

Mit einer Zeichenmenge kann man nach einzelnen Zeichen in einer Gruppe von Zeichen suchen. Um eine Zeichenmenge zu definieren, setzt man die Zeichen, nach denen gesucht werden soll, in eckige Klammern.

Beispiel: [ab] sucht nach a oder b

Man kann als Zeichenmenge auch einen Bereich definieren.

Beispiele:

[a-z] passt auf jeden Kleinbuchstaben

[A-Z] passt auf jeden Großbuchstaben

[0-9] passt auf jede Ziffer

Zeichenmengen können auch kombiniert werden.

Beispiel:

[a-z5] passt auf jeden Kleinbuchstaben und auf die Ziffer 5

Zeichenmengen können auch negiert werden, indem man unmittelbar nach der ersten Klammer das Zeichen ^ verwendet.

Beispiel:

[^A-Z] findet alles außer Großbuchstaben

Weitere Beispiele:

Büro[PN]T findet "BüroNT", "BüroPT".

Büro[^PN]T findet alles was nicht "BüroNT", "BüroPT" ist.

Vorsicht bei "-", "[" und "]"

Diese Zeichen müssen entweder ein Escape Zeichen bekommen (siehe oben) oder an den Anfang oder das Ende der Menge gestellt werden.

Beispiel:

[-az] findet 'a', 'z' und '-' (am Anfang)

[az-] findet 'a', 'z' und '-' (am Ende)

[a\-z] findet 'a', 'z' und '-' (mit Escape Zeichen "\")

[a-z] findet alle Kleinbuchstaben von "a" bis "z"

[]-a] findet irgendein Zeichen von "]"..."a".

vordefinierte Zeichenmengen

\w	ein alphanumerisches Zeichen inklusive "_"
\W	kein alphanumerisches Zeichen, auch kein "_"
\d	ein numerisches Zeichen
\D	kein numerisches Zeichen

\s	irgendein wörtertrennendes Zeichen (entspricht [\t\n\r\f])
\S	kein wörtertrennendes Zeichen
(?i)	Groß- und Kleinschreibung wird nicht unterschieden

Metazeichen

Metazeichen erfüllen eine bestimmte Bedingung, z. B. Zeilenseparatoren

^	Beginn einer Zeile
$	Ende einer Zeile
\A	Beginn des Textes
\Z	Ende des Textes
.	irgendein beliebiges Zeichen

Beispiel:

^BüroPlus => findet "BüroPlus" nur, wenn es am Zeilenanfang vorkommt

BüroPlus$ => findet "BüroPlus" nur, wenn es am Zeilenende vorkommt

^BüroPlus$ => findet "BüroPlus" nur, wenn er der einzige String in der Zeile ist.

BüroPl.s => findet "BüroPlus", "BüroPlas", "BüroPlxs" usw.

Standardmäßig garantiert das Metazeichen "^" nur, dass das Suchmuster sich am Anfang des Zielstrings befinden muss oder am Ende des Zielstrings mit dem Metazeichen "$". Kommen im Zielstring Zeilenumbrüche vor, so werden diese von "^" oder "$" nicht gefunden.

Iteratoren (=Häufigkeitsoperatoren)

Wenn Sie in einer Zeichenkette nach einem Zeichen suchen, können Sie mit einem Häufigkeitsoperator, genannt Interator, festlegen, wie oft nach einer Übereinstimmung gesucht werden soll.

Beispiel: Das Muster a+ verlangt mindestens ein a, dem dann noch 0 bis beliebig viele andere a folgen können.

Dieser Häufigkeitsoperator gilt jeweils für das davor stehende Zeichen, das Metazeichen oder den Teilausdruck.

Nachfolgende Tabelle enthält die Iteratoren zum Auffinden wiederholt auftretender Zeichen.

	Vorkommen	gleichbedeutend wie	'Gierig' oder 'Genügsam'
*	kein- oder mehrmalig	{0,}	gierig
+	ein- oder mehrmalig	{1,}	gierig
?	kein- oder einmalig	{0,1}	gierig
{n}	genau n-malig		gierig
{n,m}	mindestens n-, aber höchstens m-malig		gierig
*?	kein- oder mehrmaliges	{0,}?	genügsam
+?	ein- oder mehrmalig	{1,}?	genügsam
??	kein- oder einmalige	{0,1}?	genügsam
{n}?	genau n-malig		genügsam
{n,}?	mindestens n-malig		genügsam
{n,m}?	mindestens n-malig, aber höchstens m-malig		genügsam

Bedeutung von 'Gierig' und 'Genügsam'

"Genügsam": Sobald ein Ergebnis (Suchmuster wird erfüllt) gefunden wurde, wird nicht mehr weiter gesucht.

"Gierig": nimmt soviel wie möglich - Die Suche wird auch nach dem ersten Erfüllen des Suchmusters fortgesetzt.

Grenzen

Die in geschweiften Klammern stehenden Zahlen werden Grenzen genannt. Mit einer Grenze können Sie genau festlegen, wie oft ein Zeichen gesucht werden soll.

Die Ziffern in den geschweiften Klammern in der Form {n,m} geben an, wie viele Male das Suchmuster im Zielstring gefunden werden muss, um einen Treffer zu ergeben.

Die Angabe {n} ist gleichbedeutend mit {n,n} und findet genau n Vorkommen.

Die Form {n,} findet n oder mehrere Vorkommen.

Beispiel: a{4,5}

sucht eine Zeichenkette, die mindestens 4 Mal und höchstens 5 Mal das Zeichen a in Folge enthält.

Es gibt keine Begrenzung für die Zahlen n und m. Aber je größer sie sind, desto mehr Speicher und Zeit wird benötigt, um den regulären Ausdruck auszuwerten.

Hinweis

Beispiel: 'b+' und 'b*' angewandt auf den Zielstring 'abbbbc' findet 'bbbb' 'b+?' findet 'b' 'b*?' findet den leeren String 'b{2,3}?' findet 'bb' 'b{2,3}' findet 'bbb'

Falls eine geschweifte Klammer in einem anderen als dem eben vorgestellten Kontext vorkommt, wird es wie ein normales Zeichen behandelt.

Beispiele:

foob.*r	findet Strings wie 'foobar', 'foobalkjdflkj9r' und 'foobr'
foob.+	findet Strings wie 'foobar', 'foobalkjdflkj9r', aber nicht 'foobr'
foob.?r	findet Strings wie 'foobar', 'foobbr' und 'foobr', aber nicht 'foobalkj9r'
fooba{2}r	findet den String 'foobaar'
fooba{2,}r	findet Strings wie 'foobaar', 'foobaaar', 'foobaa aar' etc.
fooba{2,3}r	findet Strings wie 'foobaar', or 'foobaaar', aber nicht 'foobaaaar'

Alternativen

Man kann eine Serie von Alternativen für Suchmuster angeben, indem man diese mit einem "|" (Alt GR + <>| (Taste links neben Y)) trennt. Auf diese Art findet das Suchmuster fee|fie|foe eines von "fee", "fie", oder "foe" im Zielstring - dies würde auch mit f(e|i|o)e erreicht.

Die erste Alternative beinhaltet alles vom letzten Muster-Limiter ("(", "[" oder natürlich den Anfang des Suchmusters) bis zum ersten "|". Die letzte Alternative beinhaltet alles vom letzten "|" bis zum nächsten Muster-Limiter.

Aus diesem Grunde ist es allgemein eine gute Gewohnheit, die Alternativen in Klammern anzugeben, um möglichen Missverständnissen darüber vorzubeugen, wo die Alternativen beginnen oder enden.

Alternativen werden von links nach rechts geprüft, so dass der Treffer im Zielstring aus den jeweils zuerst passenden Alternativen zusammengesetzt ist. Das bedeutet, dass Alternativen nicht notwendigerweise "gierig" sind.
Ein Beispiel: Wenn man mit "(foo|foot)" im Zielstring "barefoot" sucht, so passt bereits die erste Variante. Diese Tatsache mag nicht besonders wichtig erscheinen, aber es ist natürlich wichtig, wenn der gefundene Text weiter verwendet wird. Im Beispiel zuvor würde der Benutzer nicht "foot" erhalten, wie er eventuell beabsichtigt hatte, sondern nur "foo". Ergänzend bleibt zu sagen, dass "|" innerhalb von eckigen Klammern wie ein normales Zeichen behandelt wird, so dass z. B. [fee|fie|foe] dasselbe bedeutet wie [feio|]. Beispiel: foo(bar|foo) findet die Strings 'foobar' oder 'foofoo'.

Teilausdrücke

Das Klammernkonstrukt (...) wird auch dazu benutzt, reguläre Teilausdrücke zu definieren. Teilausdrücke werden von links nach rechts nummeriert, jeweils in der Reihenfolge ihrer öffnenden Klammer. Der erste Teilausdruck hat die Nummer 1, der gesamte reguläre Ausdruck hat die Nummer 0. Beispiel: (foobar){8,10} findet Strings, die 8, 9 oder 10 Vorkommen von 'foobar' beinhalten foob([0-9]|a+)r findet 'foob0r', 'foob1r' , 'foobar', 'foobaar', 'foobaar' etc.

Weitere Themen

- Beispiele für "Reguläre Ausdrücke" für Adressnummern
- Beispiele für "Reguläre Ausdrücke" für Belegnummern