

We capture the future.

Janich & Klass

D P U

scanner value pack



# PlugIn for DpuScan Classification

Supplement to the DpuScan Reference Manual

## Copyrights

© 1997 to 2011 Janich & Klass Computertechnik GmbH and J&K Imaging, Marietta/USA.

All rights reserved.

Printed in Germany.

The information contained in this documentation is the property of J&K Imaging, Marietta and Janich & Klass, Wuppertal. Neither receipt nor possession hereof confers or transfers any right to reproduce or disclose any part of the contents hereof, without the prior written consent of J&K Imaging, Marietta and Janich & Klass, Wuppertal.

## Trademarks

The DPU logo is a registered trademark of Janich & Klass Computertechnik GmbH. DpuScan is a trademark of J&K Imaging, Marietta/USA. All other product names and logos are trademarks or registered trademarks of their representative companies.

## Disclaimer

The instruction and description in this manual were accurate at the time of this manual's printing. However, we reserve the right to alter the description and/or the product at anytime without prior notice.

Janich & Klass and J&K Imaging assume no liability for damages incurred directly or indirectly from errors, omissions, or discrepancies between this manual and the product.

## Actuality

It may happen that a more recent version of this Reference Manual for DpuScan is available for download from the Internet. Therefore, it is recommended that you should compare the version by means of the date printed on this page with the version on the Internet. You should use the most up-to-date version of the manual.

The internet version of this expansion to the DpuScan Reference Manual is found on the Web at the following address:

<http://www.jkimaging.com/pdf/PlugIns/Classification.pdf>

© 2011 Janich & Klass Computertechnik GmbH, Wuppertal, Germany

21 March 2011

## Table of Contents

1	Overview .....	4
2	Classes .....	5
3	How to Configure the PlugIn .....	6
4	Field Types for Classification .....	11
4.1	Classify Condition .....	12
4.2	Text.....	12
4.3	Barcode .....	12
4.4	Patchcode.....	12
5	Field Types for Data Extraction .....	13
5.1	Text Field.....	13
5.2	List Fields .....	14
5.3	Relative Search .....	16
6	How to Use the PlugIn in DpuScan .....	17
7	Regular Expressions.....	18
7.1	Syntax.....	18
7.2	Examples for Regular Expressions .....	22

# 1 Overview

This PlugIn serves for Classification of images that are put available by DpuScan.

The target is to arrange an image to a document class.

Document classes (later just called "class") are defined in the PlugIn by one or several Classify Conditions. search a definition compares the OCR results of a specific image area with stored data. These data may be fix texts, or also Regular Expressions.

Further to Classification, the PlugIn can also extract data from an image and put them available for the later process of DpuScan, with user-defined Variables.

The field types must therefore be separated in two groups.

- Classification fields, and
- Extraction fields.

For every class to be detected, one or several Classify Conditions can be defined. These conditions may be joined logically.

## 2 Classes

A set of logically joined features is named a document class. The features relate to results of the prior executed OCR. This allows to, for example, search for key words like Invoice, Delivery Slip, etc, in specified areas on the images, in order to determine the class of a document. For increasing the recognition safety, these features can be logically joined.

Another, far more often used option is to define that all invoices from one supplier belong to one class. This is useful because the contents to be extratracted can be found on fix defined positions on the images.

So, for example, the sender and the key word Invoice serve for identification of an image. After its identification, the required data can be read from the known spaces of the sheet, like invoice amount, or the invoice number.

### 3 How to Configure the Plugin

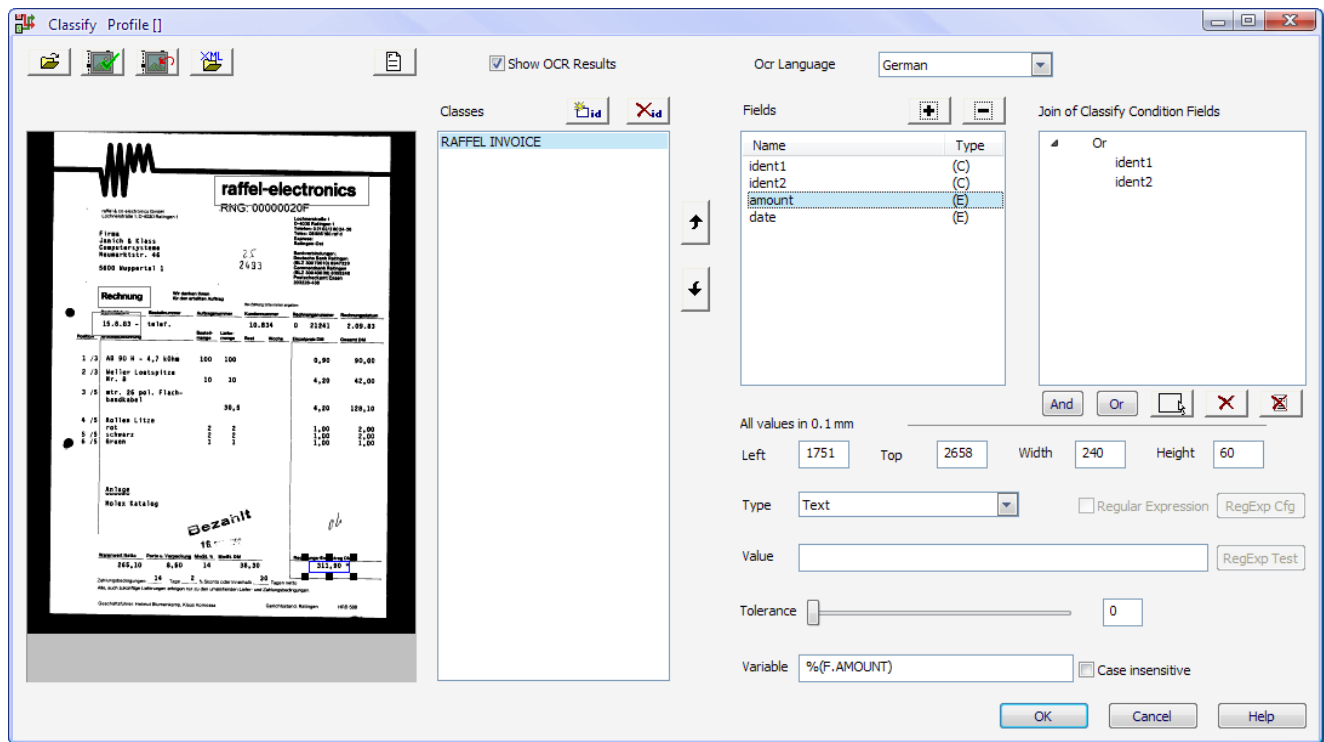


Illustration 1 – Setup Dialog



Loads a file from the disk. All usual file formats are supported.



Generates a new frame. You can modify the frame in size, or move it to another position. Keep the left mouse button pressed down while the cursor is inside the frame, in order to reposition the frame. When you click on the edge of the frame, you can change the size of the frame.



Deletes the active field from the preview window and from the list of Fields.

Attention: All defined operations will be deleted, when a field is deleted, disregarding whether the deleted field was a component of an operation.



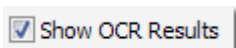
Connects the actual image to the actually active frame. This image can be reread; just click the neighboring button.





Rereads the image that is connected to the active frame. This way, you can restore the image with which you defined the actually active frame.


This is useful for checking the definition.

### Preview window



Displays a document that can be selected with  **Load file**. All usual file formats are supported. Use the mouse buttons to zoom in / zoom out the image, or keep the right mouse button pressed down to move the image when it is larger than the window. Frames that you created with  **Add field** are also shown. They can be moved, or changed in size, both with the mouse.

Executes a Classify test for the image that displays in the preview window. The results of this test are displayed in a message box.

When this checkbox is activated, and the cursor in the preview window is moved across the selected frame, the contents of the OCR search within this frame are displayed. Precondition: The  **Test** button must be clicked, for the actually active image. The OCR results can be copied from the displayed dialog box and be used for the definition of a text for the Classify condition.

A Classify test stores the result of the according OCR search in a file in the PlugIn folder. This file will be overwritten with the next Classify test. When you want to see the contents of the OCR file, you should click this button.

### OCR language

This box serves for the definition of the Classify test. You can set the OCR language that shall be utilized during a test, for the OCR search.

### Classes

All defined Classes are shown in this list. If one of these Classes is selected, all defined elements of this Class are displayed in the according dialog elements.



Creates a new Class. A newly defined Class initially has no defined field(s).



Deletes the actually selected Class



Moves the selected Class one line higher.

The sequence of the Classes may be important. An image is arranged to the first Class for that the Classification condition is fulfilled. Therefore, the sequence can influence the Classification result.



Moves the selected Class one line lower.

The sequence of the Classes may be important. An image is arranged to the first Class for that the Classification condition is fulfilled. Therefore, the sequence can influence the Classification result.

All defined Fields (independent whether they are Extraction Fields, or Classification Fields) are displayed in this list.

Every Field has a name that was defined during its creation. After a right-hand click on an element in this list, this name can be modified. When you select an element in this list, also the according frame in the preview window is selected. If this element is part of a Classify join, it is also selected in the tree (possible only for Classify types).

### Join of Classify Condition Fields

The defined operations are also shown in a tree view.

Only those Classify types can be joined with logical operations, that rule the affiliation with a Classify Class. Extraction types do not appear in this tree view.

You can also simulate the five buttons under the tree view, with your right-hand mouse button. For definition, select the according menu items.



If an element is selected in the tree view, the **And** join is added under this selection. If the tree is empty, the **And** join is added as the first element.



If an element is selected in the tree view, the **Or** join is inserted under this selection. If the tree is empty, the **Or** join is added as the first element.



All available Classify elements are displayed in a menu. Precondition: A logical join must be selected in the tree. Below Classify elements, no additional Classify elements can be inserted.

Here, no Extraction elements are shown. This tree only serves for the join of Classify types (Classify condition, Barcodes and Patchcodes).



Deletes the selected element from the tree structure. If no element is selected, nothing will be deleted.



Deletes all defined joins. This way, the tree view is emptied.

Left

Represents the left-hand coordinate of the actually selected frame in tenth millimeter.

Top

Represents the top coordinate of the actually selected frames in tenth millimeter.

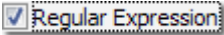
Width

Represents the width of the actually selected frames in tenth millimeter.

Height

Represents the height of the actually selected frames in tenth millimeter.



<b>Type</b>	Select from Classify condition, Text type, Barcode type, Patchcode type, and list field
<b>Classify condition</b>	<p>The defined field serves to classify the documents. This means a Text type, as Regular or non-Regular Expression.</p> <p>Classify types may be combined (by logical operations) and define, in common, the features of a Classify Class.</p>
<b>Barcodes</b>	<p>Also the barcode is a Classify type. Contrary to the Classify condition, it is no Text type that would be compared to an OCR result, but it is the value of the recognized barcode in the area of the defined field. You may define a specific Barcode value that must be recognized, at <b>Value</b>.</p> <p>Here, specially a Regular Expression is useful that allows to also define value ranges. When no value is defined, every detected barcode in the area of the field will fulfill the Classify condition.</p>
<b>Patchcodes</b>	Another Classify type. For Patchcodes, the area is irrelevant because the position of the detected patchcode will not be verified. Only six different types of Patchcodes exist, therefore they must be selected in a fix combo box. Actually it is not possible to fulfill the Classify condition with any random patchcode.
<b>Text</b>	This is an extraction type what means that the condition to be defined here will not rule the Classify Class. Instead, specific text areas can be extracted. Extraction areas area inspected only if the adhering Classification class was recognized. If by chance a text is detected on an image that fulfills an extraction condition, but the image is not arranged to this class, the extraction text will not be delivered.
<b>List Field</b>	Also the list field is an extraction type. As the definition of a list is ample, it can be made in an additional dialog. Please find more details at <a href="#">List Definition</a> .
	<p>Enable this option when you intend to use a Regular Expression as comparison value for your detected text or barcode. You have to enter this Regular Expression in the Edit field value. Leave this option disabled when you want to compare to a fix value, instead to a Regular Expression.</p> <p>You can check the Regular Expression with the <b>RegExp Test</b> button.</p>

**Value**

Here, you define the comparison string. If the Regular Expression option is active, a correct Regular Expression must be defined here. You can check whether the Regular Expression is correct, by clicking the **RegExp Test** button.

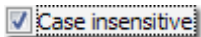
If Regular Expression is not active, you must enter a comparison text here. In this case, we recommend to set a Tolerance. One single character, recognized wrongly, would otherwise lead to a negative comparison result.

In case of Patchcodes, this edit field turns to be a combo box where you can select one of the six known Patchcode types. Tolerance is then inactive.

**Variable**

In order to report the results of a text extraction to DpuScan, a Variable can be defined. The extraction result is then stored in this Variable and can later still be used within DpuScan.

The Variable for a list field is not defined at this place, but in the list definition. Therefore it is inactive for list types.



Makes sense only for Text types. If you like to, you may disable the differentiation for higher / lower characters. In this case, you must activate the checkbox..

**Filter**

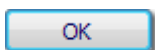
This field will become active only if a Regular Expression was defined. You may select parts of a Regular Expressions that are placed between round brackets.

\1 is the result from the first round brackets in the Regular Expression, and \2 is the result from the second round brackets in the Regular Expression.

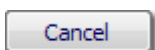
**Tolerance**

If you compare by means of a Text type, or with the Classification type, it makes sense to set a Tolerance. If the OCR search delivers only one wrongly recognized character, the comparison would otherwise have failed. This option is available only for Text types, and when no Regular Expression is used. For Patchcodes, no Tolerance can be entered. At list types, you may enter a Tolerance within the list definition (but it is not available at this place here).

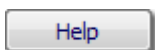
Recommended values for Tolerance range between 5 and 20, depending on the quality of the image. 0 means that the Text must match completely.



Closes the PlugIn configuration and saves all settings.



Closes the PlugIn configuration without saving.



Opens the Help screen

## 4 Field Types for Classification

Classification fields serve to determine the document class, to which the image belongs. Basing on recognition of text, barcode, or patchcode, the field is examined in order to find out whether the formulated condition matches.

Illustration 2 – Definitions for a Classification Field with Text Evaluation

The field type is determined via the drop down list **Type**.

You may select from

- Classification Condition (text comparison)
- Barcode and
- Patchcode

When more than one classification feature seems to be required to identify a class, these features must be joined logically. The joints are shown in a tree view.

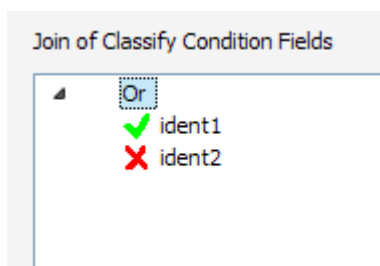


Illustration 3 – Joining Classification Conditions

You may define number of Classes in a configuration. The fields are inspected one after the other, for each of the Classes, and the list is worked off from its top to its bottom.

As soon as the conditions, or their join, are true for a class, the class is successfully determined. The further inspection of the remaining lines in the list is not executed any more. Therefore, the sequence of the Classes can be changed as you like.

The name of the recognized class is entered in the Variable `%(I.CLASSIFY)` and eventually existing fields for data extraction are processed.

Afterwards, all Variables are delivered to DpuScan and can be further processed there, for example by Event Rules, Batch Files, etc.

## 4.1 Classify Condition

The **Classify Condition** marks a field that will analyze the OCR result in the area that is defined by an according frame.

The comparison with the search pattern can be configured as a mere text comparison, or as usage of a Regular Expression.

## 4.2 Text

A **Text** comparison searches for the given text within the text that was recognized by OCR.

You may set a **Tolerance** so that a word is classified to match even if one or two characters are wrong. The Tolerance value is set by means of a slider at the bottom of the window. Here, 0 means that the value must appear exactly as written here. Typing errors, or recognition errors will not be accepted.

With rising Tolerance, more deviations from the given text will be accepted.

Another possibility for comparison is the usage of a **Regular Expression**.

When this Expression finds a match, the field is successfully detected.

Please read more about Regular Expressions in Chapter 7 [Regular Expressions](#) .

## 4.3 Barcode

For Classification, you may also use the barcode value of the DpuScan Variable `%(S.BAR1)`. This barcode value can be evaluated analog to the field type "Classify Condition". You may use a text comparison with tolerance, or a Regular Expression.

Please read more about Regular Expressions in Chapter 7 [Regular Expressions](#) .

## 4.4 Patchcode

For Classification, you may also use the patchcode value of the DpuScan Variable `%(S. PATCH)`. The patchcode value can be evaluated analog to the field type "Classify Condition". You may use a text comparison with tolerance, or a Regular Expression.

Please read more about Regular Expressions in Chapter 7 [Regular Expressions](#) .

## 5 Field Types for Data Extraction

Reading characters or values is later denominated as data extraction.

Here, in contrary to simple Classification, the read value shall be returned to the scan program, as per the defined conditions.

So, for example, you may search for a date, or for the number of an invoice, where it will not be sufficient to just state that such an Expression was found, but also the value itself must be returned.

Three different types of fields are available to fulfill this demand:

In a Text field, a search can be executed for a fix text with settable error tolerance, or with Regular Expressions.

### 5.1 Text Field

The **Text field** serves for a simple search for text elements in a specific area of the image. The area of the search field is set by a frame on the image.

When such an element could be found, the result of the search is saved in a freely definable Variable. The name of the Variable is indicated in the **Variable** input field.

If the search text could not be found, the Variable remains empty.

You can configure the search as a mere text comparison, or as usage of a Regular Expression.

At this field type, the Plugin searches in the text that was recognized by OCR for the entry in the **Value** input field.

When a mere text comparison is selected, so that the **Regular Expression** check box has not been hooked, the **Tolerance** slider is active, and a tolerance value can be entered so that a word is still interprets as a match even if, for example, one or two letters are wrong. Here, 0 means that the value must appear exactly as written here. Typing or recognition errors will not be accepted. With rising Tolerance, more deviations from the entered text will be accepted.

Another comparison possibility is the usage of a Regular Expression. This method is selected by marking the **Regular Expression** check box. When the Expression in the **Value** input field finds a match, the field is successfully found. In case that only a part of the Expression shall be stored in the Variable, you can enter it in the **Filter** input field.

Please read more about Regular Expressions in Chapter 7 [Regular Expressions](#) .

## 5.2 List Fields

A **Fields** list contains a list with any number of lines. In each of the lines, one or two conditions are entered that work like the earlier described **Text** field. If two conditions are entered in one only line, they must be joined logically.

Every single line of the list is compared to the OCR results from the search area belonging to the list. The comparison is done in the same sequence as the lines are placed in the list.

When the conditions of a line match, the Variable will be populated with the value as fixed in the line. The further inspection of the remaining lines in the list is not executed any more. The return value can freely be selected and is entered per line of the list.

The following join modes are allowed:

- And-join
- Or-join
- AndNot-join

Every line in this list can be defined individually. It consists of the return value and up to two comparison strings that are joined by a logical operation.

### Dialog elements

Line	<p>A line is defined by the <b>Add</b> or <b>Edit</b> button and consists of the return value, the first comparison string, the first comparison type of the logical operation, and the second comparison string and comparison type.</p> <p>The return value is any text that shall be returned in case of a successful comparison for the defined line.</p> <p>The type is either a Regular Expression, or a Tolerance value. These names have already been explained in the configuration of the PlugIn. Tolerance allows a value between 0 and 100.</p> <p>Logical operations allow, further to <b>And / Or</b>, also <b>No</b> (when only the first of the two strings shall be evaluated) and <b>And Not</b>.</p>
Variable	<p>Defines the name of the return variable for DpuScan. This Variable name is the same for all lines. Only the return value that can be defined separately for every line will change.</p>
Add	<p>Adds a new line at the end of the list. Another dialog will open to define the individual elements of a line.</p>
Edit	<p>Precondition for editing a line of the list: At least one line has been defined and is selected. The dialog will open that was also used for adding a line. The individual elements of a line have their actual value and can be modified.</p>
Delete	<p>Deletes the selected line from the list.</p>
Delete all	<p>Deletes all defined lines from the list.</p>

**Import**

Allows to open a text file. This way, a complete list definition can be added in one single step. The import file must follow a specific syntax:

Lines must be separated by Carriage Return/Linefeed.

The individual line elements are separated by #.

Tolerances must have a value between 0 and 100 and thus indicate that no Regular Expression shall be used. Every value that is longer than three characters will lead to an interpretation of the input as Regular Expression, independent of the contents of the string.

At these operations, it is irrelevant whether you write capitals, or low letters, but only the values **And**, **Or**, **And Not**, and **No** are accepted.

### 5.3 Relative Search

The relative search means a method where two search criteria are joined in a way that ...

- the starting position is determined by the first search Expression, and
- then, in relative distance from the first found Expression, the second Expression is searched.

This way you can reach that, for example, a definable area below the Expression **"Date"** is inspected by use of a Regular Expression for a numbers structure that might match the indication of a date.

Define relative Value

Search Key: date Tolerance:

Relative Search Text

Reg Expression: (\d\d.\d\d.\d\d) Check

Filter: \0

Variable: %(J.DATE)

Search Area relative to begin of Search Key (Values in 0.1 mm)

Left border: left of word 0	Top border: retained 0
Width: Value 0	Height: Value 0

OK Cancel

*Illustration 4 – Relative Search for the Value of a Date*

If the first Expression could be found, and if also the search for the second Expression was successful, the value of the second Expression can be stored in a Variable and be delivered to DpuScan.



## 6 How to Use the PlugIn in DpuScan

The PlugIn requires the data coming from the OCR full-page search with exported XML format as also information about the recognized barcodes and patchcodes. In order to make sure that these data are available when the PlugIn is invoked, it is necessary to work in OpenJob mode. Also, the OCR full-page search, barcode search, and patchcode search must have been executed before the invocation of the Classify PlugIn. Therefore, the indispensable Task actions are as follows:

```
Load Base Profile
Load Batch (OpenJob)
  Load from Scanner (scan process)
  Barcode Search on Images
  Patchcode Search on Images
  Set Pathname
  Set Filename
  OCR on Images
  Call PlugIn for every image (call of the Classify PlugIn)
  Save Image to Disk
```

## 7 Regular Expressions

A Regular Expression (abbreviates also: RegExp or Regex) is a search pattern that is utilized to detect a text, a specific string like, for example, a word.

The search pattern is composed following a complex syntax.

The syntax variant to be utilized can be set for the relative field.

Click on the  button.

### 7.1 Syntax

Remark: For better readability in this text, some Regular Expressions are cited in quotation marks ". ". In this case, however, these quotations marks are *not* part of the search expression and must be left out for the search.

Usually, a Regular Expression consists of wild cards and eventually characters that must exactly occur in the text to be searched. These characters are so-called literals.

#### Literals

All characters except ". ", "\*", "?", "+", "(", ")", "{", "}", "[", "]", "^", "\$" and "\" are literals. If point, asterisk, question mark, plus, roof, Dollar, round or curly or square brackets, and also the back slash shall be treated as literals nevertheless, they must be „masked“ by a preceding back slash.

If, for example, you want to search for the filename `C:\temp\file.txt`, you must enter the back slash twice in the Regular Expression. Also, the point before the file type must be masked:

```
"C:\\Temp\\file\\.txt"
```

***In the Classification Plugin, the search always respects high / low letters.***

#### Wildcards

The dot ". " stands for any single character.

#### Repetitions, Curly Brackets

A Repetition Expression is an Expression that indicates how often the heading Expression may occur.

"\*" The Asterisk behind an Expression means that this Expression may occur often, or not at all.

"+" The Plus means that this Expression may occur often, but at least once.

"?" The Question Mark indicates that an Expression may occur once at max (or not at all).

"{x,y}" The values in curly brackets indicate how often an Expression may occur: at least x or y at max. Here, the minimum value x must always be set, but the maximum value y may be left open. When no maximum value is indicated, the searched Expression may occur any number of times.

Examples:

"Ab\*"

Search for a single A probably followed by b's, will find "A", "Ab", "Abbb" etc.

"Ab+"

Search for an A followed by at least one b, will find "Ab" or "Abbbb" but not "A".

"Ab?"

Search for an A followed by at least one b, will find "A" or "Ab".

"Ab{2,4}"

Search for an A followed by at least two and at max four b's, will find

"Abb", "Abbb" and "Abbbb".

"Ab{,5}"

Search for an A followed by maximum five b's

"Ab{3,}"

Search for an A followed by at least three b's

In these examples, the repetition expressions always relate to one single character, the „b“. When you want to repeat character strings, you must utilize bracket.

## Combinations, Round Brackets

Round brackets serve for combination of search patterns, and for grouping of the results.

**In the Classify-Plugin, a search pattern must always be enclosed in brackets.**

The setup dialog offers the possibility to fetch single results groups. Here, the result filter "\1" is used as wildcard for the first partial result.

"(Ab)\*"

searches for the string Ab with any number of repetitions, and will find Ab AbAb etc.

".\*(amount).\*"

searches for the word amount, where any number of characters may be placed before or behind the word. The result is saved as the first recognized text and can be called with "\1".

**When a string is searched with the Classification-Plugin in a text, the search pattern for the text must be entered before and after the part to be searched.**

When you are interested in the texts before and behind, you can bracket them, too:

"(.\*)(amount)(.\*)"

In this case, the word amount is placed in the second part and can be called with "\2".

If, on the contrary, the expression between brackets starts with "?:" then the sub-expression is nevertheless detected, but not returned.

"(?:.\*)(amount)(?:.\*)"

Now, the word amount is again found in the first part "\1". It is important to mention that the search now no longer works eroding.

**Alternatives, Vertical Lines**

Vertical lines can be used in order to select between two each possible Expressions

```
" ( Jan | January ) "
```

searches for January, or Jan, in a text,

**Classes, Square Brackets, Roof**

Classes are combinations of characters to be searched for. They are enclosed by square brackets and contain the characters that shall be allowed.

```
" ( [ UuMmWmSsTt ] + ) "
```

searches in a text for a word that consists of the letters UMWST, higher or lower, and finds, for example: UST, Ust, MWST, MWSt but also stumm.

Instead of listing the characters, you can also define an area:

```
" ( [ A-Z ] + ) "
```

searches for all higher letters in a text.

```
" ( [ A-z ] + ) "
```

searches in a text for a word that consists of letters, but not of numbers, for example.

The roof means the inversion: only characters are used that are *not* listed.

```
" ( [ ^a-z ] + ) "
```

searches in a text for a word that contains no lower letters.

Instead of listing letters, you may also use the names of Classes. For doing so, these constants are foreseen:

" [ [:alnum:] ] "	letters or numbers
" [ [:word:] ] "	letters, numbers or base lines
" [ [:alpha:] ] "	letters
" [ [:digit:] ] "	numbers
" [ [:punct:] ] "	punctuation signs (dot, comma, semicolon, all brackets, ...)
" [ [:print:] ] "	any printable character
" [ [:space:] ] "	blank spaces, tabs, line feeds, page feeds
" [ [:upper:] ] "	higher letters
" [ [:lower:] ] "	lower letters

For often used Classes, there are even shortcuts

"\d" numbers

"\w" letters, numbers, or base lines

"\s" blank spaces, tabs, line feeds, page feeds

"\u" higher letters

"\l" lower letters

"([[:alpha:]]+)"

searches for a sequence of letters, what means a word.

"(\d{1,4})"

searches for a one-digit up to four-digit number, and finds 123 or 5

## 7.2 Examples for Regular Expressions

### VAT ID (Germany)

In Germany, the VAT ID starts with the string "DE", followed by 9 numbers.

`"DE\d{9}"`

`DE` searches for an exact match with the string `DE`.

`\d{9}` searches for exactly 9 numbers

### Amount

You search for the number behind the word Total or Subtotal.

For example 123,45 or 67,--

`"(Total|Subtotal) (\s*) (\d{1,} [.,] {1} [-] |\d {2}) "`

`(Total|Subtotal)` searches in the text for an exact match with the word `Total` or the word `Subtotal`

`(\s*)` searches for any number of blank spaced

`(\d{1,} [.,] {1} [-] |\d {2})` searches for at least one number, followed by exactly one dot, or comma, followed by two minus signs, or numbers.

This last part will find, for example: `122,18` but also `150, --`

In order to get the number, the third part of the Expression must be referenced. Therefore the entry `\3` in field "Filter" delivers the amount.

### Order Number

You search for the string after the word part `umber` or `nr`. The order number may – depending on the supplier – have up to 30 digits and may contain separation signs.

`"(umber|nr\.) (\s*) ([[:digit:]][:punct:]]{1,30}"`

`(umber|nr\.)` searches for an exact match in the text with the word part `umber` or the shortcut `nr`

`(\s*)` searches for any number of blank spaces

`([[:digit:]][:punct:]]{1,30})` searches for at least one number or punctuation mark, but for 30 such characters at max.

This last part will find, for example: `122.18.4432.1234` or also `122/18/4432.1234`

**Hint:**

It may occur that an Expression delivers no result although results are shown in the window for testing the Regular Expressions. The reason may be that the OCR characters were found earlier than the structure itself. Please try to place pattern like (.\*) before the search pattern, or to add them.



Janich & Klass Computertechnik GmbH  
Zum Alten Zollhaus 24  
D-42281 Wuppertal  
Germany

Tel.: +49 (0)202 2708-0  
Fax: +49 (0)202 700 625  
<http://www.janichklass.com>  
<http://www.JKImaging.com>

408.201002.001 1