Regular expression examples

Cloud Insights

Tony Lavoie
April 02, 2021

Table of Contents

Regular expression examples

If you have selected the regular expression approach as your source naming strategy, you can use the regular expression examples as guides for your own expressions used in the Cloud Insights automatic resolution methods.

Formatting regular expressions

When creating regular expressions for Cloud Insights automatic resolution, you can configure output format by entering values in a field named *FORMAT*.

The default setting is \1, which means that a zone name that matches the regular expression is replaced by the contents of the first variable created by the regular expression. In a regular expression, variable values are created by parenthetical statements. If multiple parenthetical statements occur, the variables are referenced numerically, from left to right. The variables can be used in the output format in any order. Constant text can also be inserted in the output, by adding it to the FORMAT field.

For example, you might have the following zone names for this zone naming convention:

```
[Zone number]_[data center]_[hostname]_[device type]_[interface number]
```

- S123_Miami_hostname1_filer_FC1
- S14_Tampa_hostname2_switch_FC4
- S3991_Boston_hostname3_windows2K_FC0
- S44_Raleigh_hostname4_solaris_FC1

And you might want the output to be in the following format:

```
[hostname]-[data center]-[device type]
```

To do this, you need to capture the host name, data center, and device type fields in variables, and use them in the output. The following regular expression would do this:

```
.*?_([a-zA-Z0-9]+)_([a-zA-Z0-9]+)_([a-zA-Z0-9]+)_.*
```

Because there are three sets of parentheses, the variables \1, \2 and \3 would be populated.

You could then use the following format to receive output in your preferred format:

```
\2-\1-\3
```

Your output would be as follows:

```
hostname1-Miami-filer
hostname2-Tampa-switch
hostname3-Boston-windows2K
hostname4-Raleigh-solaris
```

The hyphens between the variables provide an example of constant text that is inserted in the formatted output.

Example 1 showing zone names

In this example, you use the regular expression to extract a host name from the zone name. You could create a regular expression if you have something similar to the following zone names:

- S0032_myComputer1Name-HBA0
- S0434_myComputer1Name-HBA1
- S0432_myComputer1Name-HBA3

The regular expression that you could use to capture the host name would be:

```
S[0-9]+_([a-zA-Z0-9]*)[_-]HBA[0-9]
```

The outcome is a match of all zones beginning with S that are followed by any combination of digits , followed by an underscore, the alphanumeric hostname (myComputer1Name), an underscore or hyphen, the capital letters HBA, and a single digit (0-9). The hostname alone is stored in the **\1** variable.

The regular expression can be broken into its components:

- "S" represents the zone name and begins the expression. This matches only an "S" at the beginning of the zone name.
- The characters [0-9] in brackets indicate that what follows "S" must be a digit between 0 and 9, inclusive.
- The + sign indicates that the occurrence of the information in the preceding brackets has to exist 1 or more times.
- The _ (underscore) means that the digits after S must be followed immediately by only an underscore character in the zone name. In this example, the zone naming convention uses the underscore to separate the zone name from the host name.
- After the required underscore, the parentheses indicate that the pattern contained within will be stored in the \1 variable.
- The bracketed characters [a-zA-Z0-9] indicate that the characters being matched are all letters (regardless of case) and numbers.
- The * (asterisk) following the brackets indicates that the bracketed characters occur 0 or more times.
- The bracketed characters [_-] (underscore and dash) indicate that the alphanumeric pattern must be followed by an underscore or a dash.
- The letters HBA in the regular expression indicate that this exact sequence of characters must occur in the zone name.
- The final set of bracketed characters [0-9] match a single digit from 0 through 9, inclusive.

Example 2

In this example, skip up to the first underscore "*", then match E and everything after that up to the second "*", and then skip everything after that.

Zone: Z_E2FHDBS01_E1NETAPP

Hostname: E2FHDBS01

RegExp: .**?*(E.*?)**.*?

Example 3

The parentheses "( )" around the last section in the Regular Expression (below) identifies which part is the hostname. If you wanted VSAN3 to be the host name, it would be: _([a-zA-Z0-9]).*

Zone: A_VSAN3_SR48KENT_A_CX2578_SPA0

Hostname: SR48KENT

RegExp: _[a-zA-Z0-9]+_([a-zA-Z0-9]).*

Example 4 showing a more complicated naming pattern

You could create a regular expression if you have something similar to the following zone names:

- myComputerName123-HBA1_Symm1_FA3
- myComputerName123-HBA2_Symm1_FA5
- myComputerName123-HBA3_Symm1_FA7

The regular expression that you could use to capture these would be:

```
([a-zA-Z0-9]*)_.*
```

The \1 variable would contain only *myComputerName123* after being evaluated by this expression.

The regular expression can be broken into its components:

- The parentheses indicate that the pattern contained within will be stored in the \1 variable.
- The bracketed characters [a-zA-Z0-9] mean that any letter (regardless of case) or digit will match.
- The * (asterisk) following the brackets indicates that the bracketed characters occur 0 or more times.
- The _ (underscore) character in the regular expression means that the zone name must have an underscore immediately following the alphanumeric string matched by the preceding brackets.
- The . (period) matches any character (a wildcard).
- The * (asterisk) indicates that the preceding period wildcard may occur 0 or more times.

  In other words, the combination .* indicates any character, any number of times.

Example 5 showing zone names without a pattern

You could create a regular expression if you have something similar to the following zone names:

- myComputerName_HBA1_Symm1_FA1
- myComputerName123_HBA1_Symm1_FA1

The regular expression that you could use to capture these would be:

```
(.*?)_.*
```

The \1 variable would contain *myComputerName* (in the first zone name example) or *myComputerName123* (in the second zone name example). This regular expression would thus match everything prior to the first underscore.

The regular expression can be broken into its components:

- The parentheses indicate that the pattern contained within will be stored in the \1 variable.
- The .* (period asterisk) match any character, any number of times.
- The * (asterisk) following the brackets indicates that the bracketed characters occur 0 or more times.
- The ? character makes the match non-greedy. This forces it to stop matching at the first underscore, rather than the last.
- The characters _.* match the first underscore found and all characters that follow it.

Example 6 showing computer names with a pattern

You could create a regular expression if you have something similar to the following zone names:

- Storage1_Switch1_myComputerName123A_A1_FC1
- Storage2_Switch2_myComputerName123B_A2_FC2
- Storage3_Switch3_myComputerName123T_A3_FC3

The regular expression that you could use to capture these would be:

```
.*?_.*?_([a-zA-Z0-9]*[ABT])_.*
```

Because the zone naming convention has more of a pattern, we could use the above expression, which will match all instances of a hostname (myComputerName in the example) that ends with either an A, a B, or a T, placing that hostname in the \1 variable.

The regular expression can be broken into its components:

- The .* (period asterisk) match any character, any number of times.
- The ? character makes the match non-greedy. This forces it to stop matching at the first underscore, rather than the last.
- The underscore character matches the first underscore in the zone name.

- Thus, the first .*?_ combination matches the characters Storage1_ in the first zone name example.
- The second .*?_ combination behaves like the first, but matches Switch1_ in the first zone name example.
- The parentheses indicate that the pattern contained within will be stored in the \1 variable.
- The bracketed characters [a-zA-Z0-9] mean that any letter (regardless of case) or digit will match.
- The * (asterisk) following the brackets indicates that the bracketed characters occur 0 or more times.
- The bracketed characters in the regular expression [ABT] match a single character in the zone name which must be A, B, or T.
- The _ (underscore) following the parentheses indicates that the [ABT] character match must be followed up an underscore.
- The .* (period asterisk) match any character, any number of times.

The result of this would therefore cause the \1 variable to contain any alphanumeric string which:

- was preceded by some number of alphanumeric characters and two underscores
- was followed by an underscore (and then any number of alphanumeric characters)
- had a final character of A, B or T, prior to the third underscore.

Example 7

Zone: myComputerName123_HBA1_Symm1_FA1

Hostname: myComputerName123

RegExp: ([a-zA-Z0-9]+)_.*

Example 8

This example finds everything before the first _.

Zone: MyComputerName_HBA1_Symm1_FA1

MyComputerName123_HBA1_Symm1_FA1

Hostname: MyComputerName

RegExp: (.**?)_.**

Example 9
This example finds everything after the 1st _ and up to the second _.

Zone: Z_MyComputerName_StorageName

Hostname: MyComputerName

RegExp: .**?(.?)**.*?

Example 10

This example extracts "MyComputerName123" from the zone examples.

Zone: Storage1_Switch1_MyComputerName123A_A1_FC1

Storage2_Switch2_MyComputerName123B_A2_FC2

Storage3_Switch3_MyComputerName123T_A3_FC3

Hostname: MyComputerName123

RegExp: .**?.?**([a-zA-Z0-9]+)**[ABT]_.**

Example 11

Zone: Storage1_Switch1_MyComputerName123A_A1_FC1

Hostname: MyComputerName123A

RegExp: .**?.?**([a-zA-z0-9]+).*?

Example 12

The ^ (circumflex or caret) **inside square brackets** negates the expression, for example, [^Ff] means anything except uppercase or lowercase F, and [^a-z] means everything except lowercase a to z, and in the case above, anything except the _. The format statement adds in the "-" to the output host name.

Zone: mhs_apps44_d_A_10a0_0429

Hostname: mhs-apps44-d

RegExp: ()_([AB]).*Format in Cloud Insights: \1-\2 ([^_])_
()_([^_]).*Format in Cloud Insights: \1-\2-\3

Example 13

In this example, the storage alias is delimited by "\" and the expression needs to use "\\" to define that there are actually "\" being used in the string, and that those are not part of the expression itself.

Storage Alias: \Hosts\E2DOC01C1\E2DOC01N1

Hostname: E2DOC01N1

RegExp: \\.**?\\.**?\\(.*?)

Example 14

This example extracts "PD-RV-W-AD-2" from the zone examples.

Zone: PD_D-PD-RV-W-AD-2_01

Hostname: PD-RV-W-AD-2

RegExp: -(.*-\d).*

Example 15

The format setting in this case adds the "US-BV-" to the hostname.

Zone: SRV_USBVM11_F1

Hostname: US-BV-M11

RegExp: SRV_USBV([A-Za-z0-9]+)_F[12]

Format: US-BV-\1