Cacti

„Autom8“ plugin

Usage Guide

Brought to you by
Reinhard Scheck alias gandalf@forums.cacti.net
The Cacti Group

Table of Contents

Purpose of this Plugin....3
Create “Meaningful” Graphs....3
Trigger Graph Creation on Multiple Events....3
Create Tree Items....3
Installing the Plugin....4
Providing Authorization to Users....6
Basic Usage....8
Graph Rules....9
Graph Rules: Matching Rule Items....11
Graph Rules: Create Graphs Items....13
Create a New Graph Rule....14
New Graph Rule: Define Matching Rule Items....15
New Graph Rule: Define Action Rule Items....16
Testing a Graph Rule....17
Tree Rules....19
Tree Rule: Host Leaf Item....20
Tree Rules (Host): Matching Rule Items....21
Tree Rules (Host): Action Rule Items....22
Tree Rule: Graph Leaf Item....25
Tree Rules (Graph): Matching Rule Items....26
Tree Rules (Graph): Action Rule Items....27
Testing a Tree Rule....28
Best Practices....29

Purpose of this Plugin

Maintaining a quite decent Cacti installation with lots of devices, graphs and tree items may result in quite a huge amount of administrative work.

Using the cli scripts allows you to write your own scripts to e.g. create Cacti device entries, graphs, tree items and more. This helps e.g. interfacing a CMDB or some other repository that holds information about devices that shall be monitored.

Create “Meaningful” Graphs

Of course, when creating a new device you will at least want to create all graphs related to “Associated Graph Templates”. But when it comes to Data Queries, it is not always desired to create graphs for all indexes; e.g. you will not want to graph each and every interface, e.g. if it's ifOperStatus is “down”. So you will make up your mind to restrict graph creation to “meaningful” indexes.

In case you are able to define “meaningful” as a (set of) rule(s), you've just caught the goal of this plugin!

In most cases, those rules depend on some device properties, e.g. a “64 bit Traffic Graph” makes no sense if the SNMP Version of the device is 1.

Trigger Graph Creation on Multiple Events

Cli scripts mostly are used when a new device is created. This plugin will be invoked each time a new device is created, be it via cli or via browser.

But this plugin uses hooks that are triggered not only when a new device is created. It features graph creation each time a new re-index occurs. This way, if you add a new specific data query that matches a given rule, all graphs will be created in about no time. This virtually eliminates visiting the page “Create Graphs for this Host”.

And more, you may call the rule execution from device management as a new dropdown action with a selected list of devices. This is of special use when testing your rules.

Create Tree Items

Using the Graph Tree is a very flexible means of grouping what you desire to be grouped. In many cases it turns out that your trees will apply to some algorithm; let's again call it a rule.

You may e.g. want to create a tree that holds all host entries. But perhaps you do not want them crowded up as a flat list of items. Instead, you may find it useful to group them by Host Template. Of course, more sophisticated and nested rule will come up to your mind.

Autom8 now turns out to be able to catch virtually any property (column) related to your device of graph to build rule based trees for you. To extend this flexibility, use of regular expressions allows it to parse virtually any string out of those properties for use in new tree headers.

These Tree Rules take respect of already created trees, sub-trees and tree items. It won't change any existing tree item but only add new ones as you define those.

Installing the plugin

As a prerequisite, the Plugin Architecture is required; this plugin requires at least plugin Architecture version 2.4 or higher. Installation of plugin Architecture is not described here, please see http://cactiusers.org/ for help regarding this topic. Then, please download this plugin to the <path_cacti>/plugin directory. Next, unpack the .tgz file. All files will now reside in the <path_cacti>/plugin/autom8 directory.

You will notice several *.patch files in a sub-directory named to match your current cacti version. Starting with Cacti 0.8.8, the required patches will be part of the standard cacti distribution. The patches mainly contain new hooks required for this plugin. And there is a code change for api_tree.php that will be part of upcoming Cacti releases but is required for the proper operation of this plugin.

Please install **all** those *.patch files from main cacti directory using e.g.

```
patch -p1 -N –dry-run < plugins/autom8/<version>/cli.patch
```

and then, if successful

```
patch -p1 -N < plugins/autom8/<version>/cli.patch
```

<version> equals “patches-087d” or “patches-087e”, depending on the Cacti version used.

CAVEAT: Please make sure not to leave old „autom8“, e.g. <path_cacti>/plugin/autom8_old!

Assuming, you already have provided authorization to Plugin Management, please go to that menu item.

autom8	
Directory:	autom8
Version:	0.20
Author:	Reinhard Scheck
Home Page:	
Status:	Not Installed
	Install | Uninstall | Enable | Disable | Check

Illustration 1: Install Autom8, Step 1

Hit „Install“:

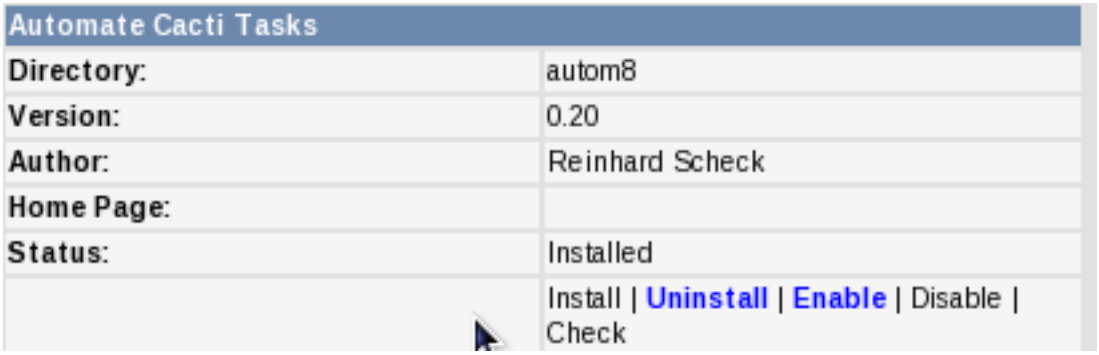

Illustration 2: Install Autom8, Step 2

Hit „Enable“:

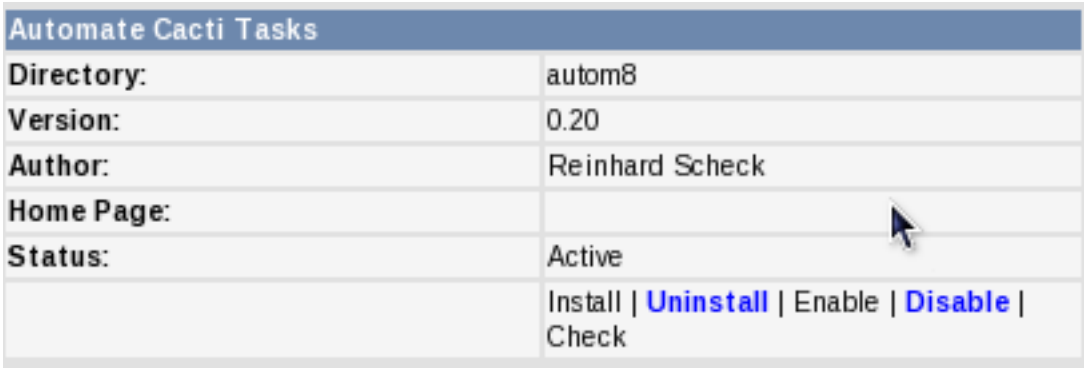

Illustration 3: Installation, Step 3 (final)

Providing Authorization to Users

As this plugin has high impact on Cacti operation, it is secured by a realm. For each user that is granted access to define an enable new rules, you will have to see the User Management page and check the appropriate check-box

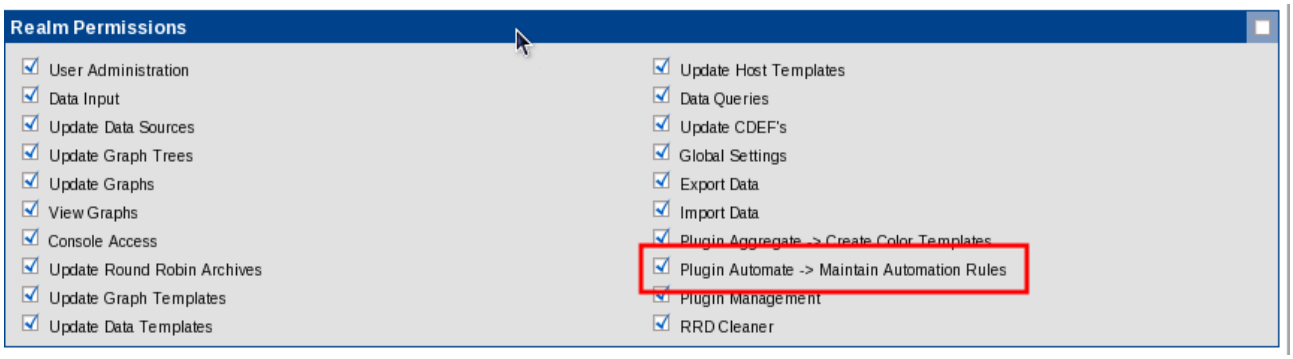

Illustration 4: Realm Permission for Autom8 Rules

This will then result in two new management options appearing on the menu bar:

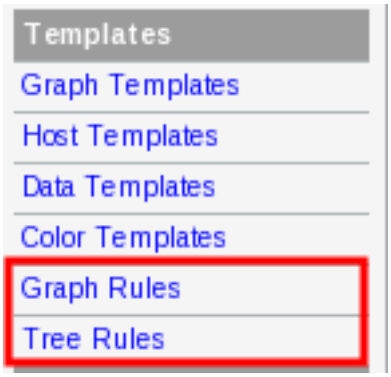

Illustration 5: New Menu Options for Rule Management

Basic Usage

The current concept of this plugin allows to create two different types of objects

- a graph
- a tree entry (along with all “parent tree items”)

In other words: the host has to be in place already; it will not be created by this plugin You may create the host using any of the standard cacti features, be via browser or via cli script.

The necessary hooks are in place to operate this plugin instantaneously. There are several occasions, when you may want this plugin to interfere:

- each time a new host is created, you will want this plugin to create all “relevant” graphs
- each time a data query is re-indexed and new indexes are discovered, you will want this plugin to create new graphs for those interfaces “if desired”
- each time a new host is added, you “may want” to automatically create the appropriate host entry for a specific tree
- each time a new graph is created, even if created by this plugin, you “may want” to add an entry to a tree

Each ticked item describes an optional action that has to be defined by a “**matching rule**”. So you will have to define **Rules** to make this plugin take control. E.g. you may want to restrict automatic graph creation to specific hosts or host related to specific host templates. Same holds for tree item creation. This is the first step of autom8 processing.

Now, that the plugin knows that it is involved by means of the first step, you will want to define the action that has to be taken. This is called an “**action rule**”. It comes in two flavors

- action rules for graph creation
- action rules for tree item creation

For ease of testing, all rules allow for displaying the list of matching objects. And for triggering this plugin, you may not want to wait until a rule matches. This is why new “actions” were defined for Device and Graph Management.

Please notice, that this plugin does not change any permissions on graphs nor does it allow to create new permissions automatically.

Graph Rules

The hooks provided for this plugin will create **all** “simple” Graph Template graphs that are associated with the host in question. It is recommended, that those Graph Templates are defined by the Host Template and no superfluous Graph Templates are listed.

It makes virtually no sense to defined rules (filters) for the Graph Templates that should be used automatically, because this is usually done by defining the Host Template appropriately. So, basically, the Graph creation covers Data Queries.

But how to make a specific host eligible for automatic graph creation? Or, changing the point of view, how to prevent the “wrong” hosts from being treated by this plugin?

After a pristine installation of this plugin, you will find two Graph Rules that we shall discuss now:

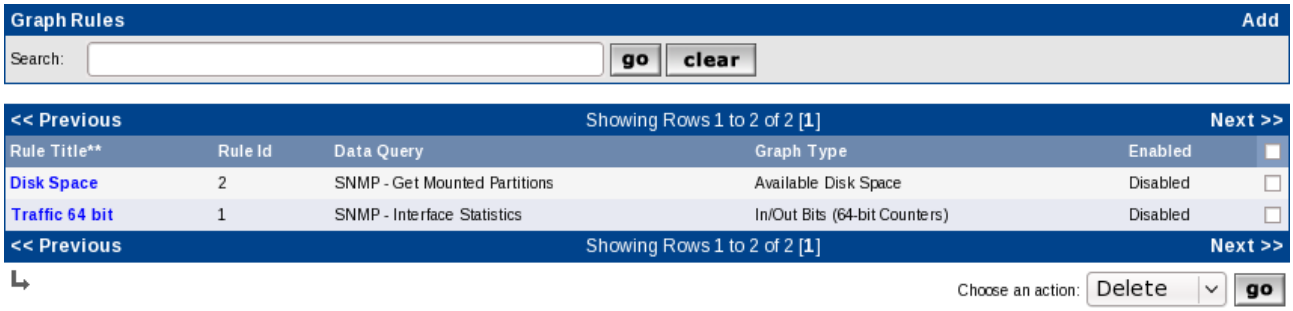

Illustration 6: Default Graph Rules

For an explanation of features, let's turn to the rule named “Traffic 64 bit”. The most important properties of this rule are listed here. Please notice, that all rules are “disabled” by default to prevent accidental and unwanted actions.

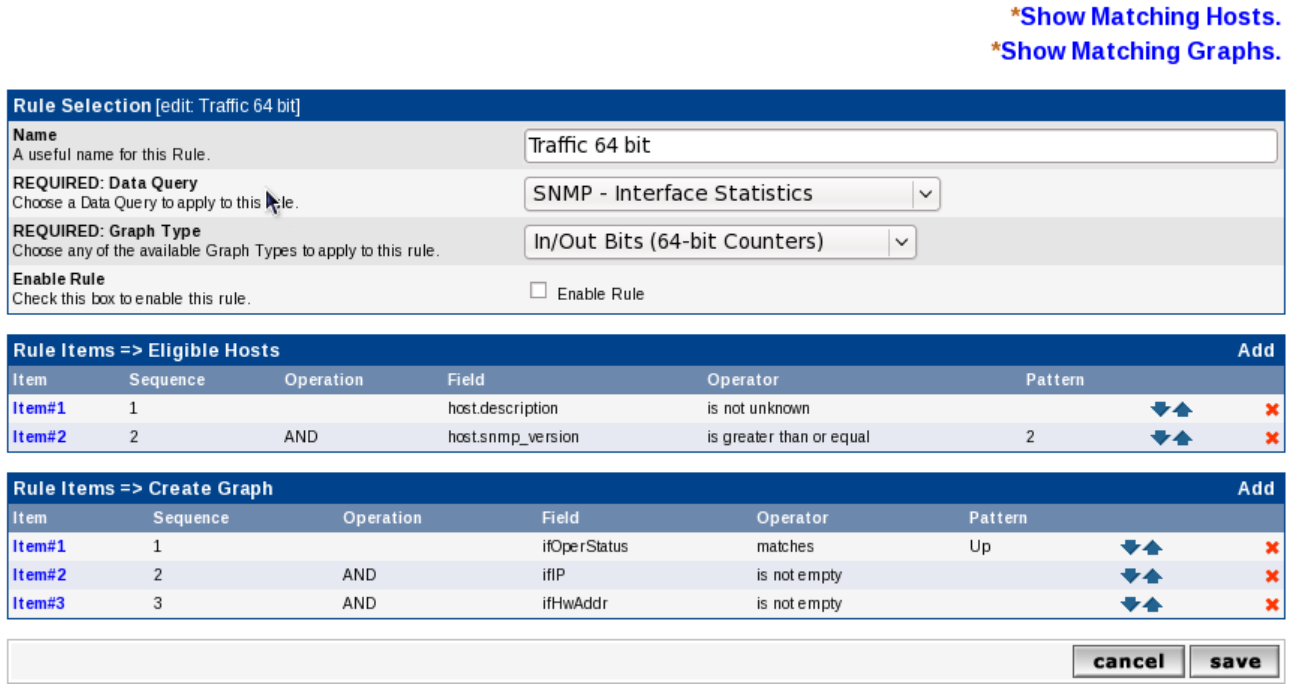

Illustration 7: “Traffic 64 bit” Graph Rule

The properties are defined as shown below

Property	Description
Name	To be defined freely
Data Query	The Data Query this rule shall act upon
Graph Type	The type of graph that shall be created
Enable Rule	Unless the Rule is complete, you will not want it to be executed. It is a good practice to go through several steps of rule verification, see below.
Rule Items => Eligible Hosts	Define properties of hosts that shall match this rule. This is called a “matching rule”
Rule Items => Create Graphs	Define the properties related to the Data Query on how to create new graphs. This is called an “action rule”

Graph Rules: Matching Rule Items

Let's have a look at some rule items:

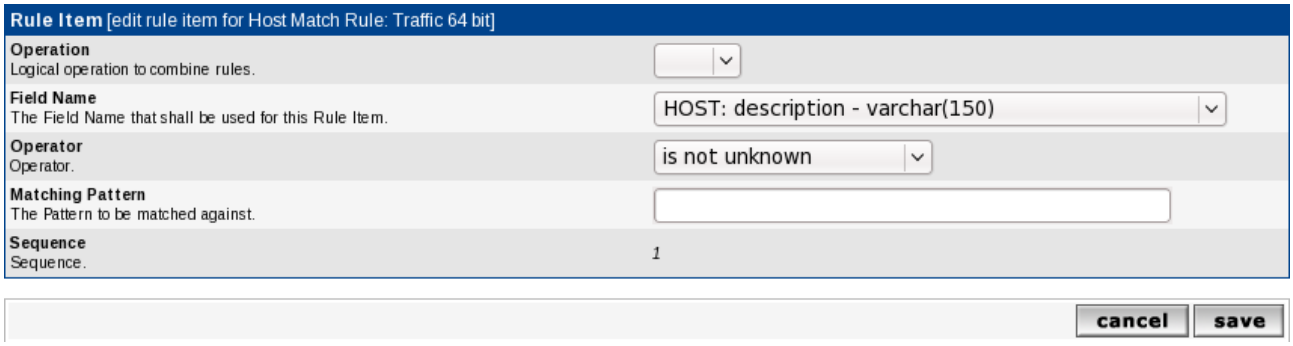

Illustration 8: Graph Rule Item 1

This rule item defines, that the description of the host must “not be unknown”. In other words, this rule matches all hosts as no host has an empty host description. This rule is of particular use when starting a new rule, because the list of matching hosts simply equals “all available hosts”. You may then add further restrictive rules to narrow the selection down. Such is the second rule item:

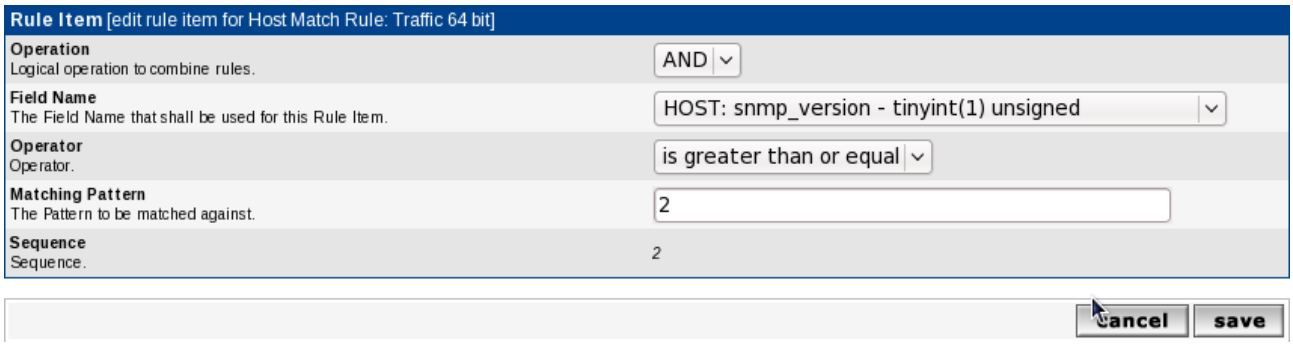

Illustration 9: Graph Rule Item 2

The operation is AND, the field operated upon is the host's SNMP version. As you may know, 64 bit traffic counters are supported only with hosts running at least SNMP V2c (or SNMP = 2 in Cacti terms). The field name is build up in three parts:

- table, HOST in this case, printed in uppercase
- table field, snmp_version in this case
- database field properties, tinyint(1) unsigned in this case

The test itself is a “>=”, the matching pattern a “2”. All in all, this rule requires a host to have at least SNMP version 2 defined in its host properties to match.

You will want to test this rule. Does it match all hosts you expected? Click “Show Matching Hosts” to see:

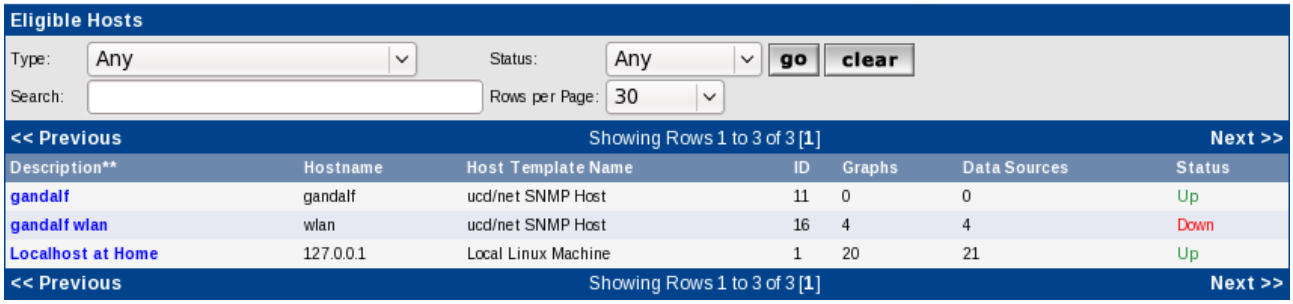

Illustration 10: Eligible Hosts

You may now check, whether all desired hosts show up in the list along with some important properties. The hosts are directly linked in the first column. So may may quickly check all host's properties if in doubt. In most cases, your list will be huge. This is where you may enter some filter conditions to find specific hosts. The filters are NOT applied as a new rule item; if you want to do so, you will have to add the rule explicitly.

Graph Rules: Create Graphs Items

This example illustrates how to create SNMP V2c traffic graphs using 64 bit counters. But you may want to restrict automatic graph creation not only to certain hosts. You may want to narrow new graphs down to “meaningful interfaces”. But how to tell Cacti about a “meaningful interface”?

That's what “action rules” are for. In this specific case, Illustration 1: “Traffic 64 bit” Graph Rule shows three parameters, that will be checked prior to creating a new traffic graph. Those parameters differ between Data Queries, e.g. there is no ifAlias when using a Disk Space data Query. Let's see the first one:

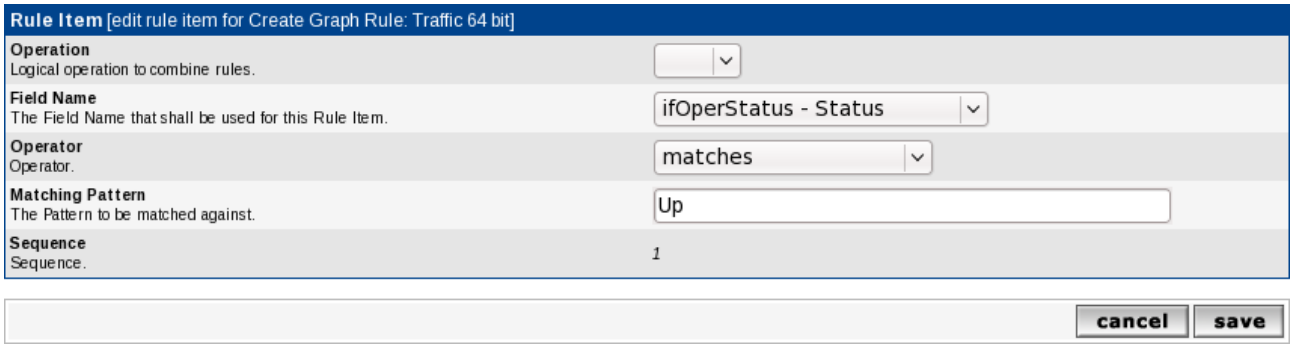

Illustration 11: Create Graph Rule Item

This very item will filter for all interfaces with ifOperStatus of “Up”. The next two rule items of this example make sure, that both ifIP and ifHwAddr are not empty (This perfectly makes sense for servers but makes no sense e.g. for switches).

Again, there's an option to see which graphs are related:

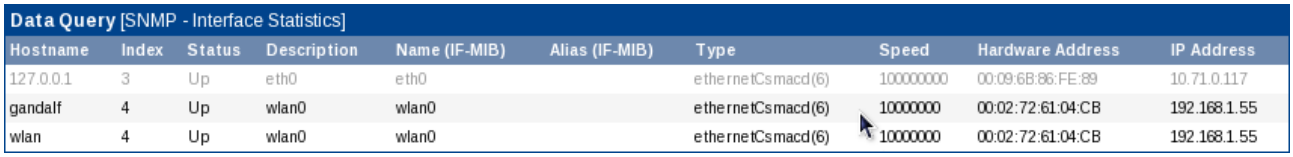

Illustration 12: Show Matching Graphs

This list looks very much like the one presented when “Create Graphs for this Host” is selected; only the first column for the associated host has been added. A grayed-out line marks a graph that already is present.

This allows you to precisely verify, that this rule matches exactly those graphs you want to create.

Create a New Graph Rule

Now that you understand how a Graph Rule works in general, let's create a new graph rule. Do so by clicking “Add” in the Graph Rule menu:

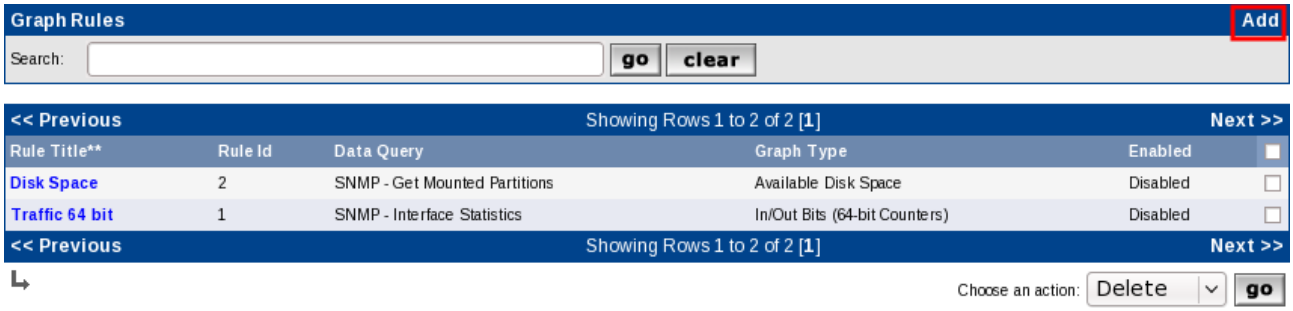

Illustration 13: Add a New Graph Rule

Now fill in the name of the new rule and select the related Data Query:

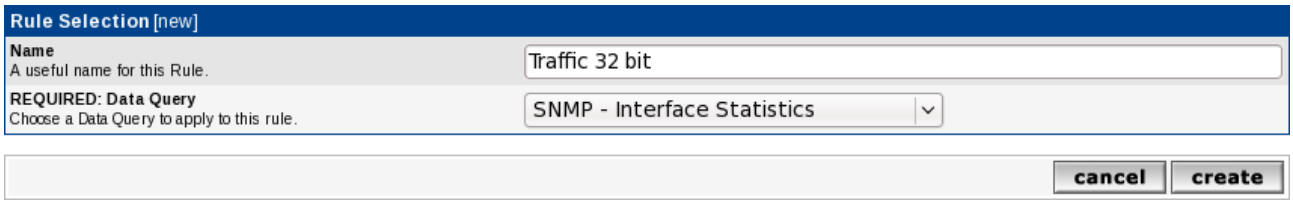

Illustration 14: Graph Rule – Select Data Query

Then “Create” and select the type of graph that shall be created, “save”:

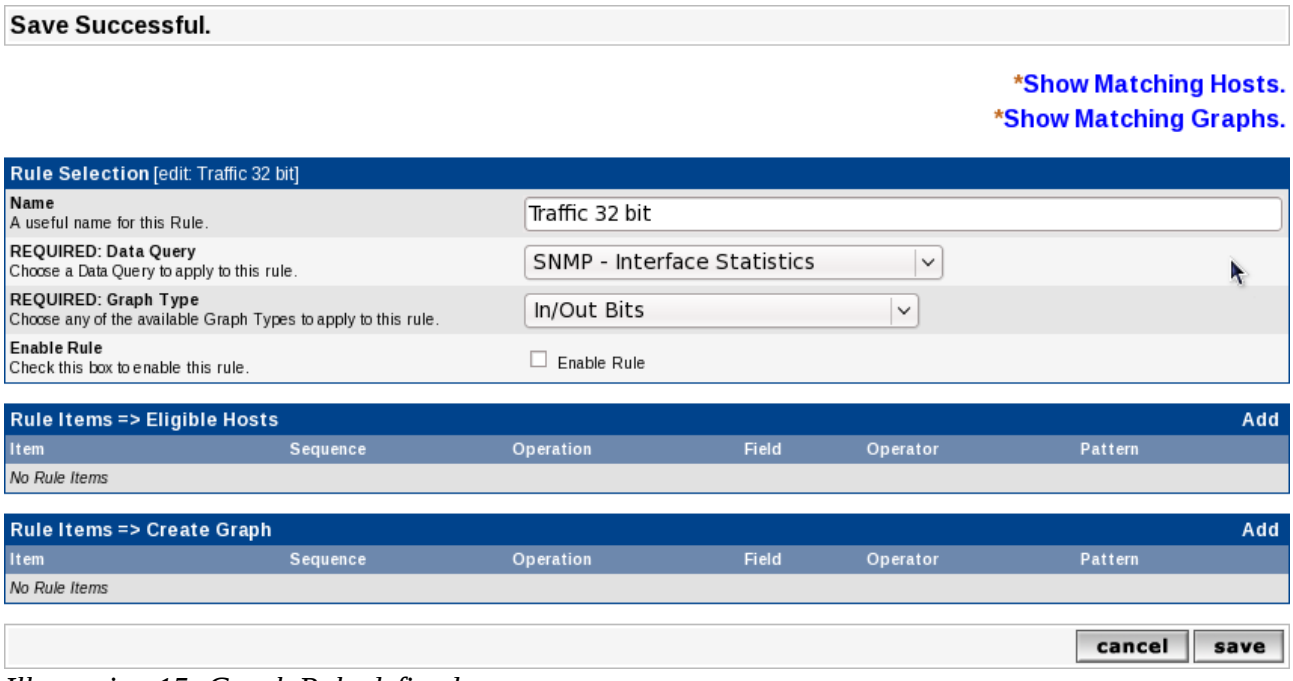

Illustration 15: Graph Rule defined

New Graph Rule: Define Matching Rule Items

Now the raw rule properties are defined. But currently it will never match, because no “Matching Rule” has been defined. So let's start to do so:

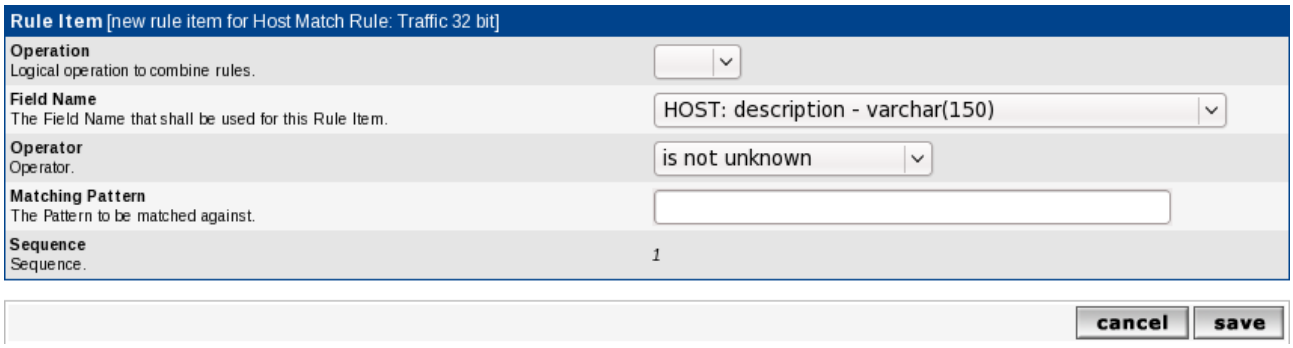

Illustration 16: Define a “Matching Rule Item”

This rule item virtually makes no sense. It's goal is simply to show the principles of operation. “Save” to see:

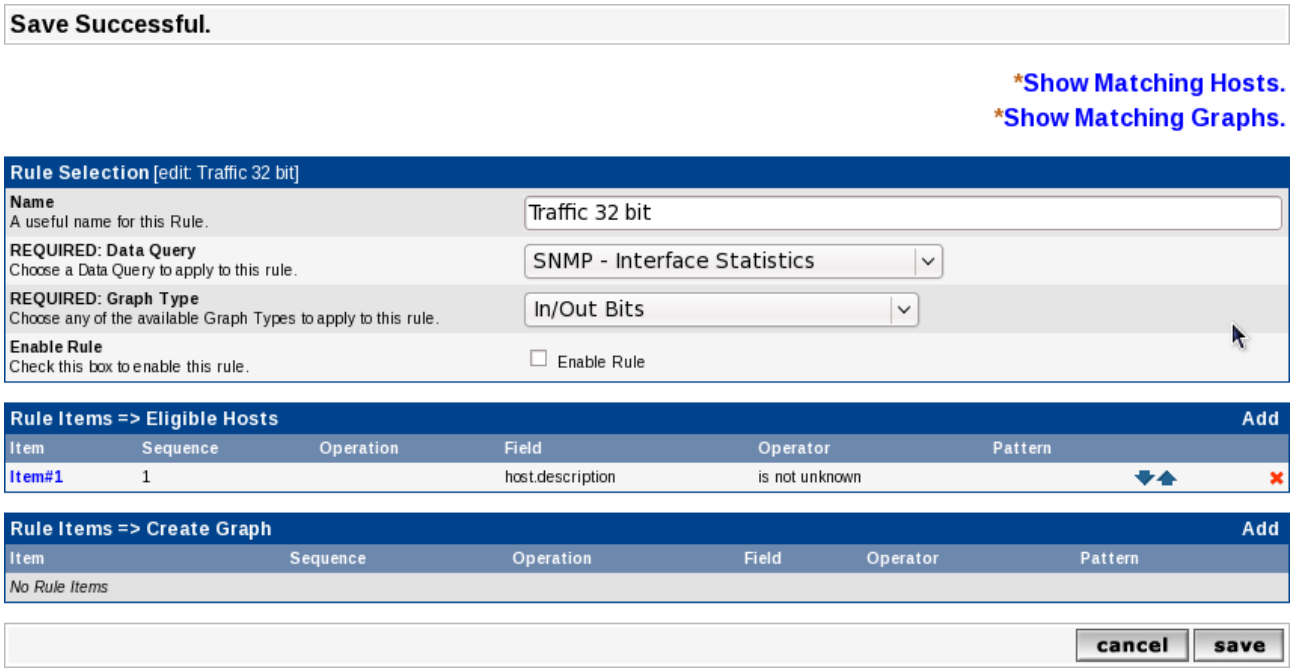

Illustration 17: Rule with a single matching rule item

Click “Show Matching Hosts” to find the list of hosts that match this rule:

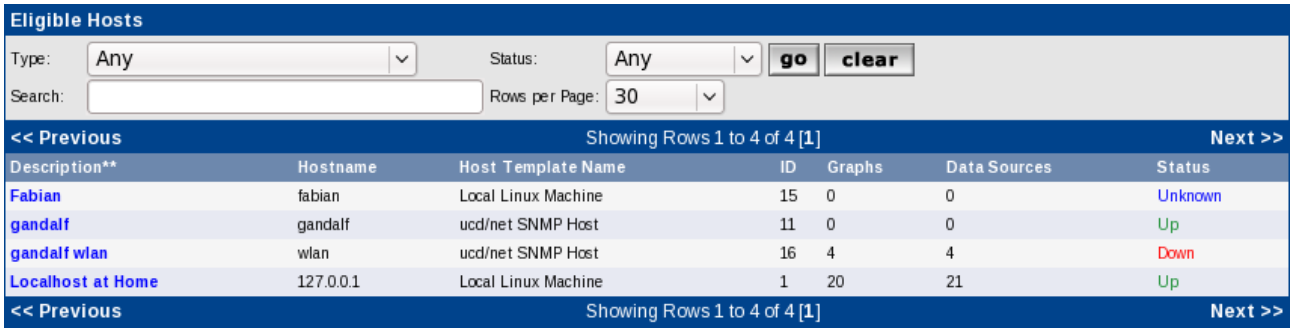

Illustration 18: Show all matching hosts

This are all hosts currently defined to my test box. Now we add a new rule to narrow down to all hosts running SNMP Version 1:

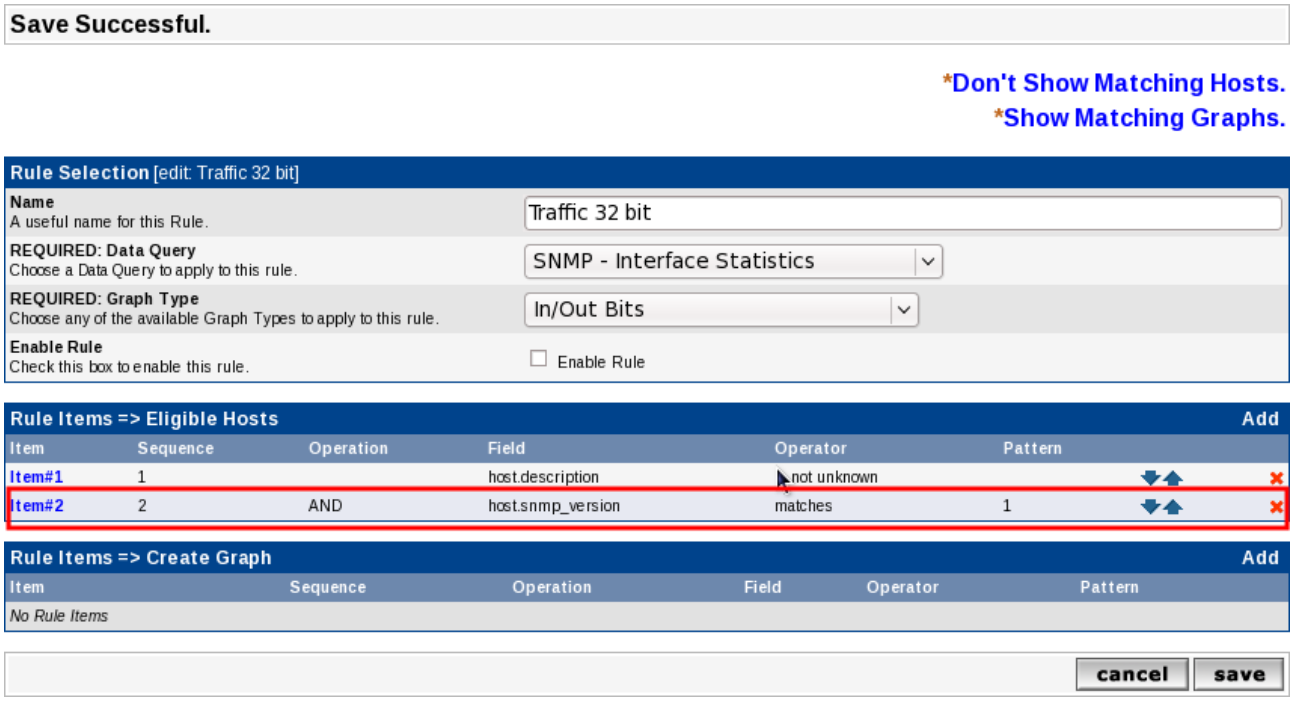

Illustration 19: Second matching rule item added

This now results in

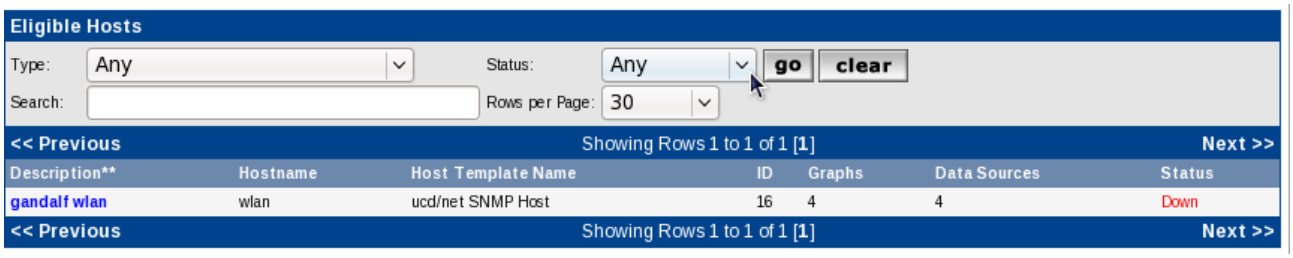

Illustration 20: Matching Host list narrowed down

This is how you want to develop host matching rules step-by-step. Still, this rule is **not** enabled.

New Graph Rule: Define Action Rule Items

Let's now defined the rule items that govern graph creation. It works quite the same way as already described for the “Traffic 64 bit” rule.

Testing a Graph Rule

Now that we have a valid Graph Rule in place, it's time to see it in action. For purpose of not breaking your installation, a two step process is recommended.

- 1st Step
  Run your new rule on dedicated and selected hosts
- 2nd Step
  Activate the rule for standard processing

At this time, you will want to know about two new “Settings -> Misc”, that are introduced by AUTOM8. These settings are “off” by default and this makes sense at this point in time:

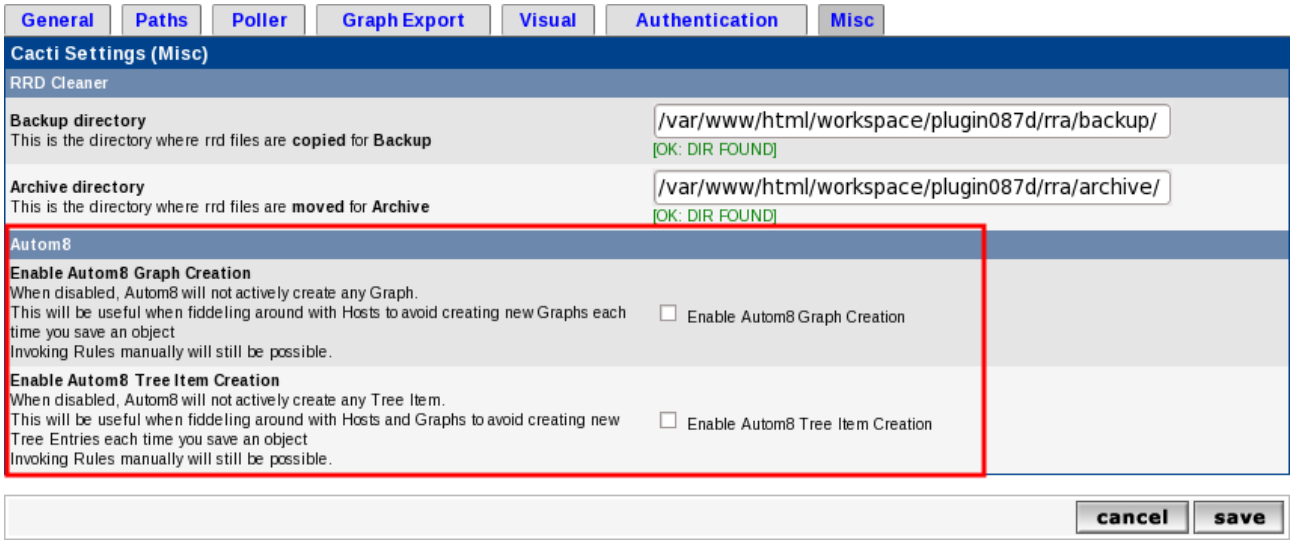

Illustration 21: Misc Settings for Autom8

Make sure that these settings are “off” unless you have finished testing for your new rule. Now turn to Devices and select any single host, or a list of hosts. Now choose the new action introduced by autom8, called “Apply Autom8 Rules”:

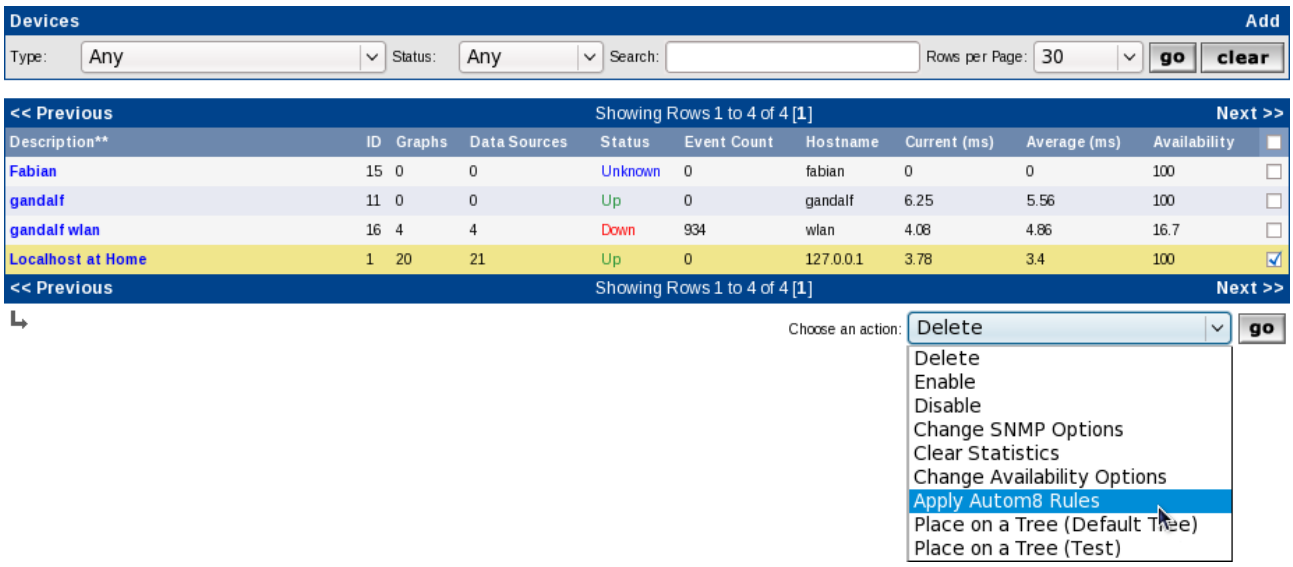

Illustration 22: Apply Autom8 Rules to specific hosts

You will be prompted by a confirmation page like:

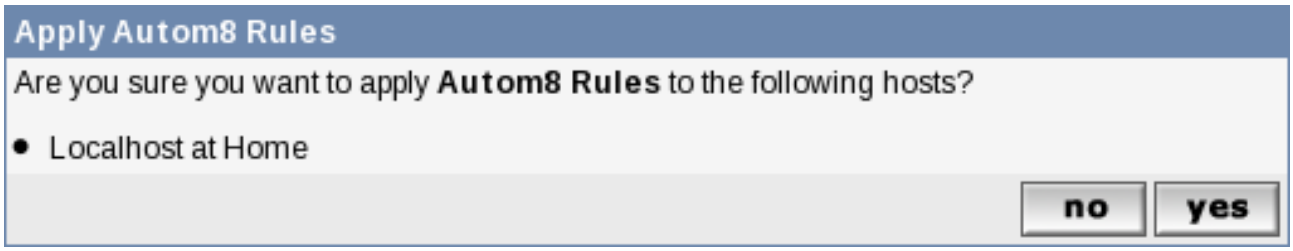

Illustration 23: Confirmation for applying Graph Rules

At time of writing, you will find lots of information in the cacti.log file. Grep for “AUTOM8” to find them.

Tree Rules

Most of the concepts explained for Graph Rules even hold for Tree Rules as well. For sake of better understanding, let's walk through both examples provided by the default installation.

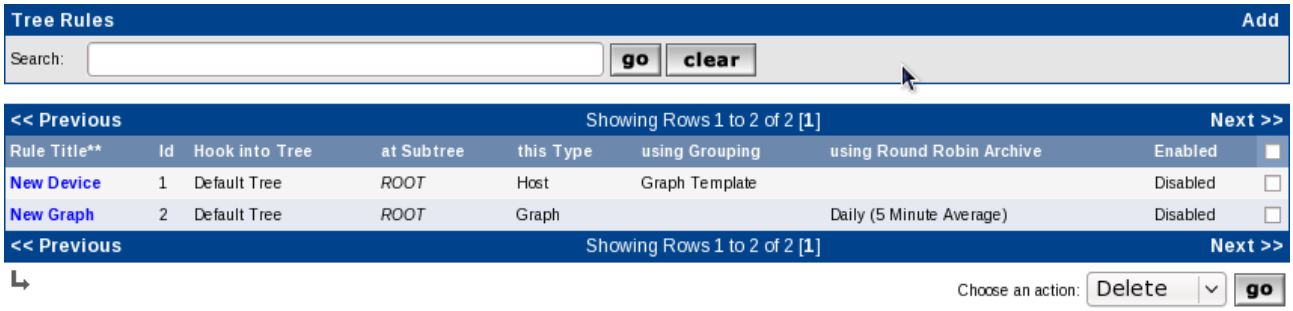

Illustration 24: Default Tree Rules provided with Autom8

The first basic issue to understand is which types of objects may be added to a tree. Currently, Cacti only knows two different types of leaf objects of a tree

- a host
- a graph

The third object, the “header”, does not make much sense as a leaf object. But you will use it to create sub-trees to group leaf items or other sub-trees. So there is one example for automatic creation of a “host leaf tree item” along with some headers. And there is another example that creates “graph leaf tree items” along with different headers.

Tree Rule: Host Leaf Item

The default Tree Rule is named “New Device”. It looks like this:

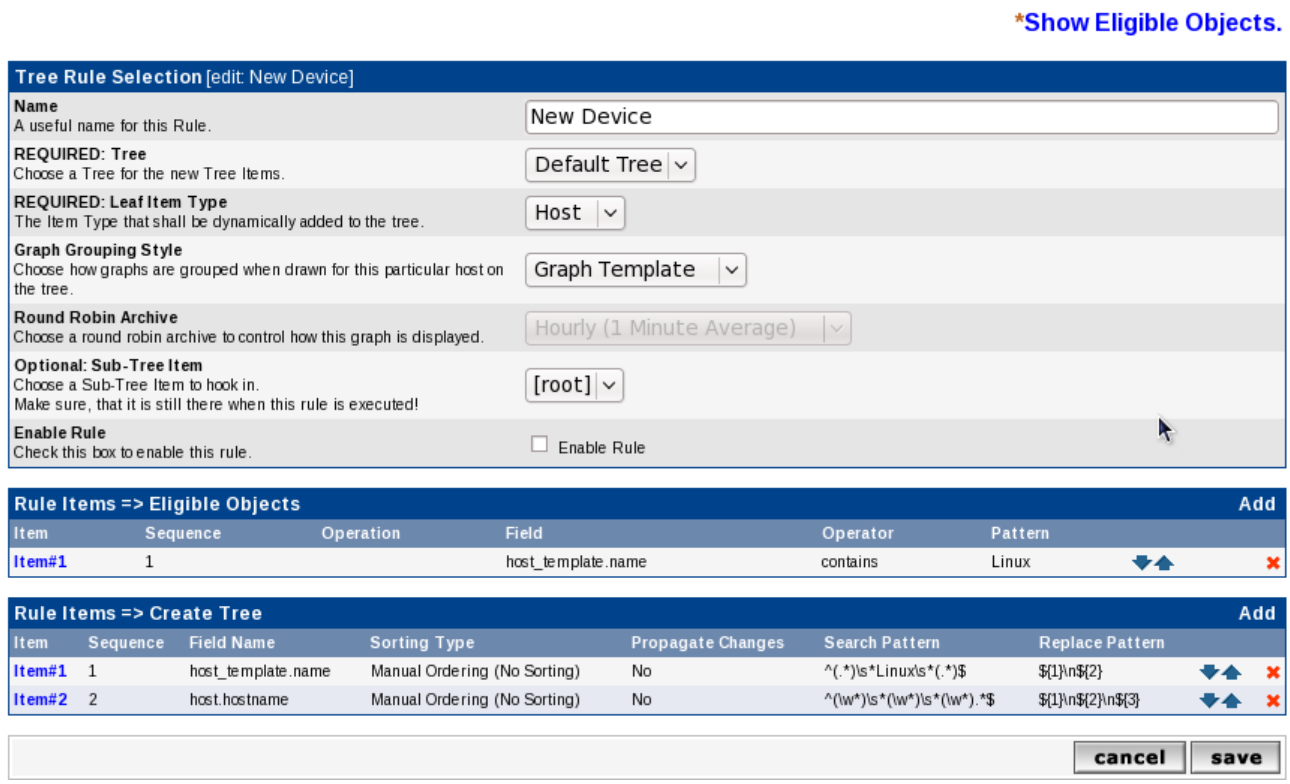

Illustration 25: Tree Rule for a Host Tree Item

Property	Description
Name	To be defined freely
Tree	The tree that shall take the new Host Tree Item. This tree has to be pre-defined. Autom8 does not create new trees, it only creates tree items
Leaf Item Type	In this example, we want to create a “Host” tree item.
Graph Grouping Style	For a host tree item type, different graph grouping options are offered. These are the same as Cacti offers itself when editing the tree manually.
Round Robin Archive	Meaningless for this type of tree item
Sub-Tree Item	You may not want to hook the new tree items directly underneath the tree's root. When defining a sub-tree here, make sure that it still exists at the time this rule is executed!
Enable Rule	Again, you may want to disable this rule until you've entered all rule items.

Tree Rules (Host): Matching Rule Items

Like the Graph Rules, you will have to define a “Matching Rule” first. This restricts the use of this Rule to specific hosts. Here's how it works:

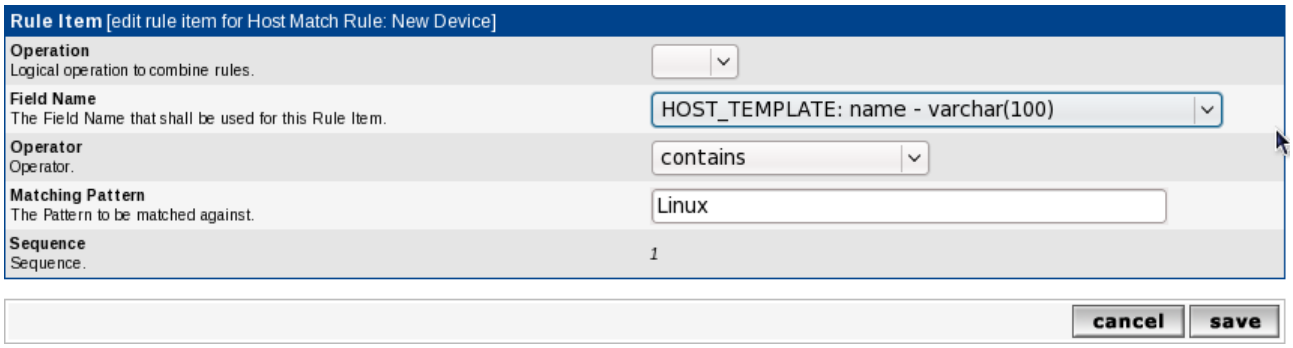

Illustration 26: Matching Rule for “New Device”

This rule reads in clear text:

1. Get the table HOST_TEMPLATE
2. Use the field “name”, which is a “varchar(100)”
3. check, whether a string “Linux” is part of the host template name

Again, this equals exactly the way that a Graph Rule Match is done. Like Graph Rules, you may list all matching hosts from the Tree Rule edit page, when choosing “Show Eligible Objects”:

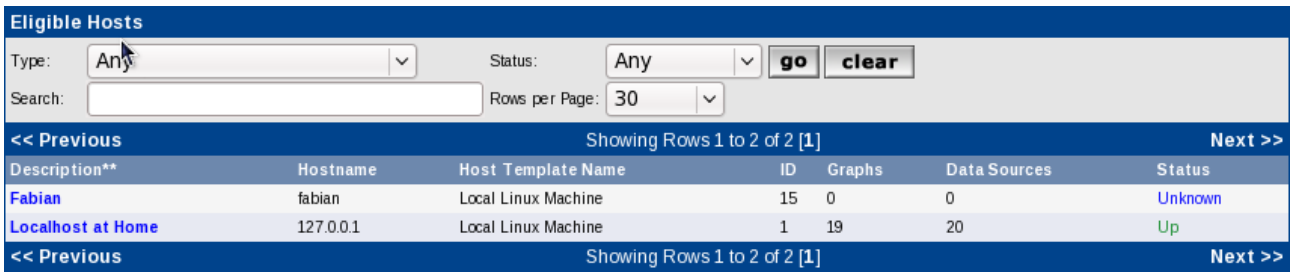

Illustration 27: Eligible Objects for Tree Rule

For sake of convenience, the target hosts are directly linked in the first column of this list. If the list is huge, you may use the opportunity to filter it by type, status or a versatile “Search” filter that matches host description, host name or host template name.

Once you've verified, that this matching rule ends up with exactly those hosts you want to act upon, you will turn to the action rules.

Tree Rules (Host): Action Rule Items

This part of Autom8 is really powerful and perhaps hard to understand on the first run. See it in action here:

*Show Created Trees.

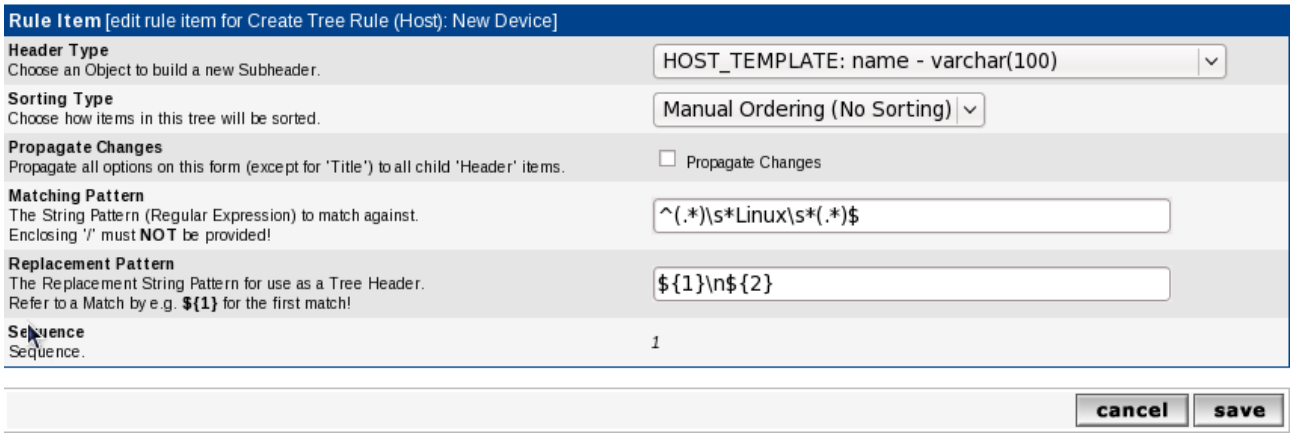

Illustration 28: Tree Rule Action

Property	Description
Header Type	You may choose any object of the drop-down list to act upon. You will be able to apply regexp magic to derive strings from any of those objects. Those strings will be used to form a new header entry. There's a special entry “Fixed String”. When used, it allows you to enter a specific string as a header.
Sorting Type	All standard Cacti sorting types are supported
Propagate Changes	Propagate sorting type to lower header levels
Matching Pattern	Define a regexp pattern for matching the given value of the variable defined by “Header Type”
Replacement Pattern	preg_replace pattern You may enter fixed strings here. Reference to variables defined by the matching pattern are done via e.g. “${1}” to denote the first variable. It is possible to split the result into multiple headers by using “\n” as a delimiter.
Sequence	Sequence number of this rule item
Enable Rule	Again, you may want to disable this rule until you've entered all rule items.

Let's explain the above example:

1. Use the table HOST_TEMPLATE, field “name” as a basis for this rule item.
2. The header derived from this rule item shall have “manual sorting” as it's sorting type
3. Sorting will not be propagated to lower levels
4. Try to find the string “Linux” in the host templates name. Put everything before that string (except for white-space) into a first variable and everything behind that string into a second variable (again skipping white-space)
5. Print the first variable as a header; then create a second header that holds the second variable

If applied to a host template name of “Local Linux Machine”, this will yield

- Local
  - Machine

as the resulting header structure. While this virtually does not make any sense, it shows how to use regexp for header definitions.

Let me explain a specific use case for this. Assume, you have a lot of different switches. You may decide to create a host template for each different manufacturer and/or model of switch. E.g.

- Switch – Cisco ...
- Switch – Enterasys N7
- Switch – Enterasys C3
- Switch – Extreme ...
- ...

are your host template names. With a matching pattern of “^(.*)\s*-\s*(.*)\s*(.*)$ and a replacement pattern of “${1}\n${2}\n${3}”, this will yield following tree structure

- Switch
  - Cisco
    - ...
    - ...
  - Enterasys
    - N7
    - C3
  - Extreme
    - ...

which may make perfect sense for a structured host tree. The second rule item works quite the same way, but it uses the HOST table, field “hostname” as a basis for generating sub-headers. The sub-headers of the second rule are placed underneath those of the first rule and so on.

But how to verify the results? You may have noticed the “Show Created Trees” option at top of each rule item. When clicking it, you will receive the following result for the first rule item:

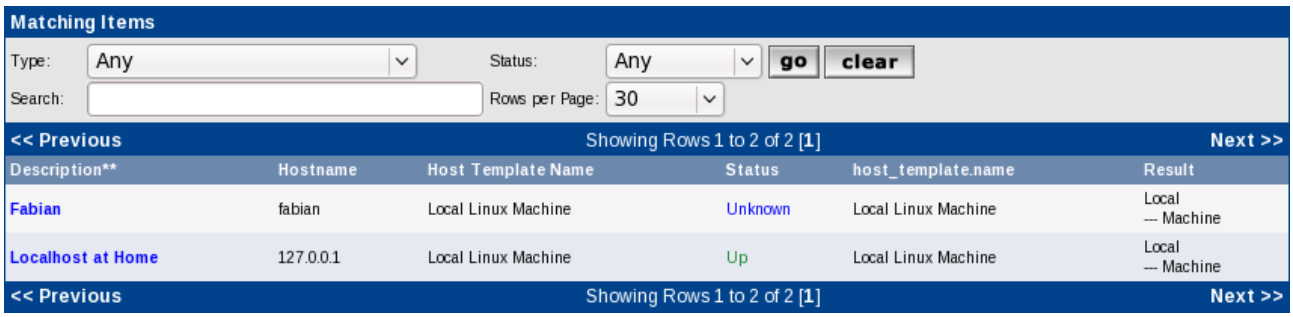

Illustration 29: Show Created Trees

The second to last column shows the variable that is used, the value printed as a table field. The last column shows which tree elements are created from that variable.

This way you again have the opportunity to verify the results of your rules before any damage may occur to your tree structure.

Tree Rule: Graph Leaf Item

There's a second default Tree Rule that handles the creation of graph tree leaves. Here is how it works:

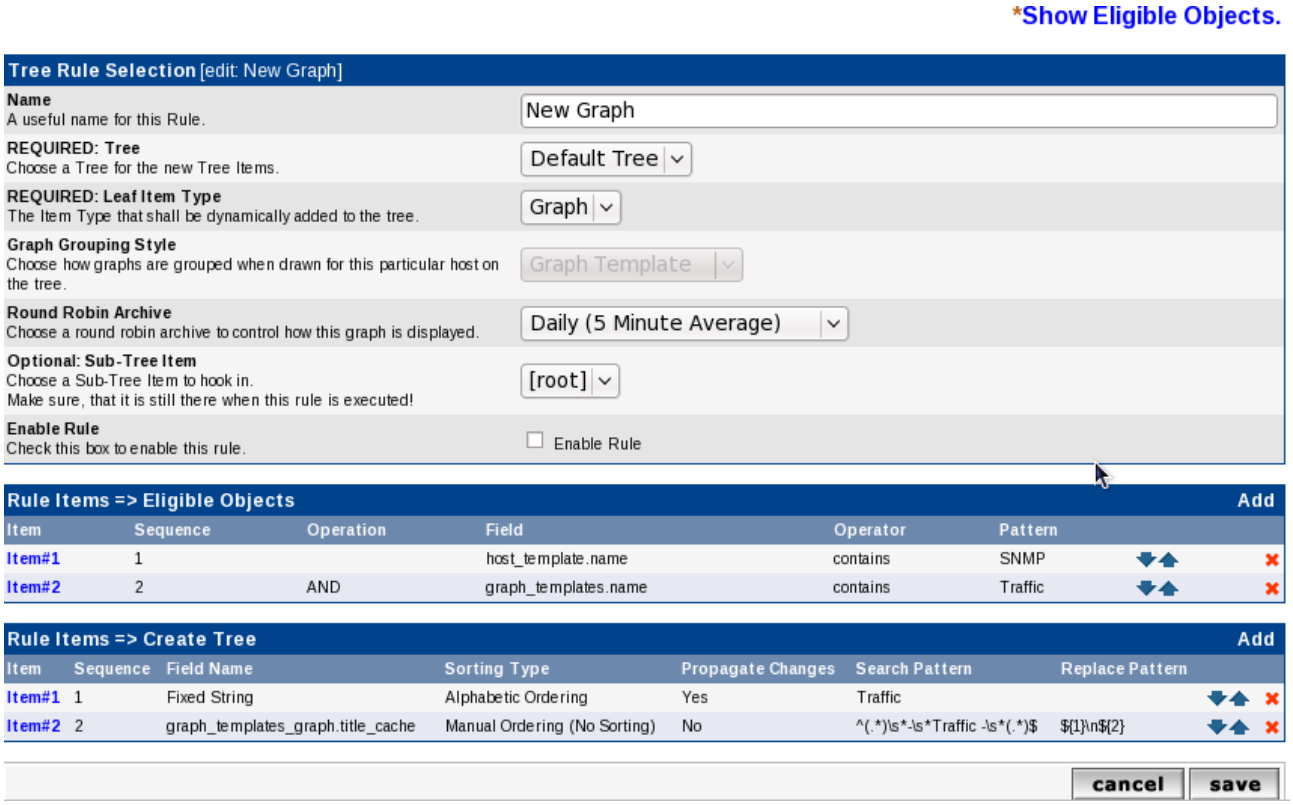

Illustration 30: Tree Rule for creating Graph Items

The main difference now is, that the “leaf item type” is a “graph”. As a result, the “graph grouping style” is grayed out as it has no meaning. Instead, you will now be able to define a default “round robin archive” to be used. The rest is like above.

Tree Rules (Graph): Matching Rule Items

Again, you will have to define a “Matching Rule” first. Using graphs as the object to add upon, there are some new tables that may be of use when defining a “match”:

- host_template: as above
- host: as above
- graph_template is new
- graph_template_graph is also available

This way, you may restrict the “match” to certain graph templates or graphs. For testing, you may want again to “Show Eligible Objects”. In this case, the resulting table will not show hosts but graphs:

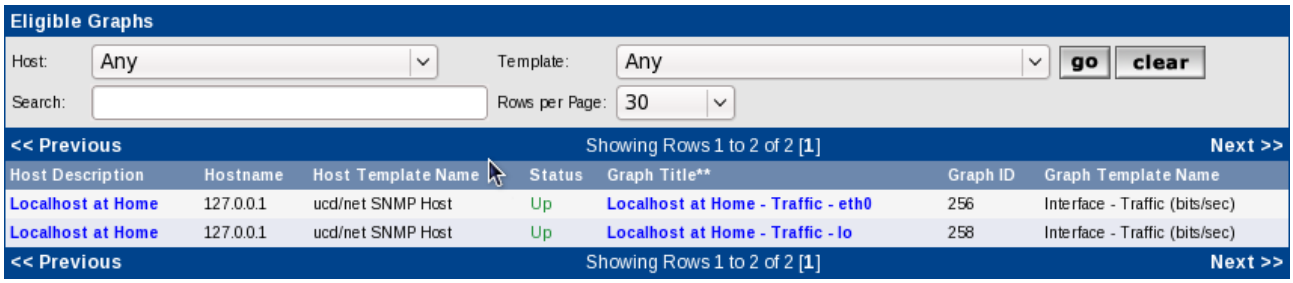

Illustration 31: Matching Objects for a Graph Type Tree Rule

Tree Rules (Graph): Action Rule Items

Here's an example for use of “Fixed String”:

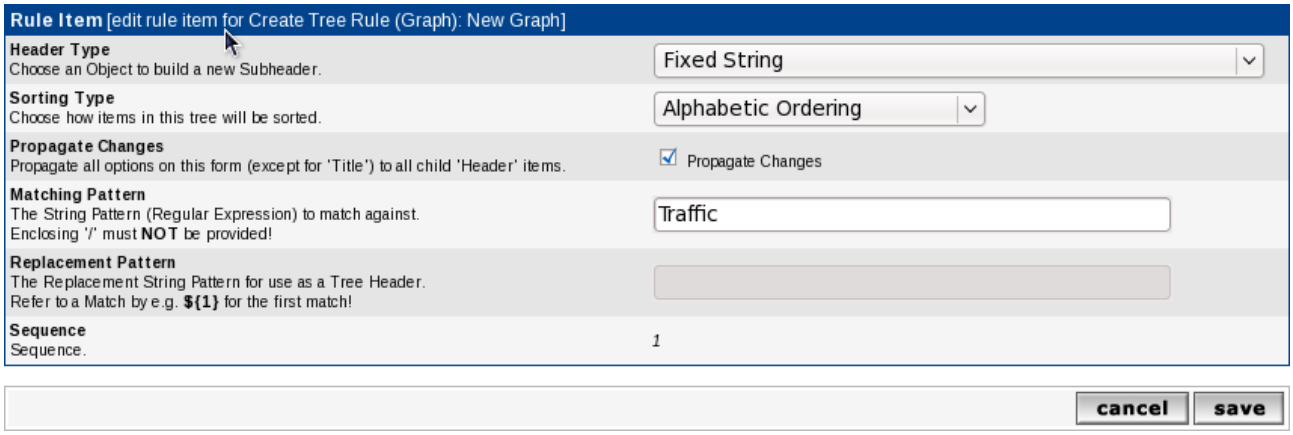

Illustration 32: Fixed String as a header type

As it makes less sense to “test” this rule item, the test link entry is suppressed. Let's turn to the second item:

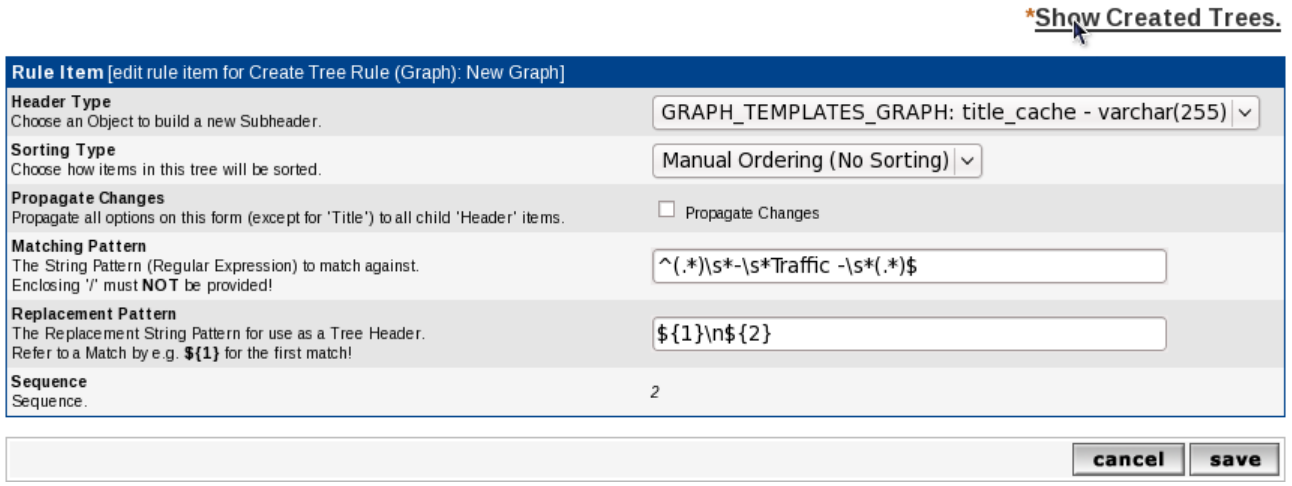

Illustration 33: Derive sub-tree header items from graph title

In this case, the graph title is searched for a match against the string “Traffic”. Again, the resulting chunks are used to form two new header entries. The result is perfectly shown when using “Show Created Trees”:

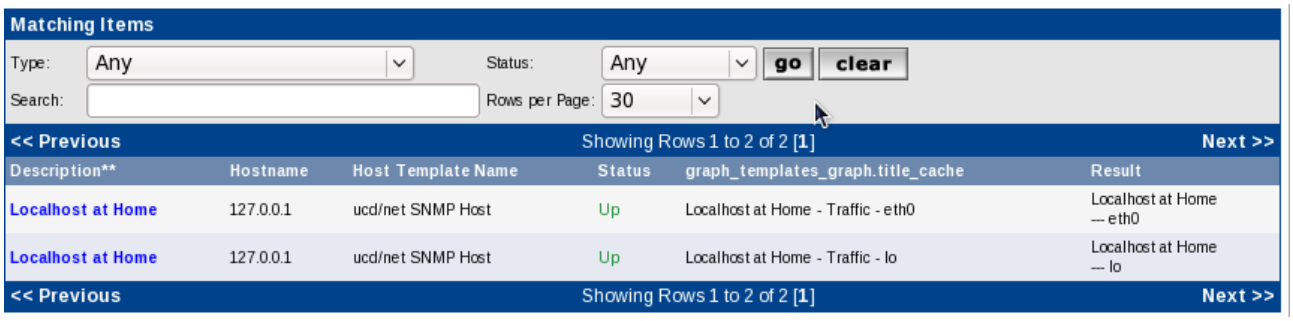

Illustration 34: Derive sub-tree header items from graph title – the results

Testing a Tree Rule

The principles of testing are just the same as above. In this case, you will start at the “Graph Management” page and again use the drop-down action “Apply Autom8 Rules”.

Best Practices

It is perfectly possible to create e.g. a set of Graph Rules that apply to one and the same host. In fact, this will be the most usual case to cope with all different data queries. So, as a rule of thumb, it is very likely that you end up with a Graph Rule for at least each different Data Query.

In most cases, you may face the need to indeed define multiple rules for the same Data Query but for e.g. different host templates. E.g. when using SNMP V2c 64bit traffic graphs, you may want to create those graphs for servers only in case the ifIP and ifHwAddr are set. But for switches, you may want to create different graphs.

This will end up not only in different “Matching Rules” related to different host templates but to different “Action Rules” as well. There's a dropdown action that allows you to duplicate a rule and modify it later. For sake of security, the “enabled” bit is reset when a rule is duplicated.

It is worth paying attention not to apply those different rules to one and the same host (which is perfectly possible from the view of this plugin).

The functions called by this plugin make sure, that if a graph is already in place, no duplicate will be created.

The same discussion holds for Tree Rules. But in fact, the most likely way to handle this will end up in e.g. creating rules to place hosts into trees. You may even drop each or only selected hosts into multiple trees by defining multiple Tree Rules matching the same set of hosts.

And additionally, you may want to drop the graphs, e.g. grouped by graph template, underneath a different tree. This way it is easily possible, to functionally join all graphs of same type (same graph template” underneath a common sub-header.

And you may define virtually any sub-header structure to further group those graphs in a meaningful way.

Again, duplicates are suppressed.

Beneath this automatism, you may perfectly keep your own, hand-grown tree structure without any impact.