

# **MRTG Implementation Manual**

Rev. 11260301

By [Florin Prunoiu](#)

Enterastream Communications Inc.

## **Table of Contents**

---

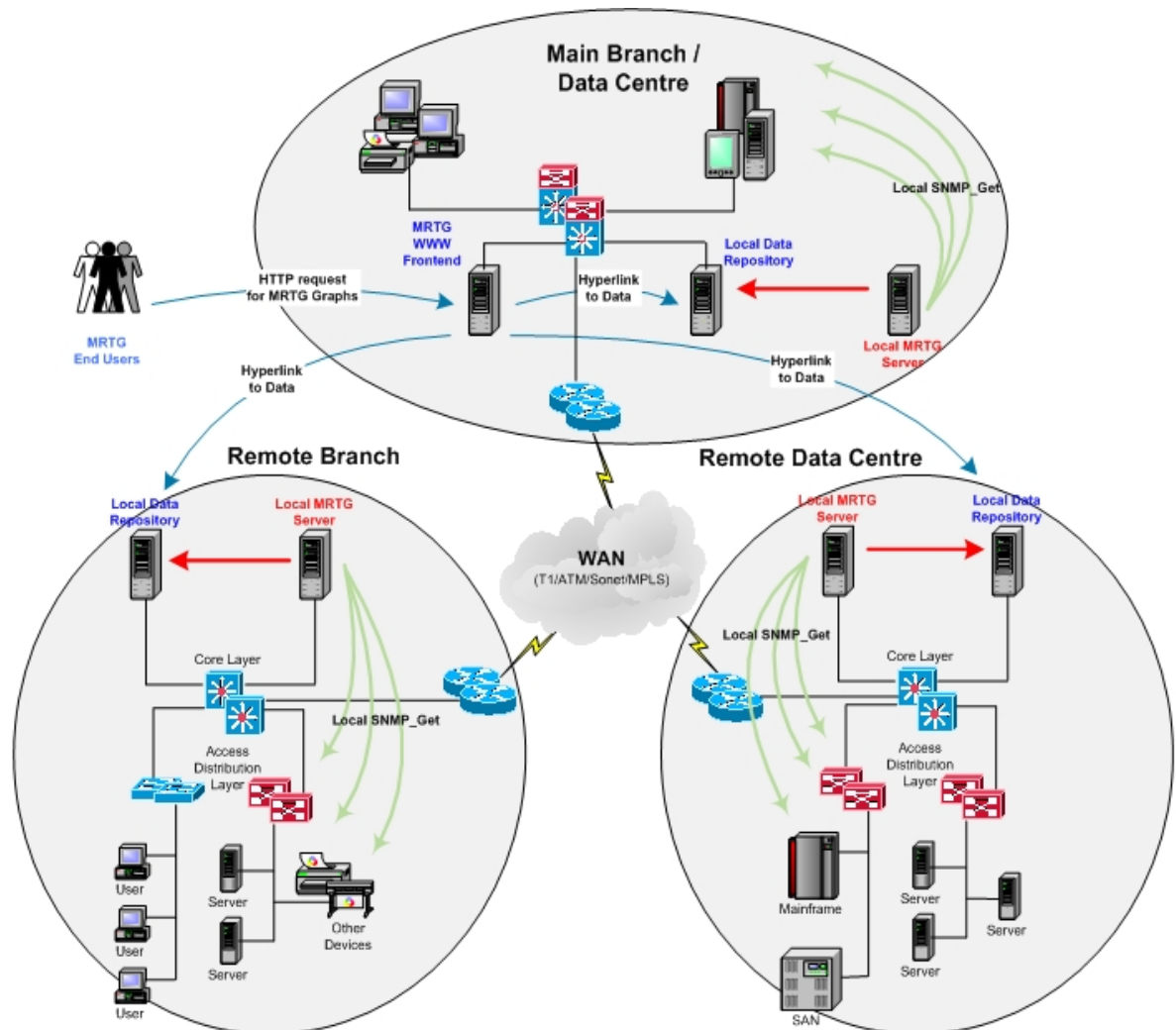
1. [MRTG architecture and related components](#)
2. [Installation process](#)
  - 2.1 [Installing Perl](#)
  - 2.2 [Installing MRTG](#)
  - 2.3 [Running the HTTP server](#)
  - 2.4 [Enabling SNMP support on the monitored hosts](#)  
[Extended SNMP support for Windows based hosts \(cpu, memory, disk, etc\)](#)
3. [Configuring MRTG tasks](#)
  - 3.1 [Building the “.cfg” configuration files](#)
  - 3.2 [Building the “index.html” files](#)
  - 3.3 [Configuration summary](#)
4. [Running MRTG tasks](#)
  - 4.1 [Running command-line MRTG instances](#)
  - 4.2 [Running MRTG instances as Windows NT/2000 Services](#)
5. [Database support for MRTG](#)
6. [Installing and configuring the RRD database support](#)
  - 6.1 [Installing the RRD database support](#)
  - 6.2 [Configuring MRTG instances with RRD database support](#)
  - 6.3 [Running MRTG instances with RRD database support](#)
  - 6.4 [RRD Database structure](#)
  - 6.5 [Resizing the database and extending it over a specific period of time](#)
7. [Exporting the database records into plain text / XML format](#)
8. [Generating on-demand MRTG graphs based on the data stored in the RRD database.](#)
9. [Monitoring generic SNMP OIDs with MRTG](#)
  - 9.1 [MRTG configuration file review](#)
  - 9.2 [Configuration file options](#)
  - 9.3 [Configuration examples for custom SNMP OIDs \(cpu, memory, disk space\)](#)

## 1. MRTG architecture and related components

MRTG is a network management application that can monitor any remote network host which has the SNMP protocol support enabled. MRTG, as a SNMP based application, runs SNMP requests against the target hosts on a regularly basis.

Originally MRTG was designed to acquire bandwidth information related to the network interfaces on a network host. Currently MRTG can interrogate a host about any supported SNMP OID and construct the variation graph. More than that, the new versions of MRTG are able to extend beyond of SNMP capabilities and collect numerical information from any host that collects and stores this kind of information (for more info on the new capabilities visit the MRTG and RRDTOOL websites). This presentation will focus on the SNMP capabilities of MRTG.

A network-distributed deployment of MRTG can be represented as follows:



MRTG acquires the SNMP information performing the following tasks:

- Interrogates the remote host and gets the value of the specific SNMP OID.
- Updates the variation graph with the new values and deletes the old graph. The graphs are images in PNG format. The new variation graph is stored in a location, which can be local or remote on a dedicated MRTG storage server.
- Stores the new value in the log file. The log file can be located on the local host or remotely on a MRTG storage server.

The classic version of MRTG builds the graphs immediately after a new SNMP value is acquired and does not store any historical data for future reference.

The newer version comes with a very flexible database support where historical data can be stored for as long as you configure the database size. MRTG does not generate the variation graphs when a new SNMP value is acquired, just stores it in database, making the whole process faster. The graphical variations are generated "on-demand" using dedicated scripts.

MTRG requires the following components:

1. The Perl interpreter needs to be installed on the MRTG machine. As MRTG is a Perl written application, it requires the Perl interpreter. A free copy of Perl can be obtained from <http://www.activeperl.com/>
2. The MRTG package. A copy can be downloaded from <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/pub/>
3. A Web Server is required on the local machine or on a remote machine (distributed installation) in order to make the MRTG graphs available over the web/http. Apache web server comes in free versions for Windows/Linux platforms. A free copy can be downloaded from: <http://www.apache.org/>

## 2. Installation process

The installation process will be presented just for a non-distributed implementation.

A distributed implementation follows the same instructions but needs modifications on path specification.

Note: This document is focused on the Windows version of MRTG. The Unix version does basically follow the same installation procedure. For UNIX instructions go to <http://people.ee.ethz.ch/~oetiker/webtools/>

### 2.1 Installing Perl

Perl has to be installed on the MRTG machine following the setup instructions. Make sure the Perl path (where you perl.exe resides) is part of your system's PATH variable.

### 2.2 Installing MRTG

MRTG comes as an archive of Perl scripts, configuration files, documentation and also the source files. It does not require specific installation tasks. Just decompress the MRTG files in the chosen MRTG folder.

Make sure your system's PATH variable contains the MRTG "bin" directory.

### 2.3 Running the HTTP server

As MRTG produces the variation .PNG graphs and .HTML files, they can be delivered over a web interface. MRTG also offers a script to construct the index.html files, which summarizes the rest of the html files for each monitored host.

The Web server setup does require the following steps:

- Create a "virtual site" for the MRTG web interface and specify the home directory where the MRTG's .html and .png files reside.
- For each monitored host, MRTG is able to construct the index.html file. It is recommended to place the files for each monitored host in a separate directory. At the root level of your MRTG folder hierarchy create a global index.html file, which makes references to the hosts' individual index.html files.

### 2.4 Enabling SNMP support on the monitored hosts Extended SNMP support for Windows based hosts (cpu, memory, disk, etc)

The majority of OS do not have the SNMP support enabled by default. The SNMP software has to be installed or enabled in order to get the SNMP OID data collection working. The SNMP support offers the SNMP client, which listens for SNMP requests coming from a NMS (network management station) and delivers the requested SNMP values.

A SNMP daemon/service requires the following generic information to be specified:

- "*community name*" // generic name which offers a basic security level. The default value is "public". A community is associated with some of the following rights: read/write/create/notify.
- "*contact*" and "*location*" are descriptive values to personalize the host.
- "*trap destinations*" // NMSs where the host is able to send SNMP trap notifications.
- "*accept SNMP packets*" // introduces a higher security level by accepting SNMP requests only from certain NMSs.

The SNMP support installation for the most popular platforms:

1. *Windows NT4* offers the SNMP service through ControlPanel -> Network -> Services where the Add/Remove Services option installs the SNMP support.
2. *Windows 2000* offers the SNMP service through ControlPanel -> AddRemoveSoftware -> WindowsComponents
3. *UNIX* based hosts need the SNMPD daemon to be configured (edit the config file) and started (edit the .RC file). Check the specific SNMPD setup in the product documentation.
4. *Cisco routers/switches* offer SNMP support running on a "public" community for CatOS based switches and no community name set for IOS based routers/switches by default.

Some sample IOS commands that change the SNMP configuration:

```
(config)#snmp-server community <name> <access-type>
(config)#snmp-server contact <text>
(config)#snmp-server enable traps [notification-type]
(config)#snmp-server host <host-addr>
```

To get the full snmp service options read the [Cisco IOS documentation](#) / [Cisco CatOS documentation](#) or get the CLI (command line) help on the "snmp-server / set snmp" commands.

Extended SNMP OID support for Windows based hosts (cpu, memory, disk, dhcp, dns, web, etc) : Microsoft Windows NT/2000 offer a limited set of SNMP OIDs by default. The most interesting ones are getting installed only by additional software services or packages. In order to install the SNMP OID support for CPU, Memory, Disks, etc you need to install the ResourceKit package which comes with additional features to the OS (requires additional license).

The SNMP OID support included in the ResourceKit can be installed separately without installing the whole ResourceKit.

A copy of the SNMP support files can be obtained from the [SNMP for the Public Community](#) website ([SNMP4PC](#)) where a standard (*no license required*) and full SNMP support (*ResourceKit license required*) packages are available.

The list of the OID counters that will become available in your Windows system is: [WindowsNT](#) and [Windows2000](#)

## 3. Configuring MRTG tasks

### 3.1 Building the “.cfg” configuration files

A .cfg file is required for each monitored host or a global .cfg file can be used for all monitored hosts, but the flexibility will decrease.

A .cfg file defines the SNMP OIDs for each entity that you intend to monitor from the destination host. MRTG parses the associated .cfg file and collects the SMNP values for all OIDs defined in the .cg file.

To build a .cfg file run the “CFGMAKER” script, which resides in the \mrtg\bin\ directory. This script scans a host for the network-interfaces only and constructs the .cfg file

The syntax is:

```
perl cfgmaker [options] [community@]router1 [[options] [community@]router2 ... ]
```

The CFGMAKER options are:

```
--ifref=nr      interface references by Interface Number (default)
--ifref=ip      ... by Ip Address
--ifref=eth     ... by Ethernet Number
--ifref=descr   ... by Interface Description
--ifref=name    ... by Interface Name
--ifref=type    ... by Interface Type
--ifdesc=nr     interface description uses Interface Number (default)
--ifdesc=ip     ... uses Ip Address
--ifdesc=eth    ... uses Ethernet Number
--ifdesc=descr  ... uses Interface Description
--ifdesc=name   ... uses Interface Name
--ifdesc=alias  ... uses Interface Alias
--ifdesc=type   ... uses Interface Type
--if-filter=f   Test every interface against filter f to decide wether or not to include that
                interface into the collection. Currently f is being evaluated as a Perl expression and it's
                truth value is used to reject or accept the interface. (Experimental, under development,
                might change)
--if-template=templatefile  Replace the normal target entries for the interfaces with an
                entry as specified by the contents in the file templatefile. The file is supposed to
                contain Perl code to be executed to generate the lines for the target in the configuration
                file.(Experimental, under development, might change)
--host-template=templatefile  In addition to creating targets for a host's interfacesdo
                also create targets for the host itself as specified by the contents in the file
                templatefile. The file is supposed to contain Perl code to be executed to generate the
                lines for the host related targets (such as CPU, ping response time measurements
                etc.) in the config-uration file. (Experimental, under development, might
                change)
--global "x: a" add global config entries
```

```

--no-down      do not look at admin or opr status of interfaces
--show-op-down show interfaces which are operatively down
--descint     describe interface instead of just 'Traffic Analysis for'
--subdirs=format give each router its own subdirectory, naming each per "format", in
which HOSTNAME and SNMPNAME will be replaced by the values of those items --
for instance,
--subdirs=HOSTNAME or --subdirs="HOSTNAME (SNMPNAME)"
--noreversedns do not reverse lookup ip numbers
--community=cmt Set the default community string to "cmt" instead of "public".
--snmp-options=[:<port>]:[:<tmout>]:[:<retr>]:[:<backoff>]:[:<ver>]]] Specify default
SNMP options to be appended to all routers following. Individual fields can be empty.
Routers following might override some or all of the options given to --snmp-options.
--dns-domain=domain Specifies a domain to append to the name of all routers
following.
--nointerfaces Don't do generate any configuration lines for interfaces, skip the step
of gathering interface information and don't run any interface template code.
--interfaces  Generate configuration lines for interfaces (this is the default). The main
purpose of this option is to negate an --nointerfaces appearing earlier on the command
line.
--help        brief help message
--man         full documentation
--version     print the version of cfmaker
--output=file output filename default is STDOUT

```

MRTG will interrogate the target host, will extract the SNMP OIDs for all network interfaces and construct the .cfg file for that specific host.

Once the .cfg file has been constructed, the following folder options must be added to the file:

```

Workdir // specifies the working directory
Syntax: Workdir: working-folder-path
HtmlDir // specifies where the html files will be copied
Syntax: HtmlDir: html-folder-path
ImageDir // specifies where the png will be copied
Syntax: ImageDir: image-folder-path
LogDir // specifies where the log files will be copied
Syntax: LogDir: html-folder-path
IconDir // specifies where the MRTG icons are
Syntax: IconDir: html-folder-path

```

Example:

```

Workdir: c:\inetpub\10.0.0.1\
HtmlDir: c:\inetpub\10.0.0.1\
ImageDir: c:\inetpub\10.0.0.1\
LogDir: c:\inetpub\10.0.0.1\
IconDir: d:\mrtg-2.9.21\images\

```

The PNG graphs are growing by default from right to left and are represented in Bytes/sec. To change this representation to Bits/sec and get the graph growing from left to right, just add the following option to the .cfg file:

```
Options[_]: growright, bits
```

CFGMAKER Example

Here is a CFGMAKER usage example and the resulted .cfg file:

```
D:\mrtg\bin\> perl cfgmaker --output d:\mrtg\branch.cfg --show-op-down --ifref=name --ifdesc=name --descint public@10.1.1.10
```

This command will interrogate the 10.1.1.10 host and parse its network interfaces.  
The result is the following .cfg file:

```
# Created by
perl cfgmaker --output d:\mrtg\branch.cfg --show-op-down --ifref=name --ifdesc=name -
-descint public@10.1.1.10

Workdir: d:\inetpub\wwwroot\
Htmldir: d:\inetpub\wwwroot\
Imagedir: d:\inetpub\wwwroot\
Logdir: d:\inetpub\wwwroot\
Icondir: /images/
Refresh: 300
RunAsDaemon: Yes
Interval: 5
# to get bits instead of bytes and graphs growing to the right
Options[_]: growright, bits

#####
#
# System: Router-10.1.1.10
# Description: Cisco Internetwork Operating System Software
#      IOS (tm) 3600 Software (C3640-JS-M), Version 12.1(3a)T1, RELEASE
SOFTWARE (fc1)
#      Copyright (c) 1986-2000 by cisco Systems, Inc.
#      Compiled Sat 29-Jul-00 11:48 by ccai
# Contact:
# Location:
#####
#

Interface 1 >> Descr: 'BRI0/0' | Name: " | Ip: " | Eth: " ###
The following interface is commented out because:
* --ifref=name is not unique for this interface
Target[10.1.254.8_1]: 1:public@10.1.254.8:
SetEnv[10.1.254.8_1]: MRTG_INT_IP="" MRTG_INT_DESCR="BRI0/0"
MaxBytes[10.1.254.8_1]: 2000
Title[10.1.254.8_1]: 1
PageTop[10.1.254.8_1]: <H1>1 </H1>
<TABLE>
  <TR><TD>System:</TD>  <TD> in </TD></TR>
  <TR><TD>Maintainer:</TD> <TD></TD></TR>
  <TR><TD>Description:</TD><TD>BRI0/0 ISDN BACK-UP TO F500_2202 and
F501_2206 </TD></TR>
  <TR><TD>ifType:</TD>  <TD>Link Access Protocol D (LAPD) (77)</TD></TR>
  <TR><TD>ifName:</TD>  <TD></TD></TR>
  <TR><TD>Max Speed:</TD> <TD>2000.0 Bytes/s</TD></TR>
</TABLE>

### Interface 2 >> Descr: 'FastEthernet0/0' | Name: 'Fa0/0' | Ip: " | Eth: " ###
```

```

Target[10.1.254.8_Fa0_0]: #Fa0/0:public@10.1.254.8:
SetEnv[10.1.254.8_Fa0_0]: MRTG_INT_IP="" MRTG_INT_DESCR="FastEthernet0/0"
MaxBytes[10.1.254.8_Fa0_0]: 12500000
Title[10.1.254.8_Fa0_0]: #Fa0/0
PageTop[10.1.254.8_Fa0_0]: <H1>#Fa0/0 -- F80CRS</H1>
<TABLE>
  <TR><TD>System:</TD>   <TD>F80CRS in </TD></TR>
  <TR><TD>Maintainer:</TD> <TD></TD></TR>
  <TR><TD>Description:</TD><TD>FastEthernet0/0 </TD></TR>
  <TR><TD>ifType:</TD>   <TD>ethernetCsmacd (6)</TD></TR>
  <TR><TD>ifName:</TD>   <TD>Fa0/0</TD></TR>
  <TR><TD>Max Speed:</TD> <TD>12.5 MBytes/s</TD></TR>
</TABLE>

```

For more information on CFGMAKER options go to  
<http://people.ee.ethz.ch/~oetiker/webtools/mrtg/cfgmaker.html>

### 3.2 Building the “index.html” files

The HTML index files are being built based on the previously created .cfg file. The role of the index.html file is to put together the html links to the individual network interfaces that are being monitored.

The index file is created by INDEXMAKER script.  
The syntax is:

```
perl indexmaker [options] [.cfg file]
```

The options are:

```

--output=filename  set output filename (default: stdout)
--filter title=~regexp  select targets by matching regexp against titles
--filter pagetop=~regexp  select targets by matching regexp against pagetop
--filter name=~regexp  select targets by matching regexp against name
--title=text        set title of generated index file
--bodyopt=text      set body tag options
--headlevel=number  use <Hnumber> at top of page (default: 1)
--pagetop=text      insert this text between <BODY> and <H1>...</H1>
--pageend=text      insert this text after the main body
--pagetopend=text   use this text for pagetop or pageend if undefined
--legend            add the Mrtg legend at the end of the page (default: add)
--columns=number    show graphs in a table with x columns (default: 2)
--compact          try to make a vertically more compact page
--optlog           log the used command line in the page (default: log)
--sort=title       sort graphs by title
--sort=name        sort graphs by their name
--sort=descr       sort graphs by their description
--sort=original    leave as is (default)
--enumerate        add a sequence number to the title of each graph

```

```
--picfirst      place pictures before text (default: text first)
--width=number  set width of graphs (default: not set)
--height=number
--sidebyside    place text / pictures side by side (default: above/below)
--bold          use bold text (default: bold)
--clicktext     make the text link to the inner page (like the image)
--show=day      pick which graph to show in the index (default)
--show=week
--show=month
--show=year
--show=none
--section=h1    h1 tag from pagetop as section heading default)
--section=title title as section headings for graphs
--section=name  graph name as section heading
--section=descr graph description as section heading
--section=portname port name entry in pagetop as section heading
--sectionhost   Try to prepend the host to the section heading if missing
--rrdviewer=path path to rrdviewer (default: /cgi-bin/14all.cgi)
--prefix=path   path from the location of the index.html to the graphs
--autoprefix    try to set prefix automatically
--<opt>-file=file read string argument for option <opt> from file
```

IndexMaker will create an html file, which has to be copied (recommended as index.html) into the folder where the html/image/log files are stored.

Example:

```
perl indexmaker --output d:\mrtg\10.1.1.10.index --compact d:\mrtg\10.1.1.10.cfg
```

### 3.3 Configuration summary

After proceeding with the previous steps, the current MRTG status should be:

- a. Perl, MRTG and the WebServer are installed.
- b. Configuration files (.cfg) for all monitored hosts have been generated using CFGMAKER. These files will be guiding the MRTG process in the SNMP acquisition.
- c. HTML index files for all hosts are generated and copied in the folders where the image/log/html files are. Mostly recommended is to name it "index.html" and use a folder for each monitored host to store its own related data.
- d. Next step: Get the MRTG process running and visualize the variation graphs.

## 4. Running MRTG tasks

### 4.1 Running command-line MRTG instances

The MRTG tasks run based on the .cfg file configuration.  
A separate MRTG instance should be started for each monitored host.  
The following options have to be added to the configuration file:

```
Workdir: d:\inetpub\wwwroot\  
Htmldir: d:\inetpub\wwwroot\  
Imagedir: d:\inetpub\wwwroot\  
Logdir: d:\inetpub\wwwroot\  
Icondir: /images/  
Refresh: 300 //Browser's refresh period in seconds  
RunAsDaemon: Yes //run in background  
Interval: 5 // run at 5min interval. The default value is 5min.  
# to get bits instead of bytes and graphs growing to the right  
Options[_]: growright, bits
```

To start a MRTG instance, run the mrtg script as follows:  
Note: the mrtg script is located in the mrtg\bin\ folder.

```
> perl mrtg [config file]
```

Example:

```
> perl d:\mrtg\bin\mrtg d:\mrtg\monitored-host.cfg
```

Note: the path to the mrtg script and mrtg config file should be specified if the command is executed in a different directory.

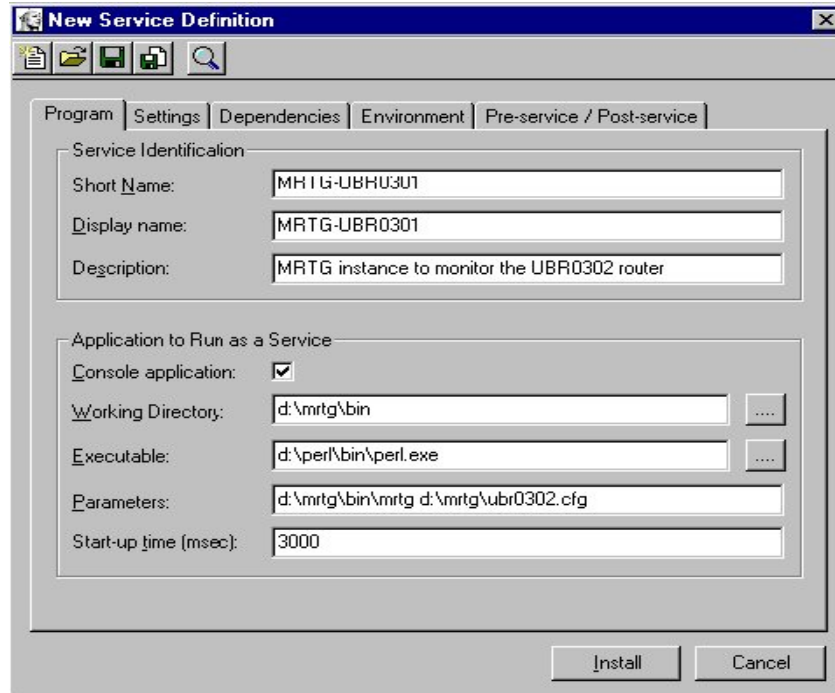
The MRTG instance will continue to run as long as the command window, where the process was started, still exists. This method is dependable on the current user session and command prompt window. The processes cannot be started in background while no user is logged-on.

### 4.2 Running MRTG instances as Windows NT/2000 Services

Running MRTG instances in background, even when no user is logged-on, is possible only by running the MRTG instances as a Service.

To register and run the MRTG instances as a service, a third party application or Microsoft's "RunAsService" service are required.

The MRTG service registration requires the execution parameters. The following service registration example is provided using FireDaemon third-party service management application (A free copy can be downloaded from <http://www.firedaemon.com/>)



In a complex environment, the MRTG setup can have multiple monitored hosts and each host can have its own settings (folders, time intervals). It is recommended to run a separate MRTG instances for each host, or group of hosts, with the same settings.

Also, a separate service should be registered for each instance to obtain a higher level of control on each individual MRTG instance.

## 5. Database support for MRTG

The classic version of MRTG does not store any historical SNMP data for future inquiries and graphs generation. It only collects the SNMP info and rebuilds the variation graphs on the fly. There are 4 types of graphs: daily, weekly, monthly and yearly for which MRTG stores the graph images only.

As the average calculation period increases (weekly, monthly, yearly) the SNMP values are calculated with an average value for that extended period. If you want to know what the network utilization was at a specific date/time, with a 5min average calculation, then it will not be available.

The database support that was given to MRTG by RRDTOOL comes with extended capabilities, which makes MRTG a more complex network analysis tool.

The backend database stores the SNMP values for as long as you configure it through the database size parameters. You will be able to generate detailed variation graphs for past times, with a very precise average calculation (lowest is 5min).

The database support is named RRDTOOL and it is a Round-Robin database. An individual database is kept for each individual monitored entity.

The database has a fixed size, which can be defined and modified by an administrator. The values are stored in the database in a sequential manner, as when the database gets full, the new values are written over the oldest values (FIFO algorithm – first in, first out).

The database can be configured to store a specific number of records, which permits to size it for a specific period of time based on the number of SNMP read actions.

The database support can be configured to run in parallel with the classic collect-and-graph MRTG functionality.

## 6. Installing and configuring RRD database support

### 6.1 Installing RRD database support

The installation steps are:

1. Download the RRDTOOL package and unzip it to the chosen RRD folder. A free copy of RRDTOOL can be downloaded from <http://www.rrdtool.com/download.html>
2. The package contains the binary files and also the source files for development reference. The src\ folder contains four subfolders where some RRD tools are available in .exe format and they are usable as they are. Copy these files in the rrdtool\bin\ folder for an easier access. Include the rrdtool\bin in the system path.
3. Register RRDTOOL package with the currently installed Perl distribution (at least Perl 5.6). Go to the “perl-shared” folder and run the following command:  
> ppm install rrdtool.ppd

RRDTOOL is now ready for use. Next step is to configure the MRTG instances to write SNMP data into the RRD databases.

### 6.2 Configuring MRTG instances with RRD database support

The MRTG configuration steps are:

1. Build the MRTG .cfg file for the monitored target using the CFGMAKER command as described in paragraph 3.1
2. Update the .cfg file with the following configuration

```
### Global Config Options
Workdir: [rrdtool repository folder]
PathAdd: [path to the rrdtool “bin” folder]
LibAdd: [path to the rrdtool “perl-shared” folder]
logformat: rrdtool
RunAsDaemon: Yes
Interval: 5 //SNMP read interval
```

### 6.3 Running MRTG instances with RRD database support

The MRTG instances with RRD database support can be ran in the same manner as the classic collect-and-graph instances.

Run from the command prompt the following command:

```
> perl mrtg [.cfg file]
```

Example:

```
> perl d:\mrtg\bin\mrtg d:\mrtg\ubr-0302-rrdformat.cfg
```

MRTG will read the SNMP data at the specified interval and will add the values to the database instead of just updating the image-graphs.

It is recommended to run these instances as NT/W2K Services (Windows version).

Follow the instructions from paragraph 4.2 to register the instance as a service and run it in background.

## 6.4 RRD Database structure

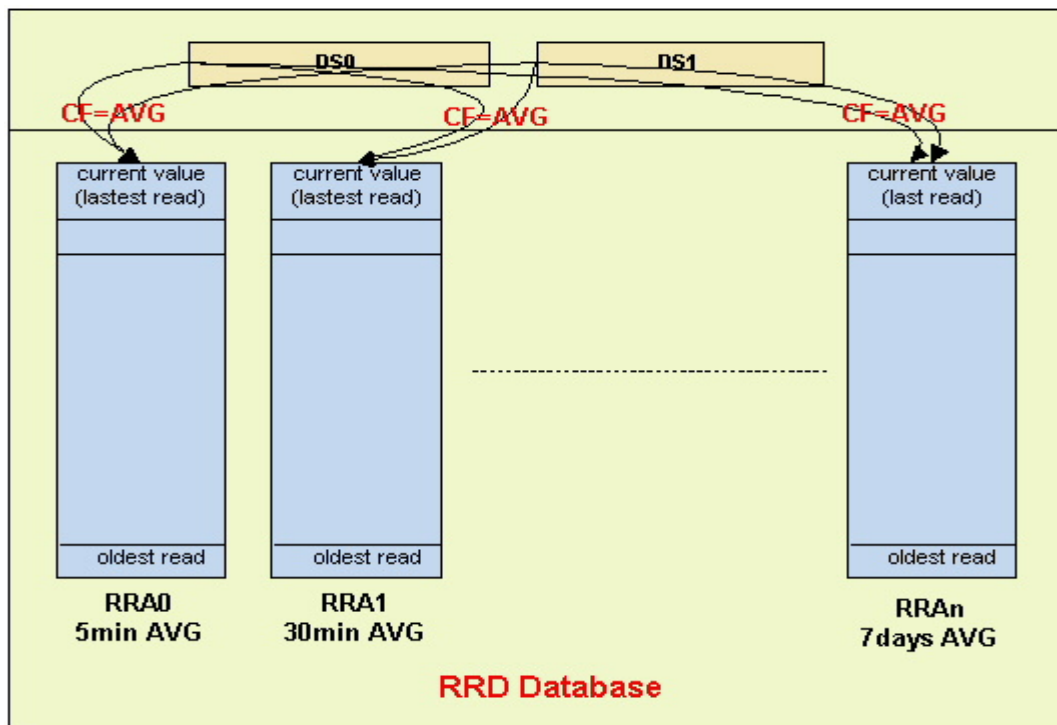
A RRD database consists of one or multiple Round Robin Archives (RRA), which stores the data for a specific frequency.

The values stored in these RRAs are calculated based on one or multiple Data Sources (DS). A DS describes the monitored entity where the SNMP information is collected.

The values stored in the RRAs are calculated using a consolidation function (CF). There are 3 CFs available: Average/Min/Max.

MRTG automatically creates two DSs when stores the bandwidth information for a network interface: DS0 is the input traffic and DS1 is the output traffic. For other type of representations the DSs can be defined as needed.

The following picture displays the RRD database generic structure:



The following function displays the structure of a database:

```
> rrdtool info [RRD database]
```

Database structure example, as shown by the above command:

```
> rrdtool info 10.0.0.51_to3_0.rrd
filename = "10.0.0.51_to3_0.rrd"
rrd_version = "0001"
step = 300
last_update = 1034865890
ds[ds0].type = "COUNTER"
ds[ds0].minimal_heartbeat = 600
ds[ds0].min = 0.0000000000e+000
ds[ds0].max = 2.0000000000e+006
ds[ds0].last_ds = "1160456235"
ds[ds0].value = 9.5316330000e+005
ds[ds0].unknown_sec = 0
ds[ds1].type = "COUNTER"
ds[ds1].minimal_heartbeat = 600
ds[ds1].min = 0.0000000000e+000
ds[ds1].max = 2.0000000000e+006
ds[ds1].last_ds = "493517813"
ds[ds1].value = 1.5046843333e+005
ds[ds1].unknown_sec = 0
rra[0].cf = "AVERAGE"
rra[0].rows = 122944
rra[0].pdp_per_row = 1
rra[0].xff = 5.0000000000e-001
rra[0].cdp_prep[0].value = NaN
rra[0].cdp_prep[0].unknown_datapoints = 0
rra[0].cdp_prep[1].value = NaN
rra[0].cdp_prep[1].unknown_datapoints = 0
rra[1].cf = "AVERAGE"
rra[1].rows = 800
rra[1].pdp_per_row = 6
rra[1].xff = 5.0000000000e-001
rra[1].cdp_prep[0].value = 7.2080766667e+003
rra[1].cdp_prep[0].unknown_datapoints = 0
rra[1].cdp_prep[1].value = 1.0866125556e+003
rra[1].cdp_prep[1].unknown_datapoints = 0
rra[2].cf = "AVERAGE"
rra[2].rows = 800
rra[2].pdp_per_row = 24
rra[2].xff = 5.0000000000e-001
rra[2].cdp_prep[0].value = 2.6916064814e+004
rra[2].cdp_prep[0].unknown_datapoints = 0
rra[2].cdp_prep[1].value = 4.1579679107e+003
rra[2].cdp_prep[1].unknown_datapoints = 0
```

Note:

When you configure MRTG to run with the RRD database support, the MRTG instance creates 8 RRAs for a database. The first RRA[0] keeps the values corresponding to the configured read frequency (ex: 5min) and the next ones RRA[1] – RRA[7] store values

that are automatically calculated for larger intervals (30min, 1hour, etc).

## 6.5 Resizing the database and extend it over a specific period of time

The MRTG instances first create the RRD databases with a standard size, which covers a short period of continuous data storage.

To extend this period to a couple of months or years you need to resize the database to a number of records which corresponds to the desired period and read frequency.

To resize a database use the following command:  
`rrdtool resize [RRD database] [RRAx] GROW [# of records to increase]`

Example:

The following command expands a database for a period of 1 year with a 5min read frequency. There are 12 readings per hour x 24 hours x 31 days x 12 months = 107136 entries for a year.

```
> rrdtool resize 10.0.0.1_fa0_0.rrd RRA[0] GROW 107136
```

## 7. Exporting the database records into plain tex / XML format

The database records can be accessed by the “rrdtool” executable, which is able to export them in an XML/ASCII file for further processing.

The export functions are:

### 1. DUMP function

```
rrdtool dump filename.rrd > filename.xml
```

This function exports the whole database in XML format. For a large database, the XML result will be a very large file.

A dump excerpt looks as follows:

```
<!-- 2002-10-05 03:10:00 Eastern Daylight Time / 1033801800 --> <row><v>
1.3837724444e+003 </v><v> 2.3811982222e+002 </v></row>
<!-- 2002-10-05 03:15:00 Eastern Daylight Time / 1033802100 --> <row><v>
1.4081976000e+003 </v><v> 2.6190866667e+002 </v></row>
<!-- 2002-10-05 03:20:00 Eastern Daylight Time / 1033802400 --> <row><v>
1.3597713301e+003 </v><v> 2.5264724650e+002 </v></row>
<!-- 2002-10-05 03:25:00 Eastern Daylight Time / 1033802700 --> <row><v>
1.4397478255e+003 </v><v> 2.4997519794e+002 </v></row>
<!-- 2002-10-05 03:30:00 Eastern Daylight Time / 1033803000 --> <row><v>
1.3969684222e+003 </v><v> 2.4970408889e+002 </v></row>
<!-- 2002-10-05 03:35:00 Eastern Daylight Time / 1033803300 --> <row><v>
1.4300162899e+003 </v><v> 2.4545897350e+002 </v></row>
<!-- 2002-10-05 03:40:00 Eastern Daylight Time / 1033803600 --> <row><v>
1.4308223434e+003 </v><v> 2.6211339317e+002 </v></row>
```

### 2. XPORT function

```
rrdtool xport [-s|--start seconds] [-e|--end seconds] [-m|--maxrows rows] --
step value [DEF:vname=rrd.ds-name: CF]
[CDEF:vname=rpn-expression]
[XPORT:vname[:legend]]
```

Where:

```
--start = start time in EpochTime
--end = end time in EpochTime
-m = max number of rows to be listed
--step = SNMP reading frequency in seconds
DEF:vname = a virtual name (user defined) for a data which corresponds to DS within
an RRD and a CF function. Multiple DEFs can be defined here.
VDEF:vname = a new virtual variable which can be actually the exported value. It can
be a calculation between multiple DEF:vname variables. The calculations are
expressed in RPN-expressions. RPN-expression example: a,b,+,8,/ is (a+b)/8
XPORT: vname = is the name of the virtual variable that will be exported. It could be a
DEF or a CDEF variable.
```

**Note:** This function performs a selective export for a specific period of time and with a specific frequency. Note that the start and end time values are expressed in Epoch Time values (the time in seconds since 1/1/1970). You need to converse the current time in EpochTime.

Sample:

```
rrdtool xport      --start 1034259600  --end 1034260300  --step 300
                  DEF:my_bandwidth=10.0.0.1_po1.rrd:ds0:AVERAGE
                  CDEF:my_bit_bandwidth= my_bandwidth,8,*
                  XPORT:my_new_bandwidth
```

An export looks as follows:

```
<xport>
  <meta>
    <start>1034259600</start>
    <step>300</step>
    <end>1034260500</end>
    <rows>4</rows>
    <columns>1</columns>
    <legend>
      <entry></entry>
    </legend>
  </meta>
  <data>
    <row><t>1034259600</t><v>1.8206442886e+005</v></row>
    <row><t>1034259900</t><v>3.0814951893e+005</v></row>
    <row><t>1034260200</t><v>2.0108738560e+005</v></row>
    <row><t>1034260500</t><v>3.6460060800e+004</v></row>
  </data>
</xport>
```

**Note:** For more details on all XPORT options go to <http://www.rrdtool.com/>

## 8. Generating on-demand MRTG graphs

Similar to the XPORT function is the GRAPH function, which builds graphs for specific past periods of time and is also able to graph multiple variables in the same graph.

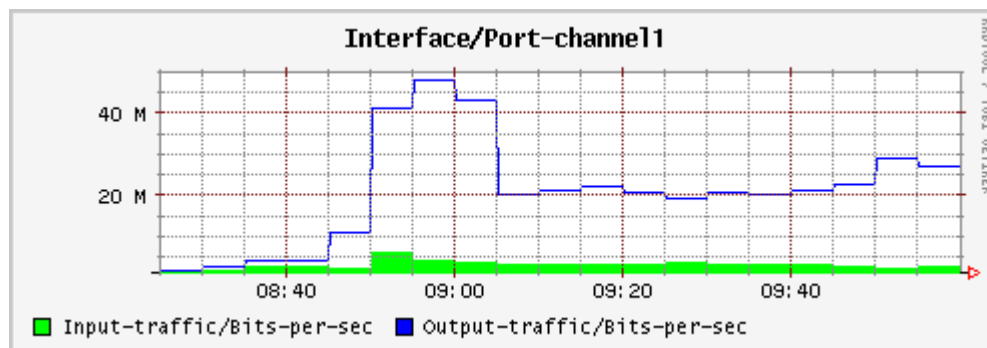
The syntax is:

```
rrdtool graph filename [-s|--start seconds] [-e|--end seconds] [-x|--x-grid x-axis grid and label]
                        [-y|--y-grid y-axis grid and label] [--alt-y-grid] [--alt-y-mrtg] [--alt-
autoscale]
                        [--alt-autoscale-max] [--units-exponent] value]> [-v|--vertical-label text]
                        [-w|--width pixels] [-h|--height pixels] [-i|--interlaced]
                        [-f|--imginfo formatstring] [-a|--imgformat GIF|PNG|GD]
                        [-B|--background value] [-O|--overlay value] [-U|--unit value] [-z|--
lazy]
                        [-o|--logarithmic] [-u|--upper-limit value] [-l|--lower-limit value]
                        [-g|--no-legend] [-r|--rigid] [--step value] [-b|--base value]
                        [-c|--color COLORTAG#rrggb] [-t|--title title]
                        [ DEF:vname=rrd:ds-name: CF] [ CDEF:vname=rpn-expression]
                        [ PRINT:vname: CF:format] [ GPRINT:vname: CF:format]
                        [ COMMENT:text] [ HRULE:value#rrggb[:legend]]
                        [ VRULE:time#rrggb[:legend]] [ LINE{1|2|3}:vname[#rrggb[:legend]]]
                        [ AREA:vname[#rrggb[:legend]]] [ STACK:vname[#rrggb[:legend]]]
```

An example:

```
rrdtool graph my-graph.gif -s 1035289500 -e 1035295200 -t Interface/Port-channel1
DEF:xx=10.0.0.1_po1.rrd:ds0:AVERAGE
DEF:yy=10.0.0.1_po1.rrd:ds1:AVERAGE
CDEF:myxx=xx,8,*
CDEF:myyy=yy,8,*
AREA:myxx#00ff00:Input-traffic/Bits-per-sec
LINE1:myyy#0000ff:Output-traffic/Bits-per-sec
```

The output of this command is the following graph:



The GRAPH function could be called from the command prompt or from a web interface to have the graphs available over http.

Note: For more details on all GRAPH options go to <http://www.rrdtool.com/>

## 9. Monitoring generic SNMP OIDs with MRTG

### 9.1 MRTG configuration file review

MRTG tasks are running based on the options present in the configuration file. A configuration file is required for each (or multiple) monitored targets. As mentioned earlier, the MRTG tasks are started as follows:

```
> perl mrtg [.cfg file]
```

As you can observe, the key element here is the configuration file, which is a collection of *MRTG option tags*.

The option tags can be generated automatically by running the CFGMAKER script against a target host. The script will build the configuration file for that target, but *only for the interface bandwidth OIDs*. The CFGMAKER script does not generate a configuration file aimed to monitor other OIDs than Input/Output bandwidth for a network interface.

MRTG has the ability to monitor any SNMP OID that a host can deliver. The CFGMAKER script has limited action range here and the configuration file has to be built manually.

The configuration file structure is made of 2 types of option tags:

- *Global options* // define the way that MRTG will handle the data collection for all targets within the file.

- *Target options* // define specific settings used to collect the data and construct the graphs for each target.

The key option tag that makes the difference between the standard MRTG configuration (input/output network bandwidth) and generic SNMP operation (monitor any SNMP OID) is the *<target>* option tag that describes the OIDs that MRTG will ask a target host for.

The format of the *<target>* option tag is:

```
> target[target-generic-name]: OID1&OID2:community-name@IP-address
```

The OID1 and OID2 are the data sources (DS) and they correspond to the original Input and Output values (the standard MRTG functionality of Input/Output bandwidth for a network interface).

OID1 & OID2 may have the following values:

- *OID1 and OID2 are different OIDs* - if you want to graph two different values per graph.

Example: Collect logical disk C and logical disk D free space OIDs and draw them on the same graph

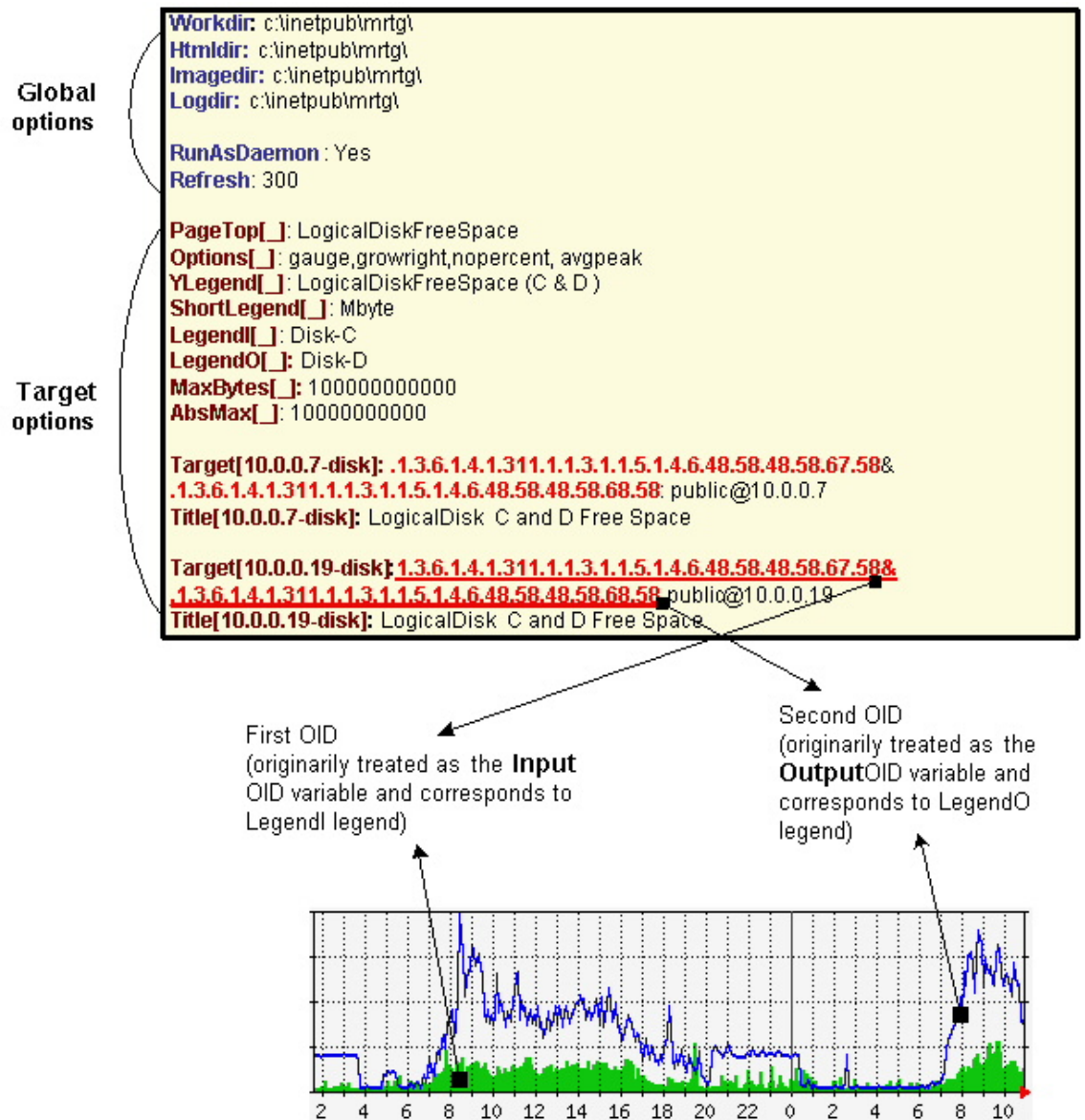
```
> Target[10.0.0.1-disk-  
cd]:.1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.58.48.58.67.58&1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.  
6.48.58.48.58.68.58:public@10.0.0.1
```

- *OID1 and OID2 are one and the same OID* - if you want to graph only one value per graph. In case when you want to represent a single OID per graph it will be represented by OID1 while OID2 (it is the same as OID1) can be ignored when drawing the graph (see paragraph 9.2)

Example: Collect logical disk C free space only.

```
> Target[10.0.0.1-disk-
c]:.1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.58.48.58.67.58&1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6
.48.58.48.58.67.58:public@10.0.0.1
```

The following diagram explains the structure of a configuration file:



## 9.2 Configuration file options

MRTG offers a complex set of option tags to be used in the configuration file.

Global options

### HtmlDir

HtmlDir specifies the directory where the html (or shtml, but we'll get on to those later,) lives.

NOTE: Workdir overrides the settings for htmldir, imagedir and logdir

Example: [HtmlDir: /www/mrtg/](#)

### ImageDir

ImageDir specifies the directory where the images live, they should be under the html directory.

Example: [Imagedir: /www/mrtg/images](#)

### LogDir

LogDir specifies the directory where the logs are stored. This need not be under htmldir directive.

Example: [Logdir: /www/mrtg/logs](#)

### Forks (UNIX only)

An a system that can fork (UNIX for example) mrtg can fork itself into multiple instances while it is acquiring data via snmp.

For situations with high latency or a great number of devices this will speed things up considerably. It will not make things faster though if you query a single switch sitting next door.

Example: [Forks: 4](#)

### Refresh

How many seconds apart should the browser (Netscape) be instructed to reload the page? If this is not defined, the default is 300 seconds (5 minutes).

Example: [Refresh: 600](#)

### Interval

How often do you call mrtg? The default is 5 minutes. If you call it less often, you should specify it here.

In this example we tell mrtg that we will be calling it every 10 minutes. If you are calling mrtg every 5 minutes, you can leave this line commented out.

Example: [Interval: 10](#)

Note: unless you are using rrdtool you can not set Interval to less than 5 minutes. If you are using rrdtool you can set interval down to 1 Minute. Note though, setting the Interval for an rrdtool/mrtg setup will influence the initial creation of the database. If you change the interval later, all existing databases will remain at the resolution they were initially created with.

### WriteExpires

With this switch mrtg will generate .meta files for CERN and Apache servers which contain Expiration tags for the html and gif files. The \*.meta files will be created in the same directory as the other files, so you will have to set ``MetaDir .'' and ``MetaFiles on'' in your apache.conf or .htaccess file for this to work

*NOTE: If you are running Apache-1.2 or later, you can use the mod\_expire to achieve the same effect ... see the file htaccess.txt*

Example: [WriteExpires: Yes](#)

**NoMib2**

Normally we ask the SNMP device for 'sysUptime', 'sysName' properties some do not have these. If you want to avoid getting complaints from mrtg about these missing properties, specify the nomib2 option. An example of agents which do not implement base mib2 attributes are Computer Associates - Unicenter TNG Agents. CA relies on using the base OS SNMP agent in addition to its own agents to supplement the management of a system.

Example: [NoMib2: Yes](#)

**SingleRequest**

Some SNMP implementations can not deal with requests asking for multiple snmp variables in one go. Set this in your cfg file to force mrtg to only ask for one variable per request.

Example: [SingleRequest: Yes](#)

**SnmpOptions**

Apart from the per target timeout options, you can also configure the behaviour of the snmpget process on a more profound level. SnmpOptions accepts a hash of options. The following options are currently supported:

```

timeout          => $default_timeout,
retries          => $default_retries,
backoff          => $default_backoff,
default_max_repetitions => $max_repetitions,
lenient_source_port_matching => 0,
lenient_source_address_matching => 1

```

The values behind the options indicate the current default value. Note that these settings OVERRIDE the per target timeout settings.

Example: [SnmpOptions: retries => 2, only\\_ip\\_address\\_matching => 0](#)

*Note: AS/400 snmp seems to be broken in a way which prevents mrtg from working with it unless*

*SnmpOptions: lenient\_source\_port\_matching => 1 is set.*

**IconDir**

If you want to keep the mrtg icons in some place other than the working (or imagedir) directory, use the IconDir variable for defining the url to the icons directory.

Example: `IconDir: /mrtgicons/`

**LoadMIBs**

Load the MIB file(s) specified and make its OIDs available as symbolic names. For better efficiency, a cache of MIBs is maintained in the WorkDir.

Example: `LoadMIBs: /dept/net/mibs/netapp.mib,/usr/local/lib/ft100m.mib`

**Language**

Switch output format to the selected Language (Check the translate directory to see which languages are supported at the moment. In this directory you can also find instructions on how to create new translations).

Currently the following languages are supported: big5 brazilian bulgarian catalan chinese croatian czech danish dutch eucjp french galician gb gb2312 german greek hungarian icelandic indonesia iso2022jp italian korean lithuanian malay norwegian polish portuguese romanian russian russian1251 serbian slovak slovenian spanish swedish turkish ukrainian

Example: [Language: danish](#)

**LogFormat**

Setting LogFormat to 'rrdtool' in your mrtg.cfg file enables rrdtool mode. In rrdtool mode, mrtg

relies on rrdtool to do its logging. Graphs and html pages will be generated on the fly by the 14all.cgi which can be found in the contrib section together with a short readme ... This feature has been contributed by Rainer Bawidamann <bawidama@users.sourceforge.net>. Please check his website for more information: <http://www.wh-hms.uni-ulm.de/~widi/14all/>

Example: [LogFormat: rrdtool](#)

### LibAdd

If you are using rrdtool mode and your rrdtool Perl module (RRDs.pm) is not installed in a location where perl can find it on its own, you can use LibAdd to supply an appropriate path.

Example: [LibAdd: /usr/local/rrdtool/lib/perl/](#)

### PathAdd

If the rrdtool executable can not be found in the normal PATH, you can use this keyword to add a suitable directory to your path.

Example: [PathAdd: /usr/local/rrdtool/bin/](#)

### RunAsDaemon

The RunAsDaemon keyword enables daemon mode operation. The purpose of daemon mode is that MRTG is launched once and not at regular basis by cron as in native mode. This behavior saves computing resources as loading and parsing of configuration files only happens once. Using daemon mode MRTG itself is responsible for timing the measurement intervals. Therefore it's important to set the Interval keyword to an appropriate value. Note that using daemon mode MRTG should no longer be started from cron by regular basis as each started process runs forever. Instead MRTG should be started from the command prompt or by a system startup script.

If you want mrtg to run under a particular user and group (it is not recommended to run MRTG as root) then you can use the *--user=user\_name and --group=group\_name options on the mrtg commandline.*

```
mrtg --user=mrtg_user --group=mrtg_group mrtg.cfg
```

Also note that in daemon mode restart of the process is required in order to activate changes in the config file.

Under UNIX, the Daemon switch causes mrtg to fork into background after checking its config file. On Windows NT the MRTG process will detach from the console, but because the NT/2000 shell waits for its children you have to use the special start sequence when you launch the program:

```
start /b perl mrtg mrtg.cfg
```

You may have to add path information equal to what you add when you run mrtg from the commandline.

Example (MRTG run as a daemon beginning data collection every 5 minutes)

```
RunAsDaemon:Yes  
Interval:5
```

### Target specific options

#### Title

Title for the HTML page which gets generated for the graph.

Example: [Title\[ezwf\]: Traffic Analysis for Our Nice Company](#)

#### PageTop

Things to add to the top of the generated HTML page. Note that you can have several lines of text as long as the first column is empty.

Note that the continuation lines will all end up on the same line in the html page. If you want linebreaks in the generated html use the '\n' sequence.

Example:

```
PageTop[ezwf]: <H1>Traffic Analysis for ETZ C95.1</H1>  
Our Campus Backbone runs over an FDDI line\n
```

with a maximum transfer rate of 12.5 megabytes per Second.

### RouterUptime

In cases where you calculate the used bandwidth from several interfaces you normally don't get the router uptime and router name displayed on the web page. If these interfaces are on the same router and the uptime and name should be displayed nevertheless you have to specify its community and address again with the RouterUptime keyword.

Example:

```
Target[kacisco.comp.edu]: 1:public@194.64.66.250 + 2:public@194.64.66.250
RouterUptime[kacisco.comp.edu]: public@194.64.66.250
```

### MaxBytes1

Same as MaxBytes, for variable 1.

### MaxBytes2

Same as MaxBytes, for variable 2.

### PageFoot

Things to add to the bottom of the generated HTML page. Note that you can have several lines of text as long as the first column is empty. Note that the continuation lines will all end up on the same line in the html page. If you want linebreaks in the generated html use the '\n' sequence. The material will be added just before the </BODY> tag:

Example:

```
PageFoot[ezwf]: Contact <A HREF="mailto:peter@x.yz";>Peter</A>
if you have questions regarding this page
```

### AddHead

Use this tag like the PageTop header, but its contents will be added between </TITLE> and </HEAD>.

Example:

```
AddHead[ezwf]: <link rev="made" href="mailto:mrtg@blabla.edu";>
```

### BodyTag

BodyTag lets you supply your very own <body ...> tag for the generated webpages.

Example:

```
BodyTag[ezwf]: <BODY LEFTMARGIN="1" TOPMARGIN="1"
BACKGROUND="/stats/images/bg.neo2.gif">
```

### AbsMax

If you are monitoring a link which can handle more traffic than the MaxBytes value. Eg, a line which uses compression or some frame relay link, you can use the AbsMax keyword to give the absolute maximum value ever to be reached. We need to know this in order to sort out unrealistic values returned by the routers. If you do not set AbsMax, rateup will ignore values higher than MaxBytes.

Example: [AbsMax\[ezwf\]: 2500000](#)

### Unscaled

By default each graph is scaled vertically to make the actual data visible even when it is much lower than MaxBytes. With the Unscaled variable you can suppress this. It's argument is a string, containing one letter for each graph you don't want to be scaled: d=day w=week m=month y=year. In the example scaling for the yearly and the monthly graph are suppressed.

Example: [Unscaled\[ezwf\]: ym](#)

### WithPeak

By default the graphs only contain the average values of the monitored variables - normally the

transfer rates for incoming and outgoing traffic. The following option instructs mrtg to display the peak 5 minute values in the [w]eekly, [m]onthly and [y]early graph. In the example we define the monthly and the yearly graph to contain peak as well as average values.

Examples: [WithPeak\[ezwf\]: ym](#)

### Suppress

By default mrtg produces 4 graphs. With this option you can suppress the generation of selected graphs. The option value syntax is analogous to the above two options. In this example we suppress the yearly graph as it is quite empty in the beginning.

Example: `Suppress[ezwf]: y`

### Extension

By default, mrtg creates .html files. Use this option to tell mrtg to use a different extension. For example you could set the extension to php3, then you will be able to enclose PHP tags into the output (usefull for getting a router name out of a database).

Example: [Extension\[ezwf\]: phtml](#)

### Directory

By default, mrtg puts all the files that it generates for each target (the GIFs, the HTML page, the log file, etc.) in WorkDir.

If the Directory option is specified, the files are instead put into a directory under WorkDir or Log-, Image- and HtmlDir). (For example the Directory option below would cause all the files for a target ezwf to be put into directory /usr/tardis/pub/www/stats/mrtg/ezwf/ .)

The directory must already exist; mrtg will not create it.

Example:

`WorkDir: /usr/tardis/pub/www/stats/mrtg`

`Directory[ezwf]: ezwf`

*NOTE: the Directory option must always be 'relative' or bad things will happen.*

### XSize and YSize

By default mrtgs graphs are 100 by 400 pixels wide (plus some more for the labels. In the example we get almost square graphs ...

Note: XSize must be between 20 and 600; YSize must be larger than 20

Example:

`XSize[ezwf]: 300`

`YSize[ezwf]: 300`

### XZoom and YZoom

If you want your graphs to have larger pixels, you can ``Zoom" them.

Example:

`XZoom[ezwf]: 2.0`

`YZoom[ezwf]: 2.0`

### XScale and YScale

If you want your graphs to be actually scaled use XScale and YScale. (Beware while this works, the results look ugly (to be frank) so if someone wants to fix this: patches are welcome.

Example:

`XScale[ezwf]: 1.5`

`YScale[ezwf]: 1.5`

### YTics and YTicsFactor

If you want to show more than 4 lines per graph, use YTics. If you want to scale the value used for the YLegend of these tics, use YTicsFactor. The default value for YTics is 4 and the default value for YTicsFactor is 1.0 .

Example:

Let's suppose you get values ranging from 0 to 700. You want to plot 7 lines and want to

show 0, 1, 2, 3, 4, 5, 6, 7 instead of 0, 100, 200, 300, 400, 500, 600, 700. You should write then:

YTics[ezwf]: 7

YTicsFactor[ezwf]: 0.01

Factor

If you want to multiply all numbers shown below the graph with a constant factor, use this directive to define it ..

Example: [Factor\[as400\]: 4096](#)

### **Step**

Change the default step from 5 \* 60 seconds to something else (I have not tested this well ...)

Example: [Step\[ezwf\]: 60](#)

### **Options**

The Options Keyword allows you to set some boolean switches:

#### **growright**

The graph grows to the left by default. This option flips the direction of growth causing the current time to be at the right edge of the graph and the history values to the left of it.

#### **bits**

All the monitored variable values are multiplied by 8 (i.e. shown in bits instead of bytes) ... looks much more impressive :-). It also affects the 'factory default' labeling and units for the given target.

#### **perminute**

All the monitored variable values are multiplied by 60 (i.e. shown in units per minute instead of units per second) in case of small values more accurate graphs are displayed. It also affects the 'factory default' labeling and units for the given target.

#### **perhour**

All the monitored variable values are multiplied by 3600 (i.e. shown in units per hour instead of units per second) in case of small values more accurate graphs are displayed. It also affects the 'factory default' labeling and units for the given target.

#### **noinfo**

Suppress the information about uptime and device name in the generated webpage.

#### **nopercent**

Don't print usage percentages

#### **transparent**

make the background of the generated gifs transparent ...

#### **integer**

Print summary lines below graph as integers without comma

#### **dorelpercent**

The relative percentage of IN-traffic to OUT-traffic is calculated and displayed in the graph as an additional line. Note: Only a fixed scale is available (from 0 to 100%). Therefore for IN-traffic greater than OUT-traffic also 100% is displayed. If you suspect that your IN-traffic is not always less than or equal to your OUT-traffic you are urged to not use this options. Note: If you use this option in combination with the Colours options, a fifth colour-name colour-value pair is required there.

**avgpeak**

There are some ISPs who use the average Peak values to bill their customers. Using this option MRTG displays these values for each graph. The value is built by averaging the max 5 minute traffic average for each 'step' shown in the graph. For the Weekly graph this means that it builds the average of all 2 hour intervals 5 minute peak values. (Confused? Though so!)

**gauge**

Treat the values gathered from target as 'current status' measurements and not as ever incrementing counters. This would be useful to monitor things like disk space, processor load, temperature, and the like ...

In the absence of 'gauge' or 'absolute' options, MRTG treats variable as a counter and calculates the difference between the current and the previous value and divides that by the elapsed time between the last two readings to get the value to be plotted.

**absolute**

This is for counter type data sources which reset their value when they are read. This means that rateup does not have to build the difference between the current and the last value read from the data source. The value obtained is still divided by the elapsed time between the current and the last reading, which makes it different from the 'gauge' option. Useful for external data gatherers.

**unknaszero**

Log unknown data as zero instead of the default behaviour of repeating the last value seen. Be careful with this, often a flat line in the graph is much more obvious than a line at 0.

**withzeroes**

Normally we ignore all values which are zero when calculating the average transfer rate on a line. If this is not desirable use this option.

**noborder**

If you are using rateup to log data, MRTG will create the graph images. Normally these images have a shaded border around them. If you do not want the border to be drawn, enable this option. This option has no effect if you are not using rateup.

**noarrow**

As with the option above, this effects rateup graph generation only. Normally rateup will generate graphs with a small arrow showing the direction of the data. If you do not want this arrow to be drawn, enable this option. This option has no effect if you are not using rateup.

**noi**

When using rateup for graph generation, you can use this option to stop rateup drawing a graph for the 'I' or first variable. This also removes entries for this variable in the HTML page MRTG generates, and will remove the peaks for this variable if they are enabled. This allows you to hide this data, or can be very useful if you are only graphing one line of data rather than two. This option is not destructive - any data received for the the variable continued to be logged, it just isn't shown.

**noo**

Same as above, except relating to the 'O' or second variable.

**nobanner**

When using rateup for graph generation, this option disables MRTG adding the MRTG banner to the HTML pages it generates.

**nolegend**

When using rateup for graph generation, this option will stop MRTG creating a legend at the

bottom of the HTML pages it generates.

Example:

Options[ezwf]: growright, bits

### **kilo**

Use this option to change the multiplier value for building prefixes. Defaultvalue is 1000. This tag is for the special case that 1kB = 1024B, 1MB = 1024kB and so far.

Example:

kilo[ezwf]: 1024

### **kMG**

Change the default multiplier prefixes (,k,M,G,T,P). In the tag ShortLegend define only the basic units. Format: Comma seperated list of prefixed. Two consecutive commas or a comma at start or end of the line gives no prefix on this item. Note: If you do not want prefixes, then leave this line blank.

Example: velocity in nm/s (nanometers per second) displayed in nm/h.

ShortLegend[ezwf]: m/h

kMG[ezwf]: n,u,m,,k,M,G,T,P

options[ezwf]: perhour

### **Colours**

The Colours tag allows you to override the default colour scheme. Note: All 4 of the required colours must be specified here. The colour name ('Colourx' below) is the legend name displayed, while the RGB value is the real colour used for the display, both on the graph and in the html doc.

Format is: Col1#RRGGBB,Col2#RRGGBB,Col3#RRGGBB,Col4#RRGGBB

Important: If you use the dorelpercent options tag a fifth colour name colour value pair is required: Col1#RRGGBB,Col2#RRGGBB,Col3#RRGGBB,Col4#RRGGBB,Col5#RRGGBB

Colour1: First variable (normally Input) on default graph

Colour2: Second variable (normally Output) on default graph

Colour3: Max first variable (input)

Colour4: Max second variable (output)

Example:

Colours[ezwf]: GREEN#00eb0c,BLUE#1000ff,DARK GREEN#006600,VIOLET#ff00ff

### **Background**

With the Background tag you can configure the background colour of the generated HTML page

Example:

Background[ezwf]: #a0a0a0a

### **YLegend, ShortLegend, Legend[1234]**

The following keywords allow you to override the text displayed for the various legends of the graph and in the HTML document

#### **YLegend**

The Y-axis label of the graph. Note that a text which is too long to fit in the graph will be silently ignored.

#### **ShortLegend**

The units string (default 'b/s') used for Max, Average and Current

#### **Legend[1234IO]**

The strings for the colour legend

Example:

```
YLegend[ezwf]: Bits per Second
ShortLegend[ezwf]: b/s
Legend1[ezwf]: Incoming Traffic in Bits per Second
Legend2[ezwf]: Outgoing Traffic in Bits per Second
Legend3[ezwf]: Maximal 5 Minute Incoming Traffic
Legend4[ezwf]: Maximal 5 Minute Outgoing Traffic
LegendI[ezwf]: &nbsp;In:
LegendO[ezwf]: &nbsp;Out:
```

Note, if LegendI or LegendO are set to an empty string with

### LegendO[ezwf]:

The corresponding line below the graph will not be printed at all.

### Timezone

If you live in an international world, you might want to generate the graphs in different timezones. This is set in the TZ variable. Under certain operating systems like Solaris, this will provoke the localtime call to give the time in the selected timezone ...

Example: [Timezone\[ezwf\]: Japan](#)

The Timezone is the standard Solaris timezone, ie Japan, Hongkong, GMT, GMT+1 etc etc.

### Weekformat

By default, mrtg (actually rateup) uses the strftime(3) '%W' option to format week numbers in the monthly graphs. The exact semantics of this format option vary between systems. If you find that the week numbers are wrong, and your system's strftime(3) routine supports it, you can try another format option. The POSIX '%V' option seems to correspond to a widely used week numbering convention. The week format character should be specified as a single letter; either W, V, or U.

Example: [Weekformat\[ezwf\]: V](#)

### RRDRowCount

This affects the creation of new rrd files. By default rrd files are created to hold about 1 days worth of high resolution data. (plus 1 week of 30 minute data, 2 month of 2 hour data and 2 years of 1 day data). With this Keyword you can change the number of base interval entries configured for new rrd files as they get created. Note that you must take the interval time into account.

Example: [RRDRowCount\[ezwf\]: 1600](#)

When multiple targets share a single configuration file (multiple interfaces of the same host), the config file can be written in an efficient manner by specifying the *target options* that are the same for all targets immediately after the global options and using a " \_ " as target generic name.

```
//global options
Workdir: c:\inetpub\mrtg\
HtmlDir: c:\inetpub\mrtg\
Imagedir: c:\inetpub\mrtg\
Logdir: c:\inetpub\mrtg\
RunAsDaemon: Yes
Refresh: 300
//target shared options
PageTop[_]: Free RAM
Options[_]: gauge,growright,nopercent, avgpeak, noo
ShortLegend[_]: Bytes;
YLegend[_]: Free RAM
LegendI[_]: &nbsp;Free RAM&nbsp;
```

```

AbsMax[_]: 500000000
MaxBytes[_]: 500000000
//individual target options
Target[10.172.10.106-
mem]:.1.3.6.1.4.1.311.1.1.3.1.1.1.1.0&.1.3.6.1.4.1.311.1.1.3.1.1.1.1.0:public@10.172.1
0.106
Title[10.172.10.106-mem]: Free RAM
Target[10.172.10.26-
mem]:.1.3.6.1.4.1.311.1.1.3.1.1.1.1.0&.1.3.6.1.4.1.311.1.1.3.1.1.1.1.0:public@10.172.1
0.26
Title[10.172.10.26-mem]: Free RAM

```

### 9.3 Custom SNMP OID configuration examples (cpu, memory, disk space)

The following example gets the logical disk free space (C,D,E drives), free RAM and CPU utilization for a host.

```

### Global Config Options
Workdir: c:\inetpub\mrtg\
Htmldir: c:\inetpub\mrtg\
Imagedir: c:\inetpub\mrtg\
Logdir: c:\inetpub\mrtg\
Icondir: /mrtg/images
Refresh: 300
RunAsDaemon: Yes
Interval: 5

### CPU Utilization
Target[10.0.0.254-
cpu]:.1.3.6.1.4.1.311.1.1.3.1.1.33.9.0&.1.3.6.1.4.1.311.1.1.3.1.1.33.9.0:public@10.0.0.254
AbsMax[10.0.0.254-cpu]: 100
MaxBytes[10.0.0.254-cpu]: 100
Title[10.0.0.254-cpu]: CPU Utilization (average)
PageTop[10.0.0.254-cpu]: CPU Utilization
Options[10.0.0.254-cpu]: gauge,growright,nopercent, noo
YLegend[10.0.0.254-cpu]: CPU Utilization
ShortLegend[10.0.0.254-cpu]: %
Legend[10.0.0.254-cpu]:&nbsp;CPU Utilization (percentage)&nbsp;

### Free disk C space
Target[10.0.0.254-
diskc]:.1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.58.48.58.67.58&1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.
58.48.58.67.58:public@10.0.0.254
AbsMax[10.0.0.254-diskc]: 100000
MaxBytes[10.0.0.254-diskc]: 100000
Title[10.0.0.254-diskc]: DISK FREE SPACE (C)
PageTop[10.0.0.254-diskc]: DISK FREE SPACE (C)
Options[10.0.0.254-diskc]: gauge,growright,nopercent, noo
YLegend[10.0.0.254-diskc]: DISK FREE SPACE (C)
ShortLegend[10.0.0.254-diskc]: MB
Legend[10.0.0.254-diskc]:&nbsp;DISK FREE SPACE (C)&nbsp;

### Free disk D space

```

```
Target[10.0.0.254-  
diskd]:1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.58.48.58.68.58&1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.  
58.48.58.68.58:public@10.0.0.254  
AbsMax[10.0.0.254-diskd]: 100000  
MaxBytes[10.0.0.254-diskd]: 100000  
Title[10.0.0.254-diskd]: DISK FREE SPACE (D)  
PageTop[10.0.0.254-diskd]: DISK FREE SPACE (D)  
Options[10.0.0.254-diskd]: gauge,growright,nopercent, noo  
YLegend[10.0.0.254-diskd]: DISK FREE SPACE (D)  
ShortLegend[10.0.0.254-diskd]: MB  
LegendI[10.0.0.254-diskd]:&nbsp;DISK FREE SPACE (D)&nbsp;
```

#### ### Free disk E space

```
Target[10.0.0.254-  
diske]:1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.58.48.58.69.58&1.3.6.1.4.1.311.1.1.3.1.1.5.1.4.6.48.  
58.48.58.69.58:public@10.0.0.254  
AbsMax[10.0.0.254-diske]: 100000  
MaxBytes[10.0.0.254-diske]: 100000  
Title[10.0.0.254-diske]: DISK FREE SPACE (E)  
PageTop[10.0.0.254-diske]: DISK FREE SPACE (E)  
Options[10.0.0.254-diske]: gauge,growright,nopercent, noo  
YLegend[10.0.0.254-diske]: DISK FREE SPACE (E)  
ShortLegend[10.0.0.254-diske]: MB  
LegendI[10.0.0.254-diske]:&nbsp;DISK FREE SPACE (E)&nbsp;
```

#### ### Free RAM

```
Target[10.0.0.254-  
mem]:.1.3.6.1.4.1.311.1.1.3.1.1.1.1.0&.1.3.6.1.4.1.311.1.1.3.1.1.1.1.0:public@10.0.0.254  
AbsMax[10.0.0.254-mem]: 900000000  
MaxBytes[10.0.0.254-mem]: 900000000  
Title[10.0.0.254-mem]: FREE RAM  
PageTop[10.0.0.254-mem]: FREE RAM  
Options[10.0.0.254-mem]: gauge,growright,nopercent, noo  
YLegend[10.0.0.254-mem]: FREE RAM  
ShortLegend[10.0.0.254-mem]: B  
LegendI[10.0.0.254-mem]:&nbsp;FREE RAM&nbsp;
```