NAME
rrdgraph_data – preparing data for graphing in rrdtool graph

SYNOPSIS
DEF:⟨*vname*⟩=⟨*rrdfile*⟩:⟨*ds−name*⟩:⟨*CF*⟩[:step=⟨*step*⟩][:start=⟨*time*⟩][:end=⟨*time*⟩][:reduce=⟨*CF*⟩][:daemon=⟨

VDEF:*vname=RPN expression*

CDEF:*vname=RPN expression*

DESCRIPTION
These three instructions extract data values out of the **RRD** files, optionally altering them (think, for example, of a bytes to bits conversion). If so desired, you can also define variables containing useful information such as maximum, minimum etcetera. Two of the instructions use a language called **RPN** which is described in its own manual page.

Variable names (*vname*) must be made up strings of the following characters `A-Z, a-z, 0-9, _, -` and a maximum length of 255 characters.

When picking variable names, make sure you do not choose a name that is already taken by an RPN operator. A safe bet is to use lowercase or mixed case names for variables since operators will always be in uppercase.

DEF
DEF:⟨*vname*⟩=⟨*rrdfile*⟩:⟨*ds−name*⟩:⟨*CF*⟩[:step=⟨*step*⟩][:start=⟨*time*⟩][:end=⟨*time*⟩][:reduce=⟨*CF*⟩][:daemon=⟨

This command fetches data from an **RRD** file. The virtual name *vname* can then be used throughout the rest of the script. By default, an **RRA** which contains the correct consolidated data at an appropriate resolution will be chosen. The resolution can be overridden with the −−step option. The resolution can again be overridden by specifying the **step size**. The time span of this data is the same as for the graph by default, you can override this by specifying **start and end**. Remember to escape colons in the time specification!

If the resolution of the data is higher than the resolution of the graph, the data will be further consolidated. This may result in a graph that spans slightly more time than requested. Ideally each point in the graph should correspond with one **CDP** from an **RRA**. For instance, if your **RRD** has an **RRA** with a resolution of 1800 seconds per **CDP**, you should create an image with width 400 and time span 400*1800 seconds (use appropriate start and end times, such as `--start end-8days8hours`).

If consolidation needs to be done, the **CF** of the **RRA** specified in the **DEF** itself will be used to reduce the data density. This behavior can be changed using `:reduce=`<**CF**>. This optional parameter specifies the **CF** to use during the data reduction phase.

It is possible to request single data sources from a specific *RRDCacheD*, see rrdcached, using the `:daemon=<address>` parameter. The value given to this parameter follows the same syntax as other means to specify the address of the caching daemon. It is described in detail in rrdcached. Beware, however, that colons (in IPv6 addresses and as a port separator, for example) need to be escaped using a backslash.

Example:

```
DEF:ds0=router.rrd:ds0:AVERAGE
DEF:ds0weekly=router.rrd:ds0:AVERAGE:step=7200
DEF:ds0weekly=router.rrd:ds0:AVERAGE:start=end-1h
DEF:ds0weekly=router.rrd:ds0:AVERAGE:start=11\:00:end=start+1h
DEF:ds0weekly=router.rrd:ds0:AVERAGE:daemon=collect1.example.com
```

VDEF
VDEF:*vname=RPN expression*

This command returns a value and/or a time according to the **RPN** statements used. The resulting *vname* will, depending on the functions used, have a value and a time component. When you use this *vname* in another **RPN** expression, you are effectively inserting its value just as if you had put a number at that place.

The variable can also be used in the various graph and print elements.

Example: `VDEF:avg=mydata,AVERAGE`

Note that currently only aggregation functions work in VDEF rpn expressions. Patches to change this are welcome.

CDEF

CDEF:*vname=RPN expression*

This command creates a new set of data points (in memory only, not in the **RRD** file) out of one or more other data series. The **RPN** instructions are used to evaluate a mathematical function on each data point. The resulting *vname* can then be used further on in the script, just as if it were generated by a **DEF** instruction.

Example: `CDEF:mydatabits=mydata,8,*`

About CDEF versus VDEF

At some point in processing, **RRDtool** has gathered an array of rates ready to display.

CDEF works on such an array. For example, *CDEF:new=ds0,8,** would multiply each of the array members by eight (probably transforming bytes into bits). The result is an array containing the new values.

VDEF also works on such an array but in a different way. For example, *VDEF:max=ds0,MAXIMUM* would scan each of the array members and store the maximum value.

When do you use VDEF versus CDEF?

Use **CDEF** to transform your data prior to graphing. In the above example, we'd use a **CDEF** to transform bytes to bits before graphing the bits.

You use a **VDEF** if you want *max(1,5,3,2,4)* to return five which would be displayed in the graph's legend (to answer, what was the maximum value during the graph period).

If you want to apply 'complex' operations to the result of a VDEF you have to use a CDEF again since VDEFs only look like RPN expressions, they aren't really.

SEE ALSO

rrdgraph gives an overview of how **rrdtool graph** works. rrdgraph_data describes **DEF**,**CDEF** and **VDEF** in detail. rrdgraph_rpn describes the **RPN** language used in the **?DEF** statements. rrdgraph_graph page describes all of the graph and print functions.

Make sure to read rrdgraph_examples for tips&tricks.

AUTHOR

Program by Tobias Oetiker <tobi@oetiker.ch>

This manual page by Alex van den Bogaerdt <alex@vandenbogaerdt.nl> with corrections and/or additions by several people