

Crontabs

[Main Page](#) > [General Information](#) > [Configuration Examples](#) > [Router control and monitoring](#) > **Crontabs**



Contents

- [1 Introduction](#)
- [2 Crontab syntax and editing overview](#)
 - [2.1 Editing environment](#)
 - [2.2 Syntax and editing](#)
- [3 Examples](#)
 - [3.1 Periodic SIM switch](#)
 - [3.2 Launching an OpenVPN server on specified hours](#)
 - [3.3 Launching a WiFi Access Point \(AP\) on specified hours](#)
- [4 External links](#)

Introduction

Crontab is a list of commands that allows tasks (programs, scripts) to be run automatically at regular time intervals. For example, in RUTxxx routers it is responsible for executing such services as Automatic Reboot functions, SIM Idle Protection, Scheduled SMS and other functions and services that operate in a periodic manner.

The crontab can be opened for editing, adding, removing or modifying scheduled tasks. This article will provide an explanation of the crontab functionality principle and present some usage examples with the hope that it may help you configure your own crontab rules.

Crontab syntax and editing overview

This section of the guide overviews the syntax of crontab rules, editing methods and editing environment choices.

Editing environment

Firstly, you must decide on what environment you're going to use for editing. Crontabs can be edited via a **command line interface (CLI)**. RUTxxx routers offer a range of options in that regard. For example, you can use the CLI present in the router's WebUI (**Services → CLI**). Or you can login via **SSH** and make edits from there. The method of logging in via SSH is different on different systems: on Linux systems you can use the **Terminal** app login with the command **ssh root@<routers_lan_ip_address>**; on **Windows** systems you can use the free **PuTTY** client.

In all cases the login information is the same (user name: **root**; password: admin password (default: **admin01**)). The syntax of the commands is also the same, so feel free to follow the guide step-by-step whichever method you choose. You can find more detailed information on RUTxxx command

line interfaces in our other wiki article [Command line interfaces](#).


Once you've chosen your preferred environment, we can begin the overview of the syntax and editing of crontabs.

Syntax and editing

Crontab entries are stored in the `/etc/crontabs/root` file. You can edit that file directly with the command `vi /etc/crontabs/root`, but it is more convenient to edit with `crontab -e`, which we'll be using for all examples in this guide. `-e` is an option of the `crontab` command; used for editing. Presented below is a list of all options that can be used with `crontab`:

```
Usage: crontab [-c DIR] [-u USER] [-ler][FILE]
  -c      Crontab directory
  -u      User
  -l      List crontab
  -e      Edit crontab
  -r      Delete crontab
  FILE    Replace crontab by FILE ('-' : stdin)
```

As a demonstration, lets try the `-l` option that shows the contents of the crontab file:



For the sake of this example, I've added a Ping Reboot rule to the router's configuration beforehand. In the figure above you can see that creating such a rule via the router's WebUI also generates an entry (encapsulated in green) in the crontab rule list. Let's examine this entry:

- `/sbin/ping_reboot 1 8.8.8.8 NULL 2 56 5 1 0 cfg02c21d` - indicates the action that is to be taken. In this case it is the `/sbin/ping_reboot` command with the options `1 8.8.8.8 NULL 2 56 5 1 0 cfg02c21d`
- `*/5 * * * *` - indicates the frequency at which the action is executed. It consists out of five segments: **minute**, **hour**, **day of month**, **month** and **day of week** in that order. A table of examples on how to specify this frequency is presented below:

Segment	minute	hour	day of month	month	day of week	Description
Possible values	0-59	0-23	1-31	1-12	0-6	
	*/5	*	*	*	*	every 5 minutes
	12	*/3	*	*	*	every 3 hours at 12 minutes
Examples	57	11	15	1,6,12	*	At 11:57 Hrs on the 15th of Jan, June & Dec.
	25	6	*	*	1-5	At 6:25 AM every weekday (Mon-Fri)
	0	0	4,12,26	*	5	At midnight on the 4th, 12th and 26th of every month plus every Friday
	5,10	9,14	*	*	0,4	At 9:05AM, 9:10AM, 2:05PM and 2:10PM every Sunday and Thursday

There are several nuances to discuss here:

- a **number** indicates the exact time at which an action should occur. For example, if the number **5** was placed in the minute section, the specified action would be taken at the fifth minute of every specified hour.
- an **asterisk** (*) stands for every possible value of the relevant time unit. For example, if placed in the hour section, it would indicate that an action should be taken every hour on specified days.
- two numbers separated by a **dash**(-) indicate a range of numbers. For example, if **4-7** was

placed in the month section, it would indicate that the specified action should take place every fourth, fifth, sixth and seventh months of a year.

- numbers separated by **commas** (,) indicate multiple specific time stamps at which an action should occur. For example, if the numbers **6,9,10** were placed in the hour section, the specified action would be taken at 6:00 AM, 9:00 AM and 10 AM every day. Ranges of numbers can also be separated by commas. For instance, **5-10,20-25** placed in the minute section would indicate that an action must occur every fifth through tenth minute as well as every twentieth through twenty fifth minute of every specified hour.
 - an asterisk and a number separated by a **forward slash** (/) indicate a periodicity. For example, ***/10** placed in the minute section would indicate that the specified action should occur every 10 minutes.
 - the **day of week** segment functions a bit differently than the others:
 - **0** indicates Sunday and **1-6** indicate Monday through Saturday.
 - since weekdays shift throughout the year, i.e., they are not dependent on specific days of a month, when they are used in conjunction with crontab, the action will be executed when the current time matches the value in either field. For instance, **5 5 5 * 5** would cause the specified command to be executed at 5:05 AM every fifth day of every month plus every Friday.
-

To edit crontab, use **crontab -e**. This is analogous to using the **vi** command on the `/etc/crontabs/root` file, so the same rules apply here as when editing with *vi*. Type `crontab -e`, press "Enter" and a text editor for the crontab file will open. To start editing, press the "I" key on your keyboard; you can then edit the crontab file much like with a regular text editor (except you can't use your mouse pointer to place the text cursor anywhere you want; use the arrow keys on your keyboard instead). To save changes after editing, press the "**Escape**" ("**Esc**") button on your keyboard, type **:x** and press "Enter". To exit the editor without saving changes, press the "Escape" button and type **:q!** on your keyboard.

A simpler yet more restricted method of editing would be using the **echo** command to add new lines to the crontab file. It can be used in such a manner: **echo "Hello, world" >> /etc/crontabs/root**. This command will add the text **Hello, world** to the crontab file, i.e., it adds whatever is specified within the quotation marks (" ") after the *echo* command to the file that is specified after the double **more than** (>>) symbol. For example, to add a rule that reboots the router everyday at midnight, we would have to use:

```
# echo "0 0 * * * reboot" >> /etc/crontabs/root
```

Using the command in such a way simply adds a new line with the denoted text at the end of the specified file. This is convenient when you just want to add new rules quickly, but unlike *vi* or *crontab -e* it doesn't offer any other editing capabilities. So if you want edit or remove existing rules, this method will not offer a solution and *crontab -e* should be used instead.

Examples

This section will provide some crontab usage examples in the hopes of helping you to get the hang of the system or even finding an example that you can use. The examples provided here will be examined thoroughly but the editing method and environment will not be discussed; you can read up on those subjects in the section above.

Periodic SIM switch

For this example we'll configure a rule that initiates a SIM switch every weekday at 6:45 PM. To execute a SIM card switch via CLI the command **sim_switch** is used, so we'll combined this with the crontab and configure the rule:

```
45 18 * * 1-5 sim_switch change
```

Let's overview what each segment indicates sequentially:

- 45 - the action must take place at minute 45 of the specified hour
- 18 - the action must take place at 6 PM
- * - all days of the month are applicable to the rule
- * - all months of the year are applicable to the rule
- 1-5 - the action must take place every Monday - Friday
- sim_switch - the command that will be executed
- change - option for the sim_switch command

To sum up, the first five segments denote the frequency of the sim_switch command. The option **change** specifies that the SIM card that was in use up until the time of the switch will be switched to the one that was inactive up until the time of the switch. So the entire entry will perform a SIM switch to the opposite SIM card every weekday at 6:45 PM.

Launching an OpenVPN server on specified hours

For this example we'll configure a system that launches an OpenVPN server at the start of every workday (8 AM) and shuts down said server at the end of every workday (6 PM). For this we'll need to add two rules. Each rule will require multiple commands to be executed. Crontab can launch multiple commands the same way as a single command; the **&&** separator should be used to divide the different commands:

```
0 8 * * 1-5 uci set openvpn.7365727665725F64656D6F.enable=1 && uci commit && /etc/init.d/openvpn start
0 18 * * 1-5 uci set openvpn.7365727665725F64656D6F.enable=0 && uci commit && /etc/init.d/openvpn stop
```

Let's overview what each segment indicates sequentially:

- **First line**
 - 0 - the action must take place at minute 0 (right after the hour has changed)
 - 8 - the action must take place at 8 AM
 - * - all days of the month are applicable to the rule
 - * - all months of the year are applicable to the rule
 - 1-5 - the action must take place every Monday - Friday
 - uci set openvpn.7365727665725F64656D6F.enable=1 - enables the OpenVPN instance
 - uci commit - commits configuration changes
 - /etc/init.d/openvpn restart - restarts the OpenVPN service
- **Second line**

- 0 - the action must take place at minute 0 (right after the hour has changed)
- 18 - the action must take place at 6 PM
- * - all days of the month are applicable to the rule
- * - all months of the year are applicable to the rule
- 1-5 - the action must take place every Monday - Friday
- `uci set openvpn.7365727665725F64656D6F.enable=0` - enables the OpenVPN instance
- `uci commit` - commits configuration changes
- `/etc/init.d/openvpn restart` - restarts the OpenVPN service
- `&&` - separates different commands

To sum up, the first line enables the OpenVPN server at 8 AM, the second disables it at 6 PM. At the end of the enabling and disabling processes the changes are committed and the related services restarted. We used UCI to change the state of the OpenVPN server. You can find more information on UCI in this article: [UCI command usage](#).

NOTE: different OpenVPN instances will have different path names (i.e., `openvpn.7365727665725F64656D6F`) to the `enable` option. You can check the path with this command:

```
uci show openvpn
```

Launching a WiFi Access Point (AP) on specified hours

For this example we'll configure a system that launches a WiFi AP at the start of every workday (8 AM) and shuts down said AP at the end of every workday (6 PM). For this we'll need to add two rules. Each rule will require multiple commands to be executed. Crontab can launch multiple commands the same way as a single command; the `&&` separator should be used to divide the different commands:

```
0 8 * * 1-5 uci set wireless.@wifi-iface[0].user_enable=1 && uci delete
wireless.@wifi-iface[0].disabled && uci commit && wifi
0 18 * * 1-5 uci set wireless.@wifi-iface[0].user_enable=0 && uci set
wireless.@wifi-iface[0].disabled=1 && uci commit && wifi
```

Let's overview what each segment indicates sequentially:

- **First line**
 - 0 - the action must take place at minute 0 (right after the hour has changed)
 - 8 - the action must take place at 8 AM
 - * - all days of the month are applicable to the rule
 - * - all months of the year are applicable to the rule
 - 1-5 - the action must take place every Monday - Friday
 - `uci set wireless.@wifi-iface[0].user_enable=1` - enables the WiFi AP
 - `uci delete wireless.@wifi-iface[0].disabled` - deletes the WiFi *disabled* option
 - `uci commit` - commits configuration changes
 - `wifi` - restarts the WiFi service
- **Second line**
 - 0 - the action must take place at minute 0 (right after the hour has changed)
 - 18 - the action must take place at 6 PM
 - * - all days of the month are applicable to the rule

- * - all months of the year are applicable to the rule
- 1-5 - the action must take place every Monday - Friday
- `uci set wireless.@wifi-iface[0].user_enable=0` - disables the WiFi AP
- `uci set wireless.@wifi-iface[0].disabled=1` - disables the WiFi AP
- `uci commit` - commits configuration changes
- `wifi` - restarts the WiFi service
- && - separates different commands

To sum up, the first line enables the WiFi AP at 8 AM, the second disables it at 6 PM. At the end of the enabling and disabling processes the changes are committed and the related services restarted. We used UCI to change the state of the WiFi AP. You can find more information on UCI in this article: [UCI command usage](#).

NOTE: if you have multiple access points or had them in the past, the path to the relevant options (i.e., `wireless.@wifi-iface[0]`) may be different. You check these paths with this command:

```
uci show wireless | grep able
```

External links

- <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> - PuTTY downloads page link