ACKSYS
COMMUNICATIONS & SYSTEMS

APPLICATION NOTE

APNUS030 How to change basic parameters with CLI commands

June 2023

Content

1. Introduction

UCI stands for Unified Configuration Interface is a small utility written in C available in WaveOS (Acksys Router Operating System to centralize and manage configuration option for ACKSYS routers settings including the main network interface configuration, wireless settings, logging functionality and remote access configuration.

UCI commands provide the user with the maximum degree of control since they can be issued via many different forms of router monitoring and administration (SSH, CLI,) and can be used to set or get any router parameter. This chapter is a guide on how to use UCI commands with ACKSYS devices.

Note: This advanced configuration interface is complex with lot of dependencies between services, we strongly advice doing basic changes with this interface.

ACKSYS doesn't provide any technical support for configuration changed by UCI commands. **We will not be liable for any loss or damage caused by an UCI command applied to configure the router that may affect your equipment. The UCI interface can change between firmware releases and therefore we recommend our customers to use the standard GUI/SNMP interface for any advanced configuration.**

2. Requirements and Prerequesites

Before we begin, let's overview the UCI command on Acksys device in general that we are attempting to achieve and the prerequisites that make it possible.

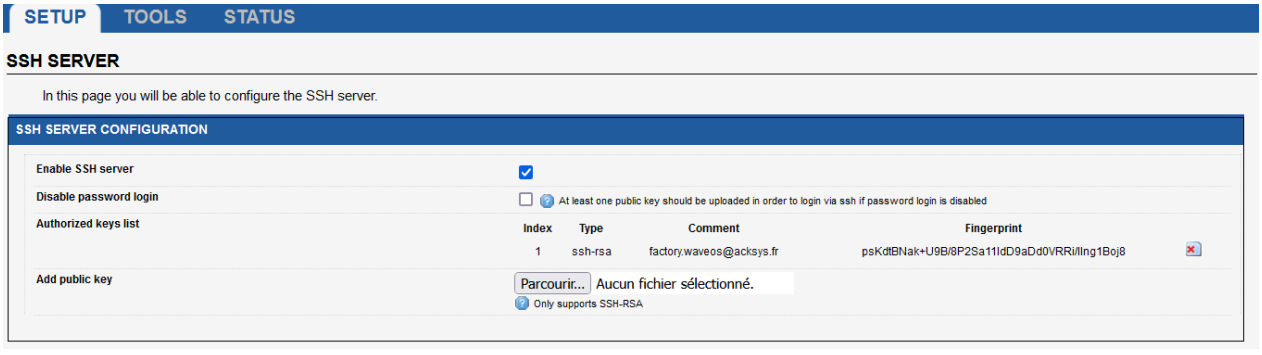
- Any ACKSYS routers
- An end device "Laptop" with Linux OS's Terminal, Putty with Windows OS after authentication on the Router.
- SSH must be enabled on the router via GUI

3. How to enable SSH Server on Acksys Router

UCI commands can be executed via SSH method with Linux OS's Terminal, Putty with Windows OS or MobaTerm after authentication on the Router.

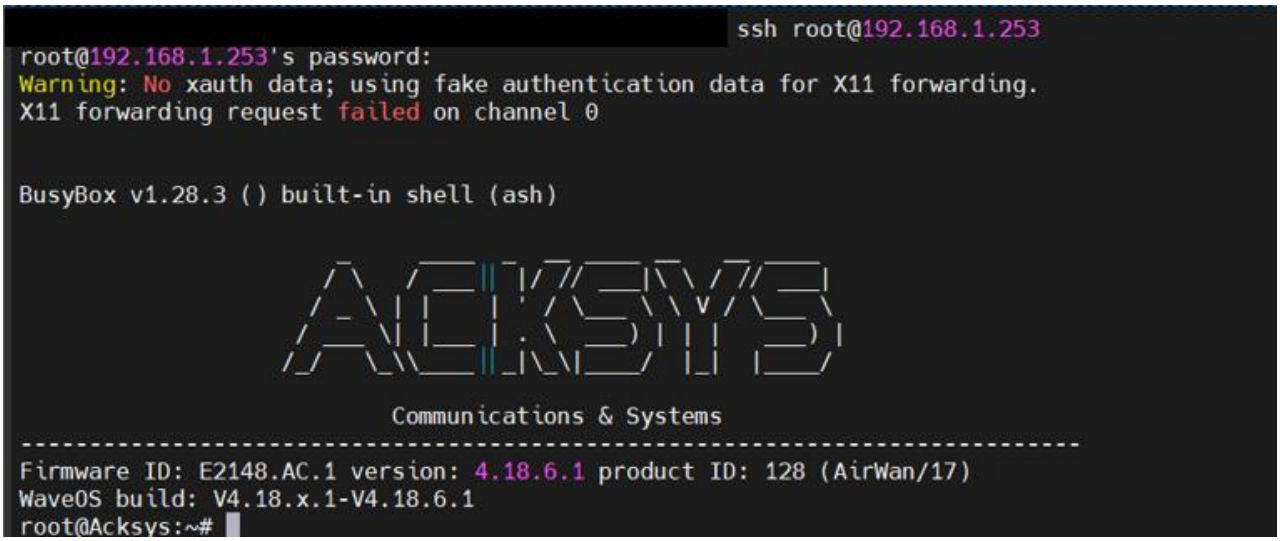
SSH service can be enabled on WaveOs in GUI or via SNMP and in this note, only Web method will be used.

To enable SSH service in WaveOS, let go in GUI and go to Setup → Service → SSH → Click on "Enable SSH server" and untick "Authorized keys" list in this example.



In this note I use a linux terminal with the below information and you will welcome with ACKSYS message after success authentication such as this:

- User name: root
- IP address: 192.168.1.253
- Password: router's admin password



Example of UCI commands

There are plenty of possible UCI commands and options available but in this note, we will only use 3 of them for better understanding (Show, Get and Set).

UCI commands		
Command	**Target**	**Description**
batch	-	Executes a multi-line UCI script which is typically wrapped into a here document syntax
export	[<config>]	Exports the configuration in a machine readable format. It is used internally to evaluate configuration files as shell scripts
import	[<config>]	Imports configuration files in UCI syntax
changes	[<config>]	Lists staged changes to the given configuration file or if none given, all configuration files
commit	[<config>]	Writes changes of the given configuration file, or if none is given, all configuration files, to the filesystem. All "uci set", "uci add", "uci rename" and "uci delete" commands are staged into a temporary location until they are written to flash with the "uci commit" command. This is used exclusively for UCI

		commands and is not needed after editing configuration files with a text editor
add	<config> <section-type>	Adds an anonymous section of type *section-type* to the given configuration
add_list	<config>.<section>.<option>=<string>	Adds the given *string* to an existing list option
del_list	<config>.<section>.<option>=<string>	Removes the given *string* from an existing list option
show	[<config>[.<section>[.<option>]]]	Shows the given option, section or configuration in compressed notation. If no option is given, shows all configuration files
get	<config>.<section>[.<option>]	Gets the value of the given option or the type of the given section
set	<config>.<section>[.<option>]=<value>	Sets the value of the given option, or add a new section with the type set to the given *value*
delete	<config>[.<section>[[.<option>][=<id>]]]	Deletes the given section or option
rename	<config>.<section>[.<option>]=<name>	Renames the given option or section to the given name

Options

```
-c  <path>  set the search path for config files (default: /etc/config)
-d  <str>   set the delimiter for list values in uci show
-f  <file>  use <file> as input instead of stdin
-m          when importing, merge data into an existing package
-n          name unnamed sections on export (default)
-N          don't name unnamed sections
-p  <path>  add a search path for config change files
-P  <path>  add a search path for config change files and use as default
-q          quiet mode (don't print error messages)
-s          force strict mode (stop on parser errors, default)
-S          disable strict mode
-X          do not use extended syntax on 'show'
```

Configuration hierarchy

UCI commands can be used to set and obtain parameters, but to do so, one has to first know the names of the **config** file, its **section** and the **option** that they are trying to interact with. Different configurations for different router functions and services are stored in config files. These config files have sections and section usually store multiple options.

The elements in the UCI model are:

Object	description
config	main configuration groups like network, system, firewall. Each configuration group has its own file in /etc/config
sections	a config is divided into sections. A section can either be named or unnamed

types	a section can have a type. E.g., in the network config we typically have sections of the type "interface"
options	each section has options that hold configuration values
values:	value of an option

4. Acksys configuration files

WaveOs (the Operating system) centralize configuration and split its into several files located in the /etc/config/ directory and each file belongs to the part of the system.

Note that these config files below from any ACKSYS router depend on the configuration done therefore some of them may not exist in your router.

```
root@Acksys:~# ls -l /etc/config
lrwxrwxrwx    1 root     root            21 Jan 10 13:26 ack_cur_wireless → /tmp/ack_cur_wireless
-rw-r--r--    1 root     root            78 Jan 10 13:26 ack_discover
-rw-rw-r--    1 root     root          1269 Jan 10 13:26 ack_firmware
-rw-r--r--    1 root     root           131 Jan 10 13:26 ackckey
-rw-r--r--    1 root     root             0 Jan 10 13:26 ackevent
-rw-------    1 root     root             1 Jan 10 13:26 acksysfilter
-rw-------    1 root     root           379 Jan 10 13:26 acksysqos
-rw-rw-r--    1 root     root            33 Jan 10 13:26 acksysvlan
-rw-rw-r--    1 root     root            39 Jan 10 13:26 async_sysupgrade
-rw-------    1 root     root            47 Jan 10 13:26 conntrack
-rw-------    1 root     root           765 Jan 10 13:26 dhcp
-rw-r--r--    1 root     root           110 Jan 10 13:42 dropbear
-rw-r--r--    1 root     root           210 Jan 10 13:26 firewall
-rw-r--r--    1 root     root           151 Jan 10 13:26 fstab
-rw-------    1 root     root           138 Jan 10 13:26 gpsd
-rw-rw-r--    1 root     root            30 Jan 10 13:26 ipsec
-rw-------    1 root     root            50 Jan 10 13:26 keepalived
-rw-r--r--    1 root     root           485 Jan 10 13:26 luci
-rw-r--r--    1 root     root          1351 Jan 10 13:26 luci_statistics
-rw-r--r--    1 root     root           496 Jan 10 13:26 network
-rw-------    1 root     root             0 Jan 10 13:26 openvpn
-rw-r--r--    1 root     root          2959 Jan 10 13:26 openvpn_recipes
-rw-r--r--    1 root     root             0 Jan 10 13:26 passpoint
-rw-r--r--    1 root     root           234 Jan 10 13:26 pimd
-rw-r--r--    1 root     root          3219 Jan  1  1970 product
-rw-------    1 root     root          1642 Jan 10 13:26 qos_queues
-rw-------    1 root     root            28 Jan 10 13:26 radius
-rw-------    1 root     root           170 Jan 10 13:26 rogueap
-rw-------    1 root     root            97 Jan 10 13:26 rpcd
-rw-------    1 root     root             0 Jan 10 13:26 rstp
-rw-r--r--    1 root     root          2382 Jan 10 13:26 snmpd
-rw-rw-r--    1 root     root           328 Jan  1  1970 system
-rw-r--r--    1 root     root             0 Jan 10 13:26 ubootenv
-rw-r--r--    1 root     root          1025 Jan 10 13:26 ucitrack
-rw-------    1 root     root           714 Jan 10 13:26 uhttpd
-rw-------    1 root     root           213 Jan 10 13:26 wacd
-rw-r--r--    1 root     root          1632 Jan 10 13:26 wireless
lrwxrwxrwx    1 root     root            22 Jan 10 13:48 zoneinfo → ../acksys/zoneinfo_uci
```

5. Obtaining parameters

This section will overview uci get and uci show commands used to obtain router parameters, option and section names and contents of entire configs or sections.

UCI get

The **uci get** command returns values for specific options. When using uci get, you have provided the correct path to the option that you are looking for.

For example, in order to obtain the Wi-Fi Access Point's SSID you would have to use a command that looks like this:

```
uci get wireless.@wifi-iface[0].ssid
```

RESPONSE

```
root@Acksys:~# uci get wireless.@wifi-iface[0].ssid

acksys
```

The command above returns the Wi-Fi Access Point's SSID. As you can see the uci get command is used. What follows after the command is the path to the value that we're looking for (SSID, in this case). The SSID value can be found in the wireless config, the @wifi-iface[0] section, stored under an option called ssid. So the basic syntax for a uci get command is this:

```
uci get <config>.<section>[.<option>]
```

UCI show

If you don't know what the exact option is called and in which section of what config file it is stored, you can use the **uci show** command.

uci show can also be used to obtain values of specific options, but it is more commonly used to display the contents of entire sections or configs. Let's modify the example above by saying that want to find out the SSID value but don't know the exact section or option under which the value is stored. In this case we'll the uci show command to view the contents of the entire wireless config:

```
# uci show wireless
```

RESPONSE

The out of the command in split in two part for better understanding.

```
root@Acksys:~# uci show wireless
wireless.radio0=wifi-device
wireless.radio0.type='mac80211'
wireless.radio0.channel='auto'
wireless.radio0.hwmode='11nbg'
wireless.radio0.path='platform/qca955x_wi
wireless.radio0.macaddr='00:09:90:02:76:
wireless.radio0.htmode='HT20'
wireless.radio0.disabled='1'
wireless.radio0.country='US'
wireless.radio0.qos_profile='default'
wireless.radio0.full11a='0'
wireless.radio0w0=wifi-iface
wireless.radio0w0.device='radio0'
wireless.radio0w0.network='lan'
wireless.radio0w0.mode='ap'
wireless.radio0w0.ssid='acksys'
wireless.radio0w0.encryption='none'
wireless.radio0w0.wds='1'
wireless.user=qos_profile
wireless.user.wmm_ac_bk_cwmin='4'
wireless.user.wmm_ac_bk_cwmax='10'
wireless.user.wmm_ac_bk_aifs='7'
wireless.user.wmm_ac_bk_txop_limit='0'
wireless.user.wmm_ac_bk_acm='0'
wireless.user.wmm_ac_be_cwmin='4'
wireless.user.wmm_ac_be_cwmax='10'
wireless.user.wmm_ac_be_aifs='3'
wireless.user.wmm_ac_be_txop_limit='0'
wireless.user.wmm_ac_bk_acm='0'
wireless.user.wmm_ac_be_cwmin='4'
wireless.user.wmm_ac_be_cwmax='10'
wireless.user.wmm_ac_be_aifs='3'
wireless.user.wmm_ac_be_txop_limit='0'
wireless.user.wmm_ac_be_acm='0'
wireless.user.wmm_ac_vi_cwmin='3'
wireless.user.wmm_ac_vi_cwmax='4'
wireless.user.wmm_ac_vi_aifs='2'
wireless.user.wmm_ac_vi_txop_limit='94'
wireless.user.wmm_ac_vi_acm='0'
wireless.user.wmm_ac_vo_cwmin='2'
wireless.user.wmm_ac_vo_cwmax='3'
wireless.user.wmm_ac_vo_aifs='2'
wireless.user.wmm_ac_vo_txop_limit='47'
wireless.user.wmm_ac_vo_acm='0'
wireless.user.tx_queue_data3_cwmin='15'
wireless.user.tx_queue_data3_cwmax='1023'
wireless.user.tx_queue_data3_aifs='7'
wireless.user.tx_queue_data3_burst='0'
wireless.user.tx_queue_data2_cwmin='15'
wireless.user.tx_queue_data2_cwmax='63'
wireless.user.tx_queue_data2_aifs='3'
wireless.user.tx_queue_data2_burst='0'
wireless.user.tx_queue_data1_cwmin='7'
wireless.user.tx_queue_data1_cwmax='15'
wireless.user.tx_queue_data1_aifs='1'
wireless.user.tx_queue_data1_burst='3.0'
wireless.user.tx_queue_data0_cwmin='3'
wireless.user.tx_queue_data0_cwmax='7'
wireless.user.tx_queue_data0_aifs='1'
wireless.user.tx_queue_data0_burst='1.5'
wireless.supplicant=supplicant
wireless.supplicant.loglevel='3'
```

As you can see, the response shows the entire wireless config and its entities. Note that instead of just showing values (like in the case of uci get) you can see the config name, section name and option name before each one.

Most config file names are simple. Wireless config is called wireless, DHCP config is called dhcp, etc. But even so one doesn't necessarily have to know what a config file is called, especially before interacting with it. if you're CLI or SSH and want to check the names of config files on the spot, you can use the **ls** command. Since ACKSYS configs are stored in **/etc/config**, the full commands should look like this:

```
# ls /etc/config
```

The ls command is used to view the contents of a directory. Here is an example of the /etc/config directory of a ACKSYS router:

```
root@ALPHA:~# ls /etc/config
ack_cur_wireless  acksysfilter      dhcp          ipsec           openvpn          qos_queues  snmpd      wacd
ack_discover      acksysqos         dropbear      keepalived      openvpn_recipes  radius      system     wireless
ack_firmware      acksysvlan        firewall      luci            passpoint        rogueap     ubootenv   zoneinfo
ackckey           async_sysupgrade  fstab         luci_statistics pimd             rpcd        ucitrack
ackevent          conntrack         gpsd          network         product          rstp        uhttpd
root@ALPHA:~#
```

So when you plan on obtaining specific parameters or setting parameter values, you should always start with finding out option and section names. To accomplish this, we recommend using the uci show <config> commands.

6. Setting parameters

UCI can also be used to set parameters, add lists of parameters and even add entire sections to config files and let having example of possible commands.

UCI set

The **uci set** command is used to set the values of specific options and let changing the Wifi SSID in this example:

```
uci set wireless.radio0w0.ssid='LAB'
uci commit
apply_config
```

As you may have noticed, the command is very similar to uci get, except it has an equals to ('=') sign added at the end and after the sign is the value that we want to assign to the option.

The next step is to commit by using the **uci commit** command and to restart all the services relevant to our configuration by using the **apply_config** command.

It is really important to do apply_config as it will automatically apply the dependencies between the different services for the given option change.

```
uci show wireless
```

RESPONSE

Let choosing only the SSID section on which we will make the SSID change.

```
wireless.radio0w0.ssid='LAB'

-----'

wireless.radio0w0.per_sta_ps_buffers='64'
```

Changing the SSID

With the SSID information, we're ready to change the SSID name. To do this, we'll issue a command using the following syntax:

```
uci set  wireless.radio0w0.ssid='Acksys'

uci commit

apply_config
```

Where the info is replaced by whatever we want to use. In our example, the command does the following:

- Set the SSID to Acksys instead of LAB

After change apply, let test if the SSID is changed as expected by using uci show pinging the new IP address.

```
root@Acksys:~# uci show wireless.radio0w0.ssid

wireless.radio0w0.ssid='Acksys'
```

How to change the default LAN IP Address

Before you change the IP address and related information, we'll need to find the network for the interface we want to change. In this example, we will change the default LAN IP address on router and let checking the network information

```
uci show network
```

Response of the Command

```
root@Acksys:~# uci show network
network.loopback=interface
network.loopback.ifname='lo'
network.loopback.proto='static'
network.loopback.ipaddr='127.0.0.1'
network.loopback.netmask='255.0.0.0'
network.globals=globals
network.globals.ula_prefix='fd42:9748:d120::/48'
network.lan=interface
network.lan.type='bridge'
network.lan.ifname='eth0 eth1'
network.lan.proto='static'
network.lan.ipaddr='192.168.1.253'
network.lan.netmask='255.255.255.0'
network.lan.ip6assign='60'
network.wwan0=interface
network.wwan0.proto='wwan'
network.wwan0.auth='none'
network.wwan0.description='Cellular'
network.wwan0.disabled='1'
```

Changing LAN IP ADDRESS

With the interface information, we're ready to change the IP Address, subnet mask, and gateway. To do this, we'll issue a command using the following syntax:

```
uci set network.lan.ipaddr='192.168.1.254'
uci commit
apply_config
```

Where the info is replaced by whatever we want to use. In our example, the command does the following:

- Set the IP address to 192.168.1.254

After change apply, let test if the network configuration is changed as expected by pinging the new IP address.

```
                                                    ping  192.168.1.254

Envoi d'une requête 'Ping'  192.168.1.254 avec 32 octets de données :
Réponse de 192.168.1.254 : octets=32 temps<1ms TTL=64
Réponse de 192.168.1.254 : octets=32 temps<1ms TTL=64
Réponse de 192.168.1.254 : octets=32 temps<1ms TTL=64
Réponse de 192.168.1.254 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.1.254:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 0ms, Moyenne = 0ms
```

STATUS:

```
root@Acksys:~# uci show network
network.loopback=interface
network.loopback.ifname='lo'
network.loopback.proto='static'
network.loopback.ipaddr='127.0.0.1'
network.loopback.netmask='255.0.0.0'
network.globals=globals
network.globals.ula_prefix='fd42:9748:d120::/48'
network.lan=interface
network.lan.type='bridge'
network.lan.ifname='eth0 eth1'
network.lan.proto='static'
network.lan.netmask='255.255.255.0'
network.lan.ip6assign='60'
network.lan.ipaddr='192.168.1.254'
network.wwan0=interface
network.wwan0.proto='wwan'
network.wwan0.auth='none'
network.wwan0.description='Cellular'
network.wwan0.disabled='1'
```

Support : https://support.acksys.fr