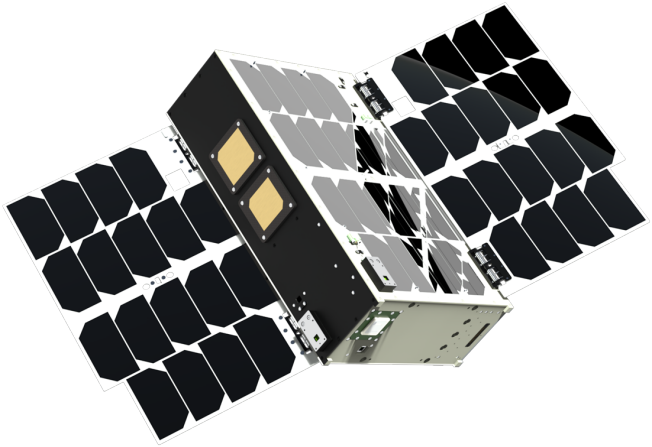
SPACE INVENTOR

WWW.SPACE-INVENTOR.COM



CSH

User Manual

Version 2

Contents

1 CSH

Command Shell for Linux PC's

1.1 Description

CSH is a Linux program designed to interface to Space Inventor modules and satellites using a CSP interface. With a CAN dongle the PC will perform as a first-class citizen on the satellite bus and have full access to all systems.

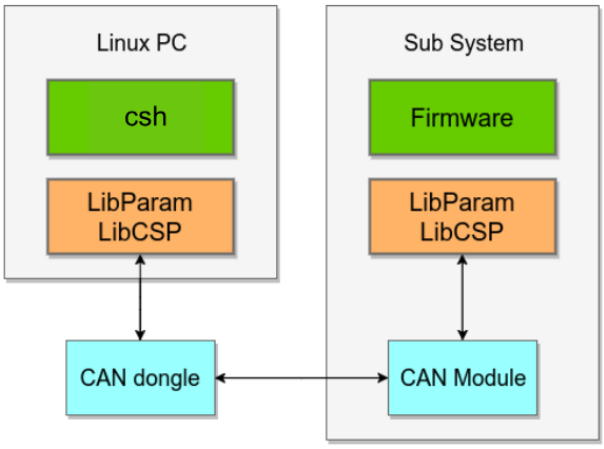
After launch, CSH is also used to operate the satellite through the radio interface, utilising CSPs routing functionalities.

CSH is written in C, and uses the same software libraries (libcsp and libparam) as Space Inventors subsystems. This ensures full compatibility with the protocol stack as well as the higher layer parameter system.

1.2 Features

- CSP version 1 and version 2 support
- CAN bus, UART, ZMQ and other interfaces
- Space Inventor shell interface
- Remote system operations
- Parameter system client
- File transfer client
- CSP node and shell commands

1.3 Physical Setup



The system consists of a Linux PC with a CAN dongle, and a device under test (the subsystem). There can be several CSP nodes on the bus as well.

2 Installation

2.1 Requirements

In order to build you need to install the following requirements:

```
$ sudo apt install git build-essential libsocketcan-dev
can-utils libzmq3-dev libyaml-dev pkg-config fonts-powerline python3-pip libelf-dev
$ sudo pip3 install meson ninja
```

Install Requirements

2.2 Build and install

The software is available as open source on github.

```
$ git clone https://github.com/spaceinventor/csh.git
$ cd csh
$ git submodule update --init --recursive
$ ./configure
$ ./build
```

Clone, configure, build and install software

2.3 Launch software

In order to launch in shell / interactive mode, start csh without a command. Run csp init. Then add an interface with one of the csp add commands and choose a CSP address that will not conflict with any devices.

```
$ csh -h
usage: csh -i init.csh[command]
$ csh
host  csp init
Version 2
Hostname: lenovo
Model: #36~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri   Feb 17 15:17:25 UTC 2
Revision: 5.19.0-35-generic
Deduplication: 3
host  csp add zmq -d 8 localhost
ZMQ init ZMQO: addr: 8, pub(tx): [tcp://localhost:6000], sub(rx): [tcp://localhost:7000]
```

Launch csh

2.4 Init configuration

In order to avoid configuring csp and interfaces on every launch an init.csh file can be created. CSH will look for one in your home directory or you can select one at launch with the -i argument. In the init folder, examples can be found for different interface configurations.

3 Operating instructions

CSH serves for module testing and operations. This section outlines the procedure for module testing, provides examples of command usage, and presents a list of commands for use in CSH.

3.1 Shell interface

CSH runs a simple command line interface (slash). After starting CSH you will see a prompt. For example to ping a node:

```
$ csh
host node 212
host 212 ping
Ping node 212 size 0 timeout 1000: Reply in 14...
host obc@212 ident
IDENT 212
 obc-hk
 FLASH-1
 v1.2-1-gd958d8e+
 Mar 17 2023 12:18:08
```

Example of ping and ident commands

3.2 In-application help texts

Most commands include a help text that is accessible by executing the command followed by -h.

3.3 Example module testing procedure using CSH

1. Connect your module to a CAN dongle (PCAN IPEH-2022 with recommended galvanic isolation) and to a PC running Linux with the socketcan driver.
2. Install our custom software csh (https://github.com/spaceinventor/csh).
3. Once this is configured, open a terminal on the Linux PC and run csh.
4. The command *ident 16383* (broadcast address) will display all nodes on the local network.
5. Once the node where your module is located is identified, simply navigate to it using the command *node <node num>*.
6. Use *list download* to download a list of parameters of your module. Use *pull* to retrieve the value of all parameters.
7. Use *set* to set a specific parameter to a value. For more examples and information about CSH, refer to the following sections of this manual.

3.4 List of commands

3.4.1 Build-in commands

Name	Usage
apm load	[OPTIONS...] -f <filename> -p <pathname>
apm info	[OPTIONS...] <search>
buffree	[OPTIONS...] [node]
cd	
cmd	
cmd add	[OPTIONS...] <name> [offset] [value]
cmd done	[OPTIONS...] -f <filename> -p <pathname>
cmd new	[OPTIONS...] <get/set> <cmd name>
cmd run	[OPTIONS...]
cmd server download	[OPTIONS...] <name>
cmd server list	[OPTIONS...]
cmd server rm	[OPTIONS...] <name>
cmd server upload	[OPTIONS...] <name>
csp add can	[OPTIONS...] <addr>
csp add kiss	[OPTIONS...] <addr>
csp add route	[OPTIONS...] <addr>/<mask> <ifname>
csp add tun	[OPTIONS...] <ifaddr> <tun src> <tun dst>
csp add udp	[OPTIONS...] <addr> <server>
csp add zmq	[OPTIONS...] <addr> <server>
csp init	[OPTIONS...]
csp scan	[OPTIONS...]
download	[OPTIONS...] <address> <length> <file>
exit	
get	[OPTIONS...] <name> [offset]
help	help [command]
history	
ident	[OPTIONS...] [node]

Name	Usage
ifstat	[OPTIONS...] <ifname>
info	
list	[OPTIONS...] [name wildcard=*]
list add	[OPTIONS...] <name> <id> <type>
list download	[OPTIONS...]
list forget	[OPTIONS...]
list save	[OPTIONS...] [name wildcard=*]
list upload	<address> [OPTIONS...]
ls	
node	
node add	[OPTIONS...] <name>
node list	
node save	
param_server start	
peek	[OPTIONS...] <addr> <len>
ping	[OPTIONS...] [node]
poke	[OPTIONS...] <addr> <data base16>
program	[OPTIONS...] <slot>
prometheus start	[OPTIONS...]
pull	
rdp opt	
reboot	[OPTIONS...][node]
resbuf	[OPTIONS...]
run	[OPTIONS...] <filename>
sleep	
schedule cmd	<server> <name> <host> <time> <latency buffer> [timeout]
schedule list	<server> [timeout]
schedule push	<server> <host> <time> <latency buffer> [timeout]
schedule reset	<server> <last id> [timeout]
schedule rm	<server> <id> [timeout]
schedule show	<server> <id> [timeout]

Name	Usage
set	[OPTIONS...] <name> [offset] <value>
shutdown	[OPTIONS...] [node]
sps	[OPTIONS...] <slot>
set	[OPTIONS...] <name> [offset] <value>
shutdown	[OPTIONS...] [node]
sps	[OPTIONS...] <slot>
set	[OPTIONS...] <name> [offset] <value>
shutdown	[OPTIONS...] [node]
sps	[OPTIONS...] <slot>
stdbuf	[OPTIONS...]
stdbuf2	[OPTIONS...]
switch	[OPTIONS...] <slot>
time	[OPTIONS...] [timestamp]
timeout	<timeout ms>
upload	[OPTIONS...] <file> <address>
uptime	[OPTIONS...] [node]
vts init	
vm start	[OPTIONS...] [node]
vm stop	
vmem	[OPTIONS...]
watch	[OPTIONS...] <command...>

3.4.2 Housekeeping APM (libcsh_hk.so) commands

Name	Usage
hk retrieve	[OPTIONS...]
hk timeoffset	[OPTIONS...]

3.4.3 Cortex APM (libcsh_cortex.so) commands

Name	Usage
csp add leafspace	[OPTIONS...] <addr> <NoradID> <mqtt-broker>

Name	Usage
csp add cortex_crt	[OPTIONS...] <addr> <server>
csp add cortex_hdr	[OPTIONS...] <addr> <server>

3.5 Command Examples

Here is a list of the most used commands together with small examples. The list is subject to change.

exit

Exit csh (ctrl+d may also be used)

history

Show command history

help

Print out list of commands

csp init

Initialize CSP in CSH. Optionally select CSP version.

```
host  csp init
Version 2
Hostname: host
```

csp add can

Register new CAN interface in CSH.

```
host  csp add can1
INIT CAN0: device:[can0],bitrate: 1000000,
...promisc:1
```

csp add zmq

Initialise a new ZMQ interface. .. class:: table

```
host  csp add zmq1 localhost
ZMQ init ZMQ0: addr: 1, pub(tx):
...[tcp://localhost:6000],sub(rx):
...[tcp://localhost:7000]
```

csp add route

Add new routes to the CSP routing table in CSH.

```
host  csp route add 64/8 CAN0
Added route 64/8 ZMQ0
...[tcp://localhost:6000],sub(rx):
...[tcp://localhost:7000]
```

csp scan

Scan all nodes for devices.

```
host  csp scan
CSP SCAN [0:16382]
Found something on addr 0...
lenovo
#36~22.04.1-Ubuntu SMP PREEMP
5.19.0-35-generic
Mar 22 2023 14:17:59

Found something on addr 212...
obc-hk
FLASH-1
v1.2-1-gd958d8e+
Mar 17 2023 12:18:08
```

prometheus start

Start the prometheus node exporter to forward all parameter request to Promotheus.

Use the hk timeoffset command to set the node of hk server for the housekeeping sniffer, to forward historical data requested from house keeping service on a satellite.

```
host  prometheus start
Prometheus exporter listening on port 9101
host  prometheus start -n 6
Initialising house keeping sniffer for HK node 6
```

vm start

Starts pushing parameters to a victoria metrics database.

Use the hk timeoffset command to set the node of hk server for the housekeeping sniffer, to forward historical data requested from house keeping service on a satellite.

```
host  vm start localhost
Connection established to http://localhost:8428
host  vm start -u username -p password -s -P 8427
hostname.com
```

info

Provides CSP info for the local node. First the routing table, then the connection table and finally interface statistics.

```
host >  info
[00 0x556b4da62120] S:0, 0 -> 0, 17 -> 1 (17) fl 1
[01 0x556b4da640f8] S:0, 0 -> 0, 18 -> 1 (18) fl 1
[02 0x556b4da660d0] S:0, 0 -> 0, 19 -> 1 (19) fl 1
...
[16 0x556b4da81ea0] S:0, 0 -> 0, 33 -> 1 (33) fl 1
[17 0x556b4da83e78] S:0, 0 -> 0, 34 -> 1 (34) fl 1
[18 0x556b4da85e50] S:0, 0 -> 0, 35 -> 1 (35) fl 1
[19 0x556b4da87e28] S:0, 0 -> 0, 36 -> 1 (36) fl 1
LOOP       addr: 0 netmask: 14
           tx: 00026 rx: 00026 txe: 00000 rxe: 00000
           drop: 00000 autherr: 00000 frame: 00000
           txb: 104 (104B) rxb: 104 (104B)

ZMQ0       addr: 107 netmask: 8
           tx: 00070 rx: 00086 txe: 00000 rxe: 00000
           drop: 00000 autherr: 00000 frame: 00000
           txb: 2117 (2K) rxb: 5033 (4K)
```

node

Sets default/environment node for most commands. Giving the node as a positional argument when running a command will take precedence over the default environment node set.

```
host >  node 6
host 6 >
```

upload

Upload a file to a memory.

```
>  hello.txt
host 6 >  upload hello.txt 0x30001000
Upload from hello.txt to node 6 addr 0x30001000 with timeout 2000
Size 12
 . - 0 K
Uploaded 12 bytes in 0.003 s at 4000 Bps
```

download

Download memory to a file.

```
host 6 >  download 0x30001000 12 hello2.txt
host 6 >  upload hello.txt 0x30001000
Download from 6 addr 0x30001000 to hello2.txt with timeout 10000
 . - 0 K
Downloaded 12 bytes in 0.007 s at 1714 Bps
host 6 >  exit
>  $ cat hello2.txt
HELLO WORLD
```

pull

Get all parameters from a remote node.

```
host 6  pull
130:6  adc_temp             = 21769
303:6  alarm_dbg            = 1
25:6  boot_cnt             = 451
24:6  boot_cur             = 0
26:6  boot_err             = 32
21:6  boot_img0            = 0
20:6  boot_img1            = 0
384:6  ch_protect           = 0
13:6  csp_can_pwrsave      = 1
11:6  csp_can_speed        = 1000000
10:6  csp_node             = 6
12:6  csp_rtable           = ""
140:6  dac_enabled          = [0 0 0 0 0 0]
164:6  efficiency           = 0.0000
```

set

Set a single parameter.

```
host 6  set gndwdt 10000
1:6  gndwdt               = 10000 uint32[1]
```

get

Get a single parameter.

```
host 6  get gndwdt 10000
   1:6  gndwdt               = 9997
```

list download

Download a list of remote parameters.

```
host 6  list download
Got param: adc_temp[1]
Got param: alarm_dbg[1]
Got param: boot_cnt[1]
Got param: boot_cur[1]
Got param: boot_err[1]
...
Got param: tlm_vmax[1]
Got param: tlm_vmin[1]
Got param: v_in[6]
Got param: v_out[1]
Received 81 parameters
```

list

Print current parameter list of selected node. Use -n -1 to list all remote parameters from all nodes.

```
host 6  list
20:6  boot_img1             = 0
21:6  boot_img0             = 0
22:6  boot_img2             = 0
23:6  boot_img3             = 0
24:6  boot_cur              = 0
25:6  boot_cnt              = 0
26:6  boot_err              = 0
1:6  gndwdt                 = 0
51:6  csp_buf_out           = 0
...
```

watch

Repeat a command periodically.

```
host 6  watch -n 1000 "ping"
Executing "ping" each 1000 ms - press <enter> to stop
Ping node 6 size 1 timeout 1000: Reply in 2 [ms]
Ping node 6 size 1 timeout 1000: Reply in 8 [ms]
Ping node 6 size 1 timeout 1000: Reply in 2 [ms]
```

time

Remote timesync.

```
host 6  time
Remote time is 1516625445.622655490 (diff 107 us)
```

poke

Manipulate remote memory (<200 bytes)

```
host 6  poke 0x30001000 DEADBEEF
Base16-decoded "DEADBEEF" to:
Poke at address 0x30001000
0x7ffc60726e67  de ad be ef
...
```

peek

Request a small (<200 bytes) piece of memory.

```
host 6  peek 0x30001000 16
Peek at address 0x30001000 len 16
 0x7ffc60726e67  48 45 4c 4c 4f 20 57 4f 52 4c 44 0a 00 00 00 00
HELLO WORLD.....
```

ifstat

Remotely request interface statistics. For a combined overview of all interfaces, use the parameter csp_print_cnf that is available on most modules.

```
host 6  ifstat CAN
CAN     tx: 75840 rx: 81818 txe: 00000 rxe: 00000
        drop: 00000 autherr: 00000 frame: 06176
        txb: 3265270 rxb: 3321911
```

ident

Responds with some system info. Hostname, Vendor, Revisions and Timestamp of build. Using ident on a broadcast node or global broadcast (16383) can be used as a csp scan to find all devices within the local network.

```
host 6  ident
IDENT 6
   obc-hk
   FLASH-1
   v1.2-1-gd958d8e+
   Mar 17 2023 12:18:08
host obc-hk@6  ident 127
IDENT 107
   lenovo
   #36-22.04.1-Ubuntu SMP PREEMP
   5.19.0-35-generic
   Mar 22 2023 14:17:59

IDENT 89
   lab
   #66-Ubuntu SMP Fri Jan 20 14:
   5.15.0-60-generic
   Oct 26 2022 16:23:29
```

uptime

Responds with the system uptime.

```
host 6  uptime
Uptime of node 6 is 10 s
```

buffree

Request the number of remaining CSP buffers on a node.

```
host 6  buffree
Free buffers at node 6 is 9
```

reboot

Reboot a remote node.

```
host 6  uptime
Uptime of node 6 is 10 s
host 6  reboot
host 6  uptime
Uptime of node 6 is 0 s
```

ping

Send a ping and wait for a response from the target.

```
host 6  ping
Ping node 6 size 1 timeout 1000: Reply in 1 [ms]
```

vmem

List vmem areas on remote node:

```
host 6  vmem
Requesting vmem list from node 6 timeout 1000 version 2
 0: sched 0x31001000 - 4096 typ 8
 1: comma 0x31002000 - 4096 typ 8
2: hk_li 0x31003000 - 20480 typ 8
3: hk_co 0x31000500 - 1280 typ 8
4: stdbu 0x2045f100 - 3584 typ 1
5: fram  0x30000000 - 32768 typ 2
6: fl3   0x580000 - 524288 typ 4
7: fl2   0x500000 - 524288 typ 4
8: fl0   0x404000 - 507904 typ 4
9: csp   0x31000000 - 84 typ 2
10: btldr 0x31000400 - 16 typ 2
```

list add

Add a remote parameter without downloading from the device.

```
host 6  list add -c "FRAM+C" -m "Rt" hk_next_timestamp 154 uint32
```

switch

Reboot into another boot image.

```
host 6  switch 1
Switching to flash 1
Will run this image 1 time
cmd new set
Rebooting.........................
|obc-hk
|FLASH-1
|v1.2-1-gd958d8e+
|Mar 17 2023 12:18:08
```

program

Program a slot, with automatic search for valid binaries in the current working directory. Optionally specify a file with the -f option.

```
host 6  program 0
Setting rdp options: 3 10000 5000 2000 2
node 16
     Requesting VMEM name: fl0...
     Found vmem
              Base address: 0x404000
  |               Size: 507904
Searching for valid binaries
0: ./obc-0.bin

ABOUT TO PROGRAM: ./obc-0.bin

|obc-hk
|FLASH-1
|v1.2-1-gd958d8e+
|Mar 17 2023 12:18:08
host 6  yes
Upload 82664 bytes to node 6 addr 0x404000
.............................. - 6 K
.............................. - 78 K
.............................. - 81 K
Uploaded 82664 bytes in 5.950 s at 13893 Bps
.............................. - 6 K
.............................. - 78 K
.............................. - 81 K
Downloaded 82664 bytes in 4.551 s at 18163 Bps
```

The normal operation of the program command is to upload the entire firmware image to the module and then download it back to the CSH terminal, for bitwise comparison. This can in some circumstances prove to be very time consuming. For this reason, the system can be instructed to use a different approach using a simple CRC-32 checksum calculation on “both sides” of the communication channel. Specifying the -c option on the command line will instruct the CSH client to do a CRC-32 calculation on the firmware file prior to uploading it to the module. When the upload process has completed, the module is instructed to do the same CRC-32 calculation on all the data received and send back the result (only 32-bits) to the CSH client for verification. For this option to succeed, the module has to support the CRC-32 calculation feature, otherwise the program operation will end with a communication error.

sps

Temporarily switch into a specific slot, program another slot and switch into the newly programmed slot. Here we are running sps while in slot 1, then rebooting into slot 0, programming slot 1 finally rebooting into slot 1.

```
host 6  ident
IDENT 6
obc-hk
FLASH-1
v1.2-1-gd958d8e+
Mar 17 2023 12:18:08
host 6  sps 0 1
Setting rdp options: 3 10000 5000 2000 2
  Switching to flash 0
  Will run this image 1 times
cmd new set
  Rebooting.....................................
|obc-hk
|FLASH-1
|v1.2-1-gd958d8e
|Feb 22 2023 13:55:31
Requesting VMEM name: fl1...
Found vmem
               Base address: 0x480000
  |                Size: 524288
Searching for valid binaries
0: ./obc-1.bin
ABOUT TO PROGRAM: ./obc-1.bin
|obc-hk
|FLASH-0
|v1.2-1-gd958d8e
|Feb 22 2023 13:55:31
Upload 82664 bytes to node 6 addr 0x404000
............................... - 6 K
............................... - 78 K
............................... - 81 K
Uploaded 82664 bytes in 5.950 s at 13893 Bps
............................... - 6 K
............................... - 78 K
............................... - 81 K
Downloaded 82664 bytes in 4.551 s at 18163 Bps
Switching to flash 1
Will run this image 1 times
cmd new set
  Rebooting.....................................
|obc-hk
|FLASH-1
|v1.2-1-gd958d8e
|Feb 22 2023 13:55:41
```

stdbuf2

Retrieve the stdout buffer of node and clear it.

```
host 6  stdbuf2
bootmsg: obc-hk Feb 15 2023 08:29:19 slot: 0, cause: SOFT
|Feb 15 2023 08:29:18
```

vts init

Send ADCS q_hat and position parameters to vts timeloop software. Specify the adcs node with -n. Server ip and port can be changed from defaults with -s and -p.

```
host 6  vts init -n 300
Streaming data to VTS at 127.0.0.1:8888
```

apm load

Load a csh apm (addin, plugin, module). Will automatically search in $HOME/.local/lib/csh folder for installed APMs.

```
host 6  apm load
Loaded: /home/user/.local/lib/csh/libcsh_hk.so
```

3.5.1 Housekeeping commands (APM)

The following commands are part of an apm that needs to be installed and then loaded with apm load.

hk retrieve

Pulls data from the housekeeping server. Giving the argument -o will set the hk timeoffset for the selected node. And print the time which can be used to manually set the hk timeoffset with the hk timeoffset command.

```
host 6  hk retrieve
Timestamp 393189
106:212  ext_temp            = 26.185358
host 6  hk retrieve -t 100 -s -N 3
Timestamp 100
106:212  ext_temp            = 25.204676
Timestamp 90
106:212  ext_temp            = 25.145563
Timestamp 80
106:212  ext_temp            = 25.116022
```

hk timeoffset

The housekeeping data's timestamp starts from the satellite epoch (when the hk server first begins pulling data). Using the current unix timestamp minus the current hk timestamp we find: 1679563632 – 395553 = 1679168079

You can set hk timeoffset for multiple hk nodes.

```
host 135  hk timeoffset 1679168079
Setting new hk node 135 EPOCH to 1679168079
host 136  hk timeoffset 1679168079
Setting new hk node 136 EPOCH to 1679168079
```

3.5.2 Cortex CSH (APM)

The following commands are part of an apm that needs to be installed and then loaded with apm load.

csh add leafspace

Adds an MQTT interface to the Leafspace ground segment operators MQTT broker. Prerequisites are to have acquired a set of credentials from Leafspace to gain access to their MQTT broker, and to know the NORADID of the satellite to establish communication with. Here is an example of how to add a MQTT interface thru Leafspace MQTT broker (host: mqtt.leaf.space) to a satellite with NORADID 25667

using <user> and <pass> as credentials. The interface is given a node address of 589. The info command can then be run to inspect the resulting interface: MQTT0-25667.

```
host  589 25667 mqtt.leaf.space
host  info
MQTT0-25667 addr: 589 netmask: 8 dfl: 0
       tx: 00000 rx: 00000 txe: 00000 rxe: 00000
       drop: 00000 autherr: 00000 frame: 00000
       txb: 0 (0B) rxb: 0 (0B)
```

This command offers a set of options for controlling how to set up the communication with the MQTT broker. Use the -h option for more information.

csh add cortex_crt

Adds an interface with a direct TCP/IP socket connection to a Cortex CRT modem at a specific IP address.

csh add cortex_hdr

Adds an interface with a direct TCP/IP socket connection to a Cortex HDR modem at a specific IP address.

3.5.3 Scheduler & Named Commands

This section describes command examples to interact with the scheduler and named commands systems of libparam.

cmd new

Create a new get or set command queue.

```
host  cmd new get example
Initialized new command: example
```

cmd add

Add a get or set command to the current queue.

```
host 2  cmd add param1
cmd new get example
cmd add -n 2 param1
host 2  cmd add param2
cmd new get example
cmd add -n 2 param1
cmd add -n 2 param2
```

cmd run

Run the current command queue.

```
host 2  cmd run
param1              = 42
param2              = 0
```

cmd done

Exit the current command queue.

cmd server upload

Upload the current command queue to the server, with <name>.

```
host 6  cmd server upload example
Command added:
cmd new set set1
cmd add -n 2 param2 21
```

cmd server download

Download a command queue from the server. NOTE: does only work with a set queue.

cmd server list

List all named commands stored at CSP node <server>.

```
host 6  cmd server list
Received list of 1 saved command names:
 1 - example
```

cmd server rm

Remove the named command <name> stored at CSP node <server>. NOTE: Inputting “RMALLCMDS” executes a “remove all” function, clearing all named commands stored at the server.

```
host 6  cmd server rm -a RMALLCMDS
Deleted n commands
```

schedule push

Push the current set queue to a scheduler service running on the CSP node <server>, to be executed at CSP node <host> in <time> seconds (or a unix timestamp in seconds). The latency buffer field is used to limit how late a queue can be executed if something prevents it from being executed at the specified time. Inputting <latency buffer> = 0 disables this feature for that schedule. The response message includes the unique ID of the schedule entry, between 0 and 65534.

```
host 6  schedule push 3 10 1673356393 0
Queue scheduled with id 5:
cmd new set example
cmd add -n 10 param1 42
```

schedule list

Request a list of the schedule currently saved in the scheduler service running at CSP node <server>. If the schedule is long, the response will be split into multiple packets. Each schedule entry in the list includes a unique ID and when it is scheduled for.

schedule show

Request the details of the schedule entry with ID <id> stored at CSP node <server>.

schedule rm

Remove the requested schedule with ID <id> from the CSP node <server>. NOTE: Inputting ID = -1 is a “remove all” command which will clear the entire schedule.

schedule reset

Reset the scheduler service meta-data, i.e. reset the last id variable to start counting new IDs from a different number. NOTE: This can result in non-unique IDs if run on a server with active schedule entries.

schedule cmd

Schedule a named command <name> stored on node ID <server> to be executed at node ID <host> in <time> seconds (or a unix timestamp in seconds). Latency buffer is described under "schedule push".

Example of scheduling a command queue named "example" on <server> 1, to be executed on <host> 8 in 100 seconds with <latency buffer>.

```
host 6  sschedule cmd 1 example 8 100 0
```

Complete scheduler example

This example combines the previous examples such that a new command is created, uploaded to a server, and scheduled for execution.

A CSH session creates a command queue locally, which is then uploaded to an On Board Computer (node 1), where it is then scheduled for execution on module (node 8).

```
host  node 8
host 8  list download
Got param: param1:8[1]
Got param: param2:8[1]
host 8  cmd new set example
host 8  cmd add param1 21
host 8  cmd add param2 42
host 8  cmd server upload -s 1 example
host 8  scheduler cmd 1 example 8 1673356393 0
```