

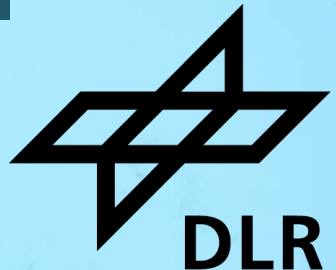
LIBS3: A NOVEL PARAMETER MANAGEMENT SYSTEM FOR DISTRIBUTED SPACE SYSTEMS

Olman Quiros Jimenez

olman.quirosjimenez@dlr.de

German Aerospace Center (DLR)

Institute for Communications and Navigation



Topic: **libs3: A novel Parameter Management System for distributed space systems**

Date: 2025-02-19

Autor: Olman Quiros Jimenez

Email: olman.quirosjimenez@dlr.de

Institute: Communications and Navigation (KN)

Image sources: All images “DLR (CC BY-NC-ND 3.0)” unless otherwise stated

Agenda



- Introduction to Parameter Management System
- The libs3 approach
- The libs3 Parameter Library
- The libs3 Workflow & Tools
- Roadmap & Conclusions

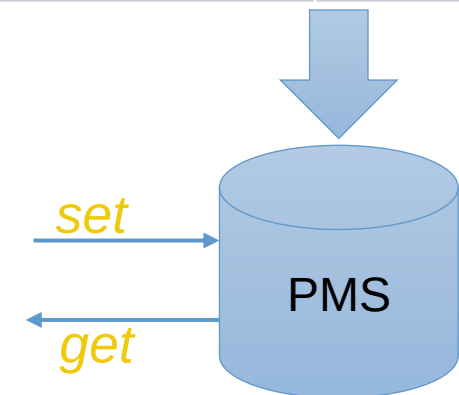
INTRODUCTION

Introduction: Parameter Management System (PMS)



- In the context of flight-software (FSW).
- Key-Value storage service.
- Maps a **name** or **ID** to a **value** = Parameter
- Typically PMS supports all primitive data types (bool, int, float), strings, blobs and arrays.
- PS allows to access (retrieve or modify) the value of Parameters from the outside-world.
 - -> *It's an interface between the outside and the inside of a System.*

Key	Value
name	"libs3"
lat.	14.6051123
long.	90.4907077
temp.	23.07
status	NOMINAL
volt.	12.34
mode	DEPLOY



Introduction: Parameter Management System (PMS)



- Some Examples...

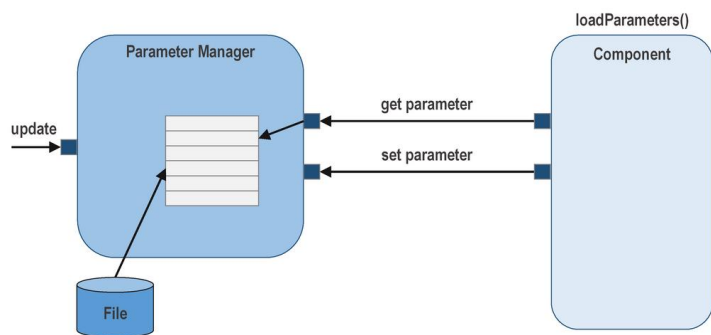
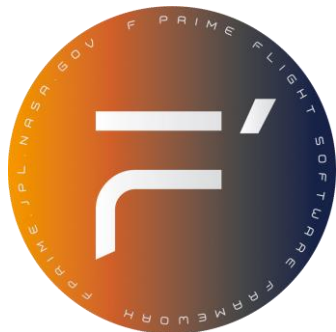
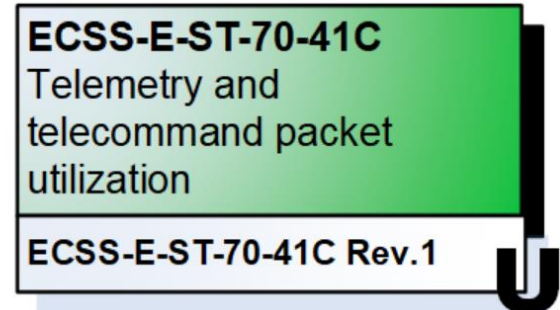


Table Service (TBL)



ST[20] parameter management

Introduction: PMS **limitations**



1. Low user-friendliness:

- Typically parameters are identified by and unique **number**. Space-systems tend to have hundreds even thousands of parameters. Making it hard for humans to remember and operate the system.
- Binary formatted.
- Coarse access.
- Poor tooling and environment.

2. Poor scalability:

- Add thousand of parameters. Try to search for one param...
- Add dozens of nodes. Try to sync or propagate changes...

3. Lack of flexibility:

- Want to add a new parameter or change an interface in orbit? A software update is required.

THE LIBS3 APPROACH

The libs3 approach

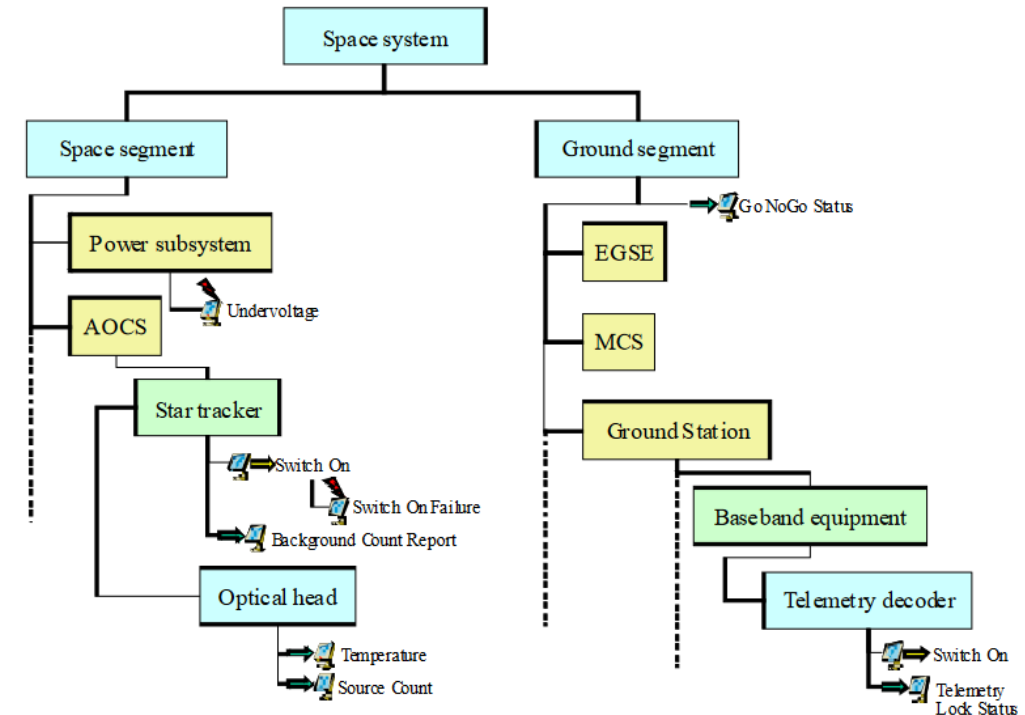


1. Human friendly Names not numbers.
-> Numbers are hard to remember and easy to confuse.
2. Params should be accessible (referenced) independently.
3. For Config and Telemetry Params
 - **Configuration**: determine the behavior of a s system.
 - **Telemetric**: describe the state of a system.

The libs3 approach

4. Flight-software should reflect the system's nature

- How to tackle complexity in space systems?
- Divide and Conquer -> Recursive decomposition.
- Inspiration from The Space System Model (ECSS)
 - representation of the space system in terms of its decomposition into system elements
- The libs3 approach: the **flight-software** shall reflect the **hierarchical** nature of the system.



Source: ECSS-E-ST-70-32C

THE LIBS3 PARAMETER SYSTEM

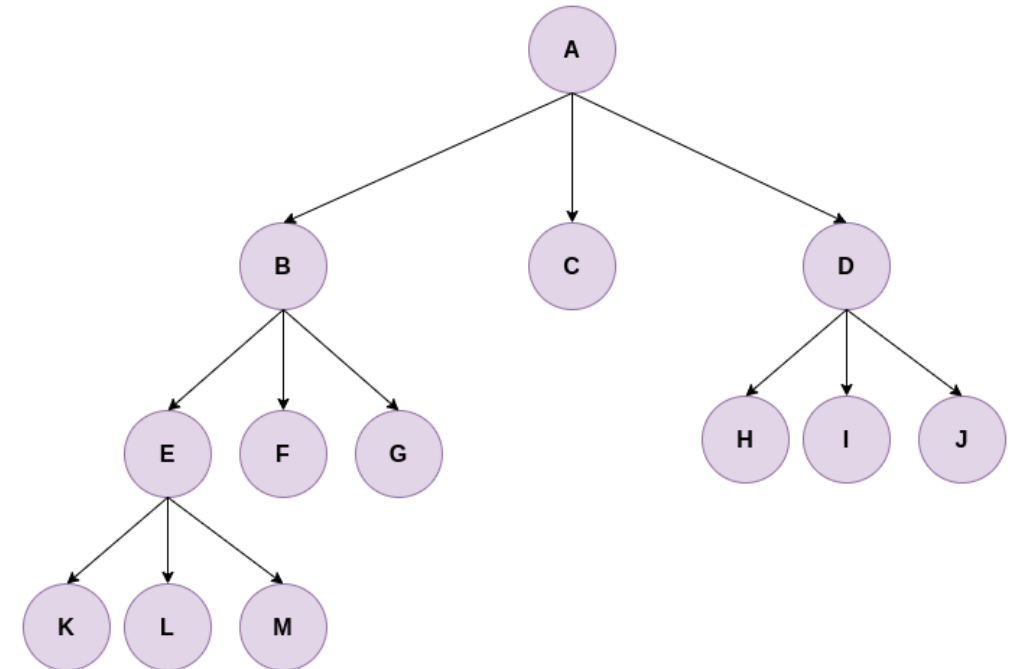
The libs3 Parameter System: **Systems**



Systems `Sys`

- Are the nodes of the tree.
- Have a
 - **Name**.
 - Can have a parent **Sys**
- Can have none or multiple subsystems.

```
Sys my_sys      ("sys");      // top-level sys
Sys my_sub_sys  (my_sys, "sub"); // a subsystem
```



The libs3 Parameter System: Parameters



Parameters `Param`

- Are a key-value pair.
- Are the properties of a **Sys**.
 - A **Sys** can have none or multiple **Params**.
- Params have:
 - A parent **Sys**
 - A **Name**
 - A **Value** of specific type (u8, i8, u16, i16, u32, i32, u64, i64, f32, f64, bool, array of any of the previous, str, blob, date).

```
Sys          my_sys ("sys");
//          Parent,  name,  default value
Param<bool>   my_bool(my_sys,  "prm1",  true);
Param<uint8_t> my_int8(my_sys,  "prm2",  -1);
Param<uint32_t> my_uint32(my_sys, "prm3",  3000);
Param<float>   my_float(my_sys,  "prm4",  0.420);
ParamStr<32>   my_str(my_sys,   "prm5",  "Hello Universe!");
ParamVec<float,3> my_vec(my_sys,  "prm6",  {1.0, 2.0, 3.0})
```

The libs3 Parameter System



- Building the hierarchy.

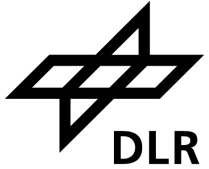
```
Sys          sys          ("sys");
Param<uint8_t> sys_Mode    (sys,      "mode");
Param<float>   sys_MCU_Temp (sys,      "temp");
Param<uint8_t> sys_Voltage (sys,      "vlt");

Sys          sys_Ch1      (sys,      "ch1");
Param<uint8_t> sys_Ch1_Enabled (sys_Ch1, "ena");
Param<uint8_t> sys_Ch1_Error  (sys_Ch1, "err");

Sys          sys_Ch1_ON   (sys_Ch1,  "on");
Param<uint8_t> sys_Ch1_ON_Cmd (sys_Ch1_ON, "cmd");
Param<uint8_t> sys_Ch1_ON_State (sys_Ch1_ON, "stat");
```

```
.
├─ sys/
│   ├── mode
│   ├── temp
│   ├── vlt
│   └─ ch1/
│       ├── ena
│       └─ err
└─ on/
    ├── cmd
    └─ stat
```

The libs3 Parameter System



- Basic utilization.

```
Sys          sys          ("sys");
Param<uint8_t> sys_Mode    (sys,      "mode");
Param<float>   sys_MCU_Temp (sys,      "temp");
Param<uint8_t> sys_Voltage (sys,      "vlt");

Sys          sys_Ch1      (sys,      "ch1");
Param<uint8_t> sys_Ch1_Enabled (sys_Ch1, "ena");
Param<uint8_t> sys_Ch1_Error  (sys_Ch1, "err");

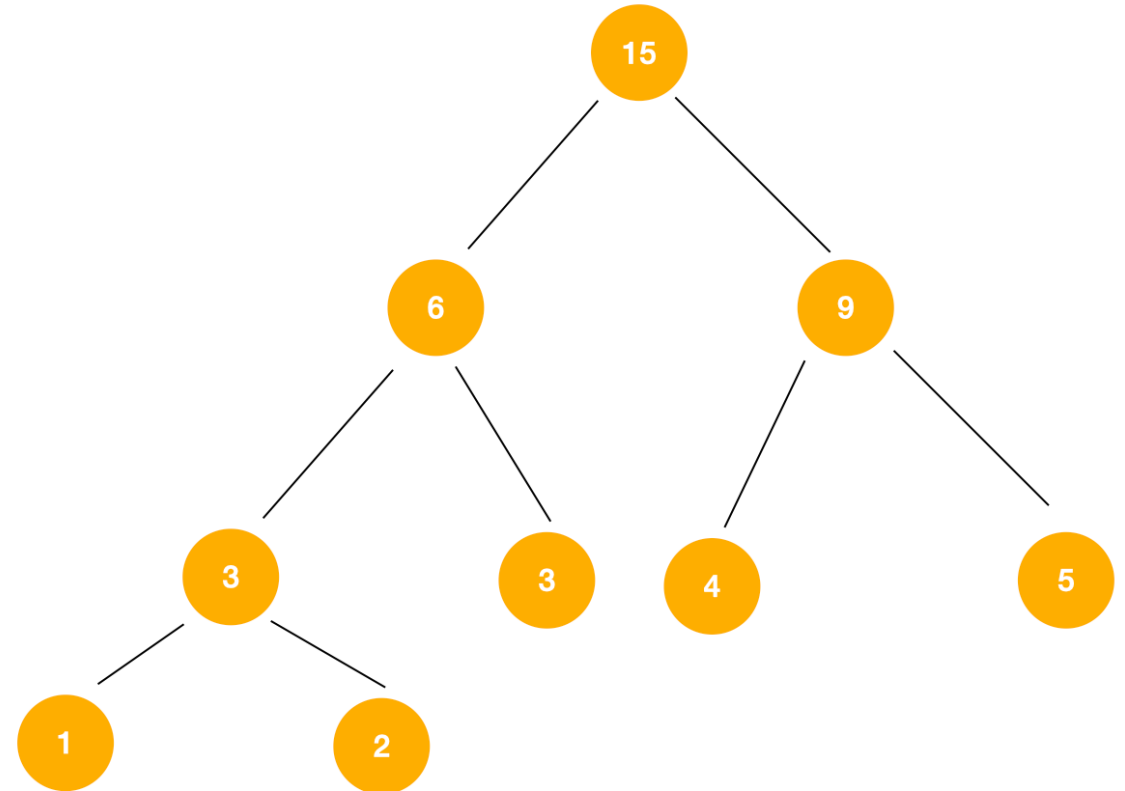
Sys          sys_Ch1_ON    (sys_Ch1,  "on");
Param<uint8_t> sys_Ch1_ON_Cmd (sys_Ch1_ON, "cmd");
Param<uint8_t> sys_Ch1_ON_State (sys_Ch1_ON, "stat");
```

```
vod my_app_task(){
    while(1){
        if(Sys_Ch1_Enabled.get() == false){
            continue;
        }
        if(Sys_Ch1_ON_Cmd.get() == true){
            // do things ...
            Sys_Ch1_ON_State.set(1);
        }
    }
}
```

The libs3 Parameter System: Indexing



- Segment Tree
- **Sys** receives an **index** based on its positions in the tree.
- **Params** receives index based on its positions in the **Sys**.
- N-ary Tree search is $O(\log(n))$.
- A lookup array for **O(1)** access can be generated.



The libs3 Parameter System: Groups



- **Sys** and **Params** can be grouped by a reference into different logical groups.
- Handy implementing interfaces, telemetry, tele-commands, logging, etc.

Interface definition

```
Group<8> param_list( "telemetry",{
    &Sys_Mode,
    &Sys_MCU_Temp,
    &Sys_Ch1_Status,
    &Sys_Ch1_Temp,
    &Sys_Ch2_Status,
    &Sys_Ch2_Temp,
    &Sys_Ch3_Status,
    &Sys_Ch3_Temp
});
```

Transmitter

```
int my_app(){
    // ...
    uint8_t buf[256];
    // Instantiate your Serializer
    Binary binary_serde();
    // Serialize param group into buffer
    binary_serde.serialize(param_list,
        buf,
        buf_size);
    // Transmit the data
    send(&buf, 256);
    // ...
}
```

Receiver

```
int my_app(){
    // ...
    uint8_t buf[256];
    // Receive command arguments
    rcv(&buf, 256);
    // Instantiate your Deserializer
    Binary binary_serde();
    // Deserialize params from data in buffer
    binary_serde.deserialize(param_list,
        buf,
        buf_size);
    // ...
}
```

The libs3 Parameter System

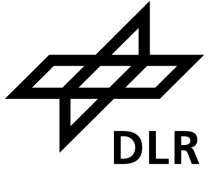


More Features

- **Callbacks**: invoke function when Param's value is modified.
- **Search** functions for Sys and Params (by path and index).
- To/from **string** conversion.
- **Serialization and Deserialization** (binary, CSV, Json, Yaml, etc).

THE LIBS3 WORKFLOW & TOOLS

The libs3 ecosystem



C++
Implementation



Python
Implementation



s3 Model
Transpiler



Command Line
Interface

The s3 model



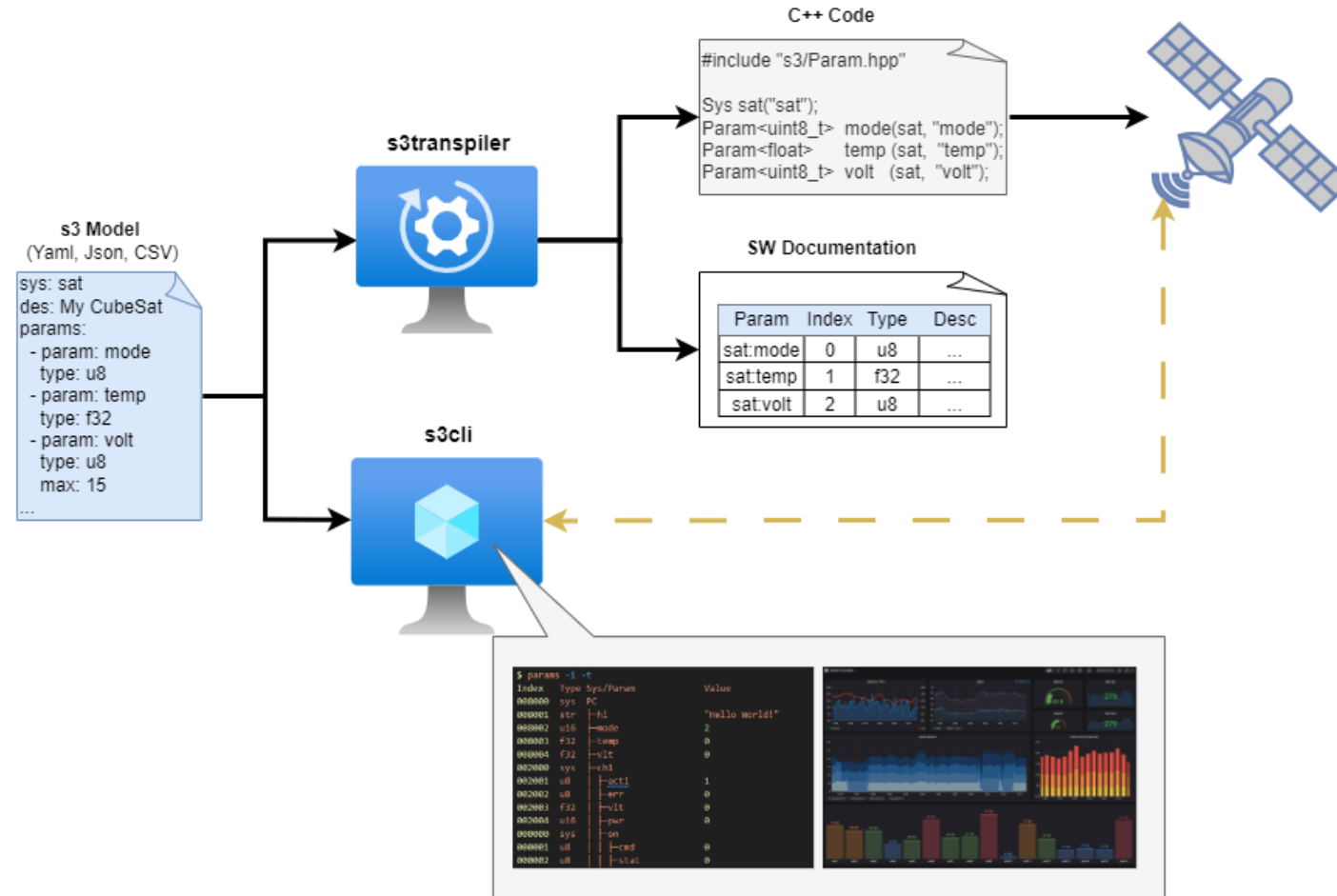
■ Yaml/Json

```
sys: obc
desc: Top level example system.
params:
  - param: time
    desc: Timestamp
    type: u32
    groups: ["telemetry", "since", "health"]
  - param: mode
    desc: Mode of operation
    type: u8
    default: 1
    access: RW
    groups: ["telemetry"]
subsys:
  - sys: app
    desc: Example application
    params:
      - param: 'on'
        type: bool
        default: True
        access: RW
      - param: flag
        type: bool
        len: 4
        default: [True, True, False, False]
        groups: ["telemetry"]
      - param: hi
        desc: Greetings
        type: str
        len: 20
        default: "Hello World!"
```

Add other sys to the tree

```
- !include: obc.yaml
- !include: csp.yaml
- !include: adcs.yaml
- !include: eps.yaml
- !include: comms.yaml
```

The s3 transpiler



The libs3 Python implementation



```
# 1. Use the ParamDB to load the s3 Model.
obc = s3.ParamDB.spawn_from_dict(S3Transpiler("obc.yaml"))

# 2. Set UDP Getaway to the system
udp_getaway = UDPGetaway("127.0.0.1", 7001)
obc.set_getaway(udp_getaway)

# 3. Get the value of the parameter obc.mode.
#    In the background, s3 will use UDP to send the request to the obc board.
mode = obc["mode"].get()
print(f"obc.mode is {mode}")

# 4. Change the value of obc.mode to 3.
#    As in the previous step, s3 will use UDP to send the request to the obc board.
obc["mode"].set(3)
```

The libs3 Command Line Interface



Navigation

- CLIs for debugging, testing and operating libs3 applications.

Print the tree of Sys and Params

```
$ params -i -t
Index  Type Sys/Param      Value
008000 sys  PC
008001 str  |hi              "Hello World!"
008002 u16  |mode            2
008003 f32  |temp            0
008004 f32  |vlt             0
002000 sys  |ch1
002001 u8   | |acti          1
002002 u8   | |err           0
002003 f32  | |vlt           0
002004 u16  | |pwr           0
000000 sys  |on
000001 u8   | |cmd           0
000002 u8   | |stat          0
001000 sys  |cur
001001 u16  | |cur           0
001002 u16  | |curL          500
007000 sys  |temp
003000 sys  |grp0
```

```
$ sys
PC

$ cs PC

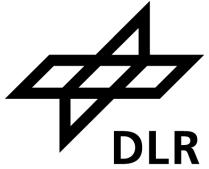
$ sys
ch1
ch2
ch3
ch4

$ cs ch1

$ params
Sys/Param      Value
ch1
|acti          1
|err           0
|vlt           0
|pwr           0

$ set acti 1
```


The libs3 Command Line Interface



■ Queries

```
# get all params named 'volt'
$ query *:vlt
Name      Value
PC:
├ch1
│  └vlt:    46
├ch2
│  └vlt:    46
├ch3
│  └vlt:    46
└ch4
  └vlt:    46
```

```
# get all sys named 'on' from a 'ch*' sys
$ query -p ch*:cmd:
Name      Value
PC:
├mode:    1
├temp:    37
├ch1
│  └on
│    └cmd:  1
│    └st:   1
├ch2
│  └on
│    └cmd:  1
│    └st:   1
├ch3
│  └on
│    └cmd:  1
│    └st:   1
└ch4
  └on
    └cmd:  1
    └st:   1
```

```
# get all 'volt' params which value exceeds a threshold
$ query *:vlt > 15
Name      Value
PC:
├ch1
│  └vlt:    46
├ch2
│  └vlt:    46
├ch3
│  └vlt:    46
└ch4
  └vlt:    46
```

ROADMAP & CONCLUSIONS

Roadmap & Conclusions



- Access control
- Thread-safety model
- Standardized trees
- Middleware leveraging the tree structure

What do you think? Feedback is appreciated!

olman.quirosjimenez@dlr.de

END

Topic: **libs3: A novel Parameter Management System for distributed space systems**

Date: 2025-02-19

Autor: Olman Quiros Jimenez

Email: olman.quirosjimenez@dlr.de

Institute: Communications and Navigation (KN)