17/04/2018

Specification REST – Project REW

Summary

System information request

This API is used to obtain the current system information.

Operation: **GET**
URI: **/services/v1/sys/info**

The request does not have any parameter. The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
pn	String	Part number
sn	String	Serial number
manufacturer	Number	ID device manufacturer (#1)
model	Number	ID device model (#1)
ostype	Number	ID Type operating system (#1)
osversion	Number	Operating system version (#1, #2)
firmwtype	Number	ID firmware type (#1)
firmwversion	Number	Firmware version (#1, #2)
usertext	String	User software name (#1)
userversion	Number	User software name (#1, #2)
attrib	Number	Device attributes (#1)
tbtype	Number	ID taskbin type (#1)
tbversion	Number	Taskbin version (#1, #2)
varsetid	Number	ID variables set (#1)
slibtype	Number	ID system library type (#1)
slibversion	Number	System library version (#1, #2)
biostype	Number	ID BIOS type (#1)
biosversion	Number	BIOS version (#1, #2)
firmwext	Number	Firmware extension (#1)
rpeversion	Number	RPE version (#1, #2)
language	Number	ID device language (#1)

Note (#1): for the field meaning consult the RTE documentation for BCC/31 protocol, command bccSysInfo.
Note (#2): the version is in the standard nvMake Robox SpA format.

Quick example of use:

```
GET /services/v1/sysinfo HTTP/1.1
{
“result” : 0,
“pn”: “AS6011.001”,
“sn”: “A100070”,
“manufacter”: 1,
“model”: 35,
```

```
“ostype”: 4,
“osversion”: 16842752,   // v1.1.0
…
“language”: 16
}
```

System state request

This API is used to obtain the current system state.

Operation: **GET**
URI: **/services/v1/sys/state**

The request does not have any parameter. The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
varsetid	Number	ID variables set
alsmask	Number	Alarms mask in stack
mode	Number	Operating mode: 0 = Initializing system 1 = 0 Cycle 2 = Programming 3 = Execution 4 = User defined 0 cycle 5 = Other

Quick example of use:

```
GET /services/v1/sysinfo HTTP/1.1
{
“result” : 0,
“varsetid”: 560561132,
“alscount: 0,
“mode”: 3
}
```

Read register integer 32bit volatile

This API is used to read the values of one or more consecutive integer 32bit volatile registers.

Operation: **GET**
URI: **/services/v1/global/vr32/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting register index
count	Number	Number of consecutive registers (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1) (#2)	Array[Number]	Array of values (I32)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 10 consecutive register, from index 1:

```
GET /services/v1/global/vr32/1/10 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
}
```

Read register real 64bit volatile

This API is used to read the values of one or more consecutive real 64bit volatile registers.

Operation: **GET**
URI: **/services/v1/global/vrr/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting register index
count	Number	Number of consecutive registers (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1) (#2)	Array[Number]	Array of values (double)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 5 consecutive register, from index 10:

```
GET /services/v1/global/vrr/10/5 HTTP/1.1
{
"result" : 0,
"values" : [0.001, -0.002, 1e10, 1000.1234, 0]
}
```

Read register string volatile

This API is used to read the values of one or more consecutive volatile string registers.

Operation: **GET**
URI: **/services/v1/global/vsr/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting register index
count	Number	Number of consecutive registers (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1) (#2)	Array[Number]	Array of values (string)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 consecutive registers, from index 5:

```
GET /services/v1/global/vsr/5/3 HTTP/1.1
{
"result" : 0,
"values" : ["registro5", "registro6", ""]
}
```

Read register integer 32bit non-volatile

This API is used to read the values of one or more consecutive integer 32bit non-volatile registers.

Operation: **GET**
URI: **/services/v1/global/nvr32/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting register index
count	Number	Number of consecutive registers (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1) (#2)	Array[Number]	Array of values (I32)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 1 register, from index 5:

```
GET /services/v1/global/nvr32/1 HTTP/1.1
{
"result" : 0,
"values" : [0]
}
```

Read register real 64bit non-volatile

This API is used to read the values of one or more consecutive real 64bit non-volatile registers.

Operation: **GET**
URI: **/services/v1/global/nvrr/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting register index
count	Number	Number of consecutive registers (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (double)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 5 consecutive register, from index 10:

```
GET /services/v1/global/nvrr/10/5 HTTP/1.1
{
"result" : 0,
"values" : [0.001, -0.002, 1e10, 1000.1234, 0]
}
```

Read register string non-volatile

This API is used to read the values of one or more consecutive non-volatile string registers.

Operation: **GET**
URI: **/services/v1/global/nvsr/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting register index
count	Number	Number of consecutive registers (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (string)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 consecutive registers, from index 5:

```
GET /services/v1/global/nvsr/5/3 HTTP/1.1
{
“result” : 0,
“values” : [“hello”, “”, “world!"]
}
```

Read logical input word 16bit

This API is used to read one or more consecutive logical input word 16bit.

Operation: **GET**
URI: **/services/v1/global/logiw16/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting word index
count	Number	Number of consecutive word (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (U16)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 5 logical input word 16bit, from index 1:

```
GET /services/v1/global/logiw16/1/5 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 2, 3, 4]
}
```

Read logical output word 16bit

This API is used to read one or more consecutive logical output word 16bit.

Operation: **GET**
URI: **/services/v1/global/logow16/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting word index
count	Number	Number of consecutive word (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (U16)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 5 logical output word 16bit, from index 1:

```
GET /services/v1/global/logow16/1/5 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 2, 3, 4]
}
```

Read physical input word 16bit

This API is used to read one or more consecutive physical input word 16bit.

Operation: **GET**
URI: **/services/v1/global/phyiw16/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting word index
count	Number	Number of consecutive word (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (U16)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 1 physical input word 16bit, from index 1:

```
GET /services/v1/global/phyiw16/1 HTTP/1.1
{
"result" : 0,
"values" : [0]
}
```

Read physical output word 16bit

This API is used to read one or more consecutive physical output word 16bit.

Operation: **GET**
URI: **/services/v1/global/phyow16/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting word index
count	Number	Number of consecutive word (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (U16)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 5 physical output word 16bit, from index 1:

```
GET /services/v1/global/phyow16/1/5 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 2, 3, 4]
}
```

Read logical input channel

This API is used to read one or more consecutive logical input channels.

Operation: **GET**
URI: **/services/v1/global/logic/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting channel index
count	Number	Number of consecutive channels (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (BOOL)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 logical input channels, from index 1:

```
GET /services/v1/global/logic/1/3 HTTP/1.1
{
"result" : 0,
"values" : [0, 0, 1]
}
```

Read logical output channel

This API is used to read one or more consecutive logical output channels.

Operation: **GET**
URI: **/services/v1/global/logoc/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting channel index
count	Number	Number of consecutive channels (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (BOOL)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 logical output channels, from index 1:

```
GET /services/v1/global/logoc/1/3 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 0]
}
```

Read physical input channel

This API is used to read one or more consecutive physical input channels.

Operation: **GET**
URI: **/services/v1/global/phyic/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting channel index
count	Number	Number of consecutive channels (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (BOOL)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 physical input channels, from index 1:

```
GET /services/v1/global/phyic/1/3 HTTP/1.1
{
"result" : 0,
"values" : [0, 0, 1]
}
```

Read physical output channel

This API is used to read one or more consecutive physical output channels.

Operation: **GET**
URI: **/services/v1/global/phyoc/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting channel index
count	Number	Number of consecutive channels (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (BOOL)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 physical output channels, from index 1:

```
GET /services/v1/global/phyoc/1/3 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 0]
}
```

Read Parameter integer 32bit

This API is used to read one or more consecutive integer 32bit parameters.

Operation: **GET**
URI: **/services/v1/global/pi32/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting Parameter index
count	Number	Number of consecutive parameters (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (I32)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 10 consecutive parameters, from index 1:

```
GET /services/v1/global/pi32/1/10 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
}
```

Read Parameter real 64bit

This API is used to read one or more consecutive real 64bit parameters.

Operation: **GET**
URI: **/services/v1/global/pr/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting Parameter index
count	Number	Number of consecutive parameters (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (double)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 5 consecutive parameters, from index 1:

```
GET /services/v1/global/pr/1/5 HTTP/1.1
{
"result" : 0,
"values" : [0.0, 10, -200.99, 1e10, -0.000001]
}
```

Read Axis parameter integer 32bit

This API is used to read one or more consecutive integer 32bit axis parameter.

Operation: **GET**
URI: **/services/v1/global/api32/<index>/<aindex>[/<acount>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	Parameter index
aindex	Number	Starting axis index
acount	Number	Number of consecutive axes (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *aindex* not valid -3 = Parameter a*count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (I32)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read Parameter 23 for axes 1 to 6:

```
GET /services/v1/global/api32/23/1/6 HTTP/1.1
{
"result" : 0,
"values" : [0, 1, 2, 3, 4, 5]
}
```

Lettura Parameter asse reale 64bit

This API is used to read one or more consecutive parametri asse reali 64bit consecutivi.

Operation: **GET**
URI: **/services/v1/global/pr32/<index>/<aindex>[/<acount>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	Parameter index
aindex	Number	Starting axis index
acount	Number	Number of consecutive axes (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *aindex* not valid -3 = Parameter a*count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (double)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read Parameter 7 for axis 5

```
GET /services/v1/global/apr/7/5 HTTP/1.1
{
"result" : 0,
"values" : [-0.0051]
}
```

Read axes state.

This API is used to get information on the axes state.

Operation: **GET**
URI: **/services/v1/axes/state/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
index	Number	Starting axis index
count	Number	Number of consecutive elements (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: < 0 = error in the request
values	Array[AxisInfo]	Array of objects AxisInfo

The structure *AxisInfo* contains the following data:

Field	Type	Details
state	Number	State mask (U32): 0x1 = enabled axis 0x2 = axis not present 0x4 = emulated axis 0x8 = axis with transductor 0x10 = powered axis 0x20 = axis in alarm
ip	Number	Ideal position (Double)
cp	Number	Current position (Double)
epos	Number	Positioning error (Double)
iv	Number	Ideal Speed (Double)
cv	Number	Current speed (Double)
ia	Number	Ideal acceleration (Double)
ca	Number	Current acceleration (Double)
ser_cthr	Number	Servo alarm threshold (Double)
rawcp	Number	Transducer read value (Double)

Example to read the axes state info

```
GET /services/v1/axes/state/1/2 HTTP/1.1
{
"result": 0,
"values": [
    {
```

```
        “state”: 3,
        “ip”: 0.1,
        “cp”: 0.11,
        “epos”: 0.02,
        “iv”: 0.3,
        “cv”: 0.33,
        “ia”: 0.4
        “iv”: 0.44,
        “ser_cthr”: 0.5,
        “rawcp”: 0.6
    },{
        “state”: 7,
        “ip”: 0.7,
        “cp”: 0.77,
        “epos”: 0.08,
        “iv”: 0.9,
        “cv”: 0.99,
        “ia”: 0.01
        “iv”: 0.011,
        “ser_cthr”: 0.02,
        “rawcp”: 0.03
    }]
}
```

Read alarms mask

This API is used to read one or more consecutive alarm mask.

Operation: **GET**
URI: **/services/v1/global/am/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	Starting mask index
count	Number	Number of consecutive masks (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid < 0 = error in the request
values (#1)(#2)	Array[Number]	Array of values (U32)

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): the number of received values might be smaller than the number of values requested.

Example to read 3 alarm masks, from index 1:

```
GET /services/v1/global/am/1/3 HTTP/1.1
{
"result" : 0,
"values" : [16384, 3, 512]
}
```

Read alarms stack

This API is used to read one or more consecutive elements of the alarms stack.

Operation: **GET**
URI: **/services/v1/alarms/stack/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	Starting index in the stack
count	Number	Number of consecutive elements (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success -1 = Parameter *index* not valid -2 = Parameter *count* not valid -3 = Error reading alarms stack < -3 = error in the request
values (#1)(#3)	Array[AlarmInfo]	Array of values (U32)
mask	Number	Alarm mask in stack **(#2)**.

Note (#1): this fields are sent only if field result equal to 0.
Note (#2): alsmask field is independent from the number of requested items.
Note (#3): the number of received values might be smaller than the number of values requested.

The structure *AlarmInfo* contains the following data:

Field	Type	Details
code	Number	Alarm code (U32)
text	String	Alarm details

Example to read alarms stack codes:

```
GET /services/v1/global/alarms/stack/1/2 HTTP/1.1
{
"result" : 0,
"values" : [
  {
      "code": 100,
      "text": "Allarme 100"
   },{
      "code": 0,
      "text": ""
   }],
"mask": 1 }
```

Read alarm history info

This API is used to get information about the alarm history.

Operation: **GET**
URI: **/services/v1/alarms/info**

The request does not need any parameter. The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
old_id	Number	ID of the older alarm
new_id	Number	ID of the most recent alarm
count	Number	Number of existing alarms
size	Number	Alarm history max dimension

Example to read the info about the alarm history:

```
GET /services/v1/global/alarms/info HTTP/1.1
{
“result” : 0,
“old_id”: 0,
“new_id”: 99,
“count”: 100,
“size”: 200
}
```

Read the alarm history

This API is used to read the alarm history.

Operation: **GET**
URI: **/services/v1/alarms/history/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	ID of the starting alarm report
count	Number	Number of consecutive elements (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
values (#1)	Array[AlarmReport]	Array of objects AlarmReport (from older to newer)

Note (#1): the number of received values might be smaller than the number of values requested.

The structure of *AlarmReport* contains the following data:

Field	Type	Details
id	Number	ID alarm
code	Number	Alarm code
time	String	String containing time information of the report: YYYY-MO-DD**T**HH:MM:SS**Z** YYYY = year MO = month DD = day **T** = separator HH = hours MM = minutes SS = seconds **Z** = closing separator
text	String	Details of the alarm

Example to read the alarm history:

```
GET /services/v1/alarms/history/1/2 HTTP/1.1
{
"result" : 0,
"values":[
    {
        "id": 1,
        "code": 99,
```

```
        “time”: “2017-04-11T10:34:11Z”,
        “text”: “”
    },{
        “id”: 2,
        “code”: 99,
        “time”: “2017-04-11T10:36:11Z”,
        “text”: “Generic Alarm”
]}
```

Read user report info

This API is used to get information about the user report.

Operation: **GET**
URI: **/services/v1/reports/usr/info**

The request does not need any parameter. The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
old_id	Number	ID older report
new_id	Number	ID most recent report
count	Number	Current number of reports
size	Number	Maximum number of reports
repid	Number	ID of the report info

The resulting IDs are consecutive. For the ID following the one in position 0xFFFFFFFF, will be used ID 0x00000000 next, and so on every time the maximum is reached.

Example to get the user report information:

```
GET /services/v1/reports/usr/info HTTP/1.1
{
“result” : 0,
“old_id”: 0,
“new_id”: 99,
“count”: 100,
“size”: 2048,
“repid”: 15
}
```

Read user report log

This API is used to read the user report log.

Operation: **GET**
URI: **/services/v1/reports/usr/list/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	Starting report index (consecutive)
count	Number	Number of consecutive elements (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
values	Array[UsrReport]	Array of objects UsrReport (from older to newer)

The structure *UsrReport* contains the following data:

Field	Type	Details
id	Number	ID usr report (U32)
time	Number	Time of report generation (Double)
mask	Number	Source mask (U16): 0x0000 Generic 0x0001 RTE/RRT 0x0002 OS 0x0004 Fieldbus 0x0008 (reserved) 0x0010 (reserved) 0x00E0 Item category: 0x00 generic 0x20 information 0x40 warning 0x80 fault/emergency 0x0100 (user) 0x0200 (user) 0x0400 (user) 0x0800 (user) 0x1000 (user) 0x2000 (user) 0x4000 (user) 0x8000 (user)
text	String	Report text

Example to read the user report log:

```
GET /services/v1/reports/usr/list/1/2 HTTP/1.1
{
“result” : 0,
“values”:[
    {
        “id”: 1,
        “time”:20156852058621,
        “mask”: 1,
        “text”: “User Report”
    },{
        “id”: 2,
        “time”:20156852068621,
        “mask”: 2,
        “text”: “User Report”
]}
```

Read system report info

This API is used to get information about the system report.

Operation: **GET**
URI: **/services/v1/reports/sys/info**

The request does not need any parameter. The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
old_id	Number	ID older report
new_id	Number	ID most recent report
count	Number	Current number of reports
size	Number	Maximum number of reports
repid	Number	ID of the report info

The resulting IDs are consecutive. For the ID following the one in position 0xFFFFFFFF, will be used ID 0x00000000 next, and so on every time the maximum is reached.

Example to get the system report information:

```
GET /services/v1/reports/sys/info HTTP/1.1
{
“result” : 0,
“old_id”: 0,
“new_id”: 99,
“count”: 100,
“size”: 2048,
“repid”: 15
}
```

Read system report log

This API is used to the system report log.

Operation: **GET**
URI: **/services/v1/reports/sys/list/<index>[/<count>]**

The request needs the following parameters:

Parameter	Type	Details
Index	Number	Starting report index (consecutive)
count	Number	Number of consecutive elements (optional, default 1)

The reply gives the following data:

Field	Type	Details
result	Number	Request result: 0 = success < 0 = error in the request
values	Array[SysReport]	Array of objects SysReport (from older to newer)

The structure *SysReport* contains the following data:

Field	Type	Details
id	Number	ID system report (U32)
time	Number	Time of report generation (Double)
mask	Number	Source mask (U16): 0x0000 Generic 0x0001 RTE/RRT 0x0002 OS 0x0004 Fieldbus 0x0008 (reserved) 0x0010 (reserved) 0x00E0 Item category: 0x00 generic 0x20 information 0x40 warning 0x80 fault/emergency 0x0100 (user) 0x0200 (user) 0x0400 (user) 0x0800 (user) 0x1000 (user) 0x2000 (user) 0x4000 (user) 0x8000 (user)
text	String	Report text

Example to read the system report log:

```
GET /services/v1/reports/sys/list/1/2 HTTP/1.1
{
“result” : 0,
“values”:[
    {
        “id”: 1,
        “time”:20156852058621,
        “mask”: 1,
        “text”: “System Report”
    },{
        “id”: 2,
        “time”:20156852068621,
        “mask”: 2,
        “text”: “System Report”
]}
```