Application Note

INSTRUMENT HEALTH & UTILIZATION MONITORING

Frederic Schütze | GFM336 | Version 0e | 01.2021

https://www.rohde-schwarz.com/appnote/GFM336

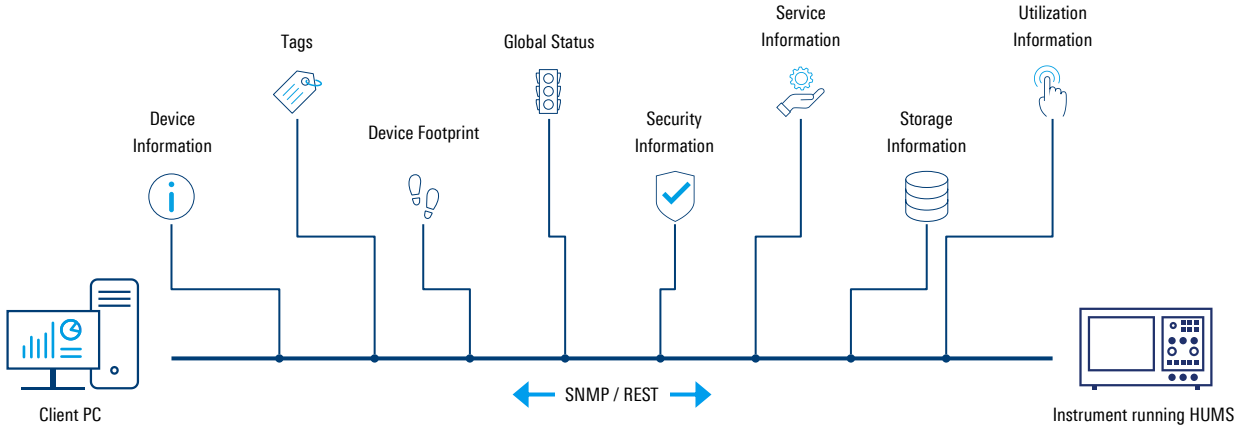
Contents

Overview

In this day and age of internet of things, more and more devices are connected to the local network, making it even more difficult for the IT department to monitor them. Rohde & Schwarz instruments are also increasingly accessed via their LAN interface, which provides additional convenience features such as remote desktop, SMB file transfer or a web interface. To make it easier to monitor the use and status of instruments, some devices offer a software option, the *Health and Utilization Monitoring Service (HUMS)*.

HUMS can be accessed via SNMP and HTTP and provides all necessary information about the health status and utilization over time in many details.

This application note describes how to access HUMS and what data is provided.



1 Introduction

HUMS is a software option that is automatically integrated with some devices or can be purchased additionally. The software runs as a service in the background on the device operating system and communicates with the operating system (OS) and the device firmware (see Figure 1). For control, HUMS opens a REST and an SNMP interface to read the monitoring and utilization data.

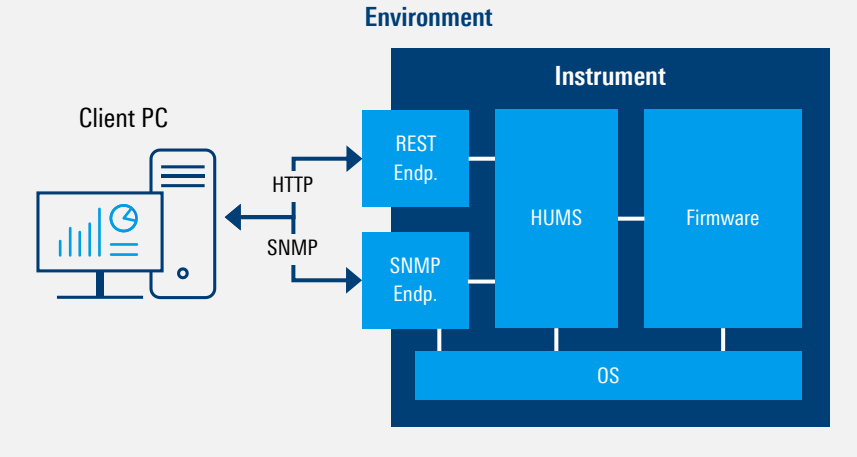


Figure 1 Remote Control Setup

Chapter 2 gives a short overview of the two technologies SNMP and REST with which the HUMS data can be retrieved and settings can be changed.

In chapter 3 the actual content of the HUMS data is explained in detail. It is shown how the data can be retrieved and in which format it is located. Simple examples of SNMP and REST are also included.

2 Protocols

The HUMS APIs can be accessed via two different protocols: SNMP and REST(HTTP).

2.1 SNMP

In today's complex network of routers, switches, and servers, it can seem like a daunting task to manage all the devices on your network and make sure they're not only up and running but performing optimally. This is where the *Simple Network Management Protocol (SNMP)* can help. SNMP was introduced in 1988 to meet the growing need for a standard for managing Internet Protocol (IP) devices. SNMP provides its users with a "simple" set of operations that allows these devices to be managed remotely. [1]

More and more measuring devices can be integrated into the IT infrastructure via ethernet, so it makes sense to monitor these devices as well and SNMP is a widely used standard, which is supported by many network monitoring software.

SNMP queries and manipulates data in the form of variables arranged in a tree structure. Each variable has an address that describes where it is located in the tree, this is called object identifier (OID). An OID is described with a sequence of numbers separated by a dot, e.g. "1.3.6.1.4.1.2566.125.2.1.2.6.2.1". The length of the row represents how deep you are in the tree and each number represents the index in the respective node. Figure 2 shown an example if the OIDs for a defined tree.

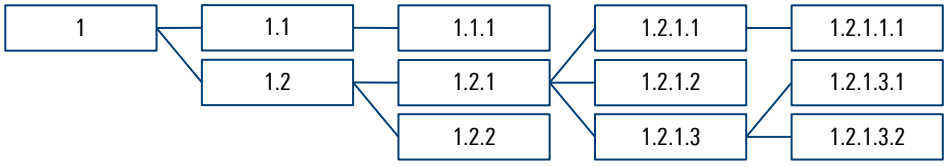


Figure 2 Example for OIDs for specific nodes

An OID has a similar format to an IP address, so it has the same problem that it is too meaningless to be remembered or understood. Therefore, as with IP addresses, there is a name resolution, but this is not done by a DNS but by so-called *Management Organization Base (MIB)*. In this case, the necessary name resolutions in the form of MIB files are provided by the device and can be loaded as an archive via the following interface:

http://<device-address>/api/hums/v1/documents?name=snmp

The SNMP interface can be accessed via SNMPv1 or SNMPv2c. For authentication, a community string is also required. The community string is like a password and can be changed in the instrument specific settings (the default value is the device serial number).

In the following chapter all variables provided by HUMS are described and examples are provided for each module. The examples require a UNIX environment (LINUX, macOS or WSL) and the package NET-SNMP, which is already installed on many systems. If it is not installed, it can be installed via the respective package manager (e.g. `apt-get install snmp snmp-mibs-downloader` on debian based distributions).

In order for SNMP to know the MIB files, they must be placed in the path `~/.snmp/mibs`.

In the examples the three programs `snmpget`, `snmpset` and `snmpwalk` are used:

- `snmpget` is used to query a variable and has the following syntax:
  `snmpget -c <community> -m <MIB> <device-address> <OID>`

- `snmpset` is used to set a variable and has the following syntax:
  `snmpset -c <community> -m <MIB> <device-address> <OID> <type> <value>`
- `snmpwalk` queries all leaves of a subtrees and has the following syntax:
  `snmpwalk -c <community> -m <MIB> <device-address> <OID>`

If you want to copy and paste the examples into your own shell, it helps to store the device address and the community string as an environment variable (`HUMS_ADDRESS`, `HUMS_COMMUNITY`) with the following commands:

```
export HUMS_ADDRESS=<device-address>
export HUMS_COMMUNITY=<community-string>
```

2.2 REST (HTTP)

Representational State Transfer (REST) is a convention how to design a HTTP interface that accesses resources. In this case, a REST interface can be used as an alternative to SNMP to read or write variable data. Using the REST interface is the best choice if you want to develop your own application that accesses the information, because nowadays almost every programming language has an HTTP library and JSON parser integrated.

The definition of the interface is provided by the OpenAPI (formerly known as Swagger) specification at:

http://<device-address>/api/hums/v1/documents?name=rest

This specification offers you various ways to address the APIs. For example, it can be included by HTTP debug programs such as Postman, or a client library can be automatically generated using Swagger Codegen.

In general, the REST interface follows the conventions: The HTTP function GET reads resources, POST creates new one and PUT or PATCH changes them.

Be warned: unlike SNMP, the REST interface has no authentication capabilities and all data provided by HUMS can be read and settings can be changed by anyone on the same network.

The UNIX program curl is used for the individual REST examples. As with SNMP, the device address can be set as an environment variable for simple copy/paste into the shell:

```
export HUMS_ADDRESS=<device-address>
```

3 APIs

HUMS offers various application programming interfaces (APIs) from which information can be retrieved. In this chapter the individual interfaces and their variables are described and the interaction is explained with examples.

3.1 Device information

Six simple interfaces are provided to read device information such as identification, BIOS, date, time, system, HUMS and software information.

For more detailed information, the equipment interface in chapter 3.3 can be used.

3.1.1 Greetings

SNMP	MIB:	RS-GREETINGS-MIB	OID: rsGreetingsMibModule
REST	GET	{endpoint}/v1/greetings	

This API provides basic device identification information.

JSON Key	SNMP OID	Description	Example Value
manufacturer	rsGreetingsManufacturer	Device Manufacturer	"Rohde&Schwarz GmbH&Co.KG"
model	rsGreetingsModel	Device Model	"CMP 200"
serial	rsGreetingsSerialNumber	Device Serial	"900001"
version	rsGreetingsVersion	Device Firmware Version	"4.0.5000.395"

3.1.1.1 Examples

SNMP: Get Greetings

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-GREETINGS-MIB $HUMS_ADDRESS rsGreetingsMibModule
```

Response

```
RS-GREETINGS-MIB::rsGreetingsManufacturer.0 = STRING: "Rohde&Schwarz GmbH&Co.KG"
RS-GREETINGS-MIB::rsGreetingsModel.0 = STRING: "CMP 200"
RS-GREETINGS-MIB::rsGreetingsSerialNumber.0 = STRING: "900001"
RS-GREETINGS-MIB::rsGreetingsVersion.0 = STRING: "4.0.5000.395"
```

REST: Get Greetings

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/greetings
```

Response

```
{
    "manufacturer": "Rohde&Schwarz GmbH&Co.KG",
    "model": "CMP 200",
    "serial": "900001",
    "version": "4.0.5000.461"
}
```

3.1.2 BIOS information

SNMP	MIB:	RS-BIOS-INFO-MIB	OID: rsBiosMibModule
REST	GET	{endpoint}/v1/equipment/bios	

This API returns the BIOS information.

JSON Key	SNMP OID	Description	Example Value
isPresent	rsBiosPresent	This variable shows if a BIOS is present in the instrument. 0 = no BIOS present 1 = BIOS present	1
manufacturer	rsBiosManufacturer	BIOS manufacturer.	“EEPD GmbH”
serialNumber	rsBiosSerialNumber	BIOS serial number.	"54L9BH2"
version	rsBiosVersion	BIOS version number.	"1.4"
caption	rsBiosCaption	BIOS caption.	"IC11REV2"
releaseDate	rsBiosReleaseDate	BIOS release date.	"06/27/2018"

3.1.2.1 Examples

SNMP: Get BIOS information

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-BIOS-INFO-MIB $HUMS_ADDRESS rsBiosMibModule
```

Response

```
RS-BIOS-INFO-MIB::rsBiosPresent.0 = INTEGER: 1
RS-BIOS-INFO-MIB::rsBiosManufacturer.0 = STRING: "EEPD GmbH"
RS-BIOS-INFO-MIB::rsBiosSerialNumber.0 = STRING: "54L9BH2"
RS-BIOS-INFO-MIB::rsBiosVersion.0 = STRING: "1.4"
RS-BIOS-INFO-MIB::rsBiosCaption.0 = STRING: "IC11REV2"
RS-BIOS-INFO-MIB::rsBiosReleaseDate.0 = STRING: "06/27/2018"
```

REST: Get BIOS information

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/equipment/bios
```

Response

```
{
    "manufacturer": "EEPD GmbH",
    "serialNumber": "54L9BH2",
    "version": "1.4",
    "releaseDate": "06/27/2018",
    "caption": "IC11REV2",
    "isPresent": true
}
```

3.1.3 Date and time information

SNMP	MIB:	RS-TIME-DATE-MIB	OID: rsTimeDateMibModule
REST	GET	{endpoint}/v1/date-time	

This API is for details regarding date and time of the R&S instrument.

JSON Key	SNMP OID	Description	Example Value
utc	rsTimeDateUTC	UTC time and date formatted according to ISO 8601.	"2020-05-04T08:48:45Z"
local	rsTimeDateLocal	Local time and date formatted according to ISO 8601.	"2020-05-04T10:48:45+0200"
timezone	rsTimeDateTimeZone	Time zone abbreviation.	"CEST"
dst	rsTimeDateDST	Daylight saving time active or not. 0 = no daylight saving time. 1 = daylight saving time.	1

3.1.3.1 Examples

SNMP: Get Date and Time information

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-TIME-DATE-MIB $HUMS_ADDRESS rsTimeDateMibModule
```

Response

```
RS-TIME-DATE-MIB::rsTimeDateUTC.0 = STRING: "2020-05-04T08:48:45Z"
RS-TIME-DATE-MIB::rsTimeDateLocal.0 = STRING: "2020-05-04T10:48:45+0200"
RS-TIME-DATE-MIB::rsTimeDateTimeZone.0 = STRING: "CEST"
RS-TIME-DATE-MIB::rsTimeDateDST.0 = INTEGER: 1
```

REST: Get Date and Time information

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/equipment/date-time
```

Response

```
{
    "utc": "2020-05-04T10:26:51Z",
    "local": "2020-05-04T12:26:51+0200",
    "timezone": "CEST",
    "dst": true
}
```

3.1.4 System information

SNMP	MIB:	RS-SYSTEM-INFO-MIB	OID: rsSystemInfoMibModule
REST	GET	{endpoint}/v1/system-info	

This API provides a basic overview (network, ressources, bios, operating system) of the R&S product that is easy to read and understand.

JSON Key	SNMP OID	Description	Example Value
hostname	rsSystemInfoHostname	Hostname of the product.	"scu-400b7429"
operatingSystemName	rsSystemInfoOperatingSystemName	Name of the operating system.	"Linux"
operatingSystemManufacturer	rsSystemInfoOperatingSystemManufacturer	Name of the operating system manufacturer.	"Microsoft Corporation"
operatingSystemConfiguration	rsSystemInfoOperatingSystemConfiguration	Configuration of the operating system.	"Standalone Workstation"
operatingSystemBuildtype	rsSystemInfoOperatingSystemBuildType	Build type of the operating system.	"Multiprocessor Free"
registeredOwner	rsSystemInfoRegisteredOwner	Registered owner.	"Rohde & Schwarz Teisnach"
registeredOrganization	rsSystemInfoRegisteredOrganisation	Registered organization.	"Rohde & Schwarz Teisnach"
productID	rsSystemInfoProductID	Product ID of the operating system.	"0329-00000-00003-AA306"
installDate	rsSystemInfoInstallDate	Original install date of the operating system.	"2019-01-16T18:09:19+120"
bootTime	rsSystemInfoBootTime	Boot time of the product.	"2020-04-21T05:10:38Z"
systemManufacturer	rsSystemInfoSystemManufacturer	The name of the system manufacturer.	"EEPD GmbH"
systemModel	rsSystemInfoSystemModel	The system model.	
systemType	rsSystemInfoSystemType	System type information.	"x86_64"
processors	rsSystemInfoProcessors	Information on the CPU used in the system.	"Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz;…"
bios	rsSystemInfoBios	Summary of BIOS information.	"EEPD GmbH IC11REV2"
bootDevice	rsSystemInfoBootDevice	Boot device information.	"/dev/sda1"
systemLocale	rsSystemInfoSystemLocale	System locale information.	"en-us;English (United States)2
inputLocale	rsSystemInfoInputLocale	Input locale information.	"en-us;English (United States)"
timeZone	rsSystemInfoTimeZone	Time zone information.	"(UTC) Coordinated Universal Time)"
totalPhysicalMemory	rsSystemInfoTotalPhysicalMemory	Total physical memory [MB].	15920
virtualMemoryMax	rsSystemInfoVirtualMemoryMax	Maximum virtual memory [MB].	15920
virtualMemoryAvailable	rsSystemInfoVirtualMemoryAvailable	Available virtual memory [MB].	15920
virtualMemoryUsed	rsSystemInfoVirtualMemoryUsed	Used virtual memory [MB].	11696
domain	rsSystemInfoDomain	Domain name.	"ad-int.net"
logonServer	rsSystemInfoLogonServer	Domain logon server.	"\\\\AD1"
hotfixes	rsSystemInfoHotfixes	List of installed hotfixes.	"KB4506998,KB4465065"
virtualization	rsSystemInfoVirtualization	Summary of virtualization.	"hyper-v"
networkCards	rsSystemInfoNetworkCardTable	Table with an overview on network cards. (see next Table)	-

The system information API contains an array of network cards with the following properties:

JSON Key	SNMP OID	Description	Example Value
name	rsSystemInfoNetworkCardName	Name of the network adapter.	"eth0"
connectionName	rsSystemInfoNetworkCardConnection	Type of the network connection.	"Ethernet"
status	rsSystemInfoNetworkCardStatus	Status of the network card or adapter.	"Media connected"
dhcpEnabled, dhcpServer	rsSystemInfoNetworkCardDHCP	DHCP summary.	"DHCP activated: yes, DHCP server: 10.11.12.13"
ipAddresses	rsSystemInfoNetworkCardIpAddresses	List of valid IP addresses (IPv4 and IPv6).	"10.112.1.119, fe80::e5b3:c1b8:44e0:ddad"

3.1.4.1 Examples

SNMP: Get System information

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-SYSTEM-INFO-MIB $HUMS_ADDRESS rsSystemInfoMibModule
```

Response

```
RS-SYSTEM-INFO-MIB::rsSystemInfoHostname.0 = STRING: "scu-400b7429"
RS-SYSTEM-INFO-MIB::rsSystemInfoOperatingSystemName.0 = STRING: "Linux"
RS-SYSTEM-INFO-MIB::rsSystemInfoBootTime.0 = STRING: "2020-04-21T05:10:38Z"
RS-SYSTEM-INFO-MIB::rsSystemInfoSystemManufacturer.0 = STRING: "EEPD GmbH"
RS-SYSTEM-INFO-MIB::rsSystemInfoSystemType.0 = STRING: "x86_64"
RS-SYSTEM-INFO-MIB::rsSystemInfoProcessors.0 = STRING: "Intel(R) Core(TM) i7-3615QE CPU @
2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @
2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @
2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @
2.30GHz; Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz"
RS-SYSTEM-INFO-MIB::rsSystemInfoBios.0 = STRING: "EEPD GmbH IC11REV2"
RS-SYSTEM-INFO-MIB::rsSystemInfoTimeZone.0 = STRING: "CEST"
RS-SYSTEM-INFO-MIB::rsSystemInfoTotalPhysicalMemory.0 = INTEGER: 15920
RS-SYSTEM-INFO-MIB::rsSystemInfoVirtualMemoryMax.0 = INTEGER: 15920
RS-SYSTEM-INFO-MIB::rsSystemInfoVirtualMemoryAvailable.0 = INTEGER: 4224
RS-SYSTEM-INFO-MIB::rsSystemInfoVirtualMemoryUsed.0 = INTEGER: 11696
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardName.1 = STRING: "eth0"
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardName.2 = STRING: "lo"
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardConnection.1 = ""
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardConnection.2 = ""
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardStatus.1 = STRING: "up"
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardStatus.2 = STRING: "unknown"
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardDHCP.1 = STRING: " "
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardDHCP.2 = STRING: " "
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardIpAddresses.1 = STRING:
"10.112.0.181;fe80::ac2d:8609:8319:5abd%eth0"
RS-SYSTEM-INFO-MIB::rsSystemInfoNetworkCardIpAddresses.2 = STRING: "127.0.0.1;::1"
```

REST: Get System information

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/system-info
```

Response


```
{
    "hostname": "scu-400b7429",
    "operatingSystemName": "Linux",
    "operatingSystemManufacturer": null,
    "operatingSystemConfiguration": null,
    "operatingSystemBuildtype": null,
    "registeredOwner": null,
    "registeredOrganization": null,
    "productID": null,
    "installDate": null,
    "bootTime": "2020-04-21T05:10:38Z",
    "systemManufacturer": "EEPD GmbH",
    "systemModel": null,
    "systemType": "x86_64",
    "bios": "EEPD GmbH IC11REV2",
    "bootDevice": null,
    "systemLocale": null,
    "inputLocale": null,
    "timeZone": "CEST",
    "totalPhysicalMemory": 15920,
    "virtualMemoryMax": 15920,
    "virtualMemoryAvailable": 4687,
    "virtualMemoryUsed": 11233,
    "domain": null,
    "logonServer": null,
    "processors": [
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
        "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz"
    ],
    "hotfixes": [],
    "networkCards": [
        {
            "name": "eth0",
            "connectionName": null,
            "status": "up",
            "dhcpEnabled": null,
            "dhcpServer": null,
            "ipAddresses": [
                "10.112.0.181",
                "fe80::ac2d:8609:8319:5abd%eth0"
            ]
        },
        {
            "name": "lo",
            "connectionName": null,
            "status": "unknown",
            "dhcpEnabled": null,
            "dhcpServer": null,
            "ipAddresses": [
                "127.0.0.1",
                "::1"
            ]
        }
    ],
    "virtualization": null
}
```

3.1.5 HUMS information

SNMP	MIB:	RS-HUMS-INFO-MIB	OID: rsHumsInfoMibModule
REST	GET	{endpoint}/v1/hums-info	

This API provides basic hums diagnostics relevant especially for development and integration.

JSON Key	SNMP OID	Description	Example Value
version	rsHumsVersion	Name of the network adapter.	"1.0.4"
startup	rsHumsStartup	Type of the network connection.	"2020-07-21T11:15:21Z"
restRequests	rsHumsRestRequests	Status of the network card or adapter.	364
snmpRequests	rsHumsSnmpRequests	DHCP summary.	423
databaseSize	rsHumsDatabaseSize	List of valid IP addresses (IPv4 and IPv6).	324968
utilizationRecordingEnabled	rsHumsUtilizationRecordingEnabled	1 = utilization history recording enabled 0 = utilization history recording disabled	1
utilizationRecordingStart	rsHumsUtilizationRecordingStart	ISO 8601 formatted timestamp of the oldest utilization history entry.	"2020-07-21T12:15:21Z"
utilizationRecordingInterval	rsHumsUtilizationRecordingInterval	The utilization recording history interval [s].	600
utilizationRecordingDuration	rsHumsUtilizationRecordingDuration	Maximum number of days the utilization recordings are stored on the device.	365
utilizationDatabaseEntries	rsHumsUtilizationDatabaseEntries	Number of history entries in the utilization database.	4565
deviceHistoryStart	rsHumsDeviceHistoryStart	ISO 8601 formatted timestamp of the oldest device history event.	"2020-07-21T12:15:21Z"
deviceHistoryEntries	rsHumsDeviceHistoryEntries	Number of device history events in the rsDeviceHistoryEventTable.	15

3.1.5.1 Examples

SNMP: Get HUMS information

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-HUMS-INFO-MIB $HUMS_ADDRESS rsHumsInfoMibModule
```

Response

```
RS-HUMS-INFO-MIB::rsHumsVersion.0 = STRING: "1.0.4"
RS-HUMS-INFO-MIB::rsHumsStartup.0 = STRING: "2020-07-21T11:15:21Z"
RS-HUMS-INFO-MIB::rsHumsRestRequests.0 = INTEGER: 364
RS-HUMS-INFO-MIB::rsHumsSnmpRequests.0 = INTEGER: 423
RS-HUMS-INFO-MIB::rsHumsDatabaseSize.0 = INTEGER: 324968
RS-HUMS-INFO-MIB::rsHumsUtilizationRecordingEnabled.0 = INTEGER: 1
RS-HUMS-INFO-MIB::rsHumsUtilizationRecordingStart.0 = STRING: "2020-07-21T12:15:21Z"
RS-HUMS-INFO-MIB::rsHumsUtilizationRecordingInterval.0 = INTEGER: 600
RS-HUMS-INFO-MIB::rsHumsUtilizationRecordingDuration.0 = INTEGER: 365
RS-HUMS-INFO-MIB::rsHumsUtilizationDatabaseEntries.0 = INTEGER: 4565
RS-HUMS-INFO-MIB::rsHumsDeviceHistoryStart.0 = STRING: "2020-07-21T12:15:21Z"
RS-HUMS-INFO-MIB::rsHumsDeviceHistoryEntries.0 = INTEGER: 15
```

REST: Get HUMS information

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/hums-info
```

Response

```
{
  "version": "1.0.4",
  "startup": "2020-07-21T11:15:21Z",
  "restRequests": 364,
  "snmpRequests": 423,
  "databaseSize": 324968,
  "utilizationRecordingEnabled": true,
  "utilizationRecordingStart": "2020-07-21T12:15:21Z",
  "utilizationRecordingInterval": 600,
  "utilizationRecordingDuration": 365,
  "utilizationDatabaseEntries": 4565,
  "deviceHistoryStart": "2020-07-21T12:15:21Z",
  "deviceHistoryEntries": 15
}
```

3.1.6 Installed software

SNMP	-	
REST	GET	{endpoint}/v1//installed-software

This API provides an **array** of information about installed software on the device.

JSON Key	SNMP OID	Description	Example Value
index	-	A unique value for each piece of software installed on the host. This value shall be in the range from 1 to the number of pieces of software installed on the host.	123
name	-	A textual description of this installed piece of software, typically including the manufacturer, revision, the name by which it is commonly known, and optionally, its serial number.	"R&S Health and Utilization Monitoring Service"
productId	-	The product ID of this installed piece of software (optional).	"UUID"
productId	-	Type of the installed software: unknown, operatingSystem, deviceDriver, application	"application"
installDate	-	The last-modification date of this application. ISO 8601 formatted.	"2020-08-01T12:00:00Z"

3.1.6.1 Example

REST: Get installed software	
Command	`curl http://$HUMS_ADDRESS/api/hums/v1/installed-software`

Response	{
	"index": 123,
	"name": "R&S Health and Utilization Monitoring Service",
	"productId": "UUID",
	"productId": "application",
	"installDate": "2020-08-01T12:00:00Z"
	}

3.2 Tags

SNMP	MIB:	RS-DEVICE-TAGS-MIB	OID: rsDevTagsTable
REST	GET	{endpoint}/v1/device-tags	
	POST	{endpoint}/v1/device-tags	
	DELETE	{endpoint}/v1/device-tags/all	
	GET	{endpoint}/v1/device-tags/:id	
	PUT	{endpoint}/v1/device-tags/:id	
	DELETE	{endpoint}/v1/device-tags/:id	

Read and write custom defined key value pairs for example an inventory number. Unlike normal key value stores, a key can be assigned here more than once.

The tag table is an **array** of maximum 32 entries with the following properties:

JSON Key	SNMP OID	Description	Example Value
id	-	Index of the device tag.	0
key	rsDevTagsKey	Short key of the device tag.	"inventory code"
value	rsDevTagsValue	Value of the device tag.	"12345"
-	rsDevTagsRowStatus	The status of the row. Rows are created by an SNMP set request setting rsDevTagsRowStatus to 'createAndGo(4)'. Rows are deleted by an SNMP set request setting rsDevTagsRowStatus to 'destroy(6)'.	1

3.2.1 Examples

SNPM: Get all tags	
Command	`snmpwalk -c $HUMS_COMMUNITY -m +RS-DEVICE-TAGS-MIB $HUMS_ADDRESS rsDevTagsTable`

Response	`RS-DEVICE-TAGS-MIB::rsDevTagsKey.1 = STRING: "Hello"`
	`RS-DEVICE-TAGS-MIB::rsDevTagsKey.2 = STRING: "Emoji"`
	`RS-DEVICE-TAGS-MIB::rsDevTagsValue.1 = STRING: "a new value"`
	`RS-DEVICE-TAGS-MIB::rsDevTagsValue.2 = Hex-STRING: F0 9F 98 80`
	`RS-DEVICE-TAGS-MIB::rsDevTagsRowStatus.1 = INTEGER: active(1)`
	`RS-DEVICE-TAGS-MIB::rsDevTagsRowStatus.2 = INTEGER: active(1)`

SNPM: Create a new tag	
Command	`snmpset -c $HUMS_COMMUNITY -m +RS-DEVICE-TAGS-MIB $HUMS_ADDRESS rsDevTagsRowStatus.2 i 4`
Response	-

SNPM: Update a tag	
Command	`snmpset -c $HUMS_COMMUNITY -m +RS-DEVICE-TAGS-MIB $HUMS_ADDRESS rsDevTagsValue.2 s "12345"`
Response	`RS-DEVICE-TAGS-MIB::rsDevTagsValue.2 = STRING: "12345"`

SNPM: Delete a tag	
Command	`snmpset -c $HUMS_COMMUNITY -m +RS-DEVICE-TAGS-MIB $HUMS_ADDRESS rsDevTagsRowStatus.2 i 6`
Response	`RS-DEVICE-TAGS-MIB::rsDevTagsRowStatus.2 = INTEGER: destroy(6)`

REST: Get all tags

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/device-tags
```

Response

```
[
    {
        "id": 0,
        "key": "InvNr",
        "value": "0815"
    },
    {
        "id": 1,
        "key": "DeviceStatus",
        "value": "satisfied 😀"
    }
]
```

REST: Create a new tag

Command

```
curl -X POST -L -H "Content-Type: application/json" -d
'{"key":"new_key","value":"new_value"}' http://$HUMS_ADDRESS/api/hums/v1/device-tags
```

Response -

REST: Delete all tags

Command

```
curl -X DELETE http://$HUMS_ADDRESS/api/hums/v1/device-tags/all
```

Response -

REST: Update a tag

Command

```
curl -X PUT -H "Content-Type: application/json" -d
'{"key":"new_key_updated","value":"new_value_updated"}'
http://$HUMS_ADDRESS/api/hums/v1/device-tags/0
```

Response -

REST: Delete a tag

Command

```
curl -X DELETE http://$HUMS_ADDRESS/api/hums/v1/device-tags/0
```

Response -

3.3 Equipment

SNMP	MIB:	RS-EQUIPMENT-MIB	OID: rsEquipmentMibModule
REST	GET	{endpoint}/v1/equipment	

This API provides the equipment configuration data of the device (device footprint). This information can be used to obtain information about the configuration of the device (hardware, software, software licenses, etc.) and automate further processing.

JSON Key	SNMP OID	Description	Example Value
@formatVersion	rsEquipmentFormatVersion	The Device-Footprint format version.	"03.04"
@creationDate	rsEquipmentCreationDate	Creation date of the RS-EQUIPMENT-MIB information.	"2020-05-06T12:05:39"
@checksum	-	Checksum of the Device Footprint. Currently the MD5 algorithm is used to calculate the value. The checksum can be used to detect changes to the Device Footprint either from malicious manipulation or from transmission errors.	"a761fc5acfb0351c8631390d856c92ff"
deviceData	rsEquipmentDeviceData	Object of device data. See 3.3.1	-
hardwareData	rsEquipmentHardware	Object of hardware See 3.3.2	-
softwareData	rsEquipmentSoftware	Object of software See 3.3.3	-
licenseData	rsEquipmentLicenses	Object of licenses See 3.3.4	-

3.3.1 Device data

Contains device-related information which is not assigned to specific modules, but to the device as a whole.

JSON Key	SNMP OID	Description	Example Value
deviceId	rsEqDevDataDeviceID	Contains the device ID of the device.	"1201.0002K20-900001-eF"
partChangeIndex	rsEqDevDataPartChangeIndex	PartChangeIndex of a device. As this entry is maintained only for some models, it is optional.	"01.00"
type	rsEqDevDataType	Contains the type designation of the device.	"CMP 200"
partName	rsEqDevDataPartName	Contains a device description which supplements the type i.e. on the device pass.	"CMP200"
productDate	rsEqDevDataProductDate	Production date of the device in UTC. The format is YYYY-MM-DD.	"2019-09-03"
upgradeType	rsEqDevDataUpgradeType	If the device has been upgraded to another model, this object contains the new type designation.	"CMW 550"
upgradePartNumber	rsEqDevDataUpgradePartNumber	If the device has been upgraded to another model, this object contains the new part number.	"1201.0002K55"
hostName	rsEqDevDataHostName	The hostname or IP address, how the device appeared in the network, when the data for this MIB module was created.	"scu-400b7429"
aliasName	rsEqDevDataAliasName	A human readable alias for the hostname, which can be defined by the user.	"TestDevice"
-	rsEqTestInstructionsTable	See 3.3.1.1	-
-	rsEqDevDataMaintenanceData	See 3.3.1.2	-
-	rsEqServiceDataTable	See 3.3.1.3	-
statusMessageList	rsEqStatusMessageTable	See 3.3.1.4	-
networkInterfaces	-	See 3.3.1.5	-

3.3.1.1 Test instructions table

Contains the list of test instruction documents used for the device test by the test department.

JSON Key	SNMP OID	Description	Example Value
name	rsEqTestInstructionName	Contains an identifier which uniquely identifies this test instruction document. May be empty only when one and only one test instruction document exists.	"RF Test"
version	rsEqTestInstructionsVersion	Version of the test instruction document.	"1.56"

3.3.1.2 Maintenance data

Contains all the data of relevance to device maintenance. This includes internal counters for operating hours, switching procedures, battery status etc.

This section contains information about the performed maintenance tasks on this device. These allow statements concerning the “illness of a patient” and therefore allow to identify recurrent problems.

Furthermore this section contains operating data including operating hours, temperature distribution and makeAndBreakCylces. Those data allow to determine the usage conditions and to predict e.g. the probability of failure.

JSON Key	SNMP OID	Description	Example Value
operatingHours	rsEquipmentOperatingHours	Contains the total operating hours for the device/module.	"120"
makeAndBreakCycles	rsEquipmentMakeAndBreakCycles	Contains the number of make and break cycles for the device/module.	"262"
maintenances	rsEqMaintenanceTable	See 3.3.1.2.1	-
-	rsEqMaintenanceTasksTable	See 3.3.1.2.1.1	-
maintenanceInfos	rsEqMaintenanceInfoTable	See 3.3.1.2.2	-
-	rsEqMaintenancePropertyTable	See 3.3.1.2.2.1	-

3.3.1.2.1 Maintenance table

List of maintenance tasks performed on this device.

JSON Key	SNMP OID	Description	Example Value
responsiblePerson	rsEqMaintenanceResponsiblePerson	Name of the person, who performed the maintenance task.	"genericPerson"
maintenanceDate	rsEqMaintenanceDate	Date of the maintenance task given in UTC.	"2013-08-21"
generatedBy	rsEqMaintenanceGeneratedBy	Defines the origin that created the maintenance entry.	"device"
maintenanceTasks	-	See 3.3.1.2.1.1	-

3.3.1.2.1.1 Maintenance tasks table

List of tasks which have been performed during this maintenance.

JSON Key	SNMP OID	Description	Example Value
name	rsEqMaintenanceTaskName	Description of the maintenance task.	"BatteryChanged"
done	rsEqMaintenanceTaskDone	Defines whether the task has been done or not.	"true"
result	rsEqMaintenanceTaskResult	Description of the result of the task.	"within specification"

3.3.1.2.2 Maintenance info table

List of maintenance information relevant for this device.

JSON Key	SNMP OID	Description	Example Value
name	rsEqMaintenanceInfoName	Name of the maintenance information.	"operatingHours"
value	rsEqMaintenanceInfoValue	Value of the maintenance information. In case there is a list of values, properties of the child table are used.	213
properties	-	See 3.3.1.2.2.1	-

3.3.1.2.2.1 Maintenance property table

List of properties with name / value pairs of this maintenance info.

JSON Key	SNMP OID	Description	Example Value
name	rsEqMaintenancePropertyName	Description of the property.	"Warm"
value	rsEqMaintenancePropertyData	Value of the property.	97

3.3.1.3 Service data table

Contains the service data for non-module-specific service circulars which have been implemented by Service.

JSON Key	SNMP OID	Description	Example Value
serviceCode	rsEqServiceDataCode	Service Code of the performed service.	34
serviceDate	rsEqServiceDataDate	Date of the performed service given in UTC.	"2013-06-12"

3.3.1.4 Status message table

Contains a list of status messages that are recognized for the whole device.

JSON Key	SNMP OID	Description	Example Value
message	rsEqStatusMessageText	Status message text.	"PCSCReader"
details	rsEqStatusMessageDetails	Additional status information, which might contain technical data.	"OMNIKEY AG Smart Card Reader 00 00: 162(has card, exclusively)"
provider	rsEqStatusMessageProvider	Name of the software module which has reported the status message.	"OptionKeyLib"

3.3.1.5 Network interface table

Contains the data of the network interfaces.

This table is only available in REST, for SNMP please use the system information MIB.

JSON Key	SNMP OID	Description	Example Value
ipAddress	-	Currently used IPv4 address of the network interface	"10.112.0.181"
ipAddressV6	-	Optional IPv6 address of the network interface	"fe80::ac2d:8609:8319:5abd"
macAddress	-	MAC address of the network adapter	"00:e0:33:05:c7:9f"
type	-	Type of the network interface	"LAN"
description	-	Longer description of the network type	"Ethernet"

3.3.2 Hardware

Contains information on automatically detectable modules in the device and their characteristics.

JSON Key	SNMP OID	Description	Example Value
cpu	-	Name of the CPU.	"Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz"
ramSize	-	Ram Size in Byte.	16693514240
ramManufacturer	-	Manufacturer or list of Manufacturers of the Ram.	"Kingston,Samsung"
hardwareList	rsEqHardwareTable	See 3.3.2.1	-
-	rsEqHwBPartIdentificationTable	See 3.3.2.1.1	-
-	rsEqHwProgrammablePartsTable	See 3.3.2.1.2	-
-	rsEqHwMaintenanceTable	See 3.3.2.1.3.1	-
-	rsEqHwMaintenanceTasksTable	See 3.3.2.1.3.1.1	-
-	rsEqHwMaintenanceInfoTable	See 3.3.2.1.3.2	-
-	rsEqHwMaintenancePropertyTable	See 3.3.2.1.3.2.1	-
-	rsEqHwServiceDataTable	See 3.3.2.1.4	-
-	rsEqHwPropertiesTable	See 3.3.2.1.5	-
productList	rsEqHwProductTable	See 3.3.2.2	-
computerComponents	-	See 3.3.2.3	-

3.3.2.1 Hardware table

Contains a list of the automatically detectable modules in the device and their characteristics.

JSON Key	SNMP OID	Description	Example Value
id	rsEqHardwareId	Numeric identifier, unique within the RS-EQUIPMENT-MIB module.	65655
productId	rsEqHardwareProductId	Identifier of the product which contains the hardware modules. Several modules can be contained in one and the same product.	75860
partNumber	rsEqHardwarePartNumber	Part number and module model.	"1234.5678.90"
sn	rsEqHardwareSN	Serial number of the module.	"104440/000"
partChangeIndex	rsEqHardwarenPartChangeIndex	PartChangeIndex (PCI for short) of the module.	"00.00"
hardwareCode	rsEqHardwareCode	Hardware code of the module.	"2"
productDate	rsEqHardwareProductDate	Production date of the module.	"2018-10-8"
readCode	rsEqHardwareReadCode	Read code of the module.	"0"
testInstruction	rsEqHardwareTestInstruction	Test instruction of the module.	"01.00"
name	rsEqHardwareName	Name of the module.	"IPC"
partName	rsEqHardwarePartName	Readable module name intended for display in programs that carry out further processing.	"CONTROLLER IPC11/4"
parentId	rsEqHardwareParentId	ID of the parent module, i.e. the id of the module to which the module in question belongs.	
externConnected	rsEqHardwareExternConnected	Indicates whether the hardware is connected to the device via an external or an internal interface. Hardware which is connected to the device via USB or IQ is assigned the value 'True'. Plug-in cards (i.e. hardware permanently connected to the device) is assigned the value 'False'. This object gives Service means of deciding unambiguously whether hardware is just connected to the device or permanently installed in the device.	
connectorName	rsEqHardwareConnectorName	Name of the connection between the module in question and the parent module.	"A500"
-	rsEqHardwareOperatingHours	Contains the total operating hours for the hardware module.	
-	rsEqHardwareMakeAndBreakCycles	Contains the number of make and break cycles for the hardware module.	
bPartIdentification	-	See 3.3.2.1.1	-
programmableParts	-	See 3.3.2.1.2	-
maintenanceData	-	See 3.3.2.1.3	-
serviceData	-	See 3.3.2.1.4	-
properties	-	See 3.3.2.1.5	-

3.3.2.1.1 Hardware part identification table

Contains the manufacturer's identification of bought-in parts in the device.

JSON Key	SNMP OID	Description	Example Value
name	rsEqHwBPartIdentificationName	Gives the name used by the manufacturer to refer to the part.	"R&S PC extended"
partNumber	rsEqHwBPartIdentificationPartNumber	Part-type designation used by the manufacturer to uniquely identify this module type.	"4218.2382.00"
sn	rsEqHwBPartIdentificationSN	Serial number assigned to the module by the manufacturer.	872382
partChangeIndex	rsEqHwBPartIdentificationPartChangeIndex	Part-type version designation used by the manufacturer to uniquely identify the module production batch.	1.03
vendorName	rsEqHwBPartIdentificationVendorName	Designation of the manufacturer.	"EEPD"
additionalInfo	rsEqHwBPartIdentificationAdditionalInfo	Field assigned no specific function. Supplementary information from the manufacturer is stored in text form in this field.	"EEPD serialized standard 2.0"

3.3.2.1.2 Hardware programmable parts table

List of the programmed software components of the hardware in question.

JSON Key	SNMP OID	Description	Example Value
name	rsEqHwProgrammablePartsName	Name of the programmed software.	"RFMOBO-FPGA (Hw)"
version	rsEqHwProgrammablePartsVersion	Version of programmed software.	"5"
partNumber	rsEqHwProgrammablePartsPartNumber	Part number, including model of the software.	"1234.5678.90"
partChangeIndex	rsEqHwProgrammablePartsPartChangeIndex	PartChangeIndex of the software.	"01.00"

3.3.2.1.3 Hardware maintenance data

Contains all the data of relevance to device maintenance. This includes internal counters for operating hours, switching procedures, battery status.

JSON Key	SNMP OID	Description	Example Value
operatingHours	-	Contains the total operating hours for the device/module.	123
makeAndBreakCycles	-	Contains the number of make and break cycles for the device/module.	6
maintenances	-	See 3.3.2.1.3.1	-
maintenanceInfos	-	See 3.3.2.1.3.2	-

3.3.2.1.3.1 Hardware maintenance table

List of maintenance tasks performed on this device.

JSON Key	SNMP OID	Description	Example Value
responsiblePerson	rsEqHwMaintenanceResponsiblePerson	Name of the person, who performed the maintenance task.	"genericPerson"
maintenanceDate	rsEqHwMaintenanceDate	Date of the maintenance task given in UTC.	"2014-08-20"
generatedBy	rsEqHwMaintenanceGeneratedBy	Defines the origin that created the maintenance entry.	"device"
maintenanceTasks	-	See 3.3.2.1.3.1.1	-

3.3.2.1.3.1.1 Hardware maintenance tasks table

List of tasks which have been performed during this maintenance.

JSON Key	SNMP OID	Description	Example Value
-	rsEqHwMaintenanceTasksParent	Index of the maintenace table entry this maintenance task corresponds to.	-
name	rsEqHwMaintenanceTaskName	Description of the maintenance task.	"BatteryChanged"
done	rsEqHwMaintenanceTaskDone	Defines whether the task has been done or not.	"true"
result	rsEqHwMaintenanceTaskResult	Description of the result of the task.	"within specification"

3.3.2.1.3.2 Hardware maintenance info table

List of maintenance informations relevant for this device.

JSON Key	SNMP OID	Description	Example Value
name	rsEqHwMaintenanceInfoName	Name of the maintenance information.	
value	rsEqHwMaintenanceInfoValue	Value of the maintenance information. In case there is a list of values, properties of the child table are used.	
properties	-	See 3.3.2.1.3.2.1	-

3.3.2.1.3.2.1 Hardware maintenance property table

List of properties with name / value pairs of this maintenance info.

JSON Key	SNMP OID	Description	Example Value
name	rsEqHwMaintenancePropertyName	Description of the property.	"OperatingHours_Hot"
value	rsEqHwMaintenancePropertyData	Value of the property.	

3.3.2.1.4 Hardware service data table

List of services performed on this hardware.

JSON Key	SNMP OID	Description	Example Value
serviceCode	rsEqHwServiceDataCode	Service Code of the performed service.	11614
serviceDate	rsEqHwServiceDataDate	Date of the performed service given in UTC.	"2013-06-12"

3.3.2.1.5 Hardware properties table

List of freely defined properties.

JSON Key	SNMP OID	Description	Example Value
name	rsEqHwPropertyName	Name of the property.	"ExtPartName"
value	- rsEqHwPropertyValue	Value of the property.	"IPC114"

3.3.2.2 Product table

Contains a list of the products incorporated in the device.

JSON Key	SNMP OID	Description	Example Value
id	rsEqHwProductId	Numeric identifier unique within the RS-EQUIPMENT-MIB.	75860
partNumber	rsEqHwProductPartNumber	Part-number including model of the product.	"1212.0209.02"
partChangeIndex	rsEqHwProductPartChangeIndex	PartChangeIndex of the product.	"01.00"
type	rsEqHwProductType	Contains the type designation of the product (or product representative).	"CMW H570A"
partName	rsEqHwProductPartName	Contains a product description which supplements the type i.e. on the device pass.	"RF TRX"

3.3.2.3 Computer components table

A generic list of computer components.

JSON Key	SNMP OID	Description	Example Value
type	-	Numeric identifier, denoting the ‘type’ of the computer component	
name	-	A name of the computer component as human readable string, e.g. “Baseboard” or “Processor”	
properties	-	See 3.3.2.3.1	-

3.3.2.3.1 Computer components properties table

List of freely definable properties of the computer component.

JSON Key	SNMP OID	Description	Example Value
name	-	Property's name.	
value	-	Property's value.	

3.3.3 Software

Contains information on automatically detectable software components of the device and their characteristics.

JSON Key	SNMP OID	Description	Example Value
bios	rsEqSwBios	See 3.3.3.1	-
operatingSystem	rsEqSwOperatingSystem	See 3.3.3.2	-
softwareList	rsEqSoftwareTable	See 3.3.3.3	-
-	rsEqSoftwarePropertiesTable	See 3.3.3.3.1	-

3.3.3.1 Bios

Data of the BIOS installed on the device.

JSON Key	SNMP OID	Description	Example Value
name	rsEqBiosName	Designation of the installed BIOS image.	"IPC10/2-BIOS"
version	rsEqBiosVersion	Version of the installed BIOS image.	"IPC1-01.36.4-9S04-GP01.01"
partNumber	rsEqBiosPartNumber	BIOS image part number, including model of the software.	"1234.1237.02"
partChangeIndex	rsEqBiosPartChangeIndex	PartChangeIndex of the BIOS image.	2.12

3.3.3.2 Operating System

Data of the operating system installed on the device.

JSON Key	SNMP OID	Description	Example Value
name	rsEqOsName	Designation of the installed operating system image.	"Linux"
version	rsEqOsVersion	Version of the installed operating system image.	"6.1"
partNumber	rsEqOsPartNumber	The part number, including model of the operating system.	"1234.5678.90"
partChangeIndex	rsEqOsPartChangeIndex	Part change index of the operating system image.	2.32
imageName	rsEqOsImageName	The name of the disk image used during production deployment.	"Rohde & Schwarz Mobile Radio Tester Linux"
imageVersion	rsEqOsImageVersion	The version of the disk image used during production deployment.	"2.5.3-2.395"
windowsSuitMask	rsEqOsWindowsSuiteMask	This object is Windows specific. A bit mask that identifies the product suits available on the system. See MSDN documentation of 'OSVERSIONINFOEX'.	256
windowsProductType	rsEqOsWindowsProductType	Windows specific. Any additional information about the system. See MSDN documentation of 'OSVERSIONINFOEX'.	"1"
servicePackDescription	rsEqOsWindowsServicePackDescription	Windows specific. Human readable description of the	"Service Pack 1"

		latest Service Pack installed on the system.	
servicePackMajorVersion	rsEqOsWindowsServicePackMajorVersion	Windows specific. The major version number of the latest Service Pack installed on the system. See MSDN documentation of 'OSVERSIONINFOEX' structure. In the case of Windows 10 this object is the major upgrade 'ReleaseId'.	"1703"
servicePackMinorVersion	rsEqOsWindowsServicePackMinorVersion	Windows specific. The minor version number of the latest Service Pack installed on the system. See MSDN documentation of 'OSVERSIONINFOEX' structure. In the case of Windows 10 this object is mapped to 'CurrentBuildNumber'.	"15063"
systemWordSize	rsEqOsSystemWordSize	The word size of the operating system. Either 32 or 64.	"32"
systemLanguage	rsEqOsSystemLanguage	The current system language (locale).	"en_US.UTF-8"
kernelRelease	rsEqOsLinuxKernelRelease	Linux specific. Release name of the operating system kernel.	"4.14.136"
kernvelVersion	rsEqOsLinuxKernelVersion	Linux specific. Version of the operating system kernel.	"#1 SMP PREEMPT Thu Apr 16 09:40:58 UTC 2020"
buildVersion	rsEqOsBuildVersion	Build version of the operating system.	"17134"
hotfixList	rsEqOsHotfixTable	See 3.3.3.2.1	-

3.3.3.2.1 Software OS hotfix table

A list of installed patches of the operating system.

JSON Key	SNMP OID	Description	Example Value
hotfixId	rsEqOsHotfixId	Short name of the hotfix. In case of windows: The KB number.	"KB12345678"
installedOn	rsEqOsHotfixInstalled	The installation date of this hotfix.	"2019-01-10"

3.3.3.3 Software table

Contains a list of the software installed on the device (e.g. base system software and the software packages installed for the base system software).

JSON Key	SNMP OID	Description	Example Value
name	rsEqSoftwareName	Designation of the installed software package.	"BASE"
version	rsEqSoftwareVersion	Version of the installed software package.	"3.5.0.44"
status	rsEqSoftwareStatus	Status of the software package in relation to current use when device is started.	"active"
licenseStatus	rsEqSoftwareLicenseStatus	License status of the software package.	"active"
partNumber	rsEqSoftwarePartNumber	Part number including model of the software.	"1234.5678.90"
partChangeIndex	rsEqSoftwarePartChangeIndex	PartChangeIndex of the software.	23.02
sn	rsEqSoftwareSN	Serial number of the software package.	21339
vendorName	rsEqSoftwareVendorName	Designation of the vendor of the software package.	"Rohde & Schwarz"
wordSize	rsEqSoftwareWordSize	Word size of the software package.	"64"
configuration	rsEqSoftwareConfiguration	Build configuration of the software package.	"Release"
releaseState	rsEqSoftwareReleaseState	Release state of the software package.	"beta"
architecture	rsEqSoftwareArchitecture	Machine architecture of the software package.	"amd64"
softwareList	-	Contains the software packages (plus versions) contained in this software package. This definition is recursive and so any nesting of software packages can be represented.	-
properties	-	See 3.3.3.3.1	-

3.3.3.3.1 Software property table

List of freely defined properties of the software.

JSON Key	SNMP OID	Description	Example Value
name	rsEqSoftwarePropertyName	Description of the property.	"config"
value	rsEqSoftwarePropertyData	Value of the property.	"ini"

3.3.4 Licenses

Contains information on the license keys installed on the device and their characteristics.

JSON Key	SNMP OID	Description	Example Value
conformToTvr320	rsEqLicenseDataConformToTVR320	For internal use.	"True"
activeLicenseList	rsEqActiveLicenseTable	See 3.3.4.1	-
-	rsEqQuantifiedRegistrationTable	See 3.3.4.1.1	-
inactiveLicenseList	rsEqInactiveLicenseTable	See 3.3.4.2	-
deactivationKeyList	rsDeactivationKeyTable	See 3.3.4.3	-
statusMessageList	rsEqLicenseStatusMessageTable	See 3.3.4.4	-
tvr320ManagementData	-	For internal use.	-

3.3.4.1 Active license table

List of active licenses installed on the device

JSON Key	SNMP OID	Description	Example Value
optionIndex	rsEqActiveLicenseOptionIndex	Unique number for identifying the license type.	1125
type	rsEqActiveLicenseType	Type designation of the license.	"K200"
partName	rsEqActiveLicensePartName	Type designation of the license.	"GSM Signaling"
partNumber	rsEqActiveLicensePartNumber	Part number plus model of the license.	"1234.5678.90"
keyCode	rsEqActiveLicenseKeyCode	Actual keycode of the license.	"19681765471810865490412026 2605"
format	rsEqActiveLicenseFormat	Contains the format version of the keycode.	0
privKeyType	rsEqActiveLicensePrivKeyType	Privileges of the license key. It can be assigned one of the following values: 'Customer Order', 'Demo', 'Factory', 'Beta', 'Evaluation', 'Service'	"Customer Order"
timeStamp	rsEqActiveLicenseTimeStamp	Contains the date when the license key was generated given in UTC.	"2019-09-03T12:19:00Z"
licenseCount	rsEqActiveLicenseCount	Number of licenses in this license key.	1
activationType	rsEqActiveLicenseActivationType	Activation type of the license key. It can be assigned one of the following values: 'Deactivation', 'Permanent', 'Portable Permanent', 'Temporary Operating Time', 'Temporary Duration', 'Temporary Date', 'Quantified', 'Global Deactivation', 'Software Maintenance', 'Customer Develop', 'Customer Runtime', 'Permanent'	"Permanent"
validFrom	rsEqActiveLicenseValidFrom	Only for temporary option keys. - For activation type 'Temporary Operating Time': start value of the operating hours counter - For activation type 'Temporary Duration' and	"2012-10-23 17:00"

		'Temporary Date': start date given in UTC	
validTo	rsEqActiveLicenseValidTo	Only for temporary option keys. - For activation type 'Temporary Operating Time': stop value of the operating hours counter - For activation type 'Temporary Duration' and 'Temporary Date': stop date given in UTC	"2013-10-23 17:00"
timeToExpiration	rsEqActiveLicenseTimeToExpiration	Time to expiry of the license (if license is temporary).	"400 h"
freeQuantifiedLicenses	rsEqActiveLicenseFreeQuantifiedLicenses	Number of not yet used licenses contained in the license key in question. This value is reasonable only if rsEqActiveLicenseActivationType has the value 'Quantified'.	
customerLicenseReference	rsEqActiveLicenseCustomerLicenseReference	Contains the license reference. This is for Customer Develop and Customer Runtime Licenses only.	
companyLicenseControlIndex	rsEqActiveLicenseCompanyLicenseControlIndex	Option index of the corresponding Company License Control Option. This applies to Company Licenses only.	
companyLicenseControlState	rsEqActiveLicenseCompanyLicenseControlState	Company License Control Licenses State.	"active"
quantifiedRegistrationList	-	See 3.3.4.1.1	-

3.3.4.1.1 Quantified registration table

List of registrations of the license in question stored on the device.

JSON Key	SNMP OID	Description	Example Value
registeredId	rsEqQuantifiedRegistrationRegisteredId	ID of a stored registration of the quantified license in question. The string contains a UUID without any separator character.	

3.3.4.2 Inactive license table

List of licenses installed on the device which are NOT active.

JSON Key	SNMP OID	Description	Example Value
optionIndex	rsEqInactiveLicenseOptionIndex	Unique number for identifying the license type.	
type	rsEqInactiveLicenseType	Type designation of the license.	"K200"
partName	rsEqInactiveLicensePartName	Contains a license designation that, unlike the usual cryptic types, gives a readable	"GSM Signaling"

		description of the license.	
partNumber	rsEqInactiveLicensePartNumber	Part number plus model of the license.	"1234.5678.90"
keyCode	rsEqInactiveLicenseKeyCode	Actual keycode of the license.	"19681765471…"
format	rsEqInactiveLicenseFormat	Contains the format version of the keycode.	0
privKeyType	rsEqInactiveLicensePrivKeyType	Privileges of the license key. It can be assigned one of the following values: 'Customer Order', 'Demo', 'Factory', 'Beta', 'Evaluation', 'Service'	"Customer Order"
timeStamp	rsEqInactiveLicenseTimeStamp	Contains the date when the license key was generated given in UTC.	"2019-09-03T12:19:00Z"
licenseCount	rsEqInactiveLicenseCount	Number of licenses in this license key.	1
activationType	rsEqInactiveLicenseActivationType	Activation type of the license key. It can be assigned one of the following values: 'Deactivation', 'Permanent', 'Portable Permanent', 'Temporary Operating Time', 'Temporary Duration', 'Temporary Date', 'Quantified', 'Global Deactivation', 'Software Maintenance', 'Customer Develop', 'Customer Runtime', 'Permanent'	"Permanent"
validFrom	rsEqInactiveLicenseValidFrom	Only for temporary option keys. For activation type 'Temporary Operating Time': start value of the operating hours counter - For activation type 'Temporary Duration' and 'Temporary Date': start date given in UTC	"2012-10-23 17:00"
validTo	rsEqInactiveLicenseValidTo	Only for temporary option keys. - For activation type 'Temporary Operating Time': stop value of the operating hours counter - For activation type 'Temporary Duration' and 'Temporary Date': stop date given in UTC	"2013-10-23 17:00"
invalidityReason	rsEqInactiveLicenseInvalidityReason	Reason for invalidity.	"expired"
customerLicenseReference	rsEqInactiveLicenseCustomerLicenseReference	obtains the license reference. This is for Customer Develop and Customer Runtime Licenses only.	
companyLicenseControlIndex	rsEqInactiveLicenseCompanyLicenseControlIndex	Option index of the corresponding Company License Control Option. This applies to Company Licenses only.	
companyLicenseControlState	rsEqInactiveLicenseCompanyLicenseControlState	Company License Control Licenses State.	"active"

3.3.4.3 Deactivation key table

List of deactivation keys installed on the device.

JSON Key	SNMP OID	Description	Example Value
optionIndex	rsEqDeactivationKeyOptionIndex	Unique number for identifying the license type.	0
type	rsEqDeactivationKeyType	Type designation of the license.	"K200"
partName	rsEqDeactivationKeyPartName	Contains a license designation that, unlike the usual cryptic types, gives a readable description of the license.	"GSM Signaling"
partNumber	rsEqDeactivationKeyPartNumber	Part number plus model of the license.	"1234.5678.90"
keyCode	rsEqDeactivationKeyKeyCode	Actual keycode of the license.	3.68456068125389e+29
format	rsEqDeactivationKeyFormat	Contains the format version of the keycode.	0
privKeyType	rsEqDeactivationKeyPrivKeyType	Privileges of the license key. It can be assigned one of the following values: 'Customer Order', 'Demo', 'Factory', 'Beta', 'Evaluation', 'Service'	"Demo"
timeStamp	rsEqDeactivationKeyTimeStamp	Contains the date when the license key was generated given in UTC.	"2012-10-23T11:56:00"
deactivationResponse	rsEqDeactivationKeyDeactivationResponse	Response from deactivation with which a customer can prove that a license has actually been deactivated.	"21872387"
deactivationData	rsEqDeactivationKeyDeactivationData	Deactivation data.	"21872387"

3.3.4.4 License status message table

List of status messages of the license administration module.

JSON Key	SNMP OID	Description	Example Value
message	rsEqLicenseStatusMessageText	Contains the status message text.	"everything is OK"
details	rsEqLicenseStatusMessageDetails	Additional status information, which might contain technical data.	"The Optionkey management has nothing to complain about."
provider	rsEqLicenseStatusMessageProvider	Name of the software module which has reported the status message.	"LicenseProxy"

3.3.5 Examples

SNMP: Get equipment data

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-EQUIPMENT-MIB $HUMS_ADDRESS rsEquipmentObjs
```

Response

```
RS-EQUIPMENT-MIB::rsEquipmentFormatVersion.0 = STRING: "03.04"
RS-EQUIPMENT-MIB::rsEquipmentCreationDate.0 = STRING: "2020-05-06T12:05:39"
RS-EQUIPMENT-MIB::rsEqDevDataDeviceID.0 = STRING: "1201.0002K20-900001-eF"
RS-EQUIPMENT-MIB::rsEqDevDataType.0 = STRING: "CMP 200"
RS-EQUIPMENT-MIB::rsEqDevDataPartName.0 = STRING: "CMP200"
RS-EQUIPMENT-MIB::rsEqDevDataProductDate.0 = STRING: "2019-09-03"
RS-EQUIPMENT-MIB::rsEqDevDataHostName.0 = STRING: "scu-400b7429"
RS-EQUIPMENT-MIB::rsEqStatusMessageText.1 = STRING: "PCSCReader"
RS-EQUIPMENT-MIB::rsEqStatusMessageDetails.1 = STRING: "OMNIKEY AG Smart Card Reader 00
00: 162(has card, exclusively)"
RS-EQUIPMENT-MIB::rsEqStatusMessageProvider.1 = STRING: "OptionKeyLib"
RS-EQUIPMENT-MIB::rsEqHardwareId.1 = INTEGER: 65655
RS-EQUIPMENT-MIB::rsEqHardwareId.2 = INTEGER: 65684
...
RS-EQUIPMENT-MIB::rsEqHardwareProductId.1 = INTEGER: 75860
RS-EQUIPMENT-MIB::rsEqHardwareProductId.2 = INTEGER: 76359
...
RS-EQUIPMENT-MIB::rsEqHardwarePartNumber.1 = STRING: "1206.3222.00"
RS-EQUIPMENT-MIB::rsEqHardwarePartNumber.2 = STRING: "1212.1186.02"
...
RS-EQUIPMENT-MIB::rsEqHardwareSN.1 = STRING: "104440/000"
RS-EQUIPMENT-MIB::rsEqHardwareSN.2 = STRING: "101501"
...
RS-EQUIPMENT-MIB::rsEqHardwarenPartChangeIndex.1 = STRING: "00.00"
RS-EQUIPMENT-MIB::rsEqHardwarenPartChangeIndex.2 = STRING: "02.03"
...
RS-EQUIPMENT-MIB::rsEqHardwareCode.1 = STRING: "2"
RS-EQUIPMENT-MIB::rsEqHardwareCode.2 = STRING: "2"
...
RS-EQUIPMENT-MIB::rsEqHardwareProductDate.1 = STRING: "2018-10-8"
RS-EQUIPMENT-MIB::rsEqHardwareProductDate.2 = STRING: "2019-04-09"
...
RS-EQUIPMENT-MIB::rsEqHardwareReadCode.1 = STRING: "0"
RS-EQUIPMENT-MIB::rsEqHardwareReadCode.2 = STRING: "0"
...
RS-EQUIPMENT-MIB::rsEqHardwareName.1 = STRING: "IPC"
RS-EQUIPMENT-MIB::rsEqHardwareName.2 = STRING: "CMG"
...
RS-EQUIPMENT-MIB::rsEqHardwarePartName.1 = STRING: "CONTROLLER IPC11/4"
RS-EQUIPMENT-MIB::rsEqHardwarePartName.2 = STRING: "CMG DIGITAL BOARD"
...
RS-EQUIPMENT-MIB::rsEqHardwareConnectorName.1 = STRING: "A000300"
RS-EQUIPMENT-MIB::rsEqHardwareConnectorName.2 = STRING: "A000100"
...
RS-EQUIPMENT-MIB::rsEqHwPropertyName.1.1 = STRING: "ExtPartName"
RS-EQUIPMENT-MIB::rsEqHwPropertyName.1.2 = STRING: "ExtProdCode"
RS-EQUIPMENT-MIB::rsEqHwPropertyName.1.3 = STRING: "ExtSerialNo"
RS-EQUIPMENT-MIB::rsEqHwPropertyName.1.4 = STRING: "ExtChangingNo"
RS-EQUIPMENT-MIB::rsEqHwPropertyValue.1.1 = STRING: "IPC114"
RS-EQUIPMENT-MIB::rsEqHwPropertyValue.1.2 = STRING: "IPC114"
RS-EQUIPMENT-MIB::rsEqHwPropertyValue.1.3 = STRING: "IC11432A8416966"
RS-EQUIPMENT-MIB::rsEqHwPropertyValue.1.4 = STRING: "32.A "
RS-EQUIPMENT-MIB::rsEqHwProductId.1 = INTEGER: 75860
RS-EQUIPMENT-MIB::rsEqHwProductId.2 = INTEGER: 76359
RS-EQUIPMENT-MIB::rsEqHwProductPartNumber.1 = STRING: "1212.0209.02"
RS-EQUIPMENT-MIB::rsEqHwProductPartNumber.2 = STRING: "1212.0444.02"
RS-EQUIPMENT-MIB::rsEqHwProductType.1 = STRING: "CMP-PB20A"
RS-EQUIPMENT-MIB::rsEqHwProductType.2 = STRING: "CMP-B500A"
RS-EQUIPMENT-MIB::rsEqHwProductPartName.1 = STRING: "CMP200 BASIC ASSEMBLY"
RS-EQUIPMENT-MIB::rsEqHwProductPartName.2 = STRING: "CMP IF UNIT"
RS-EQUIPMENT-MIB::rsEqOsName.0 = STRING: "Linux"
RS-EQUIPMENT-MIB::rsEqOsVersion.0 = STRING: "2.5.3-2.395"
RS-EQUIPMENT-MIB::rsEqOsImageName.0 = STRING: "Rohde & Schwarz Mobile Radio Tester Linux"
RS-EQUIPMENT-MIB::rsEqOsImageVersion.0 = STRING: "2.5.3-2.395"
RS-EQUIPMENT-MIB::rsEqOsSystemWordSize.0 = INTEGER: 32
RS-EQUIPMENT-MIB::rsEqOsSystemLanguage.0 = STRING: "en_US.UTF-8"
RS-EQUIPMENT-MIB::rsEqOsLinuxKernelRelease.0 = STRING: "4.14.136"
RS-EQUIPMENT-MIB::rsEqOsLinuxKernelVersion.0 = STRING: "#1 SMP PREEMPT Thu Apr 16
09:40:58 UTC 2020"
```

```
RS-EQUIPMENT-MIB::rsEqSoftwareName.1 = STRING: "BASE"
RS-EQUIPMENT-MIB::rsEqSoftwareVersion.1 = STRING: "4.0.5000.395"
RS-EQUIPMENT-MIB::rsEqSoftwareStatus.1 = STRING: "active"
RS-EQUIPMENT-MIB::rsEqSoftwareVendorName.1 = STRING: "Rohde & Schwarz"
RS-EQUIPMENT-MIB::rsEqSoftwareWordSize.1 = INTEGER: 64
RS-EQUIPMENT-MIB::rsEqSoftwareConfiguration.1 = STRING: "Debug"
RS-EQUIPMENT-MIB::rsEqSoftwareReleaseState.1 = STRING: "beta"
RS-EQUIPMENT-MIB::rsEqLicenseDataConformToTVR320.0 = STRING: "true"
RS-EQUIPMENT-MIB::rsEqActiveLicenseOptionIndex.1 = INTEGER: 1125
RS-EQUIPMENT-MIB::rsEqActiveLicenseType.1 = STRING: "CMP-KH500"
RS-EQUIPMENT-MIB::rsEqActiveLicensePartName.1 = STRING: "HW-KEY FOR CMP-B500"
RS-EQUIPMENT-MIB::rsEqActiveLicensePartNumber.1 = STRING: "1212.0738.02"
RS-EQUIPMENT-MIB::rsEqActiveLicenseKeyCode.1 = STRING: "196817654718108654904120262605"
RS-EQUIPMENT-MIB::rsEqActiveLicenseFormat.1 = STRING: "0"
RS-EQUIPMENT-MIB::rsEqActiveLicensePrivKeyType.1 = STRING: "Customer Order"
RS-EQUIPMENT-MIB::rsEqActiveLicenseTimeStamp.1 = STRING: "2019-09-03T12:19:00"
RS-EQUIPMENT-MIB::rsEqActiveLicenseCount.1 = INTEGER: 1
RS-EQUIPMENT-MIB::rsEqActiveLicenseActivationType.1 = STRING: "Permanent"
```

REST: Get equipment data

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/equipment
```

Response


```
{
    "@formatVersion": "03.04",
    "@creationDate": "2020-05-06T12:05:39Z",
    "@checksum": "a761fc5acfb0351c8631390d856c92ff",
    "deviceData": {
      "deviceId": "1201.0002K20-900001-eF",
      "type": "CMP 200",
      "partName": "CMP200",
      "productDate": "2019-09-03",
      "hostName": "scu-400b7429",
      "statusMessageList": [
        {
          "message": "PCSCReader",
          "details": "OMNIKEY AG Smart Card Reader 00 00: 162(has card, exclusively)",
          "provider": "OptionKeyLib"
        }
      ],
      "networkInterfaces": [
        {
          "ipAddress": "10.112.0.181",
          "ipAddressV6": "fe80::ac2d:8609:8319:5abd",
          "macAddress": "00:e0:33:05:c7:9f",
          "description": "Ethernet"
        }
      ]
    },
    "hardwareData": {
      "cpu": "Intel(R) Core(TM) i7-3615QE CPU @ 2.30GHz",
      "ramSize": 16693514240,
      "hardwareList": [
        {
          "id": 65655,
          "productId": 75860,
          "partNumber": "1206.3222.00",
          "sn": "104440/000",
          "partChangeIndex": "00.00",
          "hardwareCode": "2",
          "productDate": "2018-10-8",
          "readCode": "0",
          "name": "IPC",
          "partName": "CONTROLLER IPC11/4",
          "connectorName": "A000300",
          "properties": [
            {
              "name": "ExtPartName",
              "value": "IPC114"
            },
            ...
          ]
        },
        ...
      ],
      "productList": [
        {
          "id": 75860,
          "partNumber": "1212.0209.02",
          "type": "CMP-PB20A",
          "partName": "CMP200 BASIC ASSEMBLY"
        },
        ...
      ]
    },
    "softwareData": {
      "operatingSystem": {
        "name": "Linux",
        "version": "2.5.3-2.395",
        "imageName": "Rohde & Schwarz Mobile Radio Tester Linux",
        "imageVersion": "2.5.3-2.395",
        "systemWordSize": "32",
        "systemLanguage": "en_US.UTF-8",
        "kernelRelease": "4.14.136",
        "kernvelVersion": "#1 SMP PREEMPT Thu Apr 16 09:40:58 UTC 2020"
```

```
    },
    "softwareList": [
      {
        "name": "BASE",
        "version": "4.0.5000.395",
        "status": "active",
        "vendorName": "Rohde & Schwarz",
        "wordSize": "64",
        "configuration": "Debug",
        "releaseState": "beta"
      },
      ...
    ]
  },
  "licenseData": {
    "conformToTvr320": true,
    "activeLicenseList": [
      {
        "optionIndex": 1125,
        "type": "CMP-KH500",
        "partName": "HW-KEY FOR CMP-B500",
        "partNumber": "1212.0738.02",
        "keyCode": "196817654718108654904120262605",
        "format": 0,
        "privKeyType": "Customer Order",
        "timeStamp": "2019-09-03T12:19:00Z",
        "licenseCount": 1,
        "activationType": "Permanent"
      },
      ...
    ],
    "tvr320ManagementData": { ... }
  }
}
```

3.4 System status

SNMP	MIB:	RS-SYSTEM-STATUS-MIB	OID: rsSystemStatusMibModule
REST	GET	{endpoint}/v1/system-status	

This API is to read a system status overview.

JSON Key	SNMP OID	Description	Example Value
globalStatus	rsSystemStatusGlobal	Overall system status. 0 = no data 1 = green (ok) 2 = yellow (warning) 2 = red (error)	1
values	rsSystemStatusTable	A table with detailed status information.	-

3.4.1 System status table

The table is an **array** of entries with the following properties:

JSON Key	SNMP OID	Description	Example Value
id	rsSystemStatusIndex	Index of the rsSystemStatusTable.	1
description	rsSystemStatusDescription	Short description of the status attribute.	"Board temperature"
descriptionExtended	rsSystemStatusDescription2	Additional details (optional).	"The temperature of the board."
type	-	0 = standard, 1 = peristent, 2 = temporary, 3 = runtime	1
value	rsSystemStatusDataValue	Raw value.	1
unit	rsSystemStatusDataUnit	Unit of value.	"K"
upperLimit	rsSystemStatusUpperLimit	Upper limit of the data value.	350
lowerLimit	rsSystemStatusLowerLimit	Lower limit of the data value.	200
reference	rsSystemStatusReference	Optional reference for more information (SNMP OID).	-
severity	rsSystemStatusSeverity	Indication if the raw value is out of range. 1 = green (ok) 2 = yellow (warning) 3 = red (error)	1
timeOfDestruction	-	Only displayed for Temporary Messages	"2020-05-15T08:57:45+0100"
-	rsSystemStatusTimestamp	Timestamp of the status entry ISO 8601 formatted.	"2020-05-15T08:57:45+0100"

3.4.2 Examples

SNMP: Get system status	
Command	`snmpwalk -c $HUMS_COMMUNITY -m +RS-SYSTEM-STATUS-MIB $HUMS_ADDRESS rsSystemStatusMibModule`
Response	`RS-SYSTEM-STATUS-MIB::rsSystemStatusGlobal.0 = INTEGER: 0`

REST: Get system status	
Command	`curl http://$HUMS_ADDRESS/api/hums/v1/system-status`

Response	{
	"globalStatus": 1,
	"values": [
	{
	"id": 13,
	"description": "SCPI fails",
	"descriptionExtended": "Number of failed SCPI Commands",
	"type": 2,
	"value": "125",
	"unit": "Fails",
	"upperLimit": 200,
	"lowerLimit": 0,
	"reference": "1.3.5",
	"severity": 1,
	"timeOfDestruction": "2020-05-15T08:57:45+0100"
	}
	]
	}

3.5 Security

SNMP	MIB:	RS-SECURITY-INFO-MIB	OID: rsSecInfoMibModule
REST	GET	{endpoint}/v1/security-info	

This API allows to read security relevant settings of the R&S instrument.

JSON Key	SNMP OID	Description	Example Value
antimalware.name	rsSecInfoAntiMalwareName	Name of the anti malware software.	"Windows Defender"
antimalware.enabled	rsSecInfoAntiMalwareEnabled	Reports if the anti malware software is enabled. 1 = Enabled 0 = Disabled	false
antimalware.upToDate	rsSecInfoAntiMalwareUpToDate	Reports if the anti malware software is up to date. 1 = up to date 0 = not up to date	true
antimalware.timestamp	rsSecInfoAntiMalwareTimestamp	The timestamp of the anti malware product.	"2020-02-06T05:25:09Z"
firewallEnabled	rsSecInfoFirewallEnabled	This variable provides information on the firewall of the device. The value is the sum if firewall rules are active for public(4), domain(2) and private(1) profiles. Value: 0 = no firewall rules active 1 = private firewall rules active 2 = domain firewall rules active 3 = private and domain firewall rules active 4 = public firewall rules active 5 = public and private firewall rules active 6 = public and domain firewall rules active 7 = public, domain and private firewall rules active	true

3.5.1 Examples

SNMP: Get security information

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-SECURITY-INFO-MIB $HUMS_ADDRESS rsSecInfoMibModule
```

Response

```
RS-SECURITY-INFO-MIB::rsSecInfoAntiMalwareName.0 = STRING: "Windows Defender"
RS-SECURITY-INFO-MIB::rsSecInfoAntiMalwareEnabled.0 = INTEGER: 1
RS-SECURITY-INFO-MIB::rsSecInfoAntiMalwareUpToDate.0 = INTEGER: 1
RS-SECURITY-INFO-MIB::rsSecInfoAntiMalwareTimestamp.0 = STRING: "not supported"
RS-SECURITY-INFO-MIB::rsSecInfoFirewallEnabled.0 = INTEGER: 1
```

REST: Get security information

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/security-info
```

Response

```
{
    "antimalware": {
        "name": "Windows Defender",
        "enabled": true,
        "upToDate": true,
        "timestamp": "not supported"
    },
    "firewallEnabled": true
}
```

3.6 Service

SNMP	MIB:	RS-SERVICE-MIB	OID: rsServiceMibModule
REST	GET	{endpoint}/v1/service	

This API contains service relevant information of the device.

JSON Key	SNMP OID	Description	Example Value
last	rsServiceLastDate	Last date and time the instrument was serviced (ISO 8601 formatted UTC value).	"2020-06-01T08:10:35Z"
required	rsServiceRequired	This varibale shows if the instrument requires service. 0 = not supported 1 = no service required 2 = deviations detected 10 = service required	1
calibration.last	rsServiceLastCalibrationDate	Last date and time the instrument was calibrated (ISO 8601 formatted UTC value).	"2020-08-01T12:00:00Z"
calibration.nextDue	rsServiceNextCalibrationDate	Next date and time the instrument requires calibration (ISO 8601 formatted UTC value).	"2020-08-01T12:00:00Z"
calibration.nextDueExpired	rsServiceNextCalibrationDueExpired	This flag shows if the instrument requires calibration. This flag is set to 1 if the current date and time is later than rsServiceNextCalibrationDue (if set), 0 otherwise.	1
calibration. recommendedCalibrationInterval	rsServiceRecommendedCalibrationInterval	Recommended calibration interval (formatted as ISO 8601 duration).	"P1Y"
uptime	rsServiceUptime	Total uptime of the instrument (operating hours) in hours.	1321

3.6.1 Examples

SNMP: Get service information	
Command	`snmpwalk -c $HUMS_COMMUNITY -m +RS-SERVICE-MIB $HUMS_ADDRESS rsServiceMibModule`
Response	`RS-SERVICE-MIB::rsServiceRequired.0 = INTEGER: 0` `RS-SERVICE-MIB::rsServiceUptime.0 = INTEGER: 0`

REST: Get service information	
Command	`curl http://$HUMS_ADDRESS/api/hums/v1/service`

Response	<pre>{
	  "last": "2020-06-01T08:10:35Z",
	  "required": 1,
	  "calibration": {
	  "last": "2020-08-01T12:00:00Z",
	  "nextDue": "2020-08-01T12:00:00Z",
	  "nextDueExpired": false,
	  "recommendedCalibrationInterval": "P1Y"
	  },
	  "uptime": 1321
	}</pre>

3.7 Storage

This API provides a summary on HDD/SSD along with the *Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T.)* parameters of the storage devices.

3.7.1 Storage device table

SNMP	MIB:	RS-STORAGE-MIB	OID: rsStorageDeviceTable
REST	GET	{endpoint}/v1/storage	

The device table returns an **array** of storage devices (HDDs/SSDs) which are built into the device with the following properties:

JSON Key	SNMP OID	Description	Example Value
name	rsStorageDeviceName	Storage device name.	"/dev/sda"
temperature	rsStorageDeviceTemperature	Temperature of the storage device.	298
type	rsStorageDeviceType	Type of the storage device.	"ATA"
modelFamily	rsStorageDeviceModelFamily	Model family of the storage device.	"Samsung based SSDs"
model	rsStorageDeviceModel	Device model.	"Samsung SSD 850 EVO 500GB"
serialNumber	rsStorageDeviceSerialNumber	Serial number of the storage device.	"S2RBNXAH370921A"
id	rsStorageDeviceId	LU WWN Device ID of the storage device.	"5 002538 d40bdb6db"
firmwareVersion	rsStorageDeviceFirmwareVersion	Firmware version of the storage device.	"EMT02B6Q"
userCapacity	rsStorageDeviceUserCapacity	Capacity of the storage device.	"500107862016"
sectorSizeLogical, sectorSizePhysical	rsStorageDeviceSectorSize	Sector size.	"512/512"
rotationRate	rsStorageDeviceRotationRate	Rotation rate (or Solid State Device)	"Solid State Device"
formFactor	rsStorageDeviceFormFactor	Form factor of the storage device.	"2.5 inches"
ataVersion	rsStorageDeviceAtaVersion	ATA-Version (or empty if not an ATA-device).	"ACS-2, ATA8-ACS T13/1699-D revision 4c"
sataVersion	rsStorageDeviceSataVersion	SATA-Version (or empty if not a SATA device).	"SATA 3.1, 6.0 Gb/s (current: 6.0 Gb/s)"
smartDataTable	-	S.M.A.R.T. table.	-

3.7.1.1 Examples

SNMP: Get storage information

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-STORAGE-MIB $HUMS_ADDRESS rsStorageDeviceTable
```

Response

```
RS-STORAGE-MIB::rsStorageDeviceName.1 = STRING: "/dev/sda"
RS-STORAGE-MIB::rsStorageDeviceTemperature.1 = INTEGER: 32
RS-STORAGE-MIB::rsStorageDeviceType.1 = STRING: "sat"
RS-STORAGE-MIB::rsStorageDeviceModel.1 = STRING: "SFSA240GM1AA2TO-C-OC-216-STD"
RS-STORAGE-MIB::rsStorageDeviceSerialNumber.1 = STRING: "000060146178A7000002"
RS-STORAGE-MIB::rsStorageDeviceFirmwareVersion.1 = STRING: "SBR11015"
RS-STORAGE-MIB::rsStorageDeviceUserCapacity.1 = STRING: "240057409536"
RS-STORAGE-MIB::rsStorageDeviceSectorSize.1 = STRING: "512/512"
RS-STORAGE-MIB::rsStorageDeviceRotationRate.1 = STRING: "0"
RS-STORAGE-MIB::rsStorageDeviceAtaVersion.1 = STRING: "ACS-2 (minor revision not
indicated)"
RS-STORAGE-MIB::rsStorageDeviceSataVersion.1 = STRING: "SATA 3.1"
```

REST: Get storage information

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/storage
```

Response


```
[
    {
        "name": "/dev/sda",
        "temperature": 32,
        "type": "sat",
        "modelFamily": null,
        "model": "SFSA240GM1AA2TO-C-OC-216-STD",
        "serialNumber": "000060146178A7000002",
        "Id": null,
        "firmwareVersion": "SBR11015",
        "userCapacity": 240057409536,
        "sectorSizeLogical": 512,
        "sectorSizePhysical": 512,
        "rotationRate": "0",
        "formFactor": null,
        "ataVersion": "ACS-2 (minor revision not indicated)",
        "sataVersion": "SATA 3.1",
        "smartDataTable": [
            {
                "id": 1,
                "attributeName": "Raw_Read_Error_Rate",
                "dataFlag": 11,
                "value": 100,
                "worst": 100,
                "threshold": 0,
                "type": "Pre-Fail",
                "updated": null,
                "whenFailed": null,
                "rawValue": 0
            },
            ...
        ]
    }
]
```

3.7.2 Storage S.M.A.R.T. table

SNMP	MIB:	RS-STORAGE-MIB	OID: rsStorageSmartTable
REST	GET	{endpoint}/v1/storage	

S.M.A.R.T. is a common technology that analyzes and outputs the state of HDDs and SDDs. It is supported by most hard disk manufacturers. The return takes place via attributes that the manufacturer defines himself. A general definition of attributes that many manufacturers adhere to can be found under [2].

REST returns the S.M.A.R.T. attributes already in the device table.

The SNMP table returns a two-dimensional array of attributes. The first index is the index of the storage device and the second the index of the attribute. Each entry has following properties:

JSON Key	SNMP OID	Description	Example Value
id	rsStorageSmartDataID	ID of the S.M.A.R.T. attribute.	"1"
attributeName	rsStorageSmartDataAttributeName	Name of the S.M.A.R.T. attribute.	"Raw_Read_Error_Rate"
dataFlag	rsStorageSmartDataFlag	Flag of the S.M.A.R.T. attribute.	11
value	rsStorageSmartDataValue	S.M.A.R.T. attribute value.	100
worst	rsStorageSmartDataWorst	S.M.A.R.T. attribute value (worst).	100
threshold	rsStorageSmartDataThreshold	S.M.A.R.T. attribute threshold.	0
type	rsStorageSmartDataType	S.M.A.R.T. attribute type.	"Pre-fail"
updated	rsStorageSmartDataUpdated	S.M.A.R.T. attribute update.	"Always"
whenFailed	rsStorageSmartDataWhenFailed	S.M.A.R.T. attribute when failed.	
rawValue	rsStorageSmartRawValue	S.M.A.R.T. attribute raw value.	"5026"

3.7.2.1 Example

SNMP: Get S.M.A.R.T. data

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-STORAGE-MIB $HUMS_ADDRESS rsStorageSmartTable
```

Response

```
RS-STORAGE-MIB::rsStorageSmartDataID.1.1 = STRING: "1"
RS-STORAGE-MIB::rsStorageSmartDataID.1.2 = STRING: "5"
...
RS-STORAGE-MIB::rsStorageSmartDataAttributeName.1.1 = STRING: "Raw_Read_Error_Rate"
RS-STORAGE-MIB::rsStorageSmartDataAttributeName.1.2 = STRING: "Reallocated_Sector_Ct"
...
RS-STORAGE-MIB::rsStorageSmartDataFlag.1.1 = INTEGER: 11
RS-STORAGE-MIB::rsStorageSmartDataFlag.1.2 = INTEGER: 19
...
RS-STORAGE-MIB::rsStorageSmartDataValue.1.1 = INTEGER: 100
RS-STORAGE-MIB::rsStorageSmartDataValue.1.2 = INTEGER: 100
...
RS-STORAGE-MIB::rsStorageSmartDataWorst.1.1 = INTEGER: 100
RS-STORAGE-MIB::rsStorageSmartDataWorst.1.2 = INTEGER: 100
...
RS-STORAGE-MIB::rsStorageSmartDataThreshold.1.1 = INTEGER: 0
RS-STORAGE-MIB::rsStorageSmartDataThreshold.1.2 = INTEGER: 0
...
RS-STORAGE-MIB::rsStorageSmartDataType.1.1 = STRING: "Pre-Fail"
RS-STORAGE-MIB::rsStorageSmartDataType.1.2 = STRING: "Pre-Fail"
...
RS-STORAGE-MIB::rsStorageSmartDataUpdated.1.1 = ""
RS-STORAGE-MIB::rsStorageSmartDataUpdated.1.2 = ""
...
RS-STORAGE-MIB::rsStorageSmartDataWhenFailed.1.1 = ""
RS-STORAGE-MIB::rsStorageSmartDataWhenFailed.1.2 = ""
...
RS-STORAGE-MIB::rsStorageSmartRawValue.1.1 = STRING: "0"
RS-STORAGE-MIB::rsStorageSmartRawValue.1.2 = STRING: "0"
...
```

3.7.3 Storage partitions

SNMP	MIB:	RS-STORAGE-MIB	OID: rsStoragePartitionTable
REST	GET	{endpoint}/v1/storage/partition	

JSON Key	SNMP OID	Description	Example Value
device	rsStoragePartitionLabel	Label of the partition if supported by the filesystem.	"ACE4_2E81_702C_8491"
partitionId	rsStoragePartitionId	ID of the partition.	1
diskId	rsStoragePartitionDiskId	Disk ID of a partition.	0
letter	rsStoragePartitionDescriptor	Windows drive letter or Linux block device name.	"C:"
filesystem	rsStoragePartitionFilesystem	Fileystem type.	"NTFS"
free	rsStoragePartitionFree	Free space on the partition in byte	77735714816
total	rsStoragePartitionSize	Total amount of space of this partition	0
-	rsStoragePartitionUsed	Used size of the partition in bytes.	1024

3.7.3.1 Example

SNMP: Get partition information

Command
```
snmpwalk -c $HUMS_COMMUNITY -m +RS-STORAGE-MIB $HUMS_ADDRESS rsStoragePartitionTable
```

Response
```
RS-STORAGE-MIB::rsStoragePartitionLabel.1 = STRING: "ACE4_2E81_702C_8491"
RS-STORAGE-MIB::rsStoragePartitionId.1 = INTEGER: 1
RS-STORAGE-MIB::rsStoragePartitionDiskId.1 = INTEGER: 0
RS-STORAGE-MIB::rsStoragePartitionDescriptor.1 = STRING: "C:"
RS-STORAGE-MIB::rsStoragePartitionFilesystem.1 = STRING: "NTFS"
RS-STORAGE-MIB::rsStoragePartitionFree.1 = INTEGER: 77735714816
RS-STORAGE-MIB::rsStoragePartitionSize.1 = INTEGER: 0
RS-STORAGE-MIB::rsStoragePartitionUsed.1 = INTEGER: 1024
```

REST: Get partition information

Command
```
curl http://$HUMS_ADDRESS/api/hums/v1/storage/partition
```

Response
```
[
    {
        "device": "/dev/sda",
        "partitionId": 2,
        "diskId": 0,
        "letter": "/dev/sda2",
        "filesystem": "btrfs",
        "free": 227555090432,
        "total": 239787905024
    },
    ...
]
```

3.8 Utilization data

Contains information on the Utilization of the R&S instrument.

3.8.1 Generic utilization table

SNMP	MIB:	RS-UTILIZATION-MIB	OID: rsUtilizationTable
REST	GET PATCH	{endpoint}/v1/utilization {endpoint}/v1/utilization	 (only activity_tracking is writeable)

Contains a list of how many activities of a certain type were triggered, e.g. mouse interactions, measurements or remote commands. These activities are generic and not standardized between different instruments.

The generic utilization table is an **array** of entries with the following properties:

JSON Key	SNMP OID	Description	Example Value
id	-	Id of the Entry. Needed for PATCH function and History API.	430032289
scope	rsUtilizationDataScope	Scope of the Utilization data. Values are: (1) Common Utilization Data (2) SCPI Remote Utilization Data (3) Local Control Utilization Data (4) License Utilization Data (5) Device Specific Utilization Data	"COMMON"
reference	rsUtilizationDataReference	Optional SNMP OID of the referenced utilization value for further information.	1.3.6.1.4.1.2566.1.2.3.4.5
name	rsUtilizationDataName	Name of the utilization aspect.	"SCPI commands"
unit	rsUtilizationDataUnit	Physical unit of the utilization data value.	"counter", "s", "bytes"
description	rsUtilizationDataDescription	This variable describes in more detail the utilization value.	"Counter of SCPI commands"
value	rsUtilizationDataValue	Current value of the utilization data.	"1234567890"
startupValue	rsUtilizationDataValueStartup	Value of the utilization data on startup of the product.	"1000000000"
-	rsUtilizationDataChange	Value change of the utilization data since last read or startup.	"100"
-	rsUtilizationDataChangeTimestamp	Timestamp of the utilization data (last fetched via SNMP).	"2019-08-09T08:15:10Z"
activityTracking	rsUtilizationDataActivityTracking	This variable is to turn on/off activity tracking.	1

3.8.1.1 Examples

SNMP: Get utilization activities

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-UTILIZATION-MIB $HUMS_ADDRESS rsUtilizationTable
```

Response

```
RS-UTILIZATION-MIB::rsUtilizationDataScope.1 = INTEGER: 1
RS-UTILIZATION-MIB::rsUtilizationDataScope.2 = INTEGER: 2
...
RS-UTILIZATION-MIB::rsUtilizationDataReference.1 = OID: SNMPv2-SMI::zeroDotZero
RS-UTILIZATION-MIB::rsUtilizationDataReference.2 = OID: SNMPv2-SMI::zeroDotZero
...
RS-UTILIZATION-MIB::rsUtilizationDataName.1 = STRING: "SCPI commands"
RS-UTILIZATION-MIB::rsUtilizationDataName.2 = STRING: "Software restarts"
...
RS-UTILIZATION-MIB::rsUtilizationDataValue.1 = STRING: "8"
RS-UTILIZATION-MIB::rsUtilizationDataValue.2 = STRING: "5"
...
RS-UTILIZATION-MIB::rsUtilizationDataValueStartup.1 = STRING: "8"
RS-UTILIZATION-MIB::rsUtilizationDataValueStartup.2 = STRING: "5"
...
RS-UTILIZATION-MIB::rsUtilizationDataChange.1 = STRING: "0"
RS-UTILIZATION-MIB::rsUtilizationDataChange.2 = STRING: "0"
...
RS-UTILIZATION-MIB::rsUtilizationDataChangeTimestamp.1 = STRING: "2020-05-04T11:55:23Z"
RS-UTILIZATION-MIB::rsUtilizationDataChangeTimestamp.2 = STRING: "2020-05-04T11:55:23Z"
...
RS-UTILIZATION-MIB::rsUtilizationEntry.11.1 = INTEGER: 1
RS-UTILIZATION-MIB::rsUtilizationEntry.11.2 = INTEGER: 1
...
```

SNPM: Disable SCPI commands tracking

Command

```
snmpset -c $HUMS_COMMUNITY -m +RS-UTILIZATION-MIB $HUMS_ADDRESS
rsUtilizationDataActivityTracking.1 i 0
```

Response

```
RS-UTILIZATION-MIB:: rsUtilizationDataActivityTracking.1 = INTEGER: 0
```

REST: Get utilization activities

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/utilization
```

Response

```
[
    {
        "id": 430032289,
        "scope": 1,
        "reference": null,
        "name": "SCPI commands",
        "value": 8,
        "startup_value": 8,
        "activity_tracking": true
    },
    ...
]
```

REST: Disable SCPI commands tracking

Command

```
curl -X PATCH -H "Content-Type: application/json" -d
'{"id":13742625452605240000,"activity_tracking":false}'
http://$HUMS_ADDRESS/api/hums/v1/utilization
```

Response

-

3.8.2 SCPI connections table

SNMP	MIB:	RS-UTILIZATION-MIB	OID: rsUtScpiConnectionTable
REST	GET	{endpoint}/v1/utilization/scpi	

Contains a list of SCPI connections from remote hosts and when they occurred.
The table is an **array** of entries with the following properties:

JSON Key	SNMP OID	Description	Example Value
remoteHost	rsUtScpiConnectionRemoteHost	Hostname or IP address of the remote host. In case it is not a LAN based connection this object returns an empty string.	"scu-400b7429"
visaResource	rsUtScpiConnectionVisaResource	VISA resource of the SCPI connection that is active.	"TCPIP::scu-400b7429::inst1::INSTR"
established	rsUtScpiConnectionEstablished	UTC time the VISA connection was established.	"2020-05-04T10:05:57Z"
closed	rsUtScpiConnectionClosed	UTC time when the SCPI connection was closed.	"2020-05-04T10:05:57Z"
commandsExecuted	rsUtScpiConnectionCommandsExecuted	Number of SCPI commands executed with the SCPI connection.	42
errors	rsUtScpiConnectionErrorsDetected	Number of SCPI errors with the SCPI connection.	0

3.8.2.1 Examples

SNMP: Get SCPI connections

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-UTILIZATION-MIB $HUMS_ADDRESS rsUtScpiConnectionTable
```

Response

```
RS-UTILIZATION-MIB::rsUtScpiConnectionRemoteHost.1 = STRING: "scu-400b7429"
RS-UTILIZATION-MIB::rsUtScpiConnectionRemoteHost.2 = STRING: "scu-400b7429"
...
RS-UTILIZATION-MIB::rsUtScpiConnectionVisaResource.1 = STRING: "TCPIP::scu-
400b7429::inst1::INSTR"
RS-UTILIZATION-MIB::rsUtScpiConnectionVisaResource.2 = STRING: "TCPIP::scu-
400b7429::inst0::INSTR"
...
RS-UTILIZATION-MIB::rsUtScpiConnectionEstablished.1 = STRING: "2020-05-04T10:05:57Z"
RS-UTILIZATION-MIB::rsUtScpiConnectionEstablished.2 = STRING: "2020-05-04T10:05:57Z"
...
RS-UTILIZATION-MIB::rsUtScpiConnectionClosed.1 = STRING: "2020-05-04T10:05:57Z"
RS-UTILIZATION-MIB::rsUtScpiConnectionClosed.2 = STRING: "2020-05-04T10:05:57Z"
...
RS-UTILIZATION-MIB::rsUtScpiConnectionCommandsExecuted.1 = INTEGER: 0
RS-UTILIZATION-MIB::rsUtScpiConnectionCommandsExecuted.2 = INTEGER: 0
...
RS-UTILIZATION-MIB::rsUtScpiConnectionErrorsDetected.1 = INTEGER: 0
RS-UTILIZATION-MIB::rsUtScpiConnectionErrorsDetected.2 = INTEGER: 0
...
```

REST: Get SCPI connections

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/utilization/scpi
```

Response

```
[
    {
        "remote_host": "scu-400b7429",
        "visa_resource": "TCPIP::scu-400b7429::inst1::INSTR",
        "established": "2020-05-04T14:21:21Z",
        "commands_executed": 0,
        "errors": 0,
        "closed": "2020-05-04T14:21:21Z"
    },
    ...
]
```

3.8.3 History

SNMP	-	
REST	GET	{endpoint}/v1/utilization/history/:id?start=<integer>&end=<integer>&resolution=<integer>

HUMS records in a certain interval whether a utilization takes place or not. The interval, as well as the number of time stamps, can be defined via query parameters.

This API has no SNMP interface, because there is too much information reported for the SNMP.

The REST API has the following parameters:

Parameter	Description	Default
:id	ID of the utilization record to be queried	-
start	Start of the utilization period in seconds since 1970-01-01	Now - 30 Days
end	End of the utilization period in seconds since 1970-01-01	Now
resolution	Resolution of the data bins in seconds	360 (1h)

The API returns two arrays of equal length, one for the timestamps and one how often a utilization has taken place within the timestamp and the previous one. The timestamps are defined in UNIX time (seconds since 1970-01-01).

3.8.3.1 Example

REST: Get History

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/utilization/history
```

Response

```
{
    "timestamps": [
        1588578040,
        1588578940,
        ...
    ],
    "activity": [
        0,
        11400,
        ...
    ]
}
```

3.9 Device history

SNMP	MIB:	RS-DEVICE-HISTORY-MIB	OID: rsDeviceHistoryEventTable
REST	GET	{endpoint}/v1//device-history	

This API provides a device event history as an **array**. The last 10,000 events will be returned.

JSON Key	SNMP OID	Description	Example Value
id	rsDeviceHistoryEventIndex	Table index of a device history event.	2
timestamp	rsDeviceHistoryEventTimestamp	Timestamp of the device history event (ISO 8601 formatted).	"2020-07-29T12:47:18Z"
message	rsDeviceHistoryEventMessage	Text message of a device history event.	"Test"
details	rsDeviceHistoryEventDetails	Details of the device history event.	"Detailed description"
-	rsDeviceHistoryEventSeverity	Severity of the device history event. 1 = info 2 = warning 3 = error	1

3.9.1 Examples

SNMP: Get SCPI connections

Command

```
snmpwalk -c $HUMS_COMMUNITY -m +RS-DEVICE-HISTORY-MIB $HUMS_ADDRESS
rsDeviceHistoryEventTable
```

Response

```
RS-DEVICE-HISTORY-MIB::rsDeviceHistoryEventIndex.1 = INTEGER: 2
RS-DEVICE-HISTORY-MIB::rsDeviceHistoryEventTimestamp.1 = STRING: "2020-07-29T12:47:18Z"
RS-DEVICE-HISTORY-MIB::rsDeviceHistoryEventMessage.1 = STRING: "Test"
RS-DEVICE-HISTORY-MIB::rsDeviceHistoryEventDetails.1 = STRING: "Detailed description"
RS-DEVICE-HISTORY-MIB::rsDeviceHistoryEventSeverity.1 = INTEGER: 1
```

REST: Get device history

Command

```
curl http://$HUMS_ADDRESS/api/hums/v1/device-history
```

Response

```
[
  {
    "id": 2,
    "timestamp": "2020-07-29T12:47:18Z",
    "message": "Test",
    "details": "Detailed description"
  }
]
```

4 Literature

[1] D. R. Mauro and K. J. Schmidt, Essential SNMP, Sebastopol, CA: O'Reilly, 2001.

[2] J. Hatfield, "SMART Attribute Annex," 30 September 2005. [Online]. Available: http://www.t13.org/documents/uploadeddocuments/docs2005/e05148r0-acs-smartattributesannex.pdf.

Rohde & Schwarz

The Rohde & Schwarz electronics group offers innovative solutions in the following business fields: test and measurement, broadcast and media, secure communications, cybersecurity, monitoring and network testing. Founded more than 80 years ago, the independent company which is headquartered in Munich, Germany, has an extensive sales and service network with locations in more than 70 countries.

www.rohde-schwarz.com

Certified Quality Management
ISO 9001

Rohde & Schwarz training

www.training.rohde-schwarz.com

PAD-T-M: 3573.7380.02/04.00/EN

Rohde & Schwarz customer support

www.rohde-schwarz.com/support

GFM336 | Version 0e | 01.2021
Application Note | Instrument Health & Utilization Monitoring
Data without tolerance limits is not binding | Subject to change

www.rohde-schwarz.com