Software interface

There are several software interfaces available to monitor the status of the t.RECS® system. These are the Management WebGUI and a REST API providing XML based monitoring and management functionality.

Management WebGUI

The Management WebGUI is established on every t.RECS® unit. Accessible by any known browser on the assigned IP address and the default port 80. The following views are dependent on the device and assembly.

In general these symbols have the following meaning on every page:

✔	Everything is OK. Also indicated by a green line in a graph.
⚠	Warnung. Something is wrong, but the system is still fully functional. The system has to be checked so the problem doesn't get worse. Indicated by a yellow line in a graph.
✖	Critical Error. The system must be checked immediately and maybe has to be shut down to prevent hardware damage. indicated by a red line in a graph.

Figure 1 shows the first call of the Management WebGUI. It is organized into three columns. The first is on the left-hand side and contains the following:

Overview: General overview of all managed RCUs, RPUs, installed nodes and health status
Management: Selection of every managed RCU and RPU in the rack with a sensor view button for the Arneb
Global settings: IP filter and firmware update
Log: Logs from the management software about system health and java messages. The logs can be downloaded as a zipfile

The second colum contains the buttons and sliders to manipulate the system. While the third colum is mostly for history information like power usage and temperature graphs.

Overview

All units that are installed in the rack and that are managed by the software are summarized on this page. The total power usage is summed up over all managed units.

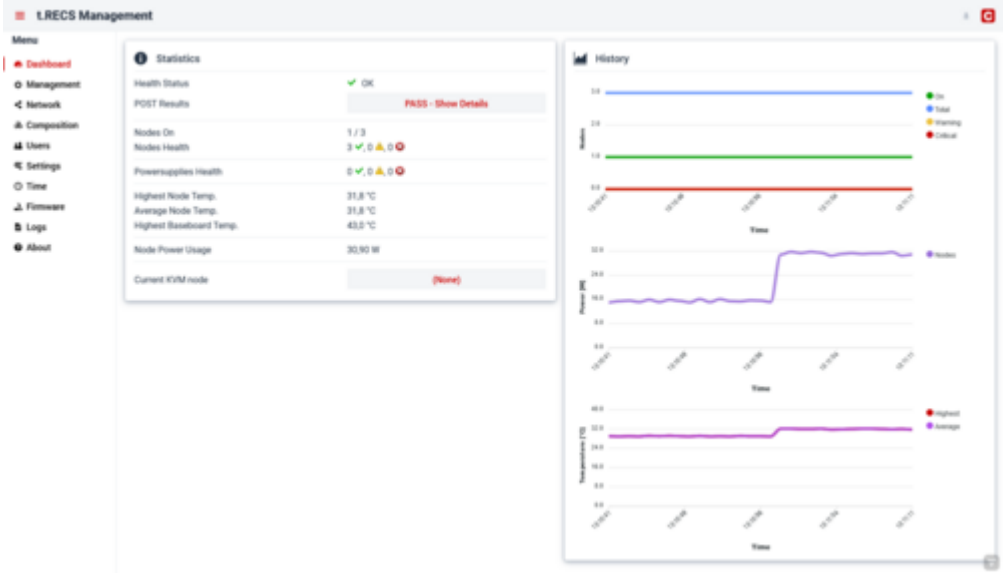

Fig. ##

Management

An overview of the selected unit can be seen in this tab. The fans can be regulated by dragging the slider to the desired percentage. And multiple nodes can be selected. By klicking on a node theNode management page of the node is shown.

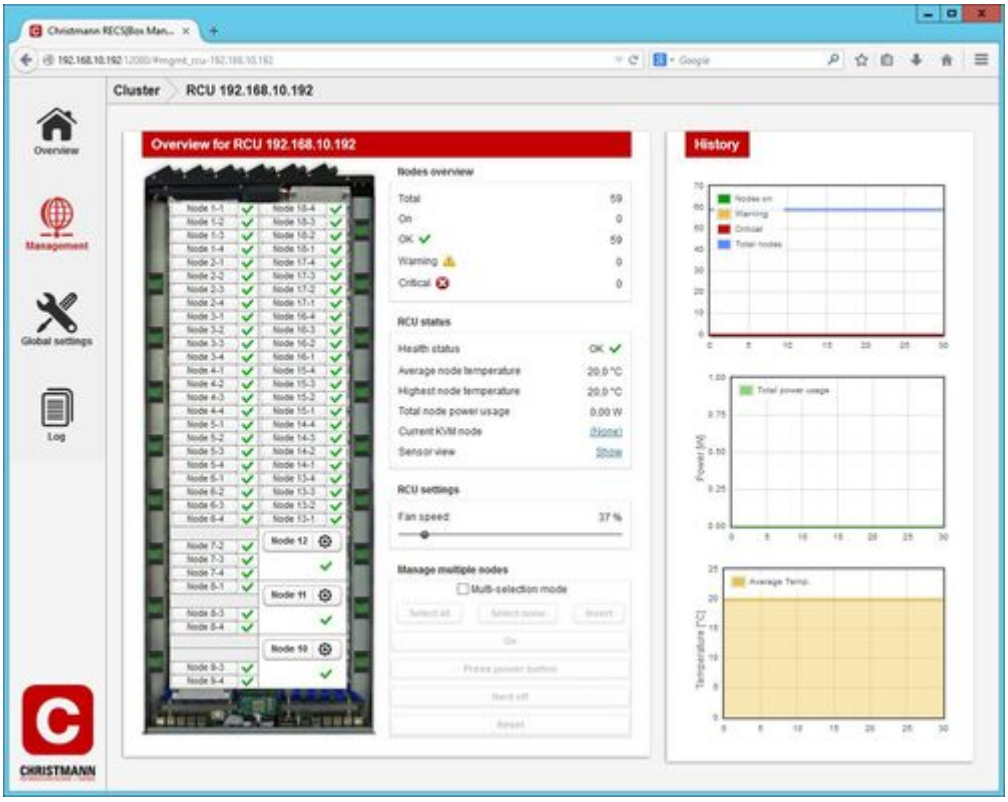

Fig. ##

A quick menu to control a node can be opened by klicking on the gear next to an CXP node. In this menu the node can be switched on and off and the KVM can be switched to the node.

Fig. ##

Apalis nodes do not show a management pop-up button due to size constraints.
Click on the node button while pressing the “Shift” key to open the management pop-up instead of navigating to the node view.

When pressing the “Shift” key while clicking, the “Select all” and “Select none” buttons select only nodes currently on or nodes currently off, respectively.

Node management

On this page the selected node can be controlled and detailed status values and graphs can be seen. By klicking on the arrow, pointing downwards in the upper bar next to the nodename, the other nodes of the unit can be chosen.

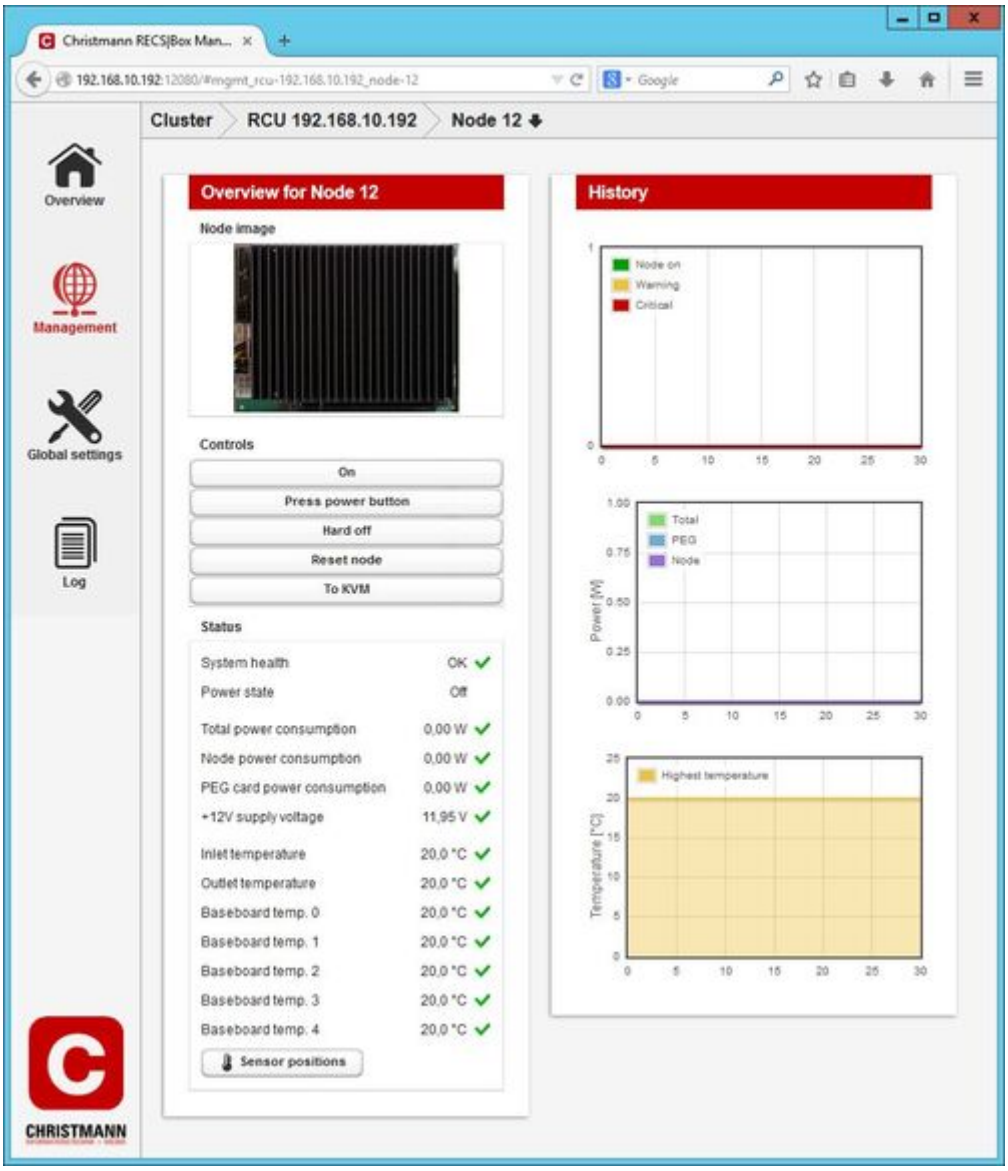

Fig. ##

Global settings

All IPs that are allowed to access the Nagios interface have to be listet here.
The firmware for the whole RECS®|Box can be uploaded here by klicking on the “Upload Firmware File” button and selecting the file. The update-process starts right after the file was uploaded.
For the update process **all modules will be powered off!**

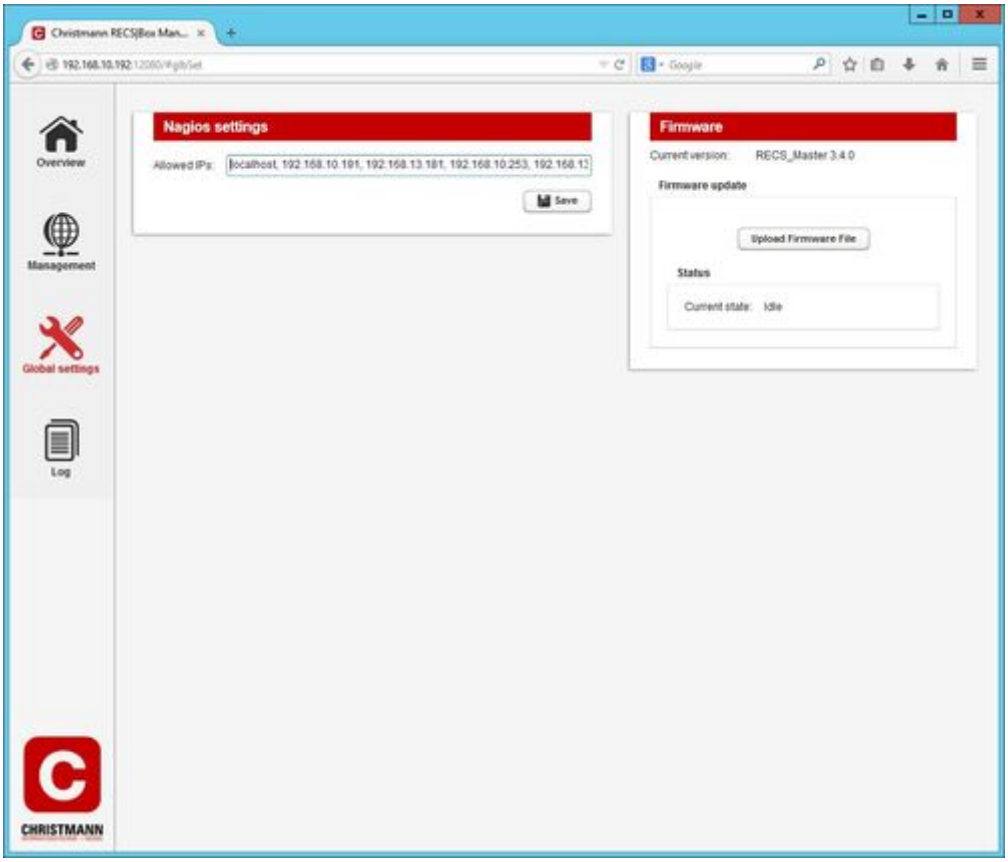

Fig. ##

Log viewer

In the system healths tab of the log page the status changes of the sensors, fan and boards can be seen.

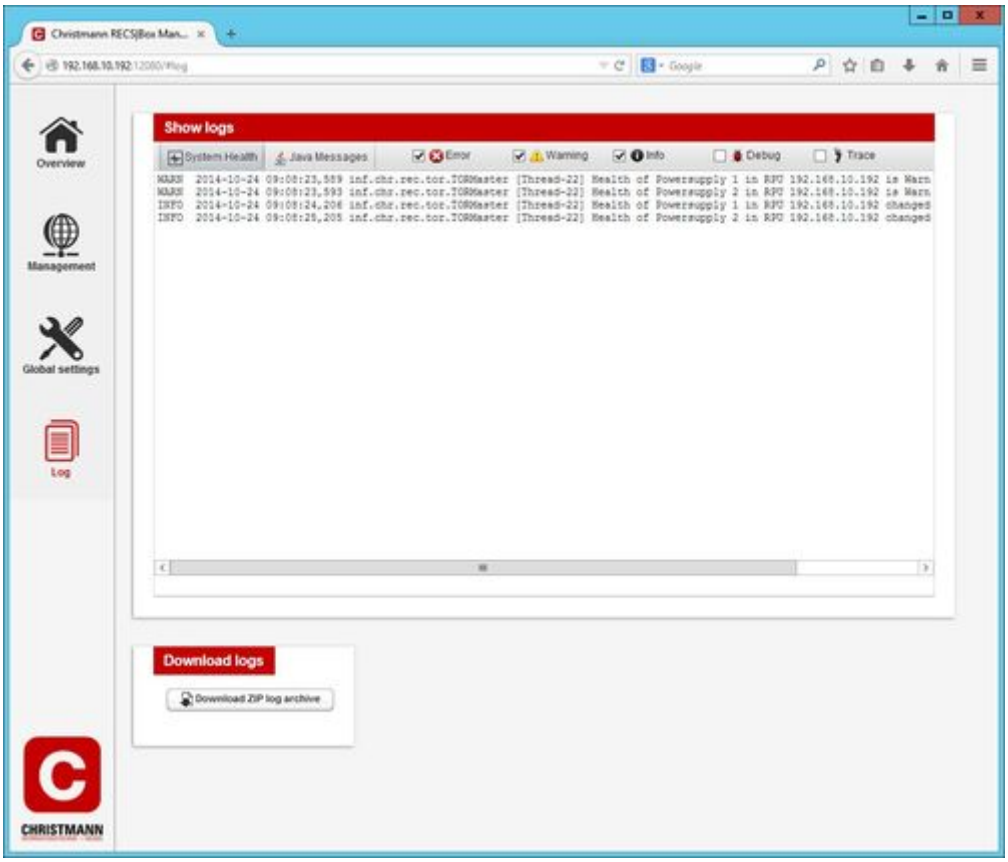

Fig. ##

In the java tab of the log page all messages regarding the software can be found.

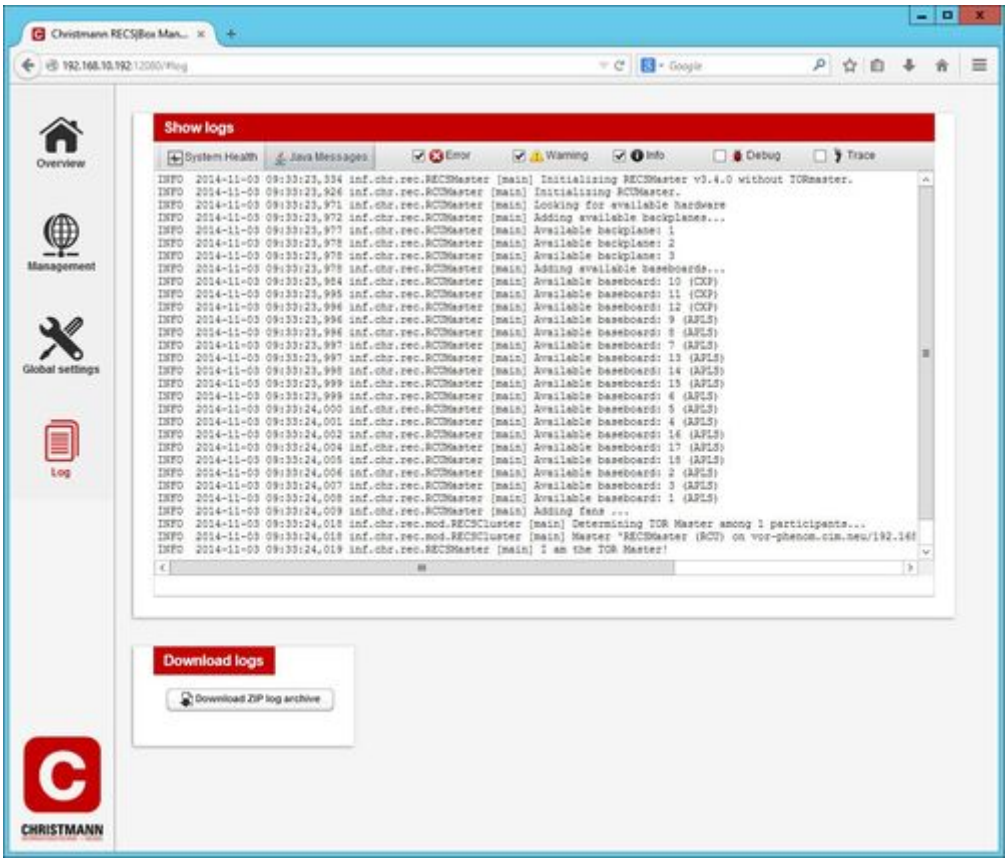

Fig. ##

Several filters can be set for both tabs at the top.
At the bottom the whole log can be downloaded as a ZIP file containing the individual logfiles.

Redfish API

The documentation of the RECS®|Box Redfish API can be seen at Github.

REST API

Access

The RECS®|Box Management API is accessible via the IP-Address or the hostname of the TOR-Master of the cluster. The basic URL of the API has the format `https://TOR-Master/REST/` or `http://TOR-Master/REST/`.

Accessing the REST API requires HTTP Basic authentication. The authenticated user has to be in the “Admin” or “User” group to be able to execute the POST/PUT management calls.

Components

The RECS®|Box Management API makes all hardware components in the cluster available as XML trees in software. The following components are supported by the API:

Attribute	Description
node	A single node
backplane	A backplane can be equipped with zero or more baseboards
baseboard	A baseboard can be equipped with zero or more nodes
rcu	A RECS®|Box Computing Unit (RCU) can be equipped with zero or more baseboards
rack	A rack consists of several RCUs

Many resources also return lists of components. These are named according to the scheme <component name>List (e.g. nodeList, rcuList) and contain the elements of the list.

Example of a backplaneList:

```
<backplaneList>
<backplane position="1" id="RCU_84055620466592_BP_1"
infrastructurePower="0.0"
lastSensorUpdate="1465470151268">
<temperatures>24.0</temperatures>
<temperatures>25.0</temperatures>
<temperatures>26.0</temperatures>
<temperatures>27.0</temperatures>
<temperatures>28.0</temperatures>
</backplane>
```

```
</backplaneList>
```

Node

Example XML:

```
<node baseboardPosition="0" maxPowerUsage="44"
actualNodePowerUsage="32.426884399865166"
actualPEGPowerUsage="15.12053962324833" actualPowerUsage="47.54742402311349"
architecture="x86"
baseboardId="RCU_84055620466592_BB_1" health="OK"
id="RCU_84055620466592_BB_1_0" inletTemperature="20.0"
lastSensorUpdate="1465470151268" macAddressCompute="70:b3:d5:56:40:48"
outletTemperature="20.0" state="1"
highestTemperature="20.0" voltage="12.072700851453936"/>
```

The following table shows the possible attributes (some are optional) and their meaning:

Attribute	Description	Unit	Data type
`id`	Unique ID for referencing the component	-	String
`actualPowerUsage`	Actual power consumption of a node (Node + PEG)	W	Double
`actualNodePowerUsage`	Actual power consumption of a node (Node only)	W	Double
`actualPEGPowerUsage`	Actual power consumption of a PEG card	W	Double
`maxPowerUsage`	Maximum power the node can draw	W	Integer
`baseboardId`	ID of the baseboard which hosts the node	-	String
`baseboardPosition`	Position of the node on the baseboard	-	Integer
`state`	Power state of the node (0=Off, 1=On, 2=Soft-off, 3=Standby, 4=Hibernate)	-	Integer
`architecture`	Architecture (x86, arm, UNKNOWN)	-	String
`health`	Health status of the node (OK, Warning, Critical)	-	String
`inletTemperature`	Temperature of the inlet air	°C	Double
`outletTemperature`	Temperature of the outlet air	°C	Double
`highestTemperature`	Highest temperature measured on the node's baseboard	°C	Double
`voltage`	Supply voltage of the baseboard	V	Double
`lastSensorUpdate`	Timestamp of the last sensor update	ms	Long
`macAddressCompute`	MAC address of the NIC connected to the compute network (optional)	-	String
`macAddressMgmt`	MAC address of the NIC connected to the management network (optional)	-	String

In accordance to the component node the API offers nodeList which returns multiple instances of node.

Backplane

Example XML:

```
<backplane position="1" id="RCU_84055620466592_BP_1"
infrastructurePower="0.0"
lastSensorUpdate="1465470151268">
<temperatures>24.0</temperatures>
<temperatures>25.0</temperatures>
<temperatures>26.0</temperatures>
<temperatures>27.0</temperatures>
<temperatures>28.0</temperatures>
</backplane>
```

The attributes have the following meaning:

Attribute	Description	Unit	Data type
`id`	Unique ID for referencing the component	-	String
`position`	Position of the backplane in the RECS®|Box Computing Unit	-	Integer
`infrastructurePower`	Power usage of the infrastructure components on the backplane	W	Double
`lastSensorUpdate`	Timestamp of the last sensor update	ms	Long
`temperatures`	List of temperatures measured on the backplane	°C	Double

In accordance to the component backplane the API offers backplaneList which returns multiple instances of backplane.

Baseboard

Example XML:

```
<baseboard rcuPosition="6" baseboardType="APLS" id="RCU_84055620466592_BB_6"
infrastructurePower="9.8"
lastSensorUpdate="1465470151268" rcuId="RCU_84055620466592">
<nodeId>RCU_84055620466592_BB_6_1</nodeId>
<nodeId>RCU_84055620466592_BB_6_2</nodeId>
<nodeId>RCU_84055620466592_BB_6_3</nodeId>
<temperatures>20.0</temperatures>
<temperatures>20.0</temperatures>
<temperatures>20.0</temperatures>
<temperatures>20.0</temperatures>
<temperatures>20.0</temperatures>
</baseboard>
```

The attributes have the following meaning:

Attribute	Description	Unit	Data type
`id`	Unique ID for referencing the component	-	String
`rcuId`	Unique ID of the RECS®|Box Computing Unit hosting the baseboard	-	String
`rcuPosition`	Position of the baseboard inside the RECS®|Box Computing Unit	-	Integer

Attribute	Description	Unit	Data type
infrastructurePower	Power usage of the infrastructure components on the baseboard	W	Double
lastSensorUpdate	Timestamp of the last sensor update	ms	Long
baseboardType	Type of the baseboard (CXP, APLS)	-	String
nodeId	List of IDs of the nodes installed on the baseboard	-	String
temperatures	List of temperatures measured on the backplane	°C	Double

In accordance to the component baseboard the API offers baseboardList which returns multiple instances of baseboard.

RCU

Example XML:

```
<rcu rcuType="ANTARES" fanSpeed="60" fanProfile="adjust_by_temperature"
rackId="RCK_1" name="RECSMaster (RCU) on 192.168.56.195" rackPosition="0"
id="RCU_84055620466592" lastSensorUpdate="1465470151268">
<backplaneId>RCU_84055620466592_BP_1</backplaneId>
<baseboardId>RCU_84055620466592_BB_1</baseboardId>
<baseboardId>RCU_84055620466592_BB_2</baseboardId>
<baseboardId>RCU_84055620466592_BB_3</baseboardId>
<baseboardId>RCU_84055620466592_BB_4</baseboardId>
<baseboardId>RCU_84055620466592_BB_5</baseboardId>
<baseboardId>RCU_84055620466592_BB_6</baseboardId>
</rcu>
```

The attributes have the following meaning:

Attribute	Description	Unit	Data type
id	Unique ID for referencing the component	-	String
rackId	ID of the rack which hosts the RECS®|Box Computing Unit	-	String
rackPosition	Position of the RECS®|Box Computing Unit in the rack	-	Integer
name	Name of the RECS®|Box Computing Unit	-	String
ip	IP address of the RECS®|Box Computing Unit	-	String
rcuType	Type of the RECS®|Box Computing Unit (SIRIUS, ARNEB, ANTARES)	-	String
kvmNode	ID of the node to which the KVM system is switched (optional)	-	String
fanSpeed	Current speed setting of the fans in the RECS®|Box Computing Unit	%	Integer
fanProfile	Current fan profileof the RECS®|Box Computing Unit	%	Integer
lastSensorUpdate	Timestamp of the last sensor update	ms	Long
backplaneId	List of IDs of backplanes which are installed in the RECS®|Box Computing Unit	-	String
baseboardId	List of IDs of baseboards which are installed in the RECS®|Box Computing Unit	-	String

In accordance to the component rcu the API offers rcuList which returns multiple instances of rcu.

Rack

Example XML:

```
<rack description="Default rack" id="RCK_1">
<rcuId>RCU_84055620466592</rcuId>
</rack>
```

The attributes have the following meaning:

Attribute	Description	Unit	Data type
`id`	Unique ID for referencing the component	-	String
`description`	Description of the rack	-	String
`rcuId`	List of IDs of RECS®|Box Computing Units which are installed in the rack	-	String

In accordance to the component rack the API offers rackList which returns multiple instances of rack.

Resources

The resources are split into monitoring resources (for pure information gathering) and management resources (for changing the system configuration or state).

Monitoring

For monitoring the following resources are available:

Attribute	Description	HTTP Method
`/node`	Returns a nodeList with all nodes of the cluster	GET
`/node/{node_id}`	Returns information about the node with the given ID	GET
`/baseboard`	Returns a baseboardList with all baseboards of the cluster	GET
`/baseboard/{baseboard_id}`	Returns information about the baseboard with the given ID	GET
`/baseboard/{baseboard_id}/node`	Returns a nodeList with all nodes that are installed on the baseboard with the given ID	GET
`/backplane`	Returns a backplaneList with all backplanes of the cluster	GET
`/backplane/{backplane_id}`	Returns information about the backplane with the given ID	GET
`/rcu`	Returns an rcuList with all RECS®|Box Computing Units of the cluster	GET

Attribute	Description	HTTP Method
`/rcu/{rcu_id}`	Returns information about the RECS®|Box Computing Unit with the given ID	GET
`/rcu/{rcu_id}/baseboard`	Returns a baseboardList with all baseboards that are installed in the RECS®|Box Computing Unit with the given ID	GET
`/rcu/{rcu_id}/backplane`	Returns a backplaneList with all backplanes that are installed in the RECS®|Box Computing Unit with the given ID	GET
`/rcu/{rcu_id}/node`	Returns a nodeList with all nodes that are installed in the RECS®|Box Computing Unit with the given ID	GET
`/rack`	Returns a rackList with all racks of the cluster	GET
`/rack/{rack_id}`	Returns information about the rack with the given ID	GET
`/rack/{rack_id}/rcu`	Returns a rcuList with all RECS®|Box Computing Units that are installed in the rack with the given ID	GET

Management

The management of individual components can be found under the “manage” path of the component.

Attribute	Description	HTTP method	Parameter
`/node/{node_id}/manage/power_on`	Turns on the node with the given ID and returns updated node XML	POST	
`/node/{node_id}/manage/power_off`	Turns off the node with the given ID and returns updated node XML	POST	
`/node/{node_id}/manage/reset`	Resets the node with the given ID and returns updated node XML	POST	
`/node/{node_id}/manage/select_kvm`	Switches the KVM port of the RECS®|Box Computing Unit containing the node to the node with the given ID and returns updated node XML	PUT	
`/rcu/{rcu_id}/manage/set_fans`	Sets the overall fan speed of the RCU with the given ID and returns the curent status of the RCU	PUT	percent={value}

Attribute	Description	HTTP method	Parameter
`/rcu/{rcu_id}/manage/set_fan_profile`	Sets the fan profile of the RCU with the given ID and returns the curent status of the RCU (Possible values: manual, increase_by_temperature, adjust_by_temperature)	PUT	percent={value}

Errors

Information about the success or failure of management requests are returned via HTTP status codes. Please have a look at RFC2616 for an overview about the defined HTTP status codes.

Prometheus

A prometheus exporter is built-in and can be enabled. It is accessable at `https://TOR-Master/metrics/` or `http://TOR-Master/metrics/` and needs a http basic authentication.

The big advantage of the Prometheus exporter compared to other APIs is that it dynamically exports its own metrics and thus, additional metrics can be added or removed during runtime after changing or hotplugging hardware. This allows to export only metrics of those microservers that are plugged in. As the RECS®|Box has a modular approach and every RECS®|Box can be equipped with different carrier blades and microserver configurations, this approach is of high relevance. Using traditional monitoring tools that don't support the export of dynamic metrics needs regular manual changes of the configuration files which is annoying.

Prometheus Configuration

Prometheus needs very little configuration to automatically parse all information and write it into a database. This makes all metrics easily accessible.

```
- job_name: 'RECS_Master'
  scrape_interval: 1s
  scrape_timeout: 1s
  static_configs:
   - targets: ['192.168.0.100']
  basic_auth:
    username: 'user'
    password: 'password'
```

Grafana Dashboard

It is recommended to use Grafana as a graphical dashboard to read out these captured metrics. A

pre-build Grafana dashboard is publicly available at https://grafana.com/grafana/dashboards/14622. It can be integrated in Grafana using the “Import” function. It automatically reads the available metrics from the database and dynamically adapts to the number of available microservers, see the following picture:

Fig. 11