PROGRAMMER GUIDE

SYMKLOUD System Manager API v3.0

Doc. Rev. 3.0

POSSIBILITIES START HERE kontron

SYMKLOUD SYSTEM MANAGER API V3.0 – PROGRAMMER GUIDE

Disclaimer

Kontron would like to point out that the information contained in this manual may be subject to alteration, particularly as a result of the constant upgrading of Kontron products. This document does not entail any guarantee on the part of Kontron with respect to technical processes described in the manual or any product characteristics set out in the manual. Kontron assumes no responsibility or liability for the use of the described product(s), conveys no license or title under any patent, copyright or mask work rights to these products and makes no representations or warranties that these products are free from patent, copyright or mask work right infringement unless otherwise specified. Applications that are described in this manual are for illustration purposes only. Kontron makes no representation or warranty that such application will be suitable for the specified use without further testing or modification. Kontron expressly informs the user that this manual only contains a general description of processes and instructions which may not be applicable in every individual case. In cases of doubt, please contact Kontron.

Kontron Ag

Lise-Meitner-Str. 3-5
86156 Augsburg
Germany
www.kontron.com

Revision History

Revision	Brief Description of Changes	Date of Issue
1.0	API V3.0 release	September 2016
2.0	Added Start SNMP and Stop SNMP commands. Fixed some cURL commands examples. SystemInformation and Logs are now secured commands.	February 2018
3.0	Added Terminate System Manager command Added Get/Set Shelf Manager Mode commands	May 2018

Customer Support

Find Kontron contacts at http://www.kontron.com/support.

Customer Service

As a trusted technology innovator and global solutions provider, Kontron extends its embedded market strengths to a services portfolio that allows companies to break the barriers of traditional product lifecycles. Proven product expertise coupled with collaborative and highly-experienced support enables Kontron to provide exceptional peace–of-mind for the building and maintenance of successful products.

For more details on Kontron's service offerings, including enhanced repair services, extended warranty, the Kontron training academy visit http://www.kontron.com/support-and-services/services.

Customer Comments

If you have any difficulty using this guide, find an error or wish to provide feedback, please contact Kontron Support. Please provide details of any errors you may find. We will correct these errors or problems as soon as possible and post the revised user guide on our website.

Thank you.

Table of Contents

List of Acronyms

API	Application Programming Interface
BMC	Base Management Controller
ECC	Error Checking and Correction
FRU	Field Replaceable Unit
IOL	IPMI-Over-LAN
IPMI	Intelligent Platform Management Interface
KVM	Keyboard Video Mouse
MEI	Management Engine Interface
SEL	System Event Log
ShMC	Shelf Management Controller

1/ About this manual

The purpose of this document is to instruct users about the API and its various logical steps, which can be performed to effectively achieve high-level goals. This document describes how to use API functions and provides the references for all API 3.0 calls, parameters and return values.

The examples shown in this document use "curl" as a command line tool to launch the HTTP API calls. The only exception is for retrieving files from the server. In this case, the command "wget" is used.

1.1. About System Manager API

The SYMKLOUD System Manager provides a rich set of features through its API. This allows you to create scripts to interact with an individual platform.

1.2. About the SYMKLOUD platform

The Kontron SYMKLOUD Series are massively scalable SDN/NFV-enabled converged modular platforms for carrier, cable, and cloud infrastructure deployments. Achieve highly efficient application workloads for Video/Content Delivery, Big Data/IoT, Mobile/Telco, and Cloud/Hosting services. It has the following characteristics:

- Designed from the ground up to integrate switching, load balancing and processing in a 3-in-1 modular approach
- Power efficiency for significant OPEX savings
- Simplified One-Click updates
- Seamless clustering and rack scalability
- Designed for cloud service providers and next-generation data centers

The SYMKLOUD platform is composed of two hubs to provide hardware redundancy. In this model, one hub is "Active" and the other is "Standby" with role switchover if a problem occurs on the "Active" Shelf Manager.

2/ Managing authentication

As of API v2.0, users must authenticate with their usernames and passwords for all API calls, except for the following calls, which do not require authentification:

- "Version"
- "WebsiteConfig"

NOTICE

The call "Version" does not require authentication. This allows users to confirm a version change following an upgrade, unimpeded by changes to the authentication method.

The following is the main principle of authentication implemented in System Manager:

1. Users must authenticate with their usernames and passwords using "Basic HTTP Authentication" to the API call "/Auth". The following shows this call:

   ```
   curl --user <username>:<password> http://<platform ip>:9090/v3/Auth
   ```

2. Upon successful authentication, the server responds with the following format :

   ```
   { "result": "Success", "token": "4eb8f7eb-7d6e-493f-b5c3-23235bc3f576", "roleKey": "admin",
   "roleText": "Administrator"}
   ```

   The token will stop being valid after 30 minutes of inactivity.

3. All subsequent access to the API must use the authenticated username and token with "Basic HTTP Authentication" to make API calls:

   ```
   curl --user admin:4eb8f7eb-7d6e-493f-b5c3-23235bc3f576 http://<platform
   ip>:9090/v3/Platform/<platform number>/AvailableModels
   ```

For possible values for URL resources, refer to 10.1 URL resource values.

3/ Managing security

You can enable secure communication using the HTTPS protocol. The HTTP protocol is enabled by default.

When using the HTTPS protocol, the server uses a self-signed certificate by default. You can upload your own certificate, key and trusted CA certificate. The trusted CA certificate is optional. It is used to secure communication between SYMKLOUD platforms.

You can enable HTTPS secured communication as follows:

1. If custom certificates are to be used (optional), upload the certificates to the platform:

```
curl --user <username>:<token> -k https://<platform ip>/v3/Security/Certificates -F
key=@<filename> -F cert=@filename -F cacert=@filename
```

NOTICE

curl "-k" option is used to perform "insecure" SSL connection. This is required since the platform will use a self-signed certificate.

2. Enable HTTPS:

```
curl --user <username>:<token> http://<platform ip>/v3/Security/HTTPS -d
"enabled=true&strictMode=false"
```

enabled (true/false): Enable or disable HTTPS protocol.

strictMode (true/false): Enable/verify certificate validation in multi-platform communications

To securely upload new certificates, HTTPS must be enabled. Once the new certificates are uploaded, you need to disable and re-enable HTTPS for them to take effect.

For possible values for URL resources, refer to 10.1 URL resource values.

4/ Managing users

In addition to authentication, you can manage user permissions by role:

- **administrator**: The administrator role provides access to all calls.
- **IPMI**: The IPMI role uses a fixed named (smmgnt). Only the password of this user can be changed. This user is only used for IPMI access.
- **technician**: The technician role provides access to all calls, except for the calls related to user management, security and the SEL settings.

If users attempt an API call without role authorization, they will get the following error response from the API:

```
{"status":"Failure", "code": 28, "message": "Invalid access level for API call"}
```

The following sections describe all API calls for user management.

4.1. Display the user list

Use this call to return a JSON object made up of two arrays: list of users and list of roles. This call also returns the list of available roles that may be used to create users. The available roles are presented in order of priority. Any given role can access anything defined under a subsequent role.

EXAMPLE

`curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/UserList`

Result:

{ "userList": [{ "name": "admin", "roleKey": "admin", "roleText": "Administrator"}], "availableRoles": [{ "key": "admin", "text": "Administrator"}, { "key": "tech", "text": "Technician"}, { "key": "ipmi", "text": "IPMI"}]}

For more information on this call, refer to 10.5.1 User List.

For possible values for URL resources, refer to 10.1 URL resource values.

4.2. Add a user

Usernames must remain unique in the system. If an attempt is made to duplicate a username, the following error will be returned:

```
{"status": "Failure", "code": 26, "message": "User already exists"}
```

You cannot add a user with the role "IPMI".

The following parameters are required for this call : name, password, roleKey

EXAMPLE

```
curl --user <username>:<token> -d "name=ricky&password=bobby&roleKey=<valid role key>"
http://<platform ip>:9090/v3/Platform/<platform number>/AddUser
```

For more information on this call, refer to 10.5.1 User List.

For possible values for URL resources, refer to 10.1 URL resource values.

4.3. Modify a user

Use this call to modify the attributes of an existing user.

All users can be deleted or modified except the user with the name "admin" and the user with the role "IPMI".

The user with the name "admin" and the user with the role "IPMI" cannot be deleted. They also cannot be renamed or be assigned another role. Only their passwords can be modified. This ensures the presence of at least one user with the "admin" and the "IPMI" roles. Other users with the "admin" role can be modified.

There can only be one user with the "IPMI" role and its name is predefined and fixed as "smmgnt". It can be used for IOL access to the system.

Users with the "admin" role can change the name, password and role of other users.

Users with the "technician" role can only change their password, not their name or their role.

Users who are able may change their names and passwords using the following parameters:

- **name**: The name of the user to be modified.
- **newName (optional)**: The new name for user. If not used, stays the same.
- **newPassword (optional)**: The new password for user. If not used, stays the same.
- **newRoleKey (optional)**: The new role key for user. If not used, stays the same.

EXAMPLE

Use the following call to change the password and role of user "ricky" but not the username.

```
curl --user <username>:<token> -d "name=ricky&newPassword=1234&newRoleKey=<valid role key>"
http://<platform ip>:9090/v3/Platform/<platform number>/ModifyUser
```

For more information on this call, refer to 10.5.3 Modify User.

For possible values for URL resources, refer to 10.1 URL resource values.

4.4. Delete a user

Use this call to delete a user from the system. Specify the form parameter "name" in the API call.

The user with the name "admin" and the user with the role "IPMI" cannot be deleted.

EXAMPLE

```
curl --user <username>:<token> -d "name=ricky" http://<platform ip>:9090/v3/Platform/<platform
number>/DeleteUser
```

For more information on this call, refer to 10.5.4 Delete User.

For possible values for URL resources, refer to 10.1 URL resource values.

5/ Monitoring sensors

A SYMKLOUD platform contains hundreds of sensors that provide critical information about the condition of the chassis. The following provides some of the API commands used to keep track of sensor values and thus simplify sensor management.

5.1. List all sensors for a node

To get a list of all sensors for a node:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node number>/Sensors
```

Result:

```
[{"number": "0", "name": "Temp Inlet", "type": "Temperature", "unit": "degrees C", "analog": "yes", "monitored": "no", "value": "0"}]
```

This command returns an array of sensor objects that can be found on the node.

"Monitored" is an important attribute of the sensors in the trending functionality. By default, this attribute is set to "false" for all sensors, so that the value of the sensor is not fetched. When set to "true" the application will retrieve the value of the sensor. The System Event Log will always log sensor threshold events whether they are set to "Monitored" or not.

For more information on this call, refer to 10.7.3 Sensors.

For possible values for URL resources, refer to 10.1 URL resource values.

5.2. Batch set the "monitored" attribute for multiple sensors

To batch set the "monitored" attribute for multiple sensors, use the following call:

EXAMPLE

```
curl --user <username>:<token> -d "list=[ 1, 2, 23, 42]&value=yes" http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node number>/MonitoredSensors
```

For more information on this call, refer to 10.10 Sensor.

For possible values for URL resources, refer to 10.1 URL resource values.

5.3. Get information concerning specific sensors

Once some sensors are set to be monitored, you may want to retrieve only the information concerning these specific sensors.

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node number>/MonitoredSensors
```

This will return the same type of list as the first call, including only the sensors that have their "Monitored" attribute set to "yes" and a valid value.

For more information on this call, refer to 10.7.4 Monitored Sensors.

For possible values for URL resources, refer to 10.1 URL resource values.

6/ Using the System Event Log (SEL)

The System Event Log (SEL) contains one entry for each event generated by the sensors from all the nodes, PSUs and ShMCs present in the system.

You can use the SEL for:

- analyzing a failure
- viewing if there were any activities at a given date and time

The following APIs are related to the SEL:

- Events
- Csv
- Clear
- Settings (GET/PUT)

NOTICE

Before using any of the SEL calls for the first time after a system upgrade to Version 2.1, you must wait at least 15 minutes for the database to retrieve all events.

6.1. Retrieve SEL events

Use the **Events** call to retrieve a range of SEL entries using the following parameters:

- **offset**: Number of records to skip. Optional, defaults to 0.
- **limit**: Maximum number of records to retrieve. Optional, defaults to 100, maximum value is 1000.

EXAMPLE

```
curl -G --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SEL/Events -d "offset=116&limit=1"
```

Result:

```
{ "numRecords": 639, "records": [{"recordId": 116, "timestamp": "2015-01-29T05:52:44Z", "generatorId": "Node1", "sensorNumber": 40, "sensorName": "ACPI State", "sensorType": "System ACPI Power State", "eventDirection": "Asserted", "eventData1": 0, "eventData2": 255, "eventData3": 255, "eventDescription": ""}]}
```

For possible values for URL resources, refer to 10.1 URL resource values.

The JSON object returned contains:

- **numRecords**: The total number of records present in the SEL database.
- **records**: The list of events. Events contain:
  - **recordId**: The ID of the record in the database
  - **timestamp**: The time at which the event occurred
  - **generatorId**: The ID of the node the event was generated from
  - **sensorNumber**: The sensor number
  - **sensorName**: The sensor name
  - **sensorType**: The sensor type as defined in the IPMI specification
  - **eventDirection**: The assertion or de-assertion (when applicable)
  - **eventData1, 2 and 3**: Any extra information from the event. This can be used if in depth analysis is required
  - **eventDescription**: The text description of the event

6.2. Retrieve all SEL in a CSV file

Use the **Csv** call to retrieve the complete SEL in a .csv file.

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SEL/Csv
```

For possible values for URL resources, refer to 10.1 URL resource values.

6.3. Clear the SEL database

Use the **Clear** call to clear the SEL database owned by the System Manager as well as the IPMI SEL.

EXAMPLE

```
curl -X POST --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SEL/Clear
```

For possible values for URL resources, refer to 10.1 URL resource values.

6.4. Modify the System Manager's SEL settings

Use the **Settings** call to modify the System Manager's SEL settings using the following parameters:

- **pollingEnabled**: Enable or disable capturing new events in the database, "true" or "false". If this setting is disabled then enabled, events that were missed will be added to the database.
- **recordsRetention**: Maximum number of events kept in the database. Old records are evicted in a first-in first-out manner. Defaults: 10000, minimum: 5000, maximum: 50000.

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SEL/Settings -X PUT -d "pollingEnabled=true&recordsRetention=10000"
```

For possible values for URL resources, refer to 10.1 URL resource values.

7/ Managing PSU statistics

The sensors from the power supply units are separate from the trending category. They have their own specific calls to obtain information and values.

7.1. Get details

The average power and peak power may be obtained for each PSU. This data can be useful to study the power consumption of the system under an application load. Use the following call:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/PSU/<psu number>/Value
```

For more information on this call, refer to 10.12.1 Details.

For possible values for URL resources, refer to 10.1 URL resource values.

7.2. Reset the PSU statistics

Power consumption statistics (average power and peak power) must be reset individually for each PSU using the following call:

EXAMPLE

```
curl --user <username>:<token>
http://<platform ip>:9090/v3/Platform/<platform number>/PSU/<psu number>/ResetStats
```

For more information on this call, refer to 10.12.2 Reset Stats.

For possible values for URL resources, refer to 10.1 URL resource values.

8/ Configuring Ports 7 and 9 speed

NOTICE This section only applies when you are using Hub model MSH8900.

To use the dual CPU MSP8020 node in SYMKLOUD slots 7 and 9, you need to apply changes to the integrated switches to modify the port speed from 10 Gb/s to 1 Gb/s.

The following shows the commands that get and set the speed settings for these ports.

8.1. Get current settings

This returns an array of two JSON objects and gives the current configuration of each slot.

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SwitchSpeed
[{ "slot": "7", "speed": "1" }, { "slot": "9", "speed": "10" }]
```

For more information on this call, refer to 10.12.2 Reset Stats.

For possible values for URL resources, refer to 10.1 URL resource values.

8.2. Set speed configurations

Use the following command to change the speed of the switch ports.

EXAMPLE

```
curl --user <username>:<token>
-d "speedConfig=[{ \"slot\": \"7\", \"speed\": \"1\"}, { \"slot\": \"9\", \"speed\": \"1\"}]"
http://<platform ip>:9090/v3/Platform/<platform number>/SwitchSpeed
```

⚠WARNING This command will reset both platform switches to apply the configuration. Before running it, make sure the application running on the SYMKLOUD platform will not be affected by the communication break.

For more information on this call, refer to 10.6.6 Switch Speed.

For possible values for URL resources, refer to 10.1 URL resource values.

9/ Upgrading firmware

9.1. Upgrading a single node

To upgrade a single node, you must have a valid "Bundle" Zip file. Once you have the file in a known location and you know its exact name, then you can proceed as follows to upgrade it using our API:

1. Upload the file to the platform.

   ```
   curl --user <username>:<token> -F file=@<full path of the file> http://<platform ip>:9090/v3/Platform/<platform number>/UploadBundle
   ```

2. Launch the upgrade for the selected node (Nodes 1 – 9 have ID 1 – 9, but Shelf Managers have IDs 10 for left and 11 for right).

   ```
   curl --user <username>:<token> -d "node=<number of the node to upgrade>" http://<platform ip>:9090/v3/Platform/<platform number>/OneClickUpgrade
   ```

3. If the launch is successful, you can then ask periodically for the status of the upgrade.

   ```
   curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/OneClickUpgrade
   ```

For possible values for URL resources, refer to 10.1 URL resource values.

9.2. Upgrading the complete platform

You can use the System Manager API to upgrade all the platform's nodes and ShMCs. This is called a "One-Click Upgrade". It upgrades all components or only a specific model of cards contained in the platform. This action uses a special "Bundle" file created by Kontron containing the files and scripts required to upgrade multiple models of cards.

Here are the steps to perform a platform upgrade:

1. Get the list of available models (optional if an upgrade for a specific model will be performed).

   ```
   curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/AvailableModels
   ```

   The answer will be a JSON object with an attribute "models" containing an array of strings of the available model names. Ex: {"models": ["MSP8000", "MSP8020"]}

2. Upload the file to the platform. This file must be a valid "Bundle" .zip file.

   ```
   curl --user <username>:<token> -F file=@<full path of the file> http://<platform ip>:9090/v3/Platform/<platform number>/UploadBundle
   ```

3. Launch the upgrade and optionally specify a model. Use the model parameter to upgrade a specific model.

   ```
   curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/OneClickUpgrade
   ```

4. Once successfully launched, you can periodically ask for the statuses of all upgrades:

   ```
   curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/OneClickUpgrade
   ```

This will return a "One-Click Upgrade" status JSON object containing:

- **nodeType**: The node type (model) currently being upgraded
- **inProgress**: An upgrade currently in progress
- **currentNode**: The node currently being upgraded (its ID)
- **currentStatus**: The status of the node currently being upgraded (the same as the status of a single upgrade)
- **allStatus**: An array of all node statuses
- **totalProgress**: The percentage of completion for the whole process
- **state**: "Running" while in progress and "Complete" or "Failed" at the end

For more information on these calls, refer to 10.6 Platform.

For possible values for URL resources, refer to 10.1 URL resource values.

10/References

10.1. URL resource values

The resources in the URL can take the following values:

- **platform ip**: The shared IP or the management IP of the active hub.
- **platform number**: 0 for the platform targeted by the IP or 1 to 8 corresponding to PlatformNumber. Refer to **Erreur ! Source du renvoi introuvable. Erreur ! Source du renvoi introuvable.**.
- **node number**: 1 to 9 for nodes, 10 for the left hub, 11 for the right hub.
- **payload number**: 1 or 2. The number of payloads depends on the node model.
- **sensor number**: Numeric value.
- **psu number**: 1 or 2.

10.2. Commands without security

10.2.1. Version

GET

Description: This is the only request that does not start with /v3. It is designed to be compatible with any version of the API so the user can set it before actually using the API.

Call: /Version

Return: Version number.

10.2.2. Website Config

GET

Description: Returns an object containing the (customizable) configuration of the visual aspects of the web interface. No authentication is required to access this information.

Call: /v3/Platform/<*platform number*>/WebsiteConfig

Return: JSON object of web configuration options.

Attributes:

- baseColor
- lightBaseColor
- altColor
- lightAltColor
- lighterAltColor
- companyName
- favicon
- logo

10.3. Authentication

10.3.1. Auth

GET / POST

Description: The API 3.0 uses a Basic http Authentication on nearly all requests. The Auth call is used to authenticate the user and generate a temporary token that expires after 30 minutes of inactivity. The token is used as the password for subsequent API requests that require authentication.

Call: /v3/Auth

Return:

Success: JSON object containing attributes:

- **status**
- **code**
- **token**: The token to use for other requests that need authentication
- **roleKey**: The role key of the authenticated user
- **roleText** : The role text for the authenticated user

Possible errors:

```
{"status":"Failure", "code": 23, "message": "Invalid password for user"}
{"status":"Failure", "code": 29, "message": "Basic HTTP Authentication required"}
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<password> http://<platform ip>:9090/v3/Auth
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.4. Security

10.4.1. Certificates

POST

Description: Used to upload SSL certificates for securing the platform.

NOTICE HTTPS mode must be enabled to use this call.

Call: /v3/Security/Certificates

Accepted files:

- **key**: The private key
- **cert**: The certificate
- **cacert**: The allowed Certificate Authorities

Possible errors:

```
{"status": "Failure", "code": 43, "message": "Certificate upload already in progress" }
{"status": "Failure", "code": 44, "message": "Failed to write uploaded file." }
{"status": "Failure", "code": 45, "message": "Uploaded .key file is protected by
passphrase.\n Please remove and re-upload." }
{"status": "Failure", "code": 46, "message": "Missing file. Must upload at least 'key' and
'cert' files." }
```

Example of cURL command:

EXAMPLE

curl --user *<username>*:*<token>* -k https://*<platform ip>*/v3/Security/Certificates -F key=@*<filename>* -F cert=@filename -F cacert=@filename

For possible values for URL resources, refer to 10.1 URL resource values.

10.4.2. HTTPS

GET

Description: Used to retrieve the HTTPS settings.

Call: /v3/Security/HTTPS

Return:

{ "enabled":"false", "strictMode":"false" }

POST

Description: Used to modify the HTTPS settings.

Call: /v3/Security/HTTPS

Parameters:

- **enabled**: To enable or disable the feature, "true" or "false"
- **strictMode**: To secure inter SYMKLOUD communications, "true" or "false". Not yet available

Possible errors:

```
{"status":"Failure", "code": 4, "message": "Bad parameters"}
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>/v3/Security/HTTPS -d
"enabled=true&strictMode=false"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.5. User Management

10.5.1. User List

GET

Description: Get a list of information for all users on the platform.

Call: /v3/Platform/*<platform number>*/UserList

Return: JSON array of user objects with attributes name, roleKey and roleText:

{ "userList": [{ "name": "admin", "roleKey": "admin", "roleText": "Administrator"}], "availableRoles": [{ "key": "admin", "text": "Administrator"}, { "key": "tech", "text": "Technician"}, { "key": "ipmi", "text": "IPMI"}]}

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/UserList
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.5.2. Add User

POST

Description: Add a new user.

NOTICE You cannot add a user with the role "IPMI".

Call: /v3/Platform/<*platform number*>/AddUser

Parameters:

- **name**: The user name (unique alpha-numeric characters).
- **password**: The user password (excluded characters: “ ‘ \ and /).
- **roleKey**: The user role. Refer to 10.5.1 User List to get available role keys.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 1, "message": "" }
{"status": "Failure", "code": 4, "message": "Bad parameters" }
{"status": "Failure", "code": 26, "message": "User already exists" }
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> -d "name=ricky&password=bobby&roleKey=<valid role key>"
http://<platform ip>:9090/v3/Platform/<platform number>/AddUser
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.5.3. Modify User

POST

Description: Modify settings of an existing user.

All users can be deleted or modified except the user with the name "admin" and the user with the role "IPMI".

The user with the name "admin" and the user with the role "IPMI" cannot be deleted. They also cannot be renamed or be assigned another role. Only their passwords can be modified. This ensures the presence of at least one user with the "admin" and the "IPMI" roles. Other users with the "admin" role can be modified.

There can only be one user with the "IPMI" role and its name is predefined and fixed as "smmgnt". It can be used for IOL access to the system.

Users with the "admin" role can change the name, password and role of other users.

Users with the "technician" role can only change their password, not their name or their role.

Call: /v3/Platform/<*platform number*>/ModifyUser

Parameters:

- **name**: The name of the user to be modified.
- **newName (optional)**: The new name for user. If not used, stays the same.
- **newPassword (optional)**: The new password for user. If not used, stays the same.

NOTICE This is the only field a "Technician" can modify independently.

- **newRoleKey (optional)**: The new role key for user. If not used, stays the same. Refer to 10.5.1 User List to get available role keys.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 20, "message": "Missing parameter" }
{"status": "Failure", "code": 25, "message": "User does not exist" }
{"status": "Failure", "code": 26, "message": "User already exists" }
{"status": "Failure", "code": 34, "message": "User only allowed to modify his password" }
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> -d "name=ricky&newPassword=1234&newRoleKey=<valid role key>"
http://<platform ip>:9090/v3/Platform/<platform number>/ModifyUser
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.5.4. Delete User

POST

Description: Delete a user from the platform. The user with name "admin" and the one with the "IPMI" role cannot be deleted.

Call: /v3/Platform/<*platform number*>/DeleteUser

Parameters:

- **name**: The name of the user to be deleted.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 4, "message": "Bad parameters" }
{"status": "Failure", "code": 25, "message": "User does not exist" }
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> -d "name=ricky" http://<platform ip>:9090/v3/Platform/<platform
number>/DeleteUser
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6. Platform

10.6.1. System Information

GET

Description: Get a human readable file containing the system information.

Call: /v3/Platform/<*platform number*>/SystemInformation.txt

Return: A file containing a human readable text (first part) and the raw JSON from which it was generated (second part).

10.6.2. Logs

GET

Description: Get the platform logs. They can be used to troubleshoot platform problems.

Call: /v3/Platform/<*platform number*>/Logs

Return: A compressed log file.

10.6.3. Store

GET

Description: Get the stored information for the platform, including all nodes.

Call: /v3/Platform/<*platform number*>/Store

platform number: 0 for current platform.

Return:

Success: JSON object representing the Store

i.e.: {"name": "Platform #6", "number": "6", "health": … }

10.6.4. Available Models

GET

Description: Get the models of the nodes and hubs present in the platform.

Call: /v3/Platform/<*platform number*>/AvailableModels

platform number: 0 for current platform.

Return:

Success: JSON array of the names of node and hub models.

i.e.:{"models": ["MSH8900","MSP8000","MSP8020"]}

10.6.5. Shelf Manager Mode

GET

Description: Get the current Shelf Manager mode (dual or single).

Call: /v3/Platform/*<platform number>*/ShelfMode

platform number: 0 for current platform.

Return:

Success:

[{ "mode":"dual" }] or [{ "mode":"single" }]

Example of cURL command:

EXAMPLE

```
curl --user <user>:<token> http://<platformIp>:9090/v3/Platform/<platform number>/ShelfMode
```

For possible values for URL resources, refer to 10.1 URL resource values.

POST

Description: Set the current ShMC mode (dual or single).

Call: /v3/Platform/*<platform number>*/ShelfMode

Parameters:

- **shelfModeConfig**: The new Shelf manager mode . See the JSON example below.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{“status”: ”Failure”, “code”: 4, “message”: “Bad parameters”}
{“status”: ”Failure”, “code”: 63, “message”: “Invalid value for Shelf mode”}
```

Example of cURL command:

EXAMPLE

```
curl --user <user>:<token> http://<platformIP>:9090/v3/Platform/<platform number>/ShelfMode
                                                -d 'shelfModeConfig=[ { "mode": "single" } ]'
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.6. Switch Speed

GET

Description: Get the platform's switch speed (for slots 7 & 9).

Call: /v3/Platform/*<platform number>*/SwitchSpeed

platform number: 0 for current platform.

Return:

Success: JSON array of slot speed objects.

i.e.: [{"slot":"7", "speed":"10" }, { "slot":"9", "speed":"1"}]

POST

Description: Set the platform's switch speed (only applicable to slots 7 and slot 9 on some models, for ex., the MSH8900).

Call: /v3/Platform/*<platform number>*/SwitchSpeed

Parameters:

- **speedConfig**: The configuration of the switch speed. See the JSON example below.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 4, "message": "Bad parameters"}
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/SwitchSpeed -d "ip=192.168.2.3&netmask=24"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.7. Fabric Current Port Mode

GET

Description: Get the Current Fabric Port Mode for all Nodes (MSH891x only).

Call: /v3/Platform/*<platform number>*/FabricCurrentPortMode

platform number: 0 for current platform.

Return:

Success: JSON array of current fabric port mode objects.

```
i.e.: [{"mode":"255", "node":"1"}, {"mode":"255", "node":"2"}, {"mode":"255", "node":"3"},
{"mode":"255", "node":"4"}, {"mode":"255", "node":"5"}, {"mode":"255", "node":"6"},
{"mode":"255", "node":"7"}, {"mode":"255", "node":"8"}, {"mode":"255", "node":"9"}]
```

10.6.8. Fabric Default Port Mode

GET

Description: Get the Default Fabric Port Mode for all Nodes (MSH891x only).

Call: /v3/Platform/*<platform number>*/FabricDefaultPortMode

Return:

Success: JSON array of default fabric port mode objects.

```
i.e.: [{"mode":"255", "node":"1"}, {"mode":"255", "node":"2"}, {"mode":"255", "node":"3"},
{"mode":"255", "node":"4"}, {"mode":"255", "node":"5"}, {"mode":"255", "node":"6"},
{"mode":"255", "node":"7"}, {"mode":"255", "node":"8"}, {"mode":"255", "node":"9"}]
```

POST

Description: Set the Default Fabric Port Mode for all Nodes (MSH891x only).

Call: /v3/Platform/*<platform number>*/FabricDefaultPortMode

Parameters:

- **fabricPortModeConfig**: Default Fabric Port Mode. See the JSON example below. 0 = Dual Port Mode, 2 = Single Port / Quad Lane Mode, 3 = Single Port / Single Lane Mode, 255 = Switch Factory Default.

Return:

Success:

```
{"status":"Success", "code": 0, "message": ""}
```

Possible errors:

```
{"status":"Failure", "code": 4, "message": "Bad parameters"}
{"status":"Failure", "code": 6, "message": "Invalid Slot Number"}
{"status":"Failure", "code": 56, "message": "Invalid Fabric Port Mode Setting"}
```

Example cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform
ip>:9090/v3/Platform/<platform number>/FabricDefaultPortMode -d 'fabricPortModeConfig=[{
"node": "1", "mode":"0" }, { "node":"2", "mode":"2"}, { "node":"3", "mode":"2"}, { "node":"8",
"mode":"255"}]'
```

For possible values of URL resources, refer to 10.1 URL resource values.

10.6.9. Shared IP

GET

Description: Get the IP the platform uses to talk to the Active Shelf Manager. It will contain the IP and the Netmask used to set the shared IP.

Call: /v3/Platform/*<platform number>*/SharedIp

Return:

Success: JSON representing shared IP information.

i.e.: {"ip":"192.168.42.42", "netmask":"16"}

Possible errors:

```
{"status": "Failure", "code": 4, "message": "Bad parameters"}
{"status": "Failure", "code": 5, "message": "Internal Error"}
```

POST

Description: Set the Shared IP configuration. This can be used afterward as a single IP to connect to the active shelf manager.

NOTICE The Shared IP will use the VLAN and Gateway information of the active Hub.

Call: /v3/Platform/*<platform number>*/SharedIp

Parameters:

- **ip**: The IP to which the active shelf manager of the platform will answer.
- **netmask**: The netmask of the network on which the shared IP is configured in CIDR or IP format.

If either of the parameters is not set, the shared IP configuration will be deleted.

Return:

Success:

```
{"status":"Success", "code": 0, "message": ""}
```

Possible errors:

```
{"status":"Failure", "code": 4, "message": "Bad parameters"}
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SharedIp
-d "ip=192.168.2.3&netmask=24"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.10. Terminate System Manager

GET

Description: Terminate the System Manager application.

Call: /v3/Platform/<platform number>/TerminateSysMan

Return:

{"status": "Success", "code": 0, "message": "" }

Example of cURL command:

EXAMPLE

```
curl --user <user>:<token> http://<platformIp>:9090/v3/Platform/<platform
number>/TerminateSysMan
```

Note: Since the System Manager application is monitored by the platform, it will be re-spawned automatically.

10.6.11. Start Snmp

MSH892x only

GET

Description: Start the SNMP server.

Call: /v3/Platform/*<platform number>*/StartSnmp

Return:

```
{"status": "Success", "code": 0, "message": "" }
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/StartSnmp
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.12. Stop Snmp

MSH892x only

GET

Description: Stop the SNMP server.

Call: /v3/Platform/<*platform number*>/StopSnmp

Return:

```
{"status": "Success", "code": 0, "message": "" }
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/StopSnmp
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.13. Restart Snmp

GET

Description: Restart the SNMP server.

Call: /v3/Platform/<*platform number*>/RestartSnmp

Return:

```
{"status": "Success", "code": 0, "message": "" }
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/RestartSnmp
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.14. Upload Snmp Config

POST

⚠WARNING

Once uploaded, the new configuration file cannot be reverted and will supersede the Kontron default SNMP configuration.

Description: Upload and apply a new configuration for SNMP.

Call: /v3/Platform/<*platform number*>/UploadSnmpConfig

Return:

```
{"status": "Success", "code": 0, "message": "" }
```

Example of cURL command:

EXAMPLE

```
curl -F file=@<filename> http://<platform ip>:9090/v3/Platform/<platform
number>/UploadSnmpConfig --user <username>:<token>
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.15. Upload Bundle

POST

Description: Upload a compressed Bundle .zip file used by the One-Click Upgrade feature. If the bundle file is uploaded to the active shelf manager, it will be copied automatically to the standby shelf manager. Using the shared IP will also upload the bundle file to the active shelf manager and copy it to the standby shelf manager.

Call: /v3/Platform/<*platform number*>/UploadBundle

Return:

- Array of supported models, i.e.: ['MSP8000', 'MSP8020']

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> -F file=@<filename> http://<platform
ip>:9090/v3/Platform/<platform number>/UploadBundle
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.16. Bundle Info

GET

Description : List the models supported by the bundle. These can be updated by a One-Click Upgrade.

Call: /v3/Platform/<*platform number*>/BundleInfo

Return:

- Array of supported models i.e : ['MSP8000', 'MSP8020']

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/BundleInfo
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.17. OneClick Upgrade

GET

Description: Get the current status of the One-Click Upgrade.

Call: /v3/Platform/<platform number>/ OneClickUpgrade

Return: The status of all the nodes upgrading for this platform.

```
i.e.: {"nodeType": "","inProgress": "0","currentNode": "","allStatus": [],"totalProgress":
0, "state": "Completed", "currentStatus": {}}
```

POST

Description: Launch a One-Click Upgrade on the platform.

Call: /v3/Platform/<*platform number*>/OneClickUpgrade

Parameters:

- **node (optional)**: Specify a single node (by ID) to upgrade.
- **model (optional)**: Specify the model to upgrade. If none is specified, it will upgrade all possible nodes with the current bundle.
- **forceUpdate (optional)**: Without this parameter, only the outdated node/hub will be updated. This is required to go to a previous version or upgrade components in the same version (i.e., forceUpdate=true).

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status":"Failure", "code": 1, "message": ""}
```

```
{"status":"Failure", "code": 38, "message": "No bundle was found. Please upload a valid Bundle before upgrading"}
```

Example of cURL command:

EXAMPLE

To upgrade a specific node model regardless of its firmware version:

```
curl --user <username>:<token> -d 'model=MSP8020&forceUpdate=true' http://<platform ip>:9090/v3/Platform/<platform number>/OneClickUpgrade
```

To upgrade all models supported by the bundle:

```
curl --user <username>:<token> -X POST http://<platform ip>:9090/v3/Platform/<platform number>/OneClickUpgrade
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.6.18. Monitored Sensors

GET

Description: Get a list of all monitored sensors along with their values.

Call: /v3/Platform/<*platform number*>/MonitoredSensors

Return:

- Array of monitored sensors

```
i.e.: { "platform": "6", "monitoredSensors": [{"node": "1", "monitoredSensors": [ {"number": "27", "name": "Health Status", "type": "Platform Alert", "unit": "", "analog": "no", "monitored": "yes", "aggregation": "yes", "value": "0x8002"}]}, {"node": "3", "monitoredSensors": [ {"number": "23", "name": "Health Error", "type": "Platform Alert", "unit": "", "analog": "no", "monitored": "yes", "aggregation": "no", "value": "0x8001"}]}, {"node": "10", "monitoredSensors": [ {"number": "70", "name": "Health Status", "type": "Platform Alert", "unit": "", "analog": "no", "monitored": "yes", "aggregation": "yes", "value": "0x8002"}]}, {"node": "11", "monitoredSensors": [ {"number": "70", "name": "Health Status", "type": "Platform Alert", "unit": "", "analog": "no", "monitored": "yes", "aggregation": "yes", "value": "0x8002"}]}]}
```

10.6.19. KVM Password

GET

Description: Get the KVM password.

Call: /v3/Platform/<*platform number*>/KVMPassword

Return: JSON object with the password of the KVM instance of the platform.

```
i.e: { "password": "<returned password>"}
```

POST

Description: Set the KVM password.

Call: /v3/Platform/<*platform number*>/KVMPassword

Parameters:

- **password**: The password to be set. It must be exactly 8 characters long and contain at least an uppercase and a lowercase letter, a digit and a symbol.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 20, "message": "Missing parameter" }
```

```
{"status": "Failure", "code": 37, "message": "The password format must have: a length of
exactly 8 characters, at least 1 lower case and 1 upper case letter, uses 1 or more number,
uses 1 or more special character."}
```

10.6.20. Alarm

GET

Description: Get the chassis' current alarm information: current severity and components in error.

Call: /v3/Platform/<*platform number*>/Alarm

Return: JSON object with the alarm level and components in alarm of the platform.

```
i.e:
```

Without alarm:

```
{"alarmlevel": "None",  "components" : [ {"nodes" : [] } , {"psu" : [] }, {"fans" : [] }],
"text": ""}
```

Missing fan:

```
{"alarmlevel": "Major", "components" : [ {"nodes" : ["10", "11"]}, {"psu" : []}, {"fans" :
[]} ], "text": "HubNode 1, HubNode 2 at alarm level Major"}
```

PSU issue:

```
{"alarmlevel": "Critical", "components" : [ {"nodes" : []}, {"psu" : ["1"]}, {"fans" : []}
], "text": "PSU 1 at alarm level Critical"}
```

10.6.21. Switch OID

GET

Description: Get the platform's switch root OID.

Call: /v3/Platform/<*platform number*>/SwitchOID

Return:

On a MSH8900 Shelf Manager:

```
{"oid": "1.3.6.1.4.1.6603"}
```

On a MSH891x Shelf Manager:

```
{"oid": "1.3.6.1.4.1.15000"}
```

10.6.22. LED Identify

GET

Description: Get the LED Identify state.

Call: /v3/Platform/*<platform number>*/LED/Identify

Return: JSON object with the LED state ON/OFF.

```
i.e.: {"status": "off"}
```

POST

Description: Turn ON/OFF the chassis LED Identify state.

Call: /v3/Platform/*<platform number>*/LED/Identify

Parameters:

- **command**: The state to apply to the LED: ON or OFF

Return:

Success:

```
{"status":"Success", "code": 0, "message": ""}
```

Example cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/LED/Identify -d "command=on"
```

10.6.23. Nodes

GET

Description: Get the Nodes list.

Call: /v3/Platform/*<platform number>*/Nodes

Return: JSON object with the present nodes list.

```
i.e: {"presentNodes": [{"type": "HubNode", "number": "10"}, {"type": "HubNode", "number":
"11"}, {"type": "Node", "number": "1"}, {"type": "Node", "number": "2"}, {"type": "Node",
"number": "3"}, {"type": "Node", "number": "4"}, {"type": "Node", "number": "5"}, {"type":
"Node", "number": "6"}, {"type": "Node", "number": "7"}, {"type": "Node", "number": "8"},
{"type": "Node", "number": "9"}]}
```

10.7. Node

10.7.1. Management Network

GET

Description: Get the management network configuration information for one of the platform's nodes.

Call: /v3/Platform/*<platform number>*/Node/*<node number>*/ManagementNetwork

Return: JSON object with the following attributes:

- **ip**
- **ip source**: "static" or "dhcp"
- **netmask:** CIDR format (0 to 32) or IP format (0-255.0-255.0-255.0-255)
- **gateway**
- **vlan**

POST

Description: Set the management network parameters for one of the platform's nodes.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/ManagementNetwork

Parameters:

- **ip (optional)**: The management IP
- **ip source (optional)**: "static" or "dhcp"
- **netmask (optional)**: CIDR format (0 to 32) or IP format (0-255.0-255.0-255.0-255)
- **gateway (optional)**
- **vlan (optional)**

If a parameter is not set, it will remain unchanged.

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Example of cURL command:

EXAMPLE

```
curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node
number>/ManagementNetwork
--user <username>:<token> -d "ip=1.1.1.1&ipsource=static&netmask=27&vlan=101"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.7.2. Power Command

POST

Description: Send a power command to a specific node.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/PowerCommand

Parameters:

- **payload (optional)**: To affect only a specific payload on nodes with multiple payloads.
- **command**: The command to send to the node. Valid commands:
  - activate
  - deactivate
  - reset

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 1, "message": "" }
{"status": "Failure", "code": 2, "message": "Target Node is not present" }
{"status": "Failure", "code": 3, "message": "Invalid payload ID" }
{"status": "Failure", "code": 4, "message": "Bad parameters" }
```

Example of cURL command:

EXAMPLE

`curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node number>/PowerCommand`

--user <*username*>:<*token*> -d "payload=2&command=deactivate"

For possible values for URL resources, refer to 10.1 URL resource values.

10.7.3. Sensors

GET

Description: Get the sensors on the specified node and their statuses. A sensor that is not monitored will not return a value.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Sensors

Return:

Success:

```
{"Sensors": [ {"number": "0", "name": "FRU0 Hot Swap", "type": "OEM Reserved", "unit": "",
"analog": "no", "monitored": "no", "aggregation": "no", "value": "0x0"}, {"number": "1",
"name": "FRU1 Hot Swap", "type": "OEM Reserved", "unit": "", "analog": "no", "monitored":
"no", "aggregation": "no", "value": "0x0"}, … ]}
```

10.7.4. Monitored Sensors

GET

Description: Get the sensors that are monitored on the specified node and their statuses.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/MonitoredSensors

Return:

Success:

```
{"node": "1", "monitoredSensors": [ {"number": "27", "name": "Health Status", "type":
"Platform Alert", "unit": "", "analog": "no", "monitored": "yes", "aggregation": "yes",
"value": "0x8002"}]}
```

POST

Description: Modify which sensors are monitored.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/MonitoredSensors

Parameters:

- **list**: A comma-separated list of the sensor numbers between square brackets (i.e.: [1,3,4,9])
- **value**: If you want to start monitoring the sensors, use "value=yes". If you want to stop monitoring the sensors, use "value=no".

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status":"Failure", "code": 15, "message": "Bad sensor number"}
```

Example of cURL command:

EXAMPLE

To start monitoring sensors 1, 23 and 35:

```
curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node
number>/MonitoredSensors
--user <username>:<token> -d "list=[1, 23, 35]&value=yes"
```

To stop monitoring sensors 1, 23 and 35:

```
curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node
number>/MonitoredSensors
--user <username>:<token> -d "list=[1, 23, 35]&value=no"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.7.5. Health Details

GET

Description: Get the list of sensors that are part of the health aggregation for a node.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/HealthDetails

Return:

Success:

```
{"sensorInfo": {"number": "27", "name": "Health Status", "type": "Platform Alert", "unit":
"", "analog": "no", "monitored": "no", "aggregation": "yes", "value": "0x8002"},
"sensorHealth": "Healthy", "aggregationDetails": [{"name": "Temp Brd Inlet", "number": "4",
"value": "28.00", "health": "Healthy"}, {"name": "Temp Brd Outlet", "number": "5", "value":
"30.00", "health": "Healthy"}, {"name": "Temp BMC", "number": "7", "value": "37.00",
"health": "Healthy"}, {"name": "Temp CPU 1", "number": "8", "value": "0.00", "health":
"Healthy"}, {"name": "Temp CPU 2", "number": "9", "value": "37.00", "health": "Healthy"}]}
```

10.7.6. Store

GET

Description: Get the stored information for the node.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Store

Return: Success: JSON object representing the Store.

```
i.e:
{"number": "1", "name": "Node 1", "health": "Green", "fruInfo": {"mfgDate": "Tue Nov  3
16:29:00 2015", "boardMfg": "Kontron", "boardProductName": "4003", "boardProductPart":
"T4003################", "boardSerialNumber": "0000000000"},"hotswapStatus":{"state": "M4",
"text": "Active"},"type":"node", "versions" : [{ "name" : "BMC", "major" : "3", "minor" :
"33"},{ "name" : "FPGA", "major" : "1", "minor" : "02"},{ "name" : "BIOS", "major" : "0",
"minor" : "42"}],"model": "MSP8000", "isUpdating" : "0", "managementIP": "172.16.143.101",
"managementMAC": "00:a0:a5:7c:ec:1a", "components": [{ "description": "Payload1       ",
"number": "1", "hotswapStatus":{"state": "M4", "text": "Active"}, "networkInterfaces": [{
"id": "1", "backplane": "Base1", "ip": ["0", "0", "0", "0"], "gateway": ["0", "0", "0",
"0"], "netmask": "0", "ipSource": "static", "mac": "00:a0:a5:7c:ec:18"}, { "id": "2",
"backplane": "Base2", "ip": ["0", "0", "0", "0"], "gateway": ["0", "0", "0", "0"],
"netmask": "0", "ipSource": "static", "mac": "00:a0:a5:7c:ec:19"}, { "id": "3",
"backplane": "Fabric1", "ip": ["0", "0", "0", "0"], "gateway": ["0", "0", "0", "0"],
"netmask": "0", "ipSource": "static", "mac": "00:a0:a5:7c:ec:16"}, { "id": "4",
```

```
"backplane": "Fabric2", "ip": ["0", "0", "0", "0"], "gateway": ["0", "0", "0", "0"],
"netmask": "0", "ipSource": "static", "mac": "00:a0:a5:7c:ec:17"}]}] }
```


10.7.7. Payloads

GET

Description: Get the stored information for the payloads on the node.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payloads

Return: Success: JSON object representing the payloads.

i.e.:

With a MSP8000 node:


```
{"payloads": [{ "number": "1", "description": "Payload1       "}]}
```


With a MSP8020 node:


```
{"payloads": [{ "number": "1", "description": "Payload1       "}, { "number": "2",
"description": " Payload2       "}]}
```


10.7.8. Alarm

GET

Description: Get the node's current alarm information: current severity and sensors list that triggered the alarm.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Alarm

Return: JSON object with the alarm level and components in alarm of the platform.

i.e.:

Without an alarm:


```
{"alarmlevel": "None", "sensors":[]}
```


With an alarm:


```
{"alarmlevel": "Major", "sensors":[{"sensorNumber":"1", "text":"Major Fault : Sensor 'Temp
Inlet' is at 23.00"}]}
```


10.7.9. Led Identify

GET

Description: Get the LED Identify state.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/LED/Identify

Return: JSON object with the Node LED state ON/OFF.

```
i.e: {"status": "off"}
```

POST

Description: Turn ON/OFF the Node LED Identify state.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/LED/Identify

Parameters:

- **command**: The state to apply to the LED: ON or OFF

Return:

Success:

```
{"status":"Success", "code": 0, "message": ""}
```

Example cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node number>/LED/Identify -d "command=on"
```

10.8. Payload

10.8.1. Provision KVM

POST

Description: Apply the configuration of the "network" (see **Erreur ! Source du renvoi introuvable. Erreur ! Source du renvoi introuvable.**) to the KVM of the payload.

NOTICE This only applies to "MSP8020" nodes.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/ProvisionKVM

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 1, "message": "" }
{"status": "Failure", "code": 3, "message": "Invalid payload ID" }
{"status": "Failure", "code": 22, "message": "Model does not support KVM provisionning" }
```

10.8.2. Power Command

POST

Description: Send a power command to a node's specific payload.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/PowerCommand

Parameters:

- **command**: The command to send to the node. Valid command:
  - activate
  - deactivate
  - reset

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 1, "message": "" }
{"status": "Failure", "code": 2, "message": "Target Node is not present" }
{"status": "Failure", "code": 3, "message": "Invalid payload ID" }
{"status": "Failure", "code": 4, "message": "Bad parameters" }
```

Example of cURL command:

EXAMPLE

```
curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node
number>/Payload/<payload number>/PowerCommand
--user <username>:<token> -d "command=deactivate"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.8.3. NICs

GET

Description: Get the NICs mapping information of the payload.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/Nics

Return: JSON object with the NICs list for the payload

i.e.:

With a MSP8000 node, on payload 1:

```
{"Nics": [{ "id": "1", "backplane": "Base1"}, { "id": "2", "backplane": "Base2"}, { "id":
"3", "backplane": "Fabric1"}, { "id": "4", "backplane": "Fabric2"}]}
```

With a MSP8020 node, on payload 1:

```
{"Nics": [{ "id": "1", "backplane": "Base2"}, { "id": "2", "backplane": "Fabric1"}]}
```

With a MSP8020 node, on payload 2:

```
{"Nics": [{ "id": "1", "backplane": "Base1"}, { "id": "2", "backplane": "Fabric2"}]}
```

10.8.4. Description

GET

Description: Get the server type field of the payload.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/Description

Return: JSON object with the payload description.

i.e.:

```
{"number": "1", "description": "UDHM"}
```

POST

Description: Set the server type field of the payload. This value is store in the FRU and also use in SNMP.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/Description

Parameters:

- **text**: Description of the payload. Limited to a maximum of 15 characters.

Return:

Success:

```
{"status":"Success", "code": 0, "message": ""}
```

Possible errors:

```
{"status":"Failure", "code": 2, "message": "Target Node is not present"}
```

```
{"status":"Failure", "code": 1, "message": ""}
```

```
{"status":"Failure", "code": 53, "message": "Maximum length of server description is 15
characters."}
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/Node/<node number>/Payload/<payload number>/Description -d "text=<description>"
```

10.9. Network Interface

10.9.1. Network

GET

Description: Obtain all the information ready to be programmed to provision MEI (via ProvisionKVM).

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/Nic/<*nic number*>/Network

Return:

```
{ "id": "1", "backplane": "Base2", "ip": ["0", "0", "0", "0"], "gateway": ["0", "0", "0",
"0"], "netmask": "0", "ipSource": "static", "mac": "00:a0:a5:7f:fa:ea"}
```

POST

Description: Associate network settings to the network interface of a payload. These settings are used by the "ProvisionKVM" API call. This setting can be used to remember an IP address configured on a server without actually provisioning it.

Call: /v3/Platform/<*platform number*>/Node/<*node number*>/Payload/<*payload number*>/Nic/<*nic number*>/Network

Parameters:

- **ip**: The management IP
- **netmask**: The CIDR format (0 to 32)
- **ipsource (optional)**: The "static" or "dhcp"
- **gateway (optional)**: The gateway IP address

Return:

Success:

```
{"status": "Success", "code": 0, "message": "" }
```

Possible errors:

```
{"status": "Failure", "code": 2, "message": "Target node is not present" }
{"status": "Failure", "code": 3, "message": "Invalid payload ID" }
{"status": "Failure", "code": 4, "message": "Bad parameters" }
```

Example of cURL command:

EXAMPLE

```
curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node
number>/Payload/<payload number>/Nic/<nic number>/Network
--user <username>:<token> -d "ip=1.1.1.1&ipsource=static&netmask=27"
```

For possible values of URL resources, refer to 10.1 URL resource values.

10.10. Sensor

10.10.1. Monitor

POST

Description: Used to add or remove a sensor to the list of monitored Sensors.

Call: /v3/Platform/*<platform number>*/Node/*<node number>*/Sensor/<sensor number>/Monitor

Parameters:

- **value**: Monitor this sensor: "yes" or "no"

Return:

```
{"status": "Success", "code": 0, "message": "" }
```

Example of cURL command:

EXAMPLE

```
curl http://<platform ip>:9090/v3/Platform/<platform number>/Node/<node number>/Sensor/<sensor number>/Monitor
--user <username>:<token> -d "value=yes"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.10.2. Details

GET

Description: Obtain all the information of this sensor and its value (if monitored, to prevent polling).

Call: /v3/Platform/*<platform number>*/Node/*<node number>*/Sensor/*<sensor number>*/Details

Return:

```
{"number": "1", "name": "FRU1 Hot Swap", "type": "OEM Reserved", "unit": "", "analog":
"no", "monitored": "no", "aggregation": "no", "value": "0x8002"}
```

10.11. System Event Log (SEL)

NOTICE

Before using any of the SEL calls for the first time after a system upgrade to Version 3.0, you must wait at least 15 minutes for the database to retrieve all events.

10.11.1. Events

GET

Description: Retrieve a number of events from the database.

Call: /v3/Platform/*<platform number>*/SEL/Events

Parameters:

- **offset**: Number of records to skip. Optional, defaults to 0.
- **limit**: Maximum number of records to retrieve. Optional, defaults to 100, maximum value is 1000.

Return: JSON object of the following format.

- **numRecords**: The total number of records present in the SEL database.
- **records**: The list of events. Events contain:
  - **recordId**: The ID of the record in the database
  - **timestamp**: The time at which the event occurred

- **generatorId**: The ID of the node the event was generated from
- **sensorNumber**: The sensor number
- **sensorName**: The sensor name
- **sensorType**: The sensor type as defined in the IPMI specification
- **eventDirection**: The assertion or de-assertion (when applicable)
- **eventData1, 2 and 3**: Any extra information from the event. This can be used if in depth analysis is required
- **eventDescription**: The text description of the event

```
{ "numRecords": 639, "records": [{"recordId": 116, "timestamp": "2015-01-29T05:52:44Z",
"generatorId": "Node1", "sensorNumber": 40, "sensorName": "ACPI State", "sensorType":
"System ACPI Power State", "eventDirection": "Asserted", "eventData1": 0, "eventData2":
255, "eventData3": 255, "eventDescription": ""}]}
```

Example of cURL command:

EXAMPLE

```
curl -G --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/SEL/Events -d "offset=116&limit=1"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.11.2. Clear

POST

Description: Clear the SEL database owned by the System Manager as well as the IPMI SEL.

Call: /v3/Platform/<*platform number*>/SEL/Clear

Return:

```
{"status":"Success", "code": 0, "message": ""}
```

EXAMPLE

```
curl -X POST --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/SEL/Clear
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.11.3. Settings

GET

Description: Get the System Manager's SEL settings.

Call: /v3/Platform/<*platform number*>/SEL/Settings

Return: JSON object of the following format.

- **pollingEnabled**: To enable or disable the IPMI event collecting feature.
- **recordsRetention**: The maximum number of events kept by database

```
{ "pollingEnabled": true, "recordsRetention": 10000 }
```

PUT

Description: Modify the System Manager’s SEL settings.

Call: /v3/Platform/<platform number>/SEL/Settings

Parameters:

- **pollingEnabled**: Enable or disable capturing new events in the database, "true" or "false". If this setting is disabled then enabled, events that were missed will be added to the database.
- **recordsRetention**: Maximum number of events kept in the database. Old records are evicted in a first-in first-out manner. Defaults: 10000, minimum: 5000, maximum: 50000.

Return:

```
{"status":"Success", "code": 0, "message": ""}
Possible errors:
{"status":"Failure", "code": 4, "message": "Bad parameters"}
{"status":"Failure", "code": 47, "message": "Database is locked."}
{"status":"Failure", "code": 51, "message": "Database Error."}
```

Example of cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform
number>/SEL/Settings -X PUT -d "pollingEnabled=true&recordsRetention=10000"
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.11.4. CSV

GET

Description: Retrieve all of the events contained in the database in a single CSV file format. Compatible with Microsoft Excel® for further event analysis.

Call: /v3/Platform/<*platform number*>/SEL/Csv

Return: CSV format file

EXAMPLE

```
curl  --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/SEL/Csv
```

For possible values for URL resources, refer to 10.1 URL resource values.

10.12. PSU

10.12.1. Details

GET

Description: Obtain all the information from this PSU and the values of its sensors.

Call: /v3/Platform/<*platform number*>/PSU/<*psu number*>/Details

Return: JSON object of the following format:

```
{"id": "PSU1", "number": "1", "name": "Power Supply 1", "type": "psu",
"hotswapStatus":{"state ": "M4", "text": "Active"},"tempInlet": "24.00", "tempOutlet":
"32.00", "vIn": "119.00", "powerOut": "48", "powerScale": "1400","peakConsumption": "104",
"averageConsumption": "45"}
```

10.12.2. Reset Stats

POST

Description: Reset the average and peak power consumption statistics for the PSU specified.

Call: /v3/Platform/*<platform number>*/PSU/*<psu number>*/ResetStats

Return:

```
{"status":"Success", "code": 0, "message": ""}
```

10.12.3. Alarm

GET

Description: Get the power supply's current alarm information: current severity and sensors list that triggered the alarm.

Call: /v3/Platform/*<platform number>*/PSU/*<psu number>*/Alarm

Return: JSON object with the alarm level and sensor information.

i.e.:

```
Without alarm: {"alarmlevel": "None"}
With an alarm: {"alarmlevel": "Fault"}
```

10.13. Fans

10.13.1. Led Identify

GET

Description: Get the LED Identify state

Call: /v3/Platform/*<platform number>*/Fan/*<fan number>*/LED/Identify

Return: JSON object with the Fan LED state ON/OFF.

```
i.e.: {"status": "off"}
```

POST

Description: Turn ON/OFF the Fan LED Identify state.

Call: /v3/Platform/*<platform number>*/ Fan/*<fan number>*/LED/Identify

Parameters:

- **command**: The state to apply to the LED: ON or OFF

Return:

Success:

```
{"status":"Success", "code": 0, "message": ""}
```

Example cURL command:

EXAMPLE

```
curl --user <username>:<token> http://<platform ip>:9090/v3/Platform/<platform number>/Fan/<fan number>/LED/Identify -d "command=on"
```

10.13.2. Alarm

GET

Description: Get the fan’s current alarm information: current severity and components in error.

Call: /v3/Platform/<*platform number*>/Fan/<*fan number*>/Alarm

Return: JSON object with the alarm level and components in alarm of the platform.

i.e.:

Without an alarm:

```
{"alarmlevel": "None"}
```

With a fault:

```
{"alarmlevel": "Fault"}
```

11/API error codes and messages

```
{"status":"Success", "code": 0, "message": ""}
{"status":"Failure", "code": 1, "message": ""}
{"status":"Failure", "code": 2, "message": "Target Node is not present"}
{"status":"Failure", "code": 3, "message": "Invalid Payload ID"}
{"status":"Failure", "code": 4, "message": "Bad parameters"}
{"status":"Failure", "code": 5, "message": "Internal Error"}
{"status":"Failure", "code": 6, "message": "Invalid Slot Number"}
{"status":"Failure", "code": 7, "message": "Invalid Speed Setting"}
{"status":"Failure", "code": 8, "message": "Error sending <FILE_NAME>"}
{"status":"Failure", "code": 9, "message": "Bad resource for platform"}
{"status":"Failure", "code": 10, "message": "Bad request for platform handler"}
{"status":"Failure", "code": 11, "message": "System not ready yet"}
{"status":"Failure", "code": 12, "message": "Invalid API call"}
{"status":"Failure", "code": 13, "message": "Upgrade in progress"}
{"status":"Failure", "code": 14, "message": "Start Upgrade fail"}
{"status":"Failure", "code": 15, "message": "Bad sensor number"}
{"status":"Failure", "code": 16, "message": "Failed to set platform number"}
{"status":"Failure", "code": 17, "message": "Failed to set IP table"}
{"status":"Failure", "code": 18, "message": "Non remote platform went through stack handler"}
{"status":"Failure", "code": 19, "message": "Error sending log Zip file"}
{"status":"Failure", "code": 20, "message": "Missing parameter"}
{"status":"Failure", "code": 21, "message": "Invalid file: Zip only"}
{"status":"Failure", "code": 22, "message": "Model does not support KVM provisionning"}
{"status":"Failure", "code": 23, "message": "Invalid password for user"}
```

```
{"status":"Failure", "code": 24, "message": "Cannot delete admin user"}

{"status":"Failure", "code": 25, "message": "User does not exist"}

{"status":"Failure", "code": 26, "message": "User already exists"}

{"status":"Failure", "code": 27, "message": "Invalid token for user"}

{"status":"Failure", "code": 28, "message": "Invalid access level for API call"}

{"status":"Failure", "code": 29, "message": "Basic HTTP Authentication required"}

{"status":"Failure", "code": 30, "message": "No token for user. Authenticate at /Auth API"}

{"status":"Failure", "code": 31, "message": "Token has expired. Reauthenticate at /Auth API"}

{"status":"Failure", "code": 32, "message": "Invalid format for website configuration"}

{"status":"Failure", "code": 33, "message": "Upload already in progress"}

{"status":"Failure", "code": 34, "message": "User not permitted to modify others' passwords"}

{"status":"Failure", "code": 35, "message": "API is not supported on this Hub Model"}

{"status":"Failure", "code": 36, "message": "Failed to set management configuration."}

{"status":"Failure", "code": 37, "message": ""The password format must have: a length of exactly 8
characters, at least 1 lower case and 1 upper case letter, 1 or more numbers, 1 or more special
characters."}

{"status":"Failure", "code": 38, "message": "No bundle was found. Please upload a valid bundle
before upgrading."}

{"status":"Failure", "code": 39, "message": "Unhandled server-side exception. See logs for
details."}

{"status":"Failure", "code": 40, "message": "Cannot create more than one IPMI user"}

{"status":"Failure", "code": 41, "message": "IPMI user cannot be deleted"}

{"status":"Failure", "code": 42, "message": "This management server is not currently Active. Please
connect to the Active management server."}

{"status":"Failure", "code": 43, "message": "Certificate upload already in progress."}

{"status":"Failure", "code": 44, "message": "Failed to write uploaded file."}

{"status":"Failure", "code": 45, "message": "Uploaded .key file is protected by passphrase.\n
Please remove and re-upload."}

{"status":"Failure", "code": 46, "message": "Missing file. Must upload at least 'key' and 'cert'
files."}
```

```
{"status":"Failure", "code": 47, "message": "Database is locked."}
```

```
{"status":"Failure", "code": 48, "message": "Invalid username. Cannot be empty and maximum 20 apha-
numeric characters."}
```

```
{"status":"Failure", "code": 49, "message": "Invalid password. Cannot be empty or contain ", \', \
and maximum 20 characters."}
```

```
{"status":"Failure", "code": 50, "message": "Invalid role selected for user."}
```

```
{"status":"Failure", "code": 51, "message": "Database Error."}
```

```
{"status":"Failure", "code": 52, "message": "You can only modify the password of the IPMI user."}
```

About Kontron

Kontron, a global leader in embedded computing technology and trusted advisor in IoT, works closely with its customers, allowing them to focus on their core competencies by offering a complete and integrated portfolio of hardware, software and services designed to help them make the most of their applications.

With a significant percentage of employees in research and development, Kontron creates many of the standards that drive the world's embedded computing platforms; bringing to life numerous technologies and applications that touch millions of lives. The result is an accelerated time-to-market, reduced total-cost-of-ownership, product longevity and the best possible overall application with leading-edge, highest reliability embedded technology

Kontron is a listed company. Its shares are traded in the Prime Standard segment of the Frankfurt Stock Exchange and on other exchanges under the symbol "KBC". For more information, please visit: **http://www.kontron.com/**

CORPORATE OFFICES

EUROPE, MIDDLE EAST & AFRICA

Lise-Meitner-Str. 3-5
86156 Augsburg
Germany
Tel.: + 49 821 4086-0
Fax: + 49 821 4086-111
info@kontron.com

NORTH AMERICA

14118 Stowe Drive
Poway, CA 92064-7147
USA
Tel.: + 1 888 294 4558
Fax: + 1 858 677 0898
info@us.kontron.com

ASIA PACIFIC

1~2F, 10 Building, No. 8 Liangshuihe 2nd Street,
Economical & Technological Development Zone,
Beijing, 100176, P.R. China
Tel.: + 86 10 63751188
Fax: + 86 10 83682438
info@kontron.cn