



Cisco Cyber Vision REST API User Guide



Owner: Cisco IoT

Author: Cisco Cyber Vision

Trademark Acknowledgments

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks.

Third party trademarks mentioned are the property of their respective owners.

The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

Publication Disclaimer

Cisco Systems, Inc. assumes no responsibility for errors or omissions that may appear in this publication. We reserve the right to change this publication at any time without notice. This document is not to be construed as conferring by implication, estoppel, or otherwise any license or right under any copyright or patent, whether or not the use of any information in this document employs an invention claimed in any existing or later issued patent. A printed copy of this document is considered uncontrolled. Refer to the online version for the latest revision.

Copyright

© 2018 Cisco and/or its affiliates. All rights reserved.

Information in this publication is subject to change without notice. No part of this publication may be reproduced or transmitted in any form, by photocopy, microfilm, xerography, or any other means, or incorporated into any information retrieval system, electronic or mechanical, for any purpose, without the express permission of Cisco Systems, Inc.

Americas Headquarters

Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters

Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters

Cisco Systems International BV Amsterdam
The Netherlands

Contents

1	About this documentation	5
1.1	Document purpose	5
1.2	Warnings and notices	5
2	Getting Started	6
2.1	Network representation	6
2.2	Network data	7
2.3	Network evolution	8
2.4	Network analyze	9
3	Performance note	10
4	Responses	11
5	Response objects	12
5.1	Sensor	12
5.2	Component	13
5.3	Flow	14
5.3.1	FlowStats	16
5.3.2	Flow Test Version	16
5.4	Group	18
5.5	Event	21
5.6	Variable	22
5.7	Error cases	24
6	Methods	26
6.1	Authentication	26
6.2	Sensors	26
6.2.1	Get sensors	26
6.3	Components	28
6.3.1	Get all components	28
6.3.2	Get a component	29
6.3.3	Set a custom name	31
6.3.4	Delete a custom name	32
6.3.5	Get a list of allowed icons	33
6.3.6	Get icon content	34
6.3.7	Set a custom icon	35
6.3.8	Delete a custom icon	36
6.3.9	Add a component to a group	37
6.3.10	Remove a component from a group	37
6.3.11	Get a component flows	38
6.3.12	Get a component variables	39
6.3.13	Get vendor names	40
6.3.14	Remove incorrect information	41
6.4	Flows	41
6.4.1	Get all flows	41
6.4.2	Get a flow	43

6.4.3	Get a flow content.....	45
6.4.4	Get a flow statistics	46
6.5	Groups	47
6.5.1	Get all groups	47
6.5.2	Get a group.....	48
6.5.3	Create a group.....	50
6.5.4	Edit a group	52
6.5.5	Explode a group.....	53
6.6	Events	53
6.6.1	Get all events.....	53
6.6.2	Get an event downloadable content	55
6.7	Variables	56
6.7.1	Get all variables.....	56
6.8	Operator, Parameters, Conditions & Actions	57
6.8.1	Tags	60
6.8.2	Property analyzer rules	63
6.8.3	Port analyzer rules.....	67
7	Examples	71
7.1	Get Component MAC by its ID.....	71
7.2	Print the last active Flow of a Component	72

1 About this documentation

1.1 Document purpose

This manual provides you with important information on how to use the Cisco Cyber Vision REST API.

IMPORTANT

Cisco Cyber Vision EAP is a snapshot of the ongoing development process and is in the qualifying phase. Testing for this program is under progress and may contain features that are incomplete or may change before the next full release.

This manual is applicable on **system version 3.1.0**.

1.2 Warnings and notices

This manual contains notices you have to observe to ensure your personal safety as well as to prevent damage to property.

The notices referring to your personal safety and to your property damage are highlighted in the manual by a safety alert symbol described below. These notices are graded according to the degree of danger.

WARNING

Indicates risks that involve industrial network safety or production failure that could possibly result in personal injury or severe property damage if proper precautions are not taken.

IMPORTANT

Indicates risks that could involve property or Cisco equipment damage and minor personal injury if proper precautions are not taken.

Note

Indicates important information on the product described in the documentation to which attention should be paid.

2 Getting Started

The main way to visualize and use the data analyzed by the sensors is to use Cisco Cyber Vision webapp offering a map for visualization, a timeline for events inspection and a significant number of other features.

The other way, explained in this document, is to use the Cisco Cyber Vision API. In this document, CCV refers to Cisco Cyber Vision.

The API exposes the same data than the data used by the Cisco Cyber Vision webapp through an HTTP protocol, a REST API, to allow the creation of third-party clients, scripts and automation:

2.1 Network representation

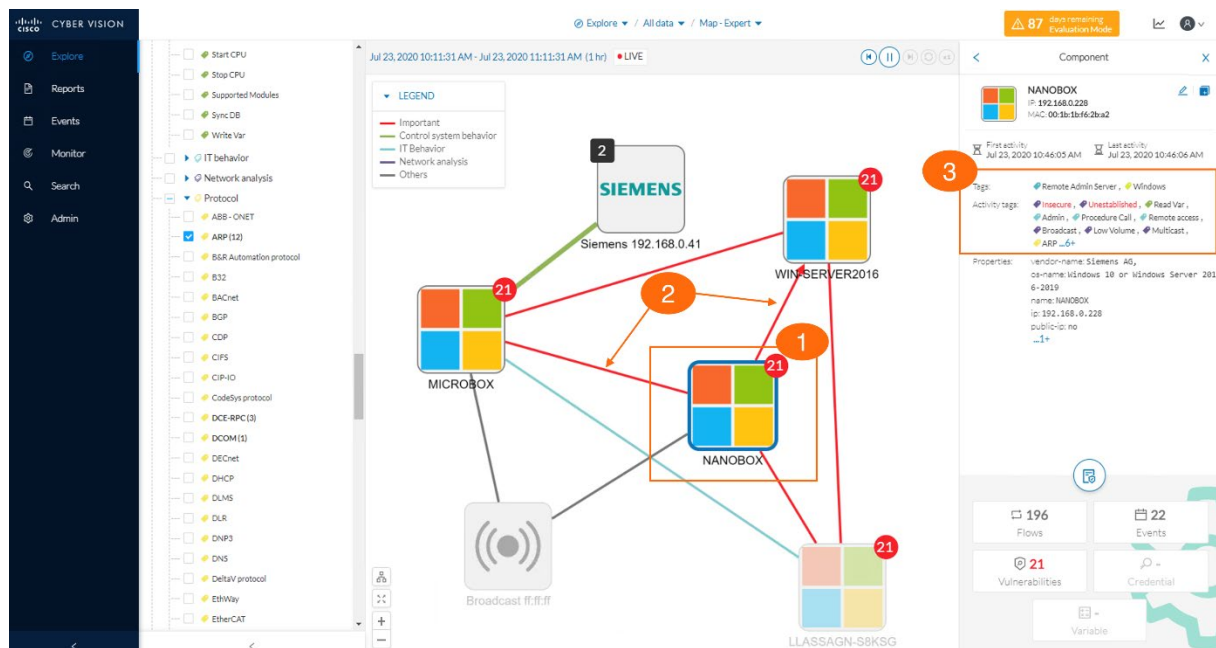
Components, flows and tags

In the Cisco Cyber Vision, a **Component (1)** symbolizes a node of a topological network: for example, a regular laptop on a network would be a node, such as a PLC or anything communicating using the ethernet protocol as an L2 protocol.

The communications observed on the network are called **Flows (2)** and are represented in the webapp by a colored link between two Components. In the user interface of Cisco Cyber Vision, the color of the Flow represents its Network Category, which is automatically set by the Cisco Cyber Vision analyzers to regroup communications and to not be flooded with information. When an API method will be used to retrieve the Flow information, a field called "network_category" will (for example) contain the value "net", which exposes the exact same information to be used for any purpose you may see fit.

When Cisco Cyber Vision detects behaviors, protocols, or critical commands, **Tags (3)** are assigned to the Components or Flows which are the source of the detection. These tags quickly improve the classification of the Component and the Flows. For example: when a PLC is detected and is using the protocol Siemens S7, Cisco Cyber Vision assigns to the corresponding component the tag 'S7' and 'PLC'. Tags also help to observe critical commands or actions happening on the network: a Flow having caused a program download on a PLC is tagged in red with the "Program Download" tag. In the same fashion, when tags are changing on a Component or on a Flow, it indicates changes in the network communications, which is something appearing in the Monitor mode of Cisco Cyber Vision.

All this information is available through methods provided by the Cisco Cyber Vision API.



2.2 Network data

Flow content, vulnerabilities, events

Apart from the communications of the network, there is more data available oriented around the "data" transiting on the network.

First there is the **Flow Content**: it represents useful data seen when **Flows** were active and that the Cisco Cyber Vision decided to store as content properties. For example, when the Cisco Cyber Vision sees Siemens S7 commands transiting through the network, it decides to store for each occurrence the name of the command + the time at which it happened. As it is the content of a Flow, you will first need to get the ID of the Flow for which you want to retrieve the content.

WARNING

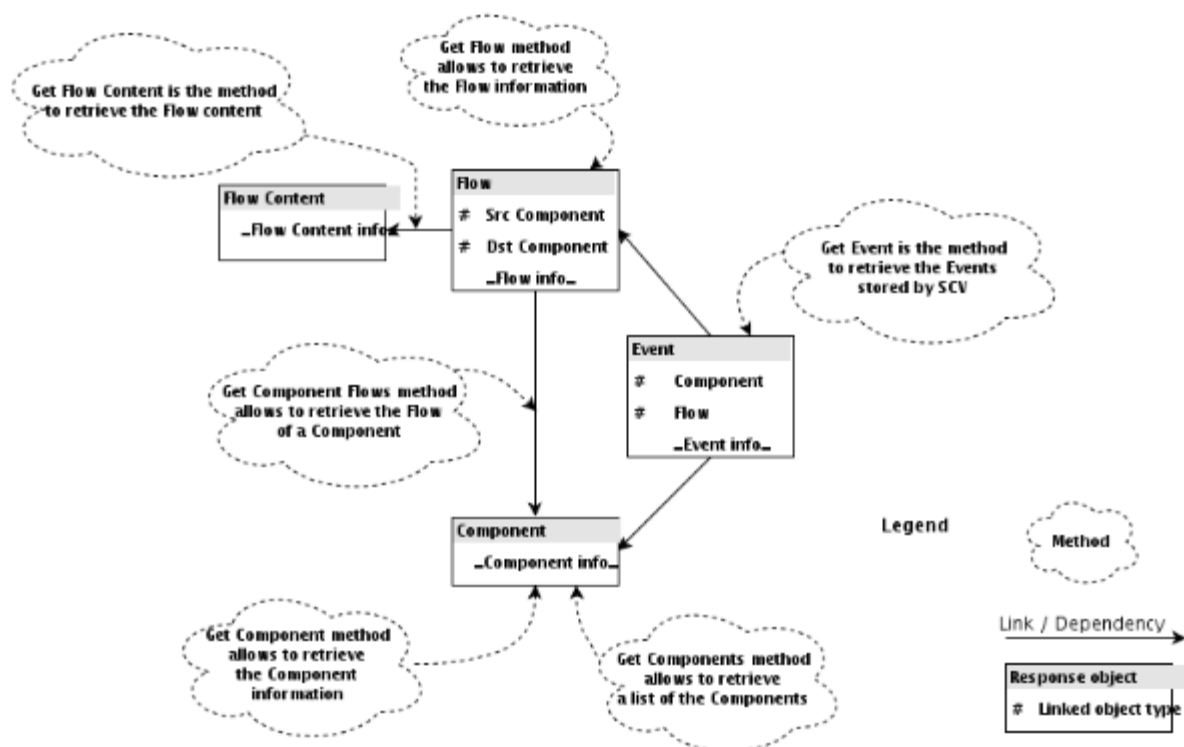
It can represent a big amount of data, ensure to use the *limit* and *offset* parameters carefully.

Vulnerabilities are objects attached to **Components** that the Cisco Cyber Vision has detected as flaws for the network. When Cisco Cyber Vision gets information on the nodes of the network, it tries to analyze as many features as possible to understand the type, version, hardware, firmware, etc., of each

component and it is able to detect Components for which there is already known vulnerabilities.

The Cisco Cyber Vision analyzes the different protocols of the traffic transiting on the network. When something important occurs, it emits **Events** containing all the information in order to be able to trace what happened on the network. In the Cisco Cyber Vision interface, all these events are available in the timeline and on a specific view of the map. The API exposes a way to get them with a complete set of filters (by time, by severity, by category, etc.).

A basic schema representing a part of the architecture of the objects and some of the capabilities of the API (not everything is exposed in this schema, it is here to help having an idea of the API):



2.3 Network evolution

References and computation of differences

Cisco Cyber Vision has a mode called the "Monitor" mode in which you can create References of the communications of the network: a reference is a set of known communications that the user has explicitly included in the reference. By doing so, you can compare different references to see whether

there are differences showing up. These differences can have a lot of different meanings but are important.

For example, let's imagine that the user has created a reference called Stable which contains all the known communications between the Component of the network. If there are new communications, for example a PLC which starts to communicate in a new protocol with the SCADA station, on a **computation of the differences** between the 'Stable' reference and the 'last 24h' of communications, the monitor mode will show and highlight this new communication.

This feature is also available in the API: the creation/edition of references is available and so is the computation of differences.

2.4 Network analyze

Tags edition, port analyzer and property analyzer

Note

This is an advanced usage of the API.

As mentioned earlier, the Tags assigned to the Components and Flows are configurable through the API. This means there is a way to list them, add, modify and remove tags which can be used with Components and Flows.

Analyzers are the part of the Cisco Cyber Vision analyzing the traffic transiting on the network and deciding which actions must be applied on the Flows, Components and others. In other words, it is a set of rules tested on each communication to see whether some actions must be taken.

To extend the capability of classification of the Cisco Cyber Vision, methods of the API have been exposed to configure some **rules on the network port of the communication and on the properties assigned to the Flows**.

For example, when Cisco Cyber Vision detects an S7 communication in a Flow such as a program download on a PLC, it automatically adds this property to the matching Flow. This is something currently done internally in the Cisco Cyber Vision and can't be changed. But the property analyzer allows to configure a rule triggered by the 'program download' being in a Flow, and this rule can for example be configured to assign the tag 'S7' to the current Flow. Among other things, these configurable rules allow to improve with your own knowledge the classification of the data. All this configuration is available through the API.

3 Performance note

Note that the API manipulates large datasets to provide complete and intelligible responses: some queries can be slow if not correctly used.

Most methods have a *limit* and an *offset* parameter to help preventing saturation on the system: **limit** limits the number of returned values and **offset** allows to paginate the data:

For example, on a system with a total of 120 components:

- A Get Components call with limit set to 50 and offset set to 0: returns the components 0 to 50 from the 120 available.
- A Get Components call with limit set to 50 and offset set to 50: returns the components 50 to 100 from the 120 available.
- A Get Components call with limit set to 50 and offset set to 100: returns the components 100 to 120 from the 120 available.

4 Responses

The API is using HTTP for method calls and method responses. It uses the HTTP return code to identify the execution state of method calls and the HTTP body to transmit the data resulting of the method calls (see chapter Responses objects).

Before reading the content (the body) of the HTTP responses, it is necessary to test the HTTP return code which can have one of these values:

HTTP return code	Description
200	Success: the method has been correctly called and it looks like everything has been successfully executed. The HTTP body can be read to retrieve the response (if any).
400	Bad request: the method hasn't been correctly called. It means that bad parameters have been provided to the method call.
401	Token error: the token is either unknown, expired, disabled or invalid.
402	License error: the current license of the Cisco Cyber Vision doesn't allow to use the API.
404	Not found: the method didn't find the Group, the Component, the Flow, etc., queried in the method call.
500	Internal server error: an unexpected error occurred during the execution of the method call. The Help Center contains documentation to diagnose the error.

5 Response objects

5.1 Sensor

A sensor is the device used to collect network activities. It has several attributes related to the network statistics or the capture mode configured.

A sensor is identified by:

- Its id
- its serial number
- its name
- its IP address
- its capture mode
- its model name (e.g. SENSOR3, SENSOR7)
- its current status (e.g. "Active", "Inactive", "Unknown", etc.)
- its firmware version

Serialized sample:

```
{
  "id": "e352adf5-6125-4194-805c-dc61b7cc7c4b",
  "name": "IC3000",
  "version": "3.1.0+202005201632",
  "model": "IC3000_MANAGED",
  "hardware_type": "IC3000",
  "status": "Connected",
  "processing_status": "waiting for data",
  "filter": {
    "capture_mode": "all",
    "custom_input": ""
  },
  "serial_number": "FCH2312Y03P",
  "auto_config": false,
  "ssh_reachable": true,
  "ssh_reachable_last_update": 1594886511000,
  "creation_time": "2020-05-13T21:37:20.683319Z",
  "last_active_time": "2020-07-16T08:02:04.404732Z",
  "uptime": "7m 25s ",
  "snort_enabled": false,
  "ip": "192.168.69.201",
  "recording": false,
  "recording_last_start": "2020-07-13T17:38:31.319209Z",
  "recording_last_stop": "2020-07-13T17:39:01.508929Z",
  "recording_last_size": 0,
  "statistics": true,
  "has_presets": false
}
```

5.2 Component



SIMATIC 300

192.168.0.1

A Component is a node of the network which has communicated. A Component is created by the Cisco Cyber Vision when the source or the destination of a Flow doesn't already match an existing component. It has several attributes such as tags and properties that the Cisco Cyber Vision has identified during its network analysis, plus user properties and user tags manually set by a user.

All these fields are normalized fields filled by the CCV and may be missing if no Flows allows the CCV to fill them:

- id
- ip
- mac
- name
- model_name
- model_ref
- fw_version
- hw_version
- serial_number
- vendor_name
- project_name
- project_version

CreationTime describes the first time that this node has been active on the network, LastActiveTime exposes the last time this node has been seen active on the network.

When the Component has been added into a group, the Group field contains the group label.

The list of Vulnerabilities are the ones impacting this Component. See Vulnerability object.

Serialized sample:

```
{
  "id": "02074cd3-dbe6-52b6-963c-7247ef33a2b1",
  "creation_time": "2015-07-09T14:17:11.370164+02:00",
  "last_active_time": "2015-09-23T14:52:53.249992+02:00",
  "ip": "192.168.105.130",
  "mac": "28:63:36:82:28:96",
  "name": "PLC_3",
  "model_name": "PLC_3",
  "model_ref": "6ES7 212-1BE40-0XB0",
  "fw_version": "V4.0",
  "hw_version": "1",
  "serial_number": "S C-E7SJ1284",
  "vendor_name": "Siemens AG - Industrial Automation - EWA",
  "project_name": "PROJECT",
  "project_version": "1",
  "tags": {
    "PLC": "info"
  },
  "properties": {
    "name-mac": "Siemens 82:28:96",
    "name-s7plus-plc": "PLC_3",
    "name-vendorip": "Siemens 192.168.105.130",
    "s7plus-hardwarerevision": "1",
    "s7plus-module-ref": "6ES7 212-1BE40-0XB0",
    "s7plus-module-ver": "V4.0",
    "s7plus-serial-number": "S C-E7SJ1284",
    "vendor": "Siemens AG - Industrial Automation - EWA"
  },
  "vulnerabilities": [ ... ]
}
```

5.3 Flow



A Flow is a communication between two Components. It has several attributes such as tags and properties that the Cisco Cyber Vision has identified during its network analysis.

A Flow is identified by:

- its source and destination component
- its MAC address
- its source port
- its content id

A Flow has two endpoints: the source, the component from which the Flow has been started and the destination (the component the Flow is going to). These endpoints objects contain:

- a MAC address: the MAC address of the component
- an IP: the IP address of the component if the Flow used IP for the L3.
- a port (if the Flow used IP for the L3).
- the ID of the component.

The NetworkCategory is a general category created by CCV to regroup the Flows, the possible values are:

- eth: this is an ethernet Flow, mainly L2 information.
- net: this Flow used protocols in the L3 layer which have been considered as "IT" protocols (snmp, http, pop, ...)
- control: this Flow used a control protocol such as S7, S7plus, etc.
- field: the CCV has seen a field protocol in this Flow (e.g. profinet)

FirstSeen identifies the first time this Flow has been active on the network, LastSeen describes the last time this flow has been seen on the network.

Serialized sample:

```
{
  "id": "3d0a4dfc-49e9-54e1-ab7e-f73090a7d47d",
  "src": {
    "mac": "52:54:dd:4c:28:74",
    "ip": "169.254.1.2",
    "component": {
      "id": "7e41e314-c0dc-5153-8ce7-10dc04d48517"
    }
  },
  "dst": {
    "mac": "d0:ec:35:59:b4:4b",
    "ip": "127.0.0.1",
    "component": {
      "id": "2a3629ae-c6c1-58cc-abf3-9e81f02ad65b"
    }
  },
  "ethertype": "IPv4",
  "protocol": "GRE",
  "sensor_id": "a419b4a3-30ce-4b79-8011-d07457084c46",
```

```

"network_category": "net",
"tags":{
  "Tunneling": "2020-07-23 12:30:53.600641+00"
},
"properties":{"erspan3-index": "6", "erspan3-switch-id": "0",
"erspan3-vlan": "507", "gre-tunneling": "ERSPAN-III"...},
"first_seen": "2020-07-23T07:51:05.745183Z",
"last_seen": "2020-07-23T12:20:12.122487Z"
},

```

5.3.1 FlowStats

A flow stats is an aggregation of information extracted during flow network-packets dissection such the number of transiting packet and the flow direction, the length of the layer 2 and 7 (see: OSI model).

Direction-method, describes the method used to identified the client and the server in the flow direction field; the possible values are:

- detected: the direction has been identified during flow processing
- inferred: the client server is chosen at best with the available information

Serialized sample:

```

{ {
  "flow_id": "ff3e0990-f9f0-5e50-8069-f5f28accac17",
  "direction": "Client→Server",
  "direction_method": "detected",
  "nb_packets": 6,
  "l2_bytes": 606,
  "l7_bytes": 282
},
  "
}

```

5.3.2 Flow Test Version



A Flow is a communication between two Components. It has several attributes such as tags and properties that the Cisco Cyber Vision has identified during its network analysis.

A Flow is identified by:

- its source and destination component
- its MAC address
- its source port

A Flow has two endpoints: the source, the component from which the Flow has been started and the destination (the component the Flow is going to). These endpoints objects contain:

- a MAC address: the MAC address of the component
- an IP: the IP address of the component if the Flow used IP for the L3.
- a port (if the Flow used IP for the L3).
- the ID of the component.

The NetworkCategory is a general category created by CCV to regroup the Flows, the possible values are:

- eth: this is an ethernet Flow, mainly L2 information.
- net: this Flow used protocols in the L3 layer which have been considered as "IT" protocols (snmp, http, pop, ...)
- control: this Flow used a control protocol such as S7, S7plus, etc.
- field: the CCV has seen a field protocol in this Flow (e.g. profinet)

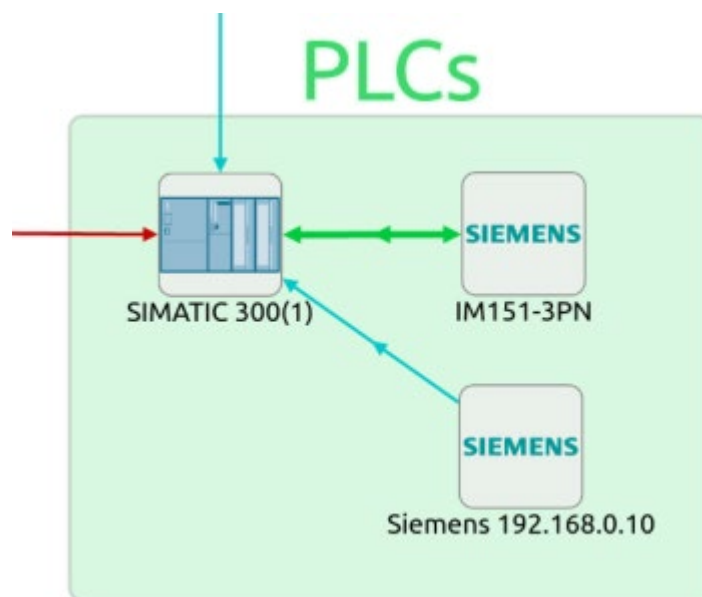
FirstSeen identifies the first time this Flow has been active on the network, LastSeen describes the last time this flow has been seen on the network.

Serialized sample:

```
{
  "id": "ff3e0990-f9f0-5e50-8069-f5f28accac17",
  "src": {
    "mac": "40:ce:24:a8:38:54",
    "ip": "10.160.41.1",
    "port": 56190,
    "component": {"id": "9d32fdae-7dbe-5e05-b5ea-5307358e0108"}
  },
  "dst": {
    "mac": "f4:54:33:a3:88:7a",
    "ip": "10.160.11.100",
    "port": 44818,
    "component": {"id": "e2a699fa-9396-537d-8965-f2524f062843"}
  },
  "ethertype": "IPv4",
  "protocol": "TCP",
  "sensor_id": "c9674309-c384-40fa-845a-b92f74cc3e55",
  "network_category": "control",
  "tags": {
    "EthernetIP": "2020-07-16 08:10:35.864944+00",
    "Start CPU": "2020-07-16 08:10:35.864944+00"
  },
}
```

```
"properties":{
  "enip-cip-request": "false",
  "enip-cpuname": "SecDemo_Cell1PLC",
  "enip-devicetype": "ProgrammableLogicController",
  "enip-event": "Generic",
  "enip-location": "Endpoint",
  "enip-name": "1769-L16ER/B LOGIX5316ER",
  "enip-productcode": "0x99",
  "enip-serial": "60b71080",
  "enip-status":
  "AtLeastOneIOConnectionInRunMode,MinorRecoverableFault,ReservedBits12-15:0x3",
  "enip-status-ra-major": "REM",
  "enip-status-ra-minor": "RUN",
  "enip-value": "RA-ProgramName",
  "enip-vendor": "Rockwell Automation/Allen-Bradley",
  "enip-version": "31.11"
},
"first_seen": "2020-07-16T08:10:06.721229Z",
"last_seen": "2020-07-16T08:10:16.777181Z"
}
```

5.4 Group



Group is a set of Components that can be created from arbitrary criteria: the physical rooms in which the PLCs/computers are located, the purpose of the Components, etc.

Groups have a unique ID and hold some information:

- Name

- Description
- Comments
- Color: purely visual, it's the color used to represent the Group in the Cisco Cyber Vision. The different values for colors are:
 - ◆ LIGHTBLUE
 - ◆ BLUE
 - ◆ PINK
 - ◆ RED
 - ◆ ORANGE
 - ◆ YELLOW
 - ◆ GREEN
- Industrial Impact: set by users of the Cisco Cyber Vision; it is meant to represent the industrial impact of Components inside the group.
- Properties: an array of information set by users of the Cisco Cyber Vision which are formatted as a key, a value, and a time at which the property has been set.

When calling methods returning Group, the API also inserts the information of all Components inside the Group in the response.

Serialized sample of a Group called 'GROUP SIEMENS' containing two Components:

```
{
  "id": "25b745dc-1d93-46ec-8bd8-c647a1d6a46e",
  "creation_time": "2016-02-17T11:47:39.198189+01:00",
  "label": "GROUP SIEMENS",
  "description": "",
  "comments": "",
  "color": "GRAY",
  "industrial_impact": "VERY_LOW",
  "properties": [
    {
      "last_update": "2016-02-17T11:59:09.284025+01:00",
      "position": 0,
      "label": "Location",
      "value": "Room 214"
    }
  ],
  "components": [
    {
      "id": "93b5cfca-77bb-5551-ad5f-e9443279c640",
      "creation_time": "2016-02-12T16:13:47.924038+01:00",
      "last_active_time": "2016-02-12T16:13:47.924038+01:00",
      "ip": "192.168.0.1",
```

```

    "mac": "00:0e:8c:84:5b:a6",
    "name": "Siemens 84:5b:a6",
    "fw_version": "6ES7 315-2EH13-0AB0",
    "hw_version": "3",
    "serial_number": "S C-V1R583472007",
    "vendor_name": "Siemens AG A&D ET",
    "tags": {
      "PLC": "info"
    },
    "properties": {
      "name-mac": "Siemens 84:5b:a6",
      "name-s7-plc": "SIMATIC 300(1)",
      "name-vendorip": "Siemens 192.168.0.1",
      "s7-bootloaderref": "Boot Loader",
      "s7-bootloaderver": "A 10.12.9",
      "s7-fwver": "V 2.5.0",
      "s7-hwref": "6ES7 315-2EH13-0AB0",
      "s7-hwver": "3",
      "s7-modulename": "CPU 315-2 PN/DP",
      "s7-moduleref": "6ES7 315-2EH13-0AB0",
      "s7-modulever": "3",
      "s7-plcname": "SIMATIC 300(1)",
      "s7-rack": "0",
      "s7-serialnumber": "S C-V1R583472007",
      "s7-slot": "2",
      "vendor": "Siemens AG A&D ET"
    },
    "group": "GROUP SIEMENS",
    "vulnerabilities": [ ... ]
  },
  {
    "id": "f36d4e1d-ef90-56b4-bd01-357bc5f4d299",
    "creation_time": "2016-02-12T16:13:47.909701+01:00",
    "last_active_time": "2016-02-12T16:13:47.970315+01:00",
    "ip": "192.168.0.10",
    "mac": "00:0e:8c:83:8b:ae",
    "name": "Siemens 192.168.0.10",
    "vendor_name": "Siemens AG A&D ET",
    "properties": {
      "name-vendorip": "Siemens 192.168.0.10",
      "vendor": "Siemens AG A&D ET"
    },
    "group": "GROUP SIEMENS"
  }
]
}

```

5.5 Event

An Event is created by the Cisco Cyber Vision every time something special occurs.

The `creation_time` represents the time at which the event has been triggered.

The severity of the event can have these values:

- Very High
- High
- Medium
- Low

The Type of the Event can have these values:

- Behavioral
- Classification
- Changes
- Software

The possible values for the Family field are:

- Component
- ICS Cybervision
- PLC Control
- Property
- Protocol Events

The Category can have these values:



- Communication
- Configuration
- Data Management
- Decode Failure
- Identification
- Run Orders
- Security
- User Management

An ID of a Flow or a Component can be embedded in the Event if it has been created by either one of these objects.

Serialized sample:

```
{
  "id": "ecee473-6532-47a1-ac13-516828564b45",
  "creation_time": "2015-10-20T12:05:33.740607+02:00",
  "severity": "low",
  "type": "Classification",
  "family": "Property",
  "category": "Identification",
  "short_message": "New properties detected",
  "message": "Found new normalized properties: model-ref=6ES7 212-1BE40-0XB0, hw-version=1, fw-version=v4.0, serial-number=S C-E7SJ1284",
  "component": {
    "id": "02074cd3-dbe6-52b6-963c-7247ef33a2b1"
  }
}
```

5.6 Variable

GENERAL	FLows (20)	VULNERABILITIES (5)	VARIABLES ACCESS (15)
M 2.0	 STATION-WINCC	READ	First at: Wednesday, September 13, 2017 11:12 AM Last at: Wednesday, September 13, 2017 11:14 AM
		WRITE	First at: Wednesday, September 13, 2017 11:12 AM Last at: Wednesday, September 13, 2017 11:12 AM
M 2.1	 STATION-WINCC	READ	First at: Wednesday, September 13, 2017 11:12 AM Last at: Wednesday, September 13, 2017 11:14 AM

A variable is a storage location paired with an associated symbolic name (an identifier) which contains some information (a value). A Variable is related to a component, a component could access to a variable of another component (variable access) and the access could be read or write (variable access type).

A variable access is created by the Cisco Cyber Vision when a dissected flow contains these kinds of information.

Serialized sample:

```
{
  "9bc31dc8-d1d9-534d-8cbb-3b11fb1e8681": [
    {
      "id": "0602b0f9-af9d-5965-b6bf-cbe69b6c8c2d",
      "name": "7",
      "authors": [
        { "id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
          "read": {"first": "2020-07-23T08:46:05.825955Z", ...}
        }
      ]
    },
    {
      "id": "94728788-e7fc-512f-bbe0-ef9d753757af",
      "name": "5",
      "authors": [
        { "id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
          "read": {"first": "2020-07-23T08:46:05.825961Z", ...}
        }
      ]
    },
    {
      "id": "f123e36f-1e0e-5bdc-82c7-187fb38b51f4",
      "name": "4",
      "authors": [
        { "id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
          "read": {"first": "2020-07-23T08:46:05.817661Z", ...}
        }
      ]
    },
    {
      "id": "00a1b9e7-37cf-5642-81a9-e2cf32243338",
      "name": "2",
      "authors": [
        { "id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
          "read": {"first": "2020-07-23T08:46:05.81718Z", ...}
        }
      ]
    },
    {
      "id": "27cb17d5-32b9-57cb-bef6-b731cd277426",
      "name": "6",
      "authors": [
        { "id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
          "read": {"first": "2020-07-23T08:46:05.825961Z", ...}
        }
      ]
    },
    {
      "id": "84a89176-9fce-5072-8653-d26625caf964",
      "name": "9",
      "authors": [
```

```

{"id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
"read":{"first": "2020-07-23T08:46:05.825936Z",...}
]
},
{
{
"id": "8be402cb-f50e-54ff-9799-9ea3647168f3",
"name": "1",
"authors":[
{"id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
"read":{"first": "2020-07-23T08:46:05.81718Z",...}
]
},
{
{
"id": "a7ff41fb-e8e4-5d35-bda9-b18e1170a4c0",
"name": "8",
"authors":[
{"id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
"read":{"first": "2020-07-23T08:46:05.820274Z",...}
]
},
{
{
"id": "acd42690-3642-5495-a7ad-16d76e7b6060",
"name": "3",
"authors":[
{"id": "cf4f8e39-f2a0-5a91-b256-53706423c445", "name": "NANOBOX",
"read":{"first": "2020-07-23T08:46:05.820289Z",...}
]
}
}
],
"b74823cc-ec08-5feb-88d0-03f17cb64fbb": [
{
{
"id": "48139500-e545-58c4-8383-ebb62b4cb32d",
"name": "MB 1000",
"authors":[
{"id": "9bc31dc8-d1d9-534d-8cbb-3b11fb1e8681", "name": "MICROBOX",
"read":{"first": "2020-07-23T08:46:05.817658Z",...}
]
}
}
]
}
}
}

```

5.7 Error cases

For each HTTP response with a HTTP return code different than a success code (code = 200), the API controller return a simple JSON response like the format below:

```

{
  "status": <An error type>,
  "message": <An error message>
}

```



```
}
```

For example, with a wrong rule action name during a tag creation:

```
{  
  "status": "error",  
  "message": "Invalid syntax for rule.\nwrong action name :  
\"TagFlow2\""  
}
```

6 Methods

This chapter details each method provided by the API.

6.1 Authentication

First thing to get started with the API, you must get a valid token from your Cisco Cyber Vision administrator.

The token must be sent with each call as an URL parameter, example using *curl*:

```
curl --cacert ca.pem
https://192.168.1.128/api/1.0/flow/fea763b5-a292-556e-a986-
af71bffb45eb?token=ics-229972468f69253c08a3ce
256341616f2adce46a-73f8da8d78c0445ebfead6919873d33d134266bb
```

For a valid HTTP/S handshake with the server, you have to use its own self-signed CA certificate. This certificate is available at **<http://192.168.1.128/ca.pem>** **<http://192.168.1.128/ca.pem>** (PEM format) or at **<http://192.168.1.128/ca.crt>** **<http://192.168.1.128/ca.pem>** (DER format). Adding this certificate to your client application key store will authenticate the server.

Alternatively use the *-k* option to disable all certificate verifications of *curl*.

6.2 Sensors

6.2.1 Get sensors

This method allows to retrieve the whole list of sensors used by Cisco Cyber Vision.

Route

```
GET /api/1.0/sensor
```

Example:

Retrieve the list of active and inactive sensors:

```
GET /api/1.0/sensor?token=YOUR_TOKEN&status=active&status=inactive
```

The response will be an array of vendor name like below:

```
{
  "318198cc-f826-4977-b510-10c2ffe37367": {
    "status": "Active",
    "recording": false,
    "recording_last_size": 0,
  }
}
```

```
"name": "My sensor 1",
"recording_last_stop": null,
"auto_config": false,
"flow_rate_day": 0,
"creation_time": "2017-11-13T17:33:33.609234+01:00",
"flows_percentage": "",
"flow_count_day": 0,
"filter": {
  "custom_input": "",
  "capture_mode": "optimal"
},
"version": "2.0.4",
"recording_last_start": null,
"ip": "192.168.69.20",
"serial_number": "SENSOR3-INT16743",
"last_active_time": "2017-11-13T17:37:38.677881+01:00",
"id": "318198cc-f826-4977-b510-10c2ffe37367"
},
"9b2a4bbd-2dbe-49d0-9555-782ae81a41be": {
  "status": "Inactive",
  "recording": false,
  "recording_last_size": 0,
  "name": "My sensor 2",
  "recording_last_stop": null,
  "auto_config": false,
  "flow_rate_day": 0,
  "creation_time": "2017-11-13T17:33:33.609234+01:00",
  "flows_percentage": "",
  "flow_count_day": 0,
  "filter": {
    "custom_input": "",
    "capture_mode": "all"
  },
  "version": "2.0.4",
  "recording_last_start": null,
  "ip": "192.168.69.23",
  "serial_number": "SENSOR3-INT16742",
  "last_active_time": "2017-11-13T17:37:38.677881+01:00",
  "id": "318198cc-f826-4977-b510-10c2ffe37367"
}
}
```

6.3 Components

6.3.1 Get all components

A Component is a node of the network having had communications. This method allows to retrieve the whole list of Components having been created by the CCV ordered by their last active time.

Route

```
GET /api/1.0/component
```

Parameters

- **limit** (mandatory): integer, max number of components to retrieve. (e.g. 100) (maximal value: 2000)
- **order**: 'desc' or 'asc', offset to apply in the list of components to retrieve. (e.g. "desc" will return the most recently active component first, "asc" will return the oldest component first). Default: asc
- **offset**: integer, offset used to move into the list of components available in the CCV. (e.g. 50)
- **start**: UTC date or datetime, low limit for the interval in which must be the returned components. (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")
- **end**: UTC date or datetime, high limit for the interval in which must be the returned components. (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")
- **network**: string, filter to component related to subnet address (e.g. "192.168.1.0/24").
- **ip**: string (may be pass several times), filter to component related to the IP address.
- **mac**: string (may be pass several times), filter to component related to the MAC address.
- **vendor**: string (may be pass several times), filter to component related to the component vendor (e.g. "Dell Inc.").
- **tag**: string (may be pass several times), filter to component related to the tag name (e.g. "Windows")
- **property**: string (may be pass several times), filter to flow related to component property name and/or flow property value (e.g. "netbios-srctype" or "netbios-srctype=WorkstationRedirector").

Example:

Retrieve up to 5 live components tagged has PLC with vendor "Siemens AG - Industrial Automation - EWA" and with has property "s7plus-modulever" equals to "V4.0" and has property "s7plus-moduleref":

```
GET /api/1.0/component?token=YOUR_TOKEN&vendor=Siemens+AG+-+Industrial+Automation+-+EWA&limit=5&generation=0&property=s7plus-modulever%3DV4.0&property=s7plus-moduleref&tag=PLC
```

The response will be an array of serialized Component objects. See "Component" for complete information. In our example:

```
[
  {
    "id": "02074cd3-dbe6-52b6-963c-7247ef33a2b1",
    "creation_time": "2015-07-09T14:17:11.370164+02:00",
    "last_active_time": "2015-09-23T14:52:53.249992+02:00",
    "ip": "192.168.105.130",
    "mac": "28:63:36:82:28:96",
    "name": "PLC_3",
    "model_name": "PLC_3",
    "model_ref": "6ES7 212-1BE40-0XB0",
    "fw_version": "V4.0",
    "hw_version": "1",
    "serial_number": "S C-E7SJ1284",
    "vendor_name": "Siemens AG - Industrial Automation - EWA",
    "project_name": "PROJECT",
    "project_version": "1",
    "tags": {
      "PLC": "info"
    },
    "properties": {
      "name-mac": "Siemens 82:28:96",
      "name-s7plus-plc": "PLC_3",
      "name-vendorip": "Siemens 192.168.105.130",
      "s7plus-hardwarerevision": "1",
      "s7plus-moduleref": "6ES7 212-1BE40-0XB0",
      "s7plus-modulever": "V4.0",
      "s7plus-serialnumber": "S C-E7SJ1284",
      "vendor": "Siemens AG - Industrial Automation - EWA"
    },
    "vulnerabilities": [ ... ]
  }
]
```

6.3.2 Get a component

A Component is a node of the network having had communications. Once a Component has been identified (e.g. its ID has been found in the syslog), this method allows to retrieve the information about the Component.

Route

```
GET /api/1.0/component/{id}
```

Note

must be replaced by the Component ID to query.

Example:

```
GET /api/1.0/component/d8cd6083-6c99-5745-9277-78506b623e4d?token=YOUR_TOKEN
```

The response will be a serialized Component object. See "Component" for complete information. In our example:

```
{
  "id": "d8cd6083-6c99-5745-9277-78506b623e4d",
  "creation_time": "2015-07-09T14:15:35.392279+02:00",
  "last_active_time": "2015-09-23T14:53:11.85282+02:00",
  "ip": "192.168.105.115",
  "mac": "00:1b:1b:b6:f6:99",
  "name": "S7-400 station_1",
  "model_name": "PLC_4",
  "model_ref": "6ES7 412-2EK06-0AB0",
  "fw_version": "V 6.0.3",
  "hw_version": "3",
  "serial_number": "SVPEN250894",
  "vendor_name": "Siemens AG,",
  "project_name": "PROJECT",
  "project_version": "1",
  "tags": {
    "PLC": "info"
  },
  "properties": {
    "name-s7-plc": "S7-400 station_1",
    "name-vendorip": "Siemens 192.168.105.115",
    "s7-bootloaderref": "Boot Loader",
    "s7-bootloaderver": "V 6.0.4",
    "s7-fwver": "V 6.0.3",
    "s7-hwref": "6ES7 412-2EK06-0AB0",
    "s7-hwver": "3",
    "s7-modulename": "PLC_4",
    "s7-moduleref": "6ES7 412-2EK06-0AB0",
    "s7-modulever": "68",
    "s7-plcname": "S7-400 station_1",
    "s7-rack": "0",
    "s7-serialnumber": "SVPEN250894",
    "s7-slot": "2",
    "vendor": "Siemens AG,"
  }
}
```

6.3.3 Set a custom name

A Component is a node of the network having had communications. This method allows to set/update a custom name.

Route

```
PUT /api/1.0/component/{id}/custom-name
```

Parameters

- id: string, the component id to update. (e.g. "13a135ad-c3c6-517b-a4a1-c627e0de46c7")

Body format

```
{
  "custom_name" : "YOU_CUSTOM_NAME"
}
```

Example:

Upsert the custom name related to the component 02074cd3-dbe6-52b6-963c-7247ef33a2b1:

```
PUT /api/1.0/component/02074cd3-dbe6-52b6-963c-7247ef33a2b1/custom-name?token=YOUR_TOKEN
{
  "custom_name" : "My PLC"
}
```

The response will be an array of serialized Component objects. See "Component" for complete information. In our example:

```
[
  {
    "id": "02074cd3-dbe6-52b6-963c-7247ef33a2b1",
    "creation_time": "2015-07-09T14:17:11.370164+02:00",
    "custom_name": "My PLC",
    "last_active_time": "2015-09-23T14:52:53.249992+02:00",
    "ip": "192.168.105.130",
    "mac": "28:63:36:82:28:96",
    "name": "PLC_3",
    "model_name": "PLC_3",
    "model_ref": "6ES7 212-1BE40-0XB0",
    "fw_version": "V4.0",
    "hw_version": "1",
    "serial_number": "S C-E7SJ1284",
    "vendor_name": "Siemens AG - Industrial Automation - EWA",
    "project_name": "PROJECT",
    "project_version": "1",
    "tags": {
      "PLC": "info"
    },
  },
]
```

```
"properties": {
  "name-mac": "Siemens 82:28:96",
  "name-s7plus-plc": "PLC_3",
  "name-vendorip": "Siemens 192.168.105.130",
  "s7plus-hardwarerevision": "1",
  "s7plus-moduleref": "6ES7 212-1BE40-0XB0",
  "s7plus-modulever": "V4.0",
  "s7plus-serialnumber": "S C-E7SJ1284",
  "vendor": "Siemens AG - Industrial Automation - EWA"
},
"vulnerabilities": [ ... ]
}
]
```

6.3.4 Delete a custom name

A Component is a node of the network having had communications. This method allows to delete the existing custom name.

Route

```
DELETE /api/1.0/component/{id}/custom-name
```

Parameters

- id: string, the component id to update. (e.g. "13a135ad-c3c6-517b-a4a1-c627e0de46c7")

Example:

Delete the custom name related to the component 02074cd3-dbe6-52b6-963c-7247ef33a2b1:

```
DELETE /api/1.0/component/02074cd3-dbe6-52b6-963c-7247ef33a2b1/custom-name?token=YOUR_TOKEN
```

The response will be an array of serialized Component objects. See "Component" for complete information. In our example:

```
[
  {
    "id": "02074cd3-dbe6-52b6-963c-7247ef33a2b1",
    "creation_time": "2015-07-09T14:17:11.370164+02:00",
    "last_active_time": "2015-09-23T14:52:53.249992+02:00",
    "ip": "192.168.105.130",
    "mac": "28:63:36:82:28:96",
    "name": "PLC_3",
    "model_name": "PLC_3",
    "model_ref": "6ES7 212-1BE40-0XB0",
    "fw_version": "V4.0",
    "hw_version": "1",
    "serial_number": "S C-E7SJ1284",
    "vendor_name": "Siemens AG - Industrial Automation - EWA",
```



```
"key": "softing",
"description": "Softing"
},
{
"key": "turck",
"description": "TURCK, Inc."
},
{
"key": "siemens_G_SY02_XX_00244",
"description": "CPU 315F-2 DP or PN/DP"
},
{
"key": "dell",
"description": "Dell"
},
]
```

6.3.6 Get icon content

The following method allows you to get the content of an icon.

Route

```
GET /api/1.0/icon/{key}
```

Parameters

- key: string, the key of an icon. (e.g. "netgear")

Body format

```
{
"key": "weintek",
"description": "Weintek Labs. Inc.",
"image": "xxxxxx=",
"content-type": "image/png"
}
```

Example:

Get the content of the custom icon named "netgear":

Request

```
GET /api/1.0/icon/netgear?token=YOUR_TOKEN
```

Response

```
{
"image": "[...]",
"content-type": "image/png",
"description": "netgear",
"key": "netgear"
}
```

Note

The content type could be "image/png", "image/gif", "image/jpeg" or "text/xml" for SVG.

6.3.7 Set a custom icon

The following method allows you to set/update a custom icon.

Route

```
PUT /api/1.0/component/{id}/custom-icon
```

Parameters

- id: string, the component id to update. (e.g. "13a135ad-c3c6-517b-a4a1-c627e0de46c7")

Body format

```
{
  "custom_icon" : "YOU_CUSTOM_ICON"
}
```

Example:

Upsert the custom icon related to the component 9485b49a-39fc-5834-ba14-436e9de26c1b:

```
PUT /api/1.0/component/9485b49a-39fc-5834-ba14-436e9de26c1b/custom-
icon?token=YOUR_TOKEN
{
  "custom_icon" : "cisco"
}
```

The response will be an array of serialized Component objects. Refer to the **Component** (page **Error! Bookmark not defined.**) section for more information.

Example:

```
{
  "vendor_name": "Cisco Systems, Inc",
  "custom_icon": "cisco",
  "name": "192.169.62.254",
  "tags": {
    "Net Management Server": "info"
  },
  "ip": "192.169.62.254",
  "creation_time": "2017-01-18T11:26:28.130892+01:00",
  "properties": {
    "vendor": "Cisco Systems, Inc",
    "name-vendorip": "192.169.62.254"
  }
}
```

```
  },
  "mac": "58:f3:9c:d1:43:60",
  "group": "GRdF Montreal (ADSL)",
  "last_active_time": "2017-01-18T14:30:02.671318+01:00",
  "id": "9485b49a-39fc-5834-ba14-436e9de26c1b"
}
```

6.3.8 Delete a custom icon

The following method allows you to delete an existing custom icon.

Route

```
DELETE /api/1.0/component/{id}/custom-icon
```

Parameters

- `id`: string, the component id to update. (e.g. "13a135ad-c3c6-517b-a4a1-c627e0de46c7")

Example:

Delete the custom icon related to the component 9485b49a-39fc-5834-ba14-436e9de26c1b:

```
DELETE /api/1.0/component/9485b49a-39fc-5834-ba14-436e9de26c1b/custom-  
icon?token=YOUR_TOKEN
```

The response will be an array of serialized Component objects. Refer to the **Component** (page **Error! Bookmark not defined.**) section for more information.

Example:

```
{
  "vendor_name": "Cisco Systems, Inc",
  "name": "192.169.62.254",
  "tags": {
    "Net Management Server": "info"
  },
  "ip": "192.169.62.254",
  "creation_time": "2017-01-18T11:26:28.130892+01:00",
  "properties": {
    "vendor": "Cisco Systems, Inc",
    "name-vendorip": "192.169.62.254"
  },
  "mac": "58:f3:9c:d1:43:60",
  "group": "GRdF Montreal (ADSL)",
  "last_active_time": "2017-01-18T14:30:02.671318+01:00",
  "id": "9485b49a-39fc-5834-ba14-436e9de26c1b"
}
```

6.3.9 Add a component to a group

This method allows to add a Component inside an existing Group.

```
POST /api/1.0/group/{id}/component/{cid}
```

- {id} must be replaced by the Group ID in which a Component will be added.
- {cid} must be replaced by the Component ID to add into the Group.

Example:

Inserting the Component having the ID d3ff2bf4-429a-4ff0-8533-4aa72f95bb9d inside the Group having the id 25b745dc-1d93-46ec-8bd8-c647a1d6a46e:

```
POST /api/1.0/group/25b745dc-1d93-46ec-8bd8-c647a1d6a46e/component/d3ff2bf4-429a-4ff0-8533-4aa72f95bb9d?token=YOUR_TOKEN
```

Responses

HTTP return code	Description
200	The method has been successfully executed and the Component has been added into the group.
309	The given Component is already into a Group and can be added to another one.
400	The provided IDs are not in the good format.
404	Either the Group or the Component doesn't exist.

6.3.10 Remove a component from a group

This method allows to remove a Component from an existing Group.

```
DELETE /api/1.0/group/{id}/component/{cid}
```

- {id} must be replaced by the Group ID in which a Component will be removed.
- {cid} must be replaced by the Component ID to remove from the Group.

Example:

Removing the Component having the ID d3ff2bf4-429a-4ff0-8533-4aa72f95bb9d from the Group having the id 25b745dc-1d93-46ec-8bd8-c647a1d6a46e:

```
DELETE /api/1.0/group/25b745dc-1d93-46ec-8bd8-
c647a1d6a46e/component/d3ff2bf4-429a-4ff0-8533-
4aa72f95bb9d?token=YOUR_TOKEN
```

Responses

HTTP return code	Description
200	The method has been successfully executed and the Component has been removed from the Group.
400	The provided IDs are not in the good format or the given Component isn't in the given Group.
404	Either the Group of the Component doesn't exist.

6.3.11 Get a component flows

A Component is created when a Flow is seen between two nodes. It is possible to retrieve the Flow of a given Component and this is the purpose of this method.

The method can retrieve Flows in a given interval of time (start, end) filtering on the last activity of the Flow, it is paginated (limit, offset) and it is ordered (order).

Route

```
GET /api/1.0/component/{id}/flow
```

- {id} must be replaced by the Component ID for which you want to retrieve the Flows.

Parameters

- limit (mandatory): integer, max number of Flows to retrieve. (e.g. 100) (maximal value: 2000)
- order: ^{'desc'} or ^{'asc'}, offset to apply in the list of Flows to retrieve. (e.g. "desc" will return the most recently active first, "asc" will return the less recently active first). Default: ^{asc}
- offset: ^{integer}, offset used to move into the list of events available in the CCV. (e.g. 50)
- start: ^{date}, low limit for the interval in which the Flow must have been active to be returned. (e.g. "2015-15-10" or "2015-15-10 10:20")
- end: ^{date}, high limit for the interval in which the Flow must have been active to be returned. (e.g. "2015-15-10" or "2015-15-10 10:20")

Example:

Retrieve the two oldest Flow of a Component:

```
GET /api/1.0/component/d8cd6083-6c99-5745-9277-78506b623e4d/flow?token=YOUR_TOKEN&order=asc&limit=2
```

6.3.12 Get a component variables

A Component is a node of the network having had communications. This method allows to retrieve all variables related to a component.

Route

```
GET /api/1.0/component/{id}/variables
```

Parameters

- **id**: string, the component id to update. (e.g. "5ad0ab04-5a74-5f48-8d15-0d46302f9f7b")
- **generation**: int, the generation of the component (reference id) to get component variables for a specific reference. By default, generation equal the live reference id, it's zero.

Example:

Retrieve all variables related to the component 5ad0ab04-5a74-5f48-8d15-0d46302f9f7b:

```
GET /api/1.0/component/5ad0ab04-5a74-5f48-8d15-0d46302f9f7b/variables?token=YOUR_TOKEN[&generation=REFERENCE_ID]
```

The response will be an array of serialized Variable objects. See "Variable" for complete information. In our example:

```
[
  {
    "id": "343fb02a-335e-5bb9-a730-8458bde3f7be",
    "name": "30721",
    "authors": [
      {
        "id": "7d675486-1b42-5e6a-9e7b-18fe164417b3",
        "name": "EMSPDSALL1",
        "read": {
          "first": "2017-03-17T09:06:38.758864+01:00",
          "last": "2017-03-17T09:06:38.758934+01:00",
          "type": "read"
        }
      }
    ]
  },
  {
    "id": "5de75197-1e68-5581-9038-e43d697323c4",
    "name": "64512",
    "authors": [
      {
```

```
        "id": "7d675486-1b42-5e6a-9e7b-18fe164417b3",
        "name": "EMSPDSALL1",
        "read": {
            "first": "2017-03-17T09:06:38.758678+01:00",
            "last": "2017-03-17T09:06:38.758782+01:00",
            "type": "read"
        }
    }
}
],
[...]
```

6.3.13 Get vendor names

A vendor is a property related to a Component. This method allows to retrieve the whole list of vendor having been defined to component by Cisco Cyber Vision.

Route

```
GET /api/1.0/vendor
```

Example:

```
GET /api/1.0/vendor?token=YOUR_TOKEN
```

The response will be an array of vendor name like below:

```
[
  "3COM CORPORATION",
  "ABB Oy / Medium Voltage Products",
  "APPLICOM INTERNATIONAL",
  "APRIL",
  "Cisco Systems, Inc",
  "Dell Inc.",
  "DIGITAL ELECTRONICS CORP.",
  "Fisher-Rosemount Systems Inc.",
  "Hewlett Packard",
  "ipcas GmbH",
  "Rockwell Automation",
  "Siemens AG,",
  "Siemens AG - Industrial Automation - EWA",
  "Siemens Electrical Apparatus Ltd., Suzhou Chengdu Branch",
  "SQUARE D COMPANY",
  "Super Micro Computer, Inc.",
  "TELEMECANIQUE ELECTRIQUE"
]
```


6.3.14 Remove incorrect information

These endpoints allow to remove incorrect information (tags, variables or vulnerabilities) for all or selected components.

Routes

Will remove all tags for component "cad069ab-c5bb-4b11-bbdd-b844db789d97":

```
DELETE /api/1.0/component/remove-tags  
["cad069ab-c5bb-4b11-bbdd-b844db789d97"]
```

Will remove all variables for component "cad069ab-c5bb-4b11-bbdd-b844db789d97":

```
DELETE /api/1.0/component/remove-variables  
["cad069ab-c5bb-4b11-bbdd-b844db789d97"]
```

Will remove all vulnerabilities for component "cad069ab-c5bb-4b11-bbdd-b844db789d97":

```
DELETE /api/1.0/component/remove-vulns  
["cad069ab-c5bb-4b11-bbdd-b844db789d97"]
```

Will remove all tags for all components:

```
DELETE /api/1.0/component/purge-tags
```

Will remove all variables for all components:

```
DELETE /api/1.0/component/purge-variables
```

Will remove all vulnerabilities for all component:

```
DELETE /api/1.0/component/purge-vulns
```

6.4 Flows

6.4.1 Get all flows

Route

```
GET /api/1.0/flows/?token=YOUR_TOKEN
```

Parameters:

- limit, offset, generation should be a positive integer and used for paginate. (maximal value: 2000)
- limit: int, allows to define the number of result per page.
- offset: int, allows to skip that many flows before beginning of the result to return.

- generation: int, filter to flow related to a reference ID (zero for live reference).
- last_seen: timestamp, should be a timestamp in ms since epoch and allows to filter flows last seen after this timestamp.
- order: string, should be "asc" or "desc" and allows to order the result set.
- ip: string (may be pass several times), filter to flow related to source/destination IP address.
- mac: string (may be pass several times), filter to flow related to source/destination MAC address.
- port: string (may be pass several times), filter to flow related to source/destination port.
- tag: string (may be pass several times), filter to flow related to tag name (i.e. see get tags route to list available keys).
- properties: string (may be pass several times), filter to flow related to flow property name and/or flow property value (e.g. "netbios-srctype" or "netbios-srctype=WorkstationRedirector").
- from / to: string, filter to flow related to the source and destination component id.
- component: string (may be pass several times), filter to flow related to the source or destination component id.
- start / end: UTC date or datetime, time interval to filter flows from first / last seen (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")

Example:

Retrieve flow between 2016-11-03 and 2017-11-14 12:05 with tag "Stop CPU" and "Write Var" with industrial as network category and has property "s7plus-moduleref" and with property "s7plus-modulever" should be equal to "V1.6" and with property "s7plus-io-ver" should be equal to "V2.0.2":

```
GET /api/1.0/flows/?token=YOUR_TOKEN&start=2016-11-03+00%3A00&tag=Stop+CPU&tag=Write+Var&limit=2&offset=0&generation=0&end=2017-11-14+12%3A05&property=s7plus-io-ver%3DV2.0.2&property=s7plus-modulever%3DV1.6&property=s7plus-moduleref&order=asc&netcat=field&netcat=control
```

The response will be an array of serialized Flow object. See "**Flow**" for complete information. In our example:

```
[
  {
    "src": {
      "ip": "192.168.105.241",
```

```

        "mac": "34:17:eb:d1:c9:97",
        "component": {
            "id": "f523ce71-19ea-5a7b-8b1a-0670433c35ab"
        },
        "port": 1613
    },
    "tags": {
        "Start CPU": "important",
        "S7Plus": "info",
        "Stop CPU": "important",
        "Read Var": "info",
        "Program Upload": "important",
        "Write Var": "info"
    },
    "dst": {
        "ip": "192.168.105.112",
        "mac": "28:63:36:85:b3:32",
        "component": {
            "id": "f18c406e-e89d-5ce7-9f49-4d03cbe6786f"
        },
        "port": 102
    },
    "id": "c2e0d272-28bf-55c6-abd6-52d3f3451289",
    "network_category": "control",
    "first_seen": "2017-11-14T11:39:32.920547+01:00",
    "properties": {
        "cotp-dst-tsap": "0",
        "s7plus-io-serialnumber": "S C-ENS824882014",
        "s7plus-io-ref": "6ES7 522-1BL00-0AB0",
        "s7plus-serialnumber": "S C-ENSK66872014",
        "s7plus-modulever": "V1.6",
        "s7plus-moduleref": "6ES7 516-3AN00-0AB0",
        "s7plus-programupload": "explore",
        "s7plus-io-ver": "V2.0.2",
        "s7plus-cpustatechange": "start-cpu",
        "s7plus-plcname": "PLC_1",
        "ipv4-ttl": "30",
        "s7plus-hardwarerevision": "4"
    },
    "last_seen": "2017-11-14T11:39:33.007821+01:00",
    "sensor_id": "08866e88-918f-48fc-a90e-a37cc9ddd318"
}
]

```

6.4.2 Get a flow

A Flow is a communication between two Components. It has several attributes such as tags and properties that the Cisco Cyber Vision has identified during its network analyze.

Once a Flow has been identified (its ID is known, by either having been read in a syslog, found in the webapp or read in a **"Get Component Flows"** query result), this method allows to retrieve the information about the Flow.

Route

```
GET /api/1.0/flow/{id}
```

- {id} must be replaced by the Flow ID to query.

Example:

```
GET /api/1.0/flow/fea763b5-a292-556e-a986-af71bffb45eb?token=YOUR_TOKEN
```

The response will be a serialized Flow object. See **"Flow"** for complete information. In our example:

```
{
  "id": "fea763b5-a292-556e-a986-af71bffb45eb",
  "src": {
    "mac": "34:17:eb:d1:c9:97",
    "ip": "192.168.1.241",
    "port": 1616,
    "component": {
      "id": "f523ce71-19ea-5a7b-8b1a-0670433c35ab"
    }
  },
  "dst": {
    "mac": "00:1b:1b:b6:f6:99",
    "ip": "192.168.1.115",
    "port": 102,
    "component": {
      "id": "d8cd6083-6c99-5745-9277-78506b623e4d"
    }
  },
  "network_category": "control",
  "tags": {
    "s7": "info"
  },
  "properties": {
    "cotp-dst-tsap": "100",
    "s7-bootloaderref": "Boot Loader",
    "s7-bootloaderver": "V 6.0.4",
    "s7-function": "NONE",
    "s7-fwver": "V 6.0.3",
    "s7-hwref": "6ES7 412-2EK06-0AB0",
    "s7-hwver": "3",
    "s7-mode": "userdata",
    "s7-modulename": "PLC_4",
    "s7-moduleref": "6ES7 412-2EK06-0AB0",
    "s7-modulever": "68",
    "s7-plcname": "S7-400 station_1",
  }
}
```

```

    "s7-serialnumber": "SVPEN250894"
  },
  "first_seen": "2015-07-09T14:24:49.90637+02:00",
  "last_seen": "2015-09-23T14:24:50.102395+02:00",
  "sensor_id": "08866e88-918f-48fc-a90e-a37cc9ddd318"
}

```

6.4.3 Get a flow content

It is sometimes necessary to not only retrieve the metadata of a Flow, but also its content transiting on the network, this is the purpose of this method.

The method can retrieve the flow content in a given interval of time (*start*, *end*) filtering when the Flow Content has transited on the network, it is paginated (*limit*, *offset*) and it is ordered (*order*).

Route

```
GET /api/1.0/flow/{id}/content
```

- {id} must be replaced by the Flow ID for which you want to retrieve the content.

Parameters

- size (mandatory): integer, ???
- start: UTC date or datetime, low limit for the interval in which the Flow Content must have transited on the network to be present in the result. (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")
- end: UTC date or datetime, high limit for the interval in which the Flow Content must have transited on the network to be present in the result. (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")

Example:

Retrieve the more recent Content of a given Flow.

```
GET /api/1.0/flow/fea763b5-a292-556e-a986-af71bffb45eb/content?token=YOUR_TOKEN&limit=1&order=desc
```

The response will be an array of serialized FlowContent object. See "**Response objects > FlowContent**" for complete information. In our example:

```

[
  {
    "id": "0cbd385f-d1fb-4203-b6bc-0635f67e0dec",
    "time": "2015-07-09T14:24:50.087495+02:00",
    "direction": "Server->Client",
    "content": {
      "s7-function": "NONE",
      "s7-mode": "userdata",
      "s7-modulename": "PLC_4",

```

```
        "s7-plcname": "s7-400 station_1",
        "s7-serialnumber": "SVPEN250894"
    }
}
]
```

6.4.4 Get a flow statistics

It is sometimes necessary to not only retrieve the metadata of a Flow, but also its network stats by flow direction, this is the purpose of this method.

Route

```
GET /api/1.0/flow/{id}/stats
```

Parameters

- {id} must be replaced by the Flow ID for which you want to retrieve the content

Example:

Retrieve stats of a given Flow.

```
GET /api/1.0/flow/fea763b5-a292-556e-a986-af71bffb45eb/stats?token=YOUR_TOKEN
```

The response will be an array of serialized FlowStats object. See "**Response objects > FlowStats**" for complete information. In our example:

```
[ {
  "flow_id": "ff3e0990-f9f0-5e50-8069-f5f28accac17",
  "direction": "Client→Server",
  "direction_method": "detected",
  "nb_packets": 6,
  "l2_bytes": 606,
  "l7_bytes": 282
},
{
  "flow_id": "ff3e0990-f9f0-5e50-8069-f5f28accac17",
  "direction": "Client→Server",
  "direction_method": "detected",
  "nb_packets": 8,
  "l2_bytes": 835,
  "l7_bytes": 403
}
]
```

6.5 Groups

6.5.1 Get all groups

A Group is a group of Component. This method allows to retrieve the whole list of Groups available created in the CCV.

```
GET /api/1.0/group
```

Example:

Retrieve all the Groups existing in the CCV:

```
GET /api/1.0/group?token=YOUR_TOKEN
```

The response will be an array of serialized Groups objects. See "Group" for complete information. In our example, a Group labeled "GROUP SIEMENS" containing two Components:

```
[
  {
    "id": "25b745dc-1d93-46ec-8bd8-c647a1d6a46e",
    "creation_time": "2016-02-17T11:47:39.198189+01:00",
    "label": "GROUP SIEMENS",
    "description": "",
    "comments": "",
    "color": "GRAY",
    "industrial_impact": "VERY_LOW",
    "properties": [
      {
        "last_update": "2016-02-17T11:59:09.284025+01:00",
        "position": 0,
        "label": "Location",
        "value": "Room 214"
      }
    ],
    "components": [
      {
        "id": "93b5cfca-77bb-5551-ad5f-e9443279c640",
        "creation_time": "2016-02-12T16:13:47.924038+01:00",
        "last_active_time": "2016-02-12T16:13:47.924038+01:00",
        "ip": "192.168.0.1",
        "mac": "00:0e:8c:84:5b:a6",
        "name": "Siemens 84:5b:a6",
        "fw_version": "6ES7 315-2EH13-0AB0",
        "hw_version": "3",
        "serial_number": "S C-V1R583472007",
        "vendor_name": "Siemens AG A&D ET",
        "tags": {
          "PLC": "info"
        }
      }
    ]
  }
]
```

```

    "properties": {
      "name-mac": "Siemens 84:5b:a6",
      "name-s7-plc": "SIMATIC 300(1)",
      "name-vendorip": "Siemens 192.168.0.1",
      "s7-bootloaderref": "Boot Loader",
      "s7-bootloaderver": "A 10.12.9",
      "s7-fwver": "V 2.5.0",
      "s7-hwref": "6ES7 315-2EH13-0AB0",
      "s7-hwver": "3",
      "s7-modulename": "CPU 315-2 PN/DP",
      "s7-moduleref": "6ES7 315-2EH13-0AB0",
      "s7-modulever": "3",
      "s7-plcname": "SIMATIC 300(1)",
      "s7-rack": "0",
      "s7-serialnumber": "S C-V1R583472007",
      "s7-slot": "2",
      "vendor": "Siemens AG A&D ET"
    },
    "group": "GROUP SIEMENS",
    "vulnerabilities": [ ... ]
  },
  {
    "id": "f36d4e1d-ef90-56b4-bd01-357bc5f4d299",
    "creation_time": "2016-02-12T16:13:47.909701+01:00",
    "last_active_time": "2016-02-
12T16:13:47.970315+01:00",
    "ip": "192.168.0.10",
    "mac": "00:0e:8c:83:8b:ae",
    "name": "Siemens 192.168.0.10",
    "vendor_name": "Siemens AG A&D ET",
    "properties": {
      "name-vendorip": "Siemens 192.168.0.10",
      "vendor": "Siemens AG A&D ET"
    },
    "group": "GROUP SIEMENS"
  }
]

```

6.5.2 Get a group

A Group is a group of Components. This method allows to retrieve a Group by its ID.

```
GET /api/1.0/group/{id}
```

- {id} must be replaced by the Group ID for which you want to retrieve the information.

Example:

Retrieve the information of the Group having the id 25b745dc-1d93-46ec-8bd8-c647a1d6a46e:

```
GET /api/1.0/group/25b745dc-1d93-46ec-8bd8-
c647a1d6a46e?token=YOUR_TOKEN
```

The response will be a serialized Group object. See "**Group**" for complete information. In our example, a Group labeled "GROUP SIEMENS" containing two Components:

```
{
  "id": "25b745dc-1d93-46ec-8bd8-c647a1d6a46e",
  "creation_time": "2016-02-17T11:47:39.198189+01:00",
  "label": "GROUP SIEMENS",
  "description": "",
  "comments": "",
  "color": "GRAY",
  "industrial_impact": "VERY_LOW",
  "properties": [
    {
      "last_update": "2016-02-17T11:59:09.284025+01:00",
      "position": 0,
      "label": "Location",
      "value": "Room 214"
    }
  ],
  "components": [
    {
      "id": "93b5cfca-77bb-5551-ad5f-e9443279c640",
      "creation_time": "2016-02-12T16:13:47.924038+01:00",
      "last_active_time": "2016-02-12T16:13:47.924038+01:00",
      "ip": "192.168.0.1",
      "mac": "00:0e:8c:84:5b:a6",
      "name": "Siemens 84:5b:a6",
      "fw_version": "6ES7 315-2EH13-0AB0",
      "hw_version": "3",
      "serial_number": "S C-V1R583472007",
      "vendor_name": "Siemens AG A&D ET",
      "tags": {
        "PLC": "info"
      },
      "properties": {
        "name-mac": "Siemens 84:5b:a6",
        "name-s7-plc": "SIMATIC 300(1)",
        "name-vendorip": "Siemens 192.168.0.1",
        "s7-bootloaderref": "Boot Loader",
        "s7-bootloaderver": "A 10.12.9",
        "s7-fwver": "V 2.5.0",
        "s7-hwref": "6ES7 315-2EH13-0AB0",
        "s7-hwver": "3",
        "s7-modulename": "CPU 315-2 PN/DP",

```

```

        "s7-moduleref": "6ES7 315-2EH13-0AB0",
        "s7-modulever": "3",
        "s7-plcname": "SIMATIC 300(1)",
        "s7-rack": "0",
        "s7-serialnumber": "S C-V1R583472007",
        "s7-slot": "2",
        "vendor": "Siemens AG A&D ET"
    },
    "group": "GROUP SIEMENS",
    "vulnerabilities": [ ... ]
},
{
    "id": "f36d4e1d-ef90-56b4-bd01-357bc5f4d299",
    "creation_time": "2016-02-12T16:13:47.909701+01:00",
    "last_active_time": "2016-02-12T16:13:47.970315+01:00",
    "ip": "192.168.0.10",
    "mac": "00:0e:8c:83:8b:ae",
    "name": "Siemens 192.168.0.10",
    "vendor_name": "Siemens AG A&D ET",
    "properties": {
        "name-vendorip": "Siemens 192.168.0.10",
        "vendor": "Siemens AG A&D ET"
    },
    "group": "GROUP SIEMENS"
}
]
}

```

6.5.3 Create a group

This method allows to create a **Group** (page **Error! Bookmark not defined.**) with a set of **Component** (page **Error! Bookmark not defined.**) IDs. The information to create the Group must be given to the call in JSON inside the HTTP query.

WARNING

It is important to set the HTTP header 'Content-Type' as 'application/json' while calling this method.

POST /api/1.0/group

JSON Body format

Field	Type	Optional	Description
label	string	false	The label to give to the group.
description	string	true	A short description of the group.

Field	Type	Optional	Description
comments	string	true	Creator comments about the group.
color	string	true	Color to assign to the group in the Cisco Cyber Vision webapp. Possible values: LIGHTBLUE BLUE PINK RED ORANGE YELLOW GREEN
components	array of ids	false	Array of IDs of the Component to initially add to the Group. At least one is mandatory.

Example:

```
POST /api/1.0/group?token=YOUR_TOKEN

{
  "label": "Room 214"
  "description": "All the PLCs installed in Room 214 of Floor 1",
  "comments": "Sentryo Sensor's in left servers rack",
  "color": "RED",
  "components": [
    "d3ff21bf4-429a-4ff0-8533-4aa72f95bb9d",
    "2162db69e-ae27-4b06-82b8-ca53fcfe3772",
    "26028b202-a87a-4485-9bd2-36c8f617a961",
    ...
  ]
}
```

Responses

On successful creation, the created Group in its serialization format (see Responses objects chapter) is returned with an HTTP 200 return code.

In other case, an empty HTTP response with one of these error codes is returned:

HTTP return code	Description
309	The Group can't be created because a Component in the list of given Components is already in a Group.
400	The provided JSON can't be properly parsed.

409 The Group can't be created because a Group with this label already exists.

6.5.4 Edit a group

This method allows to edit a **Group** (page **Error! Bookmark not defined.**) information. All fields are optional in order to be able to change only one if necessary.

WARNING

It is important to set the HTTP header 'Content-Type' as 'application/json' while calling this method.

```
PUT /api/1.0/group/{id}
```

JSON Body format

Field	Type	Optional	Description
label	string	true	To set a new name to the group.
description	string	true	To set a new description to the group.
comments	string	true	To change the comments about the group.
color	string	true	To assign a new color to the group in the Cisco Cyber Vision webapp. Possible values: LIGHTBLUE BLUE PINK RED ORANGE YELLOW GREEN

Example:

In this example we change the Comments and the Color of the Group with ID 25b745dc-1d93-46ec-8bd8-c647a1d6a46e:

```
PUT /api/1.0/group/d2abf7a0-e0e4-4a0c-a021-6c0dc62ea971?token=YOUR_TOKEN
{
  "comments": "sentryo sensor's in right servers rack",
  "color": "PINK"
}
```

Responses

On successful edition, an empty HTTP 200 response is returned. In other case, an empty HTTP response with one of these error codes is returned:

HTTP return code	Description
309	The Group can't because a Component in the list of given Components is already in a Group.
400	The provided JSON can't be properly parsed.
404	A Group with the given ID can't be found.
409	The Group can't be renamed because a Group with this label already exists.

6.5.5 Explode a group

This method allows to explode a Group of the Cisco Cyber Vision by its ID. Exploding a Group only means that the Components of the Group are no more grouped, but it doesn't delete anything except the Group itself. The Components are not deleted or anything.

Be sure of your intention with this method because it's can't be undone.

```
DELETE /api/1.0/group/{id}
```

- {id} must be replaced by the Group ID to delete permanently.

Example:

Explode the Group having the id 25b745dc-1d93-46ec-8bd8-c647a1d6a46e:

```
DELETE /api/1.0/group/25b745dc-1d93-46ec-8bd8-c647a1d6a46e?token=YOUR_TOKEN
```

An empty HTTP 200 response will be returned if the group has been successfully exploded.

6.6 Events

6.6.1 Get all events

An Event is created by the Cisco Cyber Vision every time something special occurs. This method allows to retrieve the list of all events stored in the CCV.

The method can retrieve the events in a given interval of time (*start*, *end*) filtering on the Event occurring time, it is paginated (*limit*, *offset*) and it is ordered (*order*).

Route

```
GET /api/1.0/event
```

Parameters

- **limit (mandatory):** *integer*, max number of events to retrieve. (e.g. 100) (maximal value: 2000)
- **order:** *'desc'* or *'asc'*, offset to apply in the list of events to retrieve. (e.g. "desc" will return the most recent event first, "asc" will return the oldest event first). Default: asc
- **offset:** *integer*, offset used to move into the list of events available in the CCV. (e.g. 50)
- **start:** UTC date or datetime, low limit for the interval in which must be the returned events. (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")
- **end:** UTC date or datetime, high limit for the interval in which must be the returned events. (e.g. "2015-15-10" or "2015-15-10 10:20" or "2015-15-10 10:20:05")
- **severity (or severities):** string (may be pass several times), filter to event related to severities (e.g. "veryhigh").
- **category:** string (may be pass several times), filter to event related to categories (e.g. "Identification").
- **family:** string (may be pass several times), filter to event related to families (e.g. "Component").
- **type:** string (may be pass several times), filter to event related to types.
- **group:** string (may be pass several times), filter to event related to the associated component-group ids.
- **text:** string, a pattern to match event text content (e.g. "New comp").
- **network:** string, filter to event related to the associated component subnet (e.g. "192.168.1").
- **ip:** string (may be pass several times), filter to event related to source/destination IP address.
- **mac:** string (may be pass several times), filter to event related to source/destination MAC address.
- **from / to:** string, filter to event related to the source and destination component id.

Note

Refer to the Event (page **Error! Bookmark not defined.**) section to know allowed category/family/type/severity list.

Example:

Retrieve the 5 events between the 2016-11-03 00:00 and the 2017-11-13 17:23, with "high" or "very-high" as severity and in "Identification" as category related to a component which have "d4:ae:52:aa:dc:93" as MAC address and "10.4.0.46" as IP address sorted by most recent first:

```
GET /api/1.0/event?token=YOUR_TOKEN&category=Identification&start=2016-11-03+00%3A00&mac=d4%3Aae%3A52%3Aaa%3Adc%3A93&limit=5&severity=veryhigh&severity=high&offset=0&end=2017-11-13+17%3A23&ip=10.4.0.46&order=desc
```

The response will be an array of serialized Event objects. See "**Event**" for complete information. In our example:

```
[
  {
    "category": "Identification",
    "severity": "High",
    "family": "Component",
    "component": {
      "id": "5a14574b-f191-5e8f-ac1b-2b6a08631c78"
    },
    "creation_time": "2017-11-13T17:35:54.290739+01:00",
    "message": "New component detected on the network: IP 10.4.0.46, MAC d4:ae:52:aa:dc:93, vendor Dell",
    "type": "Classification",
    "id": "a400af14-ff97-47a4-ac33-c33cc50eff99",
    "short_message": "New component detected"
  }
]
```

6.6.2 Get an event downloadable content

This method returns the downloadable content of a given event (e.g. PCAP files for decode failures).

URL/Method

```
GET /api/1.0/event-data/{id}/download
```

Example:

Request

```
GET /api/1.0/event-data/{event-id}/download?token=YOUR_TOKEN_HERE
```

Response

```
<the file linked to the event (if any)>
```

6.7 Variables

6.7.1 Get all variables

This method allows to retrieve all variables indexed by PLC/component id.

Route

```
GET /api/1.0/variables
```

Parameters

- limit (mandatory): integer, max number of events to retrieve. (e.g. 100, should be less than 2000 as threshold)
- order: 'desc' or 'asc', offset to apply in the list of events to retrieve. (e.g. "desc" will return the most recent content first, "asc" will return the oldest content first). Default: asc
- offset: integer, offset used to move into the list of events available in the CCV. (e.g. 50)

Example:

Retrieve three first variables:

```
GET /api/1.0/variables/?token=YOUR_TOKEN&limit=3&order=asc&offset=0
```

The response will be an array of serialized Variable objects indexed by PLC/component id.

See "Variable" for complete information. In our example :

```
{
  "14796b0e-d2f4-51f1-a2c1-8e67a5f05bbb": [
    {
      "authors": [
        {
          "read": {
            "type": "read",
            "last": "2017-01-18T12:20:16.331505+01:00",
            "first": "2017-01-18T11:30:16.99657+01:00"
          },
          "id": "1491688a-da68-52ac-b004-aa05b5df19c2",
          "name": "192.168.101.237"
        }
      ],
      "id": "0001ad0c-73be-53b6-a8a6-6c790adb9162",
      "name": "w414"
    }
  ],
  "62208e8c-ab62-5b57-aa5f-a9bfe7e4d96a": [
    {
      "authors": [
```



```

    {
      "read": {
        "type": "read",
        "last": "2017-01-18T12:20:15.01314+01:00",
        "first": "2017-01-18T11:30:15.863019+01:00"
      },
      "id": "1491688a-da68-52ac-b004-aa05b5df19c2",
      "name": "192.168.101.237"
    },
    {
      "read": {
        "type": "read",
        "last": "2017-01-18T14:30:14.378108+01:00",
        "first": "2017-01-18T12:30:14.377105+01:00"
      },
      "id": "5f74340a-1610-5d0f-b2a8-e507a12094b7",
      "name": "192.168.101.238"
    }
  ],
  "id": "0000d4d8-9b3c-51e9-80e2-3d88a492a30c",
  "name": " bool 133"
}

```

6.8 Operator, Parameters, Conditions & Actions

OPERATOR_TYPE

OPERATOR_TYPE is the operator used between rule's conditions for validating the rule's test.

OPERATOR_TYPE has to be one of these values: [or|and].

PARAMETERS_LIST

PARAMETERS_LIST is used to pass parameters to conditions or action (see **CONDITION_TYPE/ACTION_TYPE**)

The common JSON format for a set of parameters is:

```

{
  "parameter_name_1" : " parameter_value_1",
  "parameter_name_2" : " parameter_value_2",
  "parameter_name_3" : " parameter_value_3"
}

```

Example:

```

{
  "tag": "s7",
  "type": "info",

```

```
"throw_admin_event": true
}
```

CONDITION_TYPE

CONDITION_TYPE is an ICS internal function which returns a Boolean value. These functions will validate, or not, the rule's test. They use a **PARAMETERS_LIST** to specify arguments.

- **HasProp**: will check if component has a property. Available parameter:
property: property name
- **PropValue**: will check if component has a property and its value. Available parameters:
property: property name
value: property value

Example:

```
{
  "name": "HasProp",
  "params": {
    "property": "s7-function"
  }
}
```

ACTION_TYPE

ACTION_TYPE is an ICS internal function that will be performed if a condition is validated.

- **TagFlow**: tag a flow. Available parameters:
 - ◆ tag: tag name. Available values: all available tags from the Tags API
 - ◆ type: tag class. Available values: [info|important]
 - ◆ throw_admin_event: throw an admin event (or not). Available values: [true|false]
- **Netcat**: set flow's netcat. Available parameters:
 - ◆ value: netcat value. Available values: [field|control|eth|net]
- **TagComponent**: tag a component. Available parameters:
 - ◆ tag: tag name. Available values: all available tags from the Tags API
 - ◆ value: tag class. Available values: [info|important]
 - ◆ target: target component, Source or Destination. Available values: [src|dst]
- **SetComponentProperty**: set property of a component. Available parameters:

- ◆ target: target component, Source or Destination. Available values: [src|dst]
- ◆ property: component property name.
- ◆ value: property value.
- RemoveComponentProperty: remove property of a component. Available parameters:
 - ◆ target: target component, Source or Destination. Available values: [src|dst]
 - ◆ property: component property name.
- CopyProperties: Copy properties from flow to component. Available parameters:
 - ◆ properties: array of properties.
 - ◆ target: target component, Source or Destination. Available values: [src|dst]
 - ◆ protocol: protocol related to these properties.
- SendEvent: Send an event. Available parameters:
 - ◆ trigger: Event trigger. Available values: [PROG_DL_DETECTED|PROG_UL_DETECTED|NEW_COMM_DETECTED|EXCEPTION_DETECTED|INIT_DETECTED|START_CPU_DETECTED|STOP_CPU_DETECTED]
 - ◆ type: Event type. Available values: [Behavioral|Classification|Changes|Software]
- severity: Event severity. Available values: [Low|Medium|High|Very High]
 - ◆ category: Event category. Available values: [Decode Failure|User Management|Data Management|Security|Identification|Run Orders|Configuration|Communication]
 - ◆ family: Event family. Available values: [Property|PLC Control|Protocol Events|Component|ICS Cybervision]
- SendEventException: Send an exception event. Available parameters:
 - ◆ trigger: Event trigger. Available values: [PROG_DL_DETECTED|PROG_UL_DETECTED|NEW_COMM_DETECTED|EXCEPTION_DETECTED|INIT_DETECTED|START_CPU_DETECTED|STOP_CPU_DETECTED]
 - ◆ type: Event type. Available values: [Behavioral|Classification|Changes|Software]
 - ◆ severity: Event severity. Available values: [Low|Medium|High|Very High]

- ◆ category: Event category. Available values: [Decode Failure|User Management|Data Management|Security|Identification|Run Orders|Configuration|Communication]
- ◆ family: Event family. Available values: [Property|PLC Control|Protocol Events|Component|ICS Cybervision]

Example:

```
{
  "name": "TagFlow",
  "params": {
    "tag": "SSH",
    "type": "important",
    "throw_admin_event": true
  }
}
```

PORT_ARRAY

PORT_ARRAY is an array of value of port. Each value could be prefixed of protocol or not.

Example:

- ["TCP/80"]: TCP Flow on port 80
- ["UDP/8080"]: UDP Flow on port 8080
- ["22"]: TCP&UDP flow on port 22
- ["TCP/80", "UDP/8080", "22"]: TCP Flow on port 80 and 22, UDP Flow on port 8080 and 22

6.8.1 Tags

6.8.1.1 Get all tags

This method returns all the tags available in the Cisco Cyber Vision. It contains the ones that the Knowledge DB file has imported, but also the ones added by the API.

URL/Method

```
GET /api/1.0/tags/
```

Example:**Request**

```
GET /api/1.0/tags?token=YOUR_TOKEN_HERE
```

Response

```
[{
```

```
    "tag": "WEB",
    "label": "web",
    "desc": "web protocols"
  },{
    "tag": "WEB_SERVER",
    "label": "web server",
    "desc": "web server computer"
  },{
    "tag": "WINDOWS",
    "label": "windows",
    "desc": "windows workstation"
  }
]
```

6.8.1.2 Get a custom tag

URL/Method

```
GET /api/1.0/analyzer/tag/{tag}
```

Example:

Request

```
GET /api/1.0/analyzer/tag/WEB?token=YOUR_TOKEN_HERE
```

Response

```
{
  "tag": "WEB",
  "label": "web"
}
```

6.8.1.3 Create a tag

Creates a tag with the given key, label, description, type and domain. Only the key and label are mandatory.

Tags created via the API are grouped in the "My tag" category.

URL/Method

```
POST /api/1.0/analyzer/tag?token=YOUR_TOKEN_HERE
```

Body format

```
{
  "tag": TAG_KEY,
  "label": TAG_NAME,
  "desc": TAG_DESCRIPTION
  "type": TAG_TYPE
  "IT": TAG_DOMAIN
  "OT": TAG_DOMAIN
}
```

Note

The key can only contain char between a and z (lower or uppercase) and _.
The type can take two values : "flow" and "component".
For the domain, the field "IT" and "OT" are of type booleans.
By default:
If "type" is not filled, it takes the value "flow".
If "IT" and "OT" tag domain are not filled, they take the value "true".

Example:**Request**

```
POST /api/1.0/analyzer/tag?token=YOUR_TOKEN_HERE
```

Body

```
{
  "tag": TAG_KEY,
  "label": TAG_NAME,
  "desc": TAG_DESCRIPTION
  "type": flow
  "IT": true
  "OT": false
}
```

6.8.1.4 Update a tag**URL/Method**

```
PUT /api/1.0/analyzer/tag/{tag}
{
  "tag": TAG_KEY,
  "label": TAG_NAME
}
```

Example:**Request**

```
PUT /api/1.0/analyzer/tag/WEB_CUSTOM?token=YOUR_TOKEN_HERE
```

Body

```
{
  "tag": "WEB_CUSTOM",
  "label": "web Custom Updated"
}
```

6.8.1.5 Delete a tag

This method deletes a tag previously added with the API. Note that it doesn't delete tags imported by the Knowledge DB file.

URL/Method

```
DELETE /api/1.0/analyzer/tag/{tag}
```

Example:

Request

```
DELETE /api/1.0/analyzer/tag/WEB_CUSTOM?token=YOUR_TOKEN_HERE
```

6.8.2 Property analyzer rules

6.8.2.1 Get all property analyzer rules

URL/Method

```
GET /api/1.0/analyzer/property/rule/
```

Example:

Request

```
/api/1.0/analyzer/property/rule/?token=YOUR_TOKEN_HERE
```

Response

```
[{
  "id": "bd1c1b0b-eb8c-4740-a870-e1fac758ef2b",
  "test": {
    "operator": "or",
    "conditions": [{
      "name": "HasProp",
      "params": {
        "property": "s7-function"
      }
    }
  ]
},
  "actions": [{
    "name": "TagFlow",
    "params": {
      "tag": "S7",
      "type": "info",
      "throw_admin_event": true
    }
  }
]
}],{
  "id": "8d95c60b-c1e6-4e04-b3ed-4c8b8d339b0e",
  "test": {
    "operator": "and",
    "conditions": [{
      "name": "PropValue",
      "params": {
        "property": "dst_port",
```

```

        "value": "22"
      }
    }
  ],
  "actions": [{
    "name": "TagFlow",
    "params": {
      "tag": "SSH",
      "type": "important",
      "throw_admin_event": true
    }
  }
]}

```

6.8.2.2 Get a property analyzer rule

URL/Method

```
GET /api/1.0/analyzer/property/rule/{id}
```

Example:

Request

```
/api/1.0/analyzer/property/rule/bd1c1b0b-eb8c-4740-a870-
e1fac758ef2b?token=YOUR_TOKEN_HERE
```

Response

```

{
  "id": "bd1c1b0b-eb8c-4740-a870-e1fac758ef2b",
  "test": {
    "operator": "or",
    "conditions": [{
      "name": "HasProp",
      "params": {
        "property": "s7-function"
      }
    }
  ]
},
  "actions": [{
    "name": "TagFlow
",
    "params": {
      "tag": "S7",
      "type": "info",
      "throw_admin_event": true
    }
  }
]}

```


6.8.2.3 Create a property analyzer rule

URL/Method

```
POST /api/1.0/analyzer/property/rule/
```

Body format

```
{
  "test": {
    "operator": OPERATOR_TYPE,
    "conditions": [{
      "name": CONDITION_TYPE,
      "params": {
        PARAMETERS_LIST
      }
    }]
  },
  "actions": [{
    "name": ACTION_TYPE,
    "params": {
      PARAMETERS_LIST
    }
  }]
}
```

Note

Refer to the *Operator, Parameters, Conditions & Actions* (page **Error! Bookmark not defined.**) section for more details.

Example:

Request

```
POST /api/1.0/analyzer/property/rule/?token=YOUR_TOKEN_HERE
```

Body

```
{
  "test": {
    "operator": "or",
    "conditions": [{
      "name": "HasProp",
      "params": {
        "property": "s7plus-function"
      }
    }]
  },
  "actions": [{
    "name": "TagFlow",
    "params": {
      "tag": "s7 ",
      "type": "info",
    }
  }]
}
```

```

        "throw_admin_event": true
    }
}

```

6.8.2.4 Update a property analyzer rule

URL/Method

```
PUT /api/1.0/analyzer/property/rule/{id}
```

Body format

```

{
  "id": {id},
  "test": {
    "operator": OPERATOR_TYPE,
    "conditions": [{
      "name": CONDITION_TYPE,
      "params": {
        PARAMETERS_LIST
      }
    }]
  },
  "actions": [{
    "name": ACTION_TYPE,
    "params": {
      PARAMETERS_LIST
    }
  }]
}

```

Example:

Request

```
PUT /api/1.0/analyzer/property/rule/bd1c1b0b-eb8c-4740-a870-e1fac758ef2b?token=YOUR_TOKEN_HERE
```

Body

```

{
  "id": "bd1c1b0b-eb8c-4740-a870-e1fac758ef2b",
  "test": {
    "operator": "or",
    "conditions": [{
      "name": "HasProp",
      "params": {
        "property": "s7plus-function"
      }
    }],
    {
      "name": "HasProp",
      "params": {
        "property": "s7-function"
      }
    }
  }
}

```

```

    }
  }
},
"actions": [{
  "name": "TagFlow",
  "params": {
    "tag": "S7",
    "type": "info",
    "throw_admin_event": true
  }
}]
}
}
}

```

6.8.2.5 Delete a property analyzer rule

URL/Method

```
DELETE /api/1.0/analyzer/property/rule/{id}
```

Example:

Request

```
DELETE /api/1.0/analyzer/property/rule/bd1c1b0b-eb8c-4740-a870-
e1fac758ef2b?token=YOUR_TOKEN_HERE
```

6.8.3 Port analyzer rules

6.8.3.1 Get all port analyzer rules

URL/Method

```
GET /api/1.0/analyzer/port/rule/?token=YOUR_TOKEN_HERE
```

Response

```

[
  {
    "id": "bd1c1b0b-eb8c-4740-a870-e1fac758ef2b",
    "ports": [TCP/80, UPD/8080, 9000],
    "actions": [
      {
        "name": "TagFlow",
        "params": {
          "tag": "WEB",
          "type": "info",
          "throw_admin_event": true
        }
      }
    ]
  },
  {
    "id": "8d95c60b-c1e6-4e04-b3ed-4c8b8d339b0e",
    "ports": [22],
    "actions": [
      {
        "name": "TagFlow",

```

```
        "params": {
            "tag": "SSH",
            "type": "important",
            "throw_admin_event": true
        }
    }
}]
}]
```

6.8.3.2 Get a port analyzer rule

URL/Method

```
GET /api/1.0/analyzer/port/rule/{id}
```

Example:

Request

```
GET /api/1.0/analyzer/port/rule/bd1c1b0b-eb8c-4740-a870-
e1fac758ef2b?token=YOUR_TOKEN_HERE
```

Response

```
{
  "id": "bd1c1b0b-eb8c-4740-a870-e1fac758ef2b",
  "ports": [TCP/80, UPD/8080, 9000],
  "actions": [{
    "name": "TagFlow",
    "params": {
      "tag": "WEB",
      "type": "info",
      "throw_admin_event": true
    }
  }
}
```

6.8.3.3 Create a port analyzer rule

URL/Method

```
POST /api/1.0/analyzer/port/rule/
```

Body format

```
{
  "ports": PORT_ARRAY,
  "actions": [{
    "name": ACTION_TYPE,
    "params": {
      PARAMETERS_LIST
    }
  }
}]
}
```

Note

Refer to the (page **Error! Bookmark not defined.**) section for more details.

Example:**Request**

```
POST /api/1.0/analyzer/port/rule/?token=YOUR_TOKEN_HERE
```

Body

```
{
  "ports": ["22"],
  "actions": [{
    "name": "TagFlow",
    "params": {
      "tag": "SSH",
      "type": "important",
      "throw_admin_event": true
    }
  }]
}
```

6.8.3.4 Update a port analyzer rule**URL/Method**

```
PUT /api/1.0/analyzer/port/rule/{id}
```

Body format

```
{
  "id": {id}
  "ports": PORT_ARRAY,
  "actions": [{
    "name": ACTION_TYPE,
    "params": {
      PARAMETERS_LIST
    }
  }]
}
```

Note

Refer to the (page **Error! Bookmark not defined.**) section for more details.

Example:**Request**

```
PUT /api/1.0/analyzer/port/rule/bd1c1b0b-eb8c-4740-a870-
e1fac758ef2b?token=YOUR_TOKEN_HERE
```

Body

```
{
  "id": "bd1c1b0b-eb8c-4740-a870-e1fac758ef2b",
  "ports" : [234]
  "actions": [{
    "name": "TagFlow",
    "params": {
      "tag": "s7",
      "type": "info",
      "throw_admin_event": true
    }
  }
}]
}
```

6.8.3.5 Delete a port analyzer rule**URL/Method**

```
DELETE /api/1.0/analyzer/port/rule/{id}
```

Example:**Request**

```
DELETE /api/1.0/analyzer/port/rule/bd1c1b0b-eb8c-4740-a870-
e1fac758ef2b?token=YOUR_TOKEN_HERE
```

7 Examples

In this chapter, you can find some basic examples showing the usage of the API.

Note

These examples need at least Python 3.5.6 to be executed.

7.1 Get Component MAC by its ID

```
#!/usr/bin/python3
#
# Example using the Get Component method
# to retrieve the MAC of a component.
# Must be executed with Python >= 3.5.6

import json
import ssl
import sys
import urllib.request, urllib.parse, urllib.error

center_ip = '169.254.0.42:4443'
token = 'ics-8c9535ccce063ee56e6d3fc8d72bbd9a84f02f97-
d307d2cd42a0b77b280dac9f5d0d6bd6d03f4e7c'

def main(argv):
    if len(argv) == 0:
        print('Usage: python get_component_mac.py COMPONENT-ID')
        sys.exit(1)
    # request
    route = 'https://' + center_ip + '/api/1.0/component/{id}'
    url = route.replace('{id}', sys.argv[1])
    ctx = create_ssl_context()
    r = urllib.request.urlopen(url + '?token=' + token, context=ctx)
    # read the response
    data = ''
    for line in r.readlines():
        data += line.decode('UTF-8', 'ignore')
    # parse the JSON
    component = json.loads(data)
    # print the MAC
    print((component['mac']))

def create_ssl_context():
    ctx = ssl.create_default_context(cafile='tmp/ca.pem')
    ctx.check_hostname = False
```

```
    return ctx

if __name__ == "__main__":
    main(sys.argv[1:])
```

Note

ca.pem key file should be added on /tmp directory if not already available

Example of calling the script if the filename is `get_component_mac_by_id.py` to get the MAC of the Component having the ID '93b5cfca-77bb-5551-ad5f-e9443279c640':

```
Python3 get_component_mac_by_id.py 93b5cfca-77bb-5551-ad5f-
e9443279c640
```

7.2 Print the last active Flow of a Component

```
#!/usr/bin/python3
#
# Example using the Get last flow Component method
# to retrieve the last flow of a known component ID.
# Must be executed with Python >= 3.5.6

import json
import ssl
import sys
import urllib.request, urllib.parse, urllib.error

center_ip = '169.254.0.42:4443'
token = 'ics-8c9535ccce063ee56e6d3fc8d72bbd9a84f02f97-
d307d2cd42a0b77b280dac9f5d0d6bd6d03f4e7c'

def main(argv):
    # request
    route = 'https://' + center_ip + '/api/1.0/component/{id}/flow'
    url = route.replace('{id}', sys.argv[1])
    ctx = create_ssl_context()
    r = urllib.request.urlopen(url + '?limit=1&token=' + token,
    context=ctx)
    # read the response
    data = ''
    for line in r.readlines():
        data += line.decode('UTF-8')
    # parse the JSON
    flow = json.loads(data)
    # pretty print of the JSON
```



```

print((json.dumps(flow, sort_keys=True, indent=4)))

# Create an SSL context using the Center self-signed CA.
def create_ssl_context():
    ctx = ssl.create_default_context(cafile='tmp/ca.pem')
    ctx.check_hostname = False
    return ctx

if __name__ == "__main__":
    main(sys.argv[1:])

```

Example of calling the script if the filename is `last_active_flow.py` to get the last active Flow of the Component having the ID '93b5cfca-77bb-5551-ad5f-e9443279c640':

```
Python3 last_active_flow.py 821744de-07ac-5362-85fd-76a9594c318f
```

First access timestamp:

```

[
  {
    "dst": {
      "component": {
        "id": ""
      },
      "ip": "10.232.45.23",
      "mac": "8c:ec:4b:b3:0c:6d",
      "port": 50692
    },
    "ethertype": "IPv4",
    "first_seen": "2020-07-13T14:17:21.130341z",
    "id": "821744de-07ac-5362-85fd-76a9594c318f",
    "last_seen": "2020-07-13T14:17:21.132848z",
    "network_category": "net",
    "properties": {
      "gssspnego-mechanism-supported": "GSS-API Kerberos Version
5",
      "gssspnego-mechanism-types": "Negoex,GSS-API Kerberos
Version 5,Kerberos v5,Kerberos v5 user to user,NLTMSSP",
      "gssspnego-nego-result": "accept_completed",
      "smb-command": "smb-command-negotiate",
      "smb-dialect-proposed": "NT LM 0.12,SMB 2.002,SMB 2.???",
      "smb-dialect-selected": "2.*",
      "smb-tree-name": "\\VUFLPRNP101.emea.int.grp\\IPC$",
      "smb-version": "1",
      "smb-version1": "yes",
      "smb-version2": "yes"
    },
    "protocol": "TCP",
    "sensor_id": "8f034b99-ae02-4b1b-85ca-0b9f6955ece2",
    "src": {
      "component": {

```

```
        "id": ""
      },
      "ip": "10.232.9.130",
      "mac": "00:a0:12:34:cc:cc",
      "port": 445
    },
    "tags": {
      "Insecure": "important",
      "SMB": "info"
    }
  }
]
```