Programmable RF Switch

50Ω SPDT, DC to 6000 MHz

POE-SWITCH-6G

HIGHLIGHTS

- SPDT (Single Pole Double Throw) topology
- Broadband DC to 6000 MHz operation
- Power and control via any combination of USB or PoE-enabled Ethernet.
- Driverless: WebUI, REST, SCPI, HISLIP services all hosted on switch
- Compact size, 3.150 x 1.673 x 0.906”

APPLICATIONS

- RF signal routing and switching
- Antenna switching
- Redundancy and failover systems
- Automated test systems
- Laboratory instrumentation

Parameter	Value
Model No.	POE-SWITCH-6G
Case Style	1455D801
Connectors	SMA (female)

PRODUCT OVERVIEW

The POE-SWITCH-6G is a general-purpose, SPDT (Single Pole Double Throw) programmable RF switch designed for broadband signal routing applications from DC to 6000 MHz. It provides low insertion loss and high isolation between switched paths.

The unit is housed in a compact, rugged enclosure with SMA female connectors on all three RF ports (Common, RF1, RF2), a Power over Ethernet (IEEE 802.3af) port via RJ45, and a USB Type-C port for power and control.

An onboard web interface (WebUI) and REST API are accessible on port 80. A raw socket SCPI server runs on port 5025 and a HiSLIP-SCPI server on port 4880.

KEY FEATURES

Feature	Advantages
SPDT RF switching	Route RF signals between Common and RF1 or RF2 with low insertion loss and high isolation across a broad frequency range.
No software or driver installation required	All control software is hosted directly on the device. A web browser is all that is needed for manual control — no drivers, no installs.
Auto-detects in Keysight Connection Expert	Device uses mDNS and HiSLIP to appear during automatic discovery in commonly used LXI management tools
USB and Ethernet control	USB and Ethernet interfaces with REST, SCPI, and HiSLIP support provide broad compatibility with a wide range of programming environments and test automation frameworks.

ELECTRICAL SPECIFICATIONS

Parameter	Conditions	Freq (MHz)	Min.	Typ.	Max.	Unit
Insertion loss	RF1 or RF2 on	DC - 2000	-	1.5	2.5	dB
		2000 - 4000	-	2.5	5.0	
		4000 - 6000	-	5.0	8.0	
Isolation	RF1 or RF2 off	DC - 6000	40	50	-	dB
Return loss	Common port	DC - 2000	7	14	-	dB
Return loss	RF ports	DC - 2000	14	20	-	dB
Return loss	RF ports	2000 - 6000	5	10	-	dB
Switching speed		-	-	15	25	ms
Supply voltage (Vpoe)	Ethernet port (PoE)	-	36	48	57	V_{DC}
Supply current (Ipoe)	Ethernet port (PoE)	-	-	25	35	mA
Supply voltage (Vusb)	USB port	-	4.75	5	5.25	V_{DC}
Supply current (Iusb)	USB port	-	-	-	150	mA

ABSOLUTE MAXIMUM RATINGS

Parameter	Value
Operating temperature	0°C to 50°C
Storage temperature	-20°C to 85°C
RF input power	+30 dBm
V_{USB} MAX	6 V
V_{POE} MAX	57 V

Permanent damage may occur if any of these limits are exceeded.

CONNECTIONS

Port Name	Connector Type
Common	SMA female
RF1	SMA female
RF2	SMA female
USB	USB type C female
Network (Power Over Ethernet, Ethernet)	RJ45 socket

TYPICAL PERFORMANCE CURVES

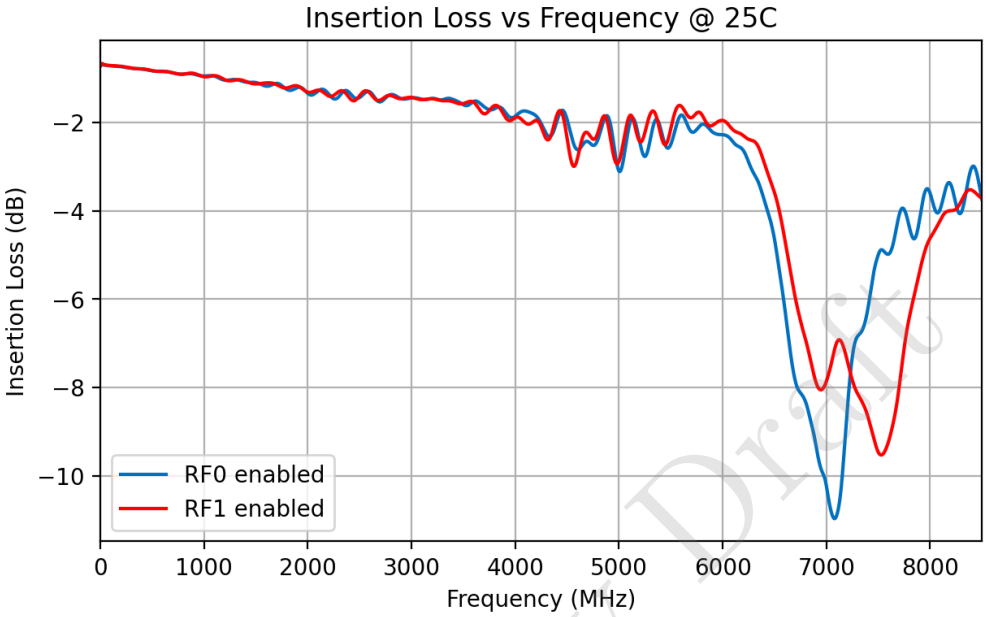

Figure 1: Insertion Loss vs. Frequency

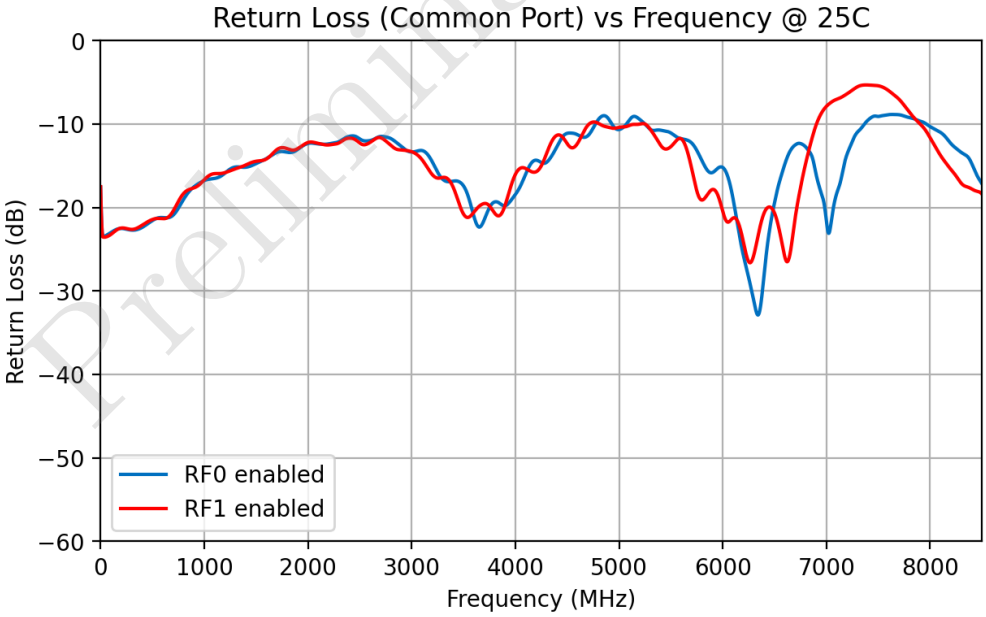

Figure 2: Return Loss (Common Port) vs. Frequency

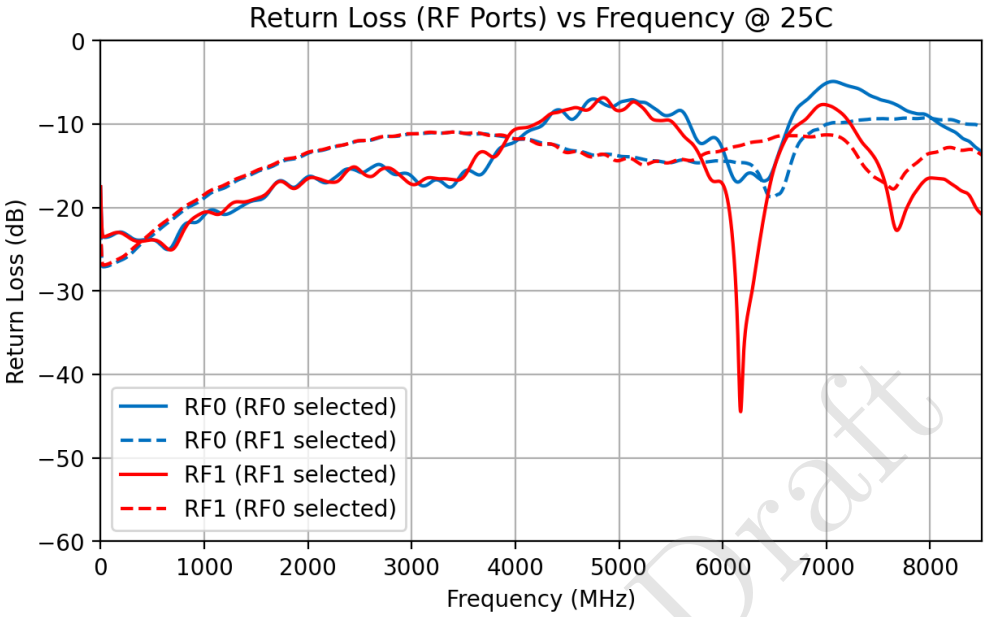

Figure 3: Return Loss (RF Ports) vs. Frequency

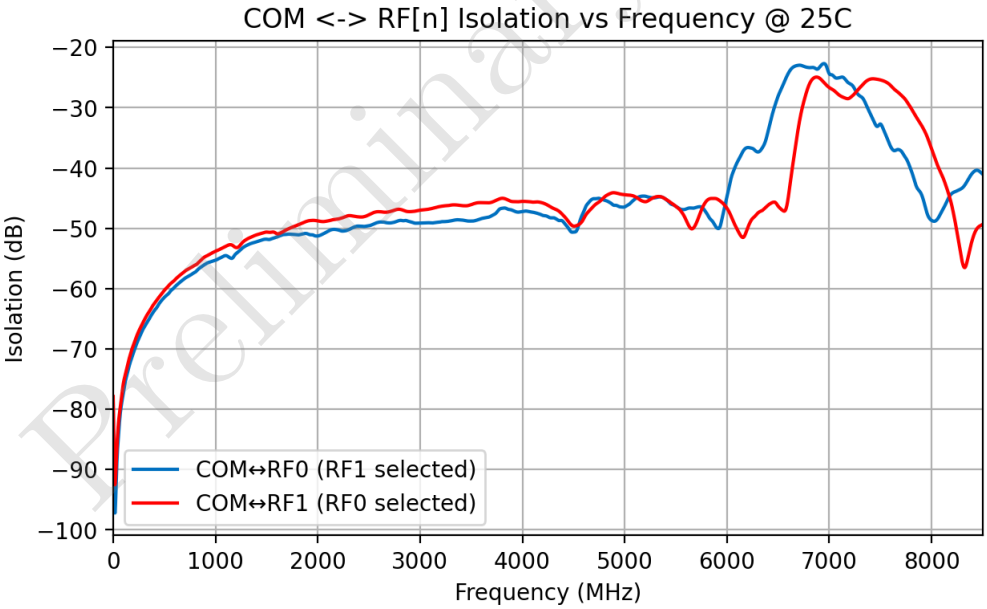

Figure 4: Isolation vs. Frequency

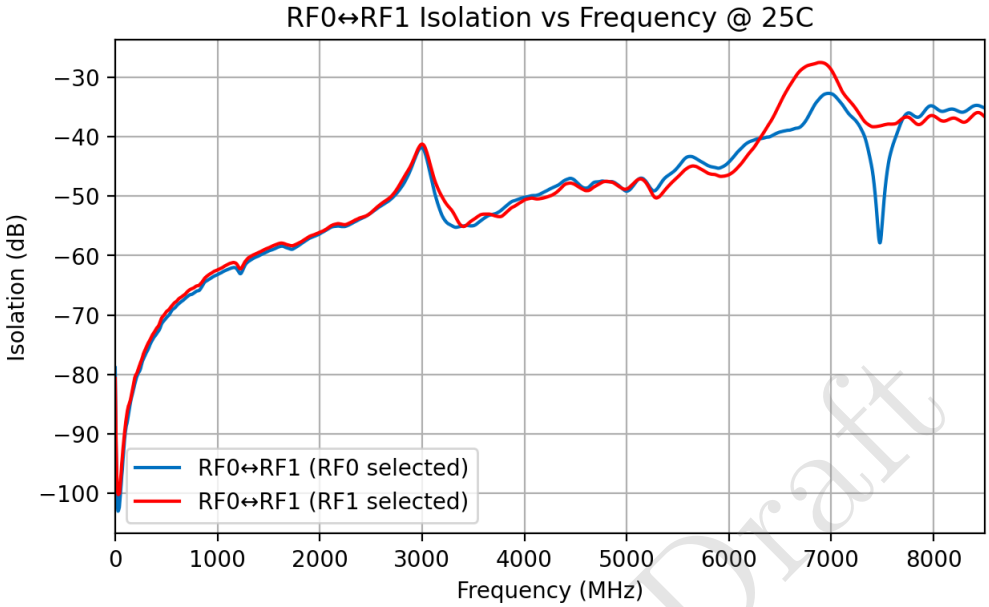

Figure 5: RF0→RF1 Isolation vs. Frequency

DEVICE DISCOVERY VIA MDNS

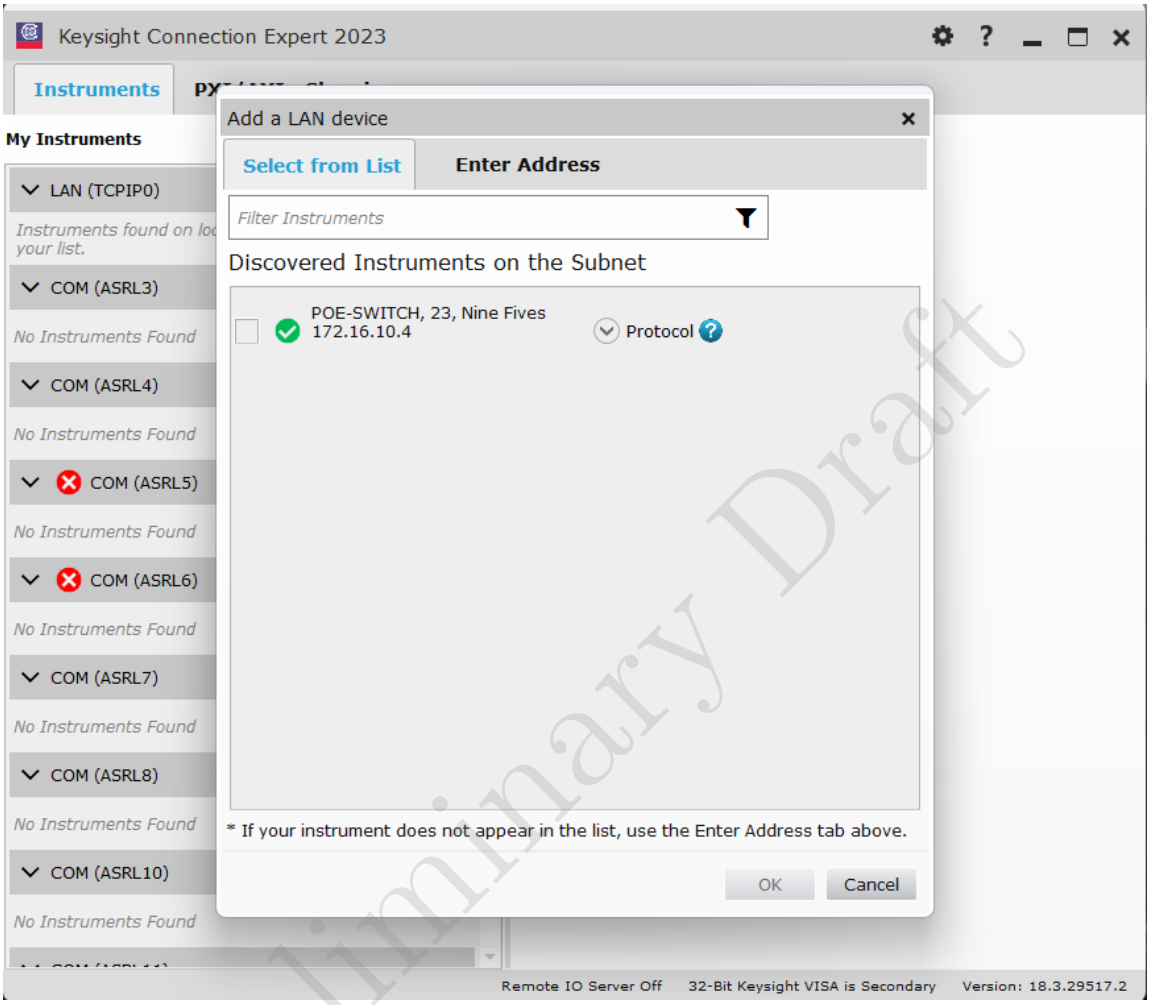

Figure 6: Device Discovery via mDNS

HTTP WEB INTERFACE (WEBUI)

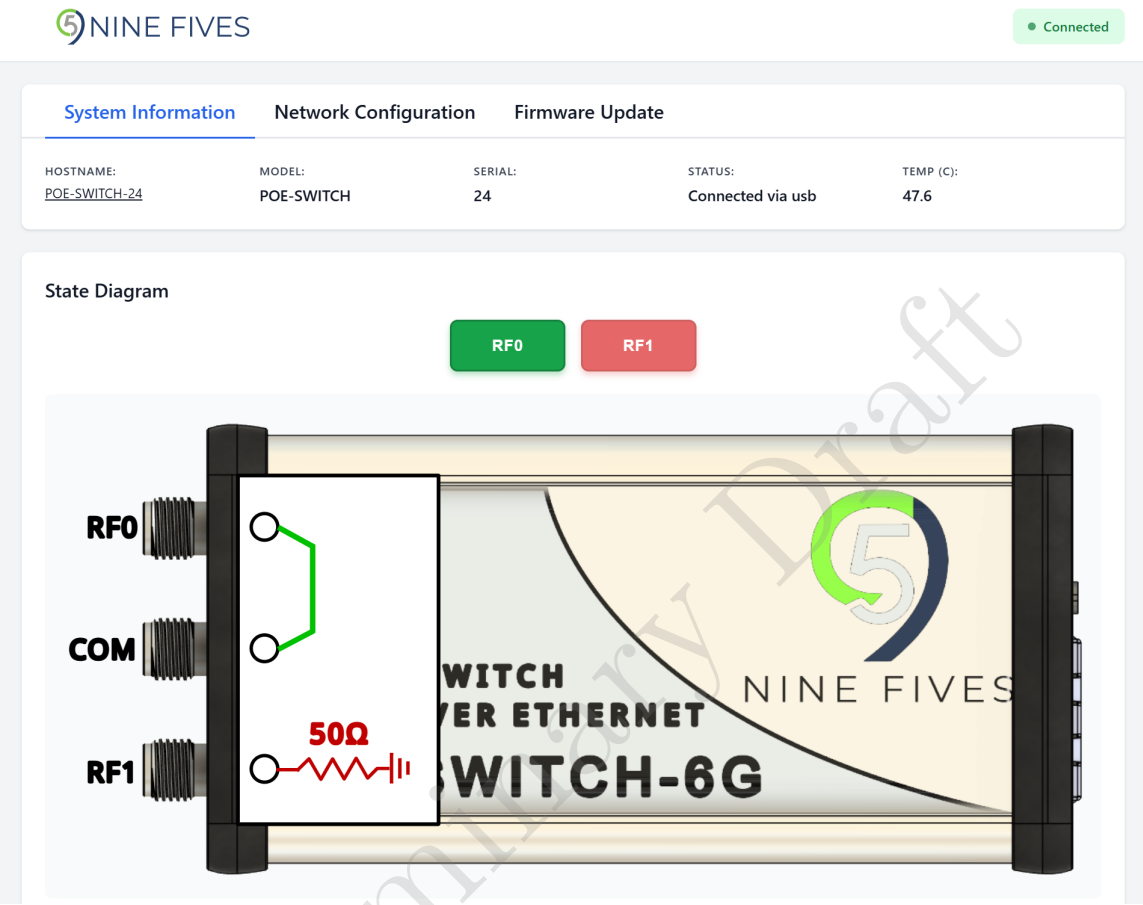

Figure 7: Web Interface (WebUI)

SCPI COMMANDING

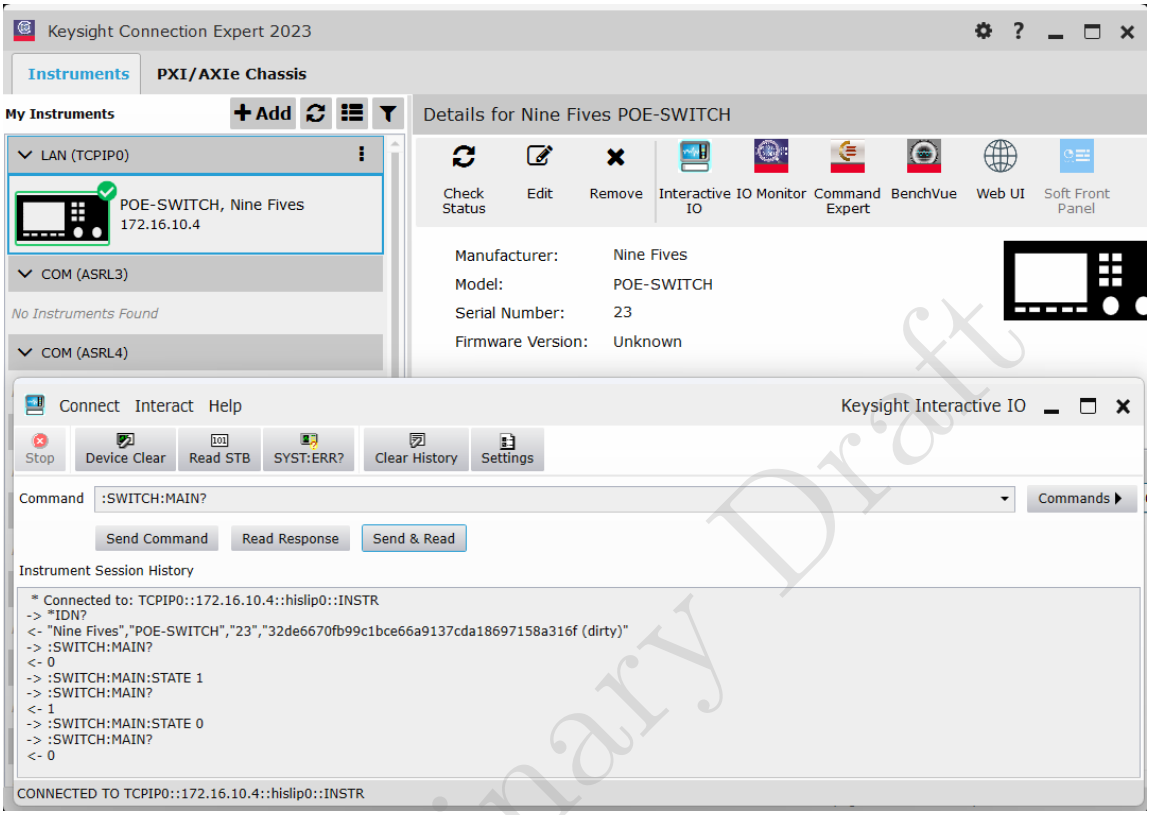

Figure 8: SCPI Commanding

COMPATIBLE WITH PYTHON

SCPI via PyVISA

```
>>> import pyvisa; rm = pyvisa.ResourceManager()
>>> rm.list_resources()
('TCPIP::172.16.10.160::hislip0,4880::INSTR',)
>>> sw = rm.open_resource(rm.list_resources()[0])
>>> sw.query('*IDN?')
'Nine Fives,Switch_Controller,0001,1.0.0\n'
>>> sw.query('SWITCH:MAIN:STATE?')
'0\n'
>>> sw.write('SWITCH:MAIN:STATE 1')
>>> sw.query('SWITCH:MAIN:STATE?')
'1\n'
```

REST API via Requests

```
>>> import requests
>>> requests.get('http://172.16.10.160/api/system/status').
json()
{'device': 'Nine Fives Switch Controller',
 'hostname': 'POE-SWITCH-23',
 'model': 'POE-SWITCH',
 'serial': '23',
 'status': 'Connected via eth',
 'temperature': 42.5,
 'connectedIface': 'eth0'}
>>> requests.get('http://172.16.10.160/api/switch').json()
{'state': 0}
>>> requests.post('http://172.16.10.160/api/switch',
...     json={'state': 1})
<Response [200]>
>>> requests.get('http://172.16.10.160/api/switch').json()
{'state': 1}
```