The bridge to possible

Manage and automate your DC network with RADKit

Wojciech Kozioł, Technical Consulting Engineer, DCRS TAC

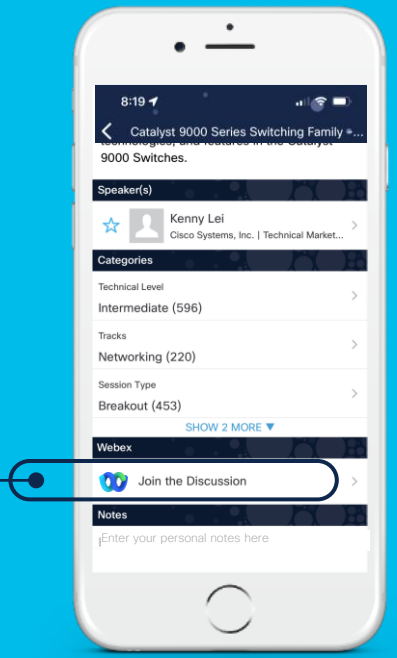
Cisco Webex App

Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space



Webex spaces will be moderated until February 24, 2023.

Agenda

- What is RADKit?
- How can I start using it?
- RADKit components overview
- Scripting capabilities

Why are we here?

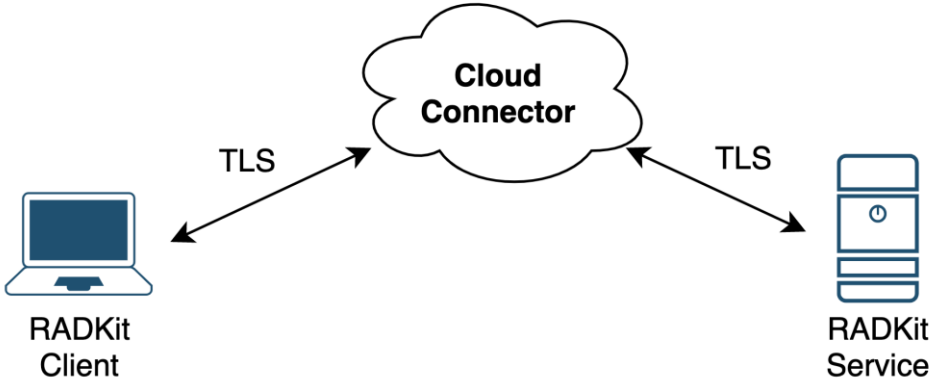
Automation is for everyone, not only for programmers.

What is RADKit?

What is RADKit?

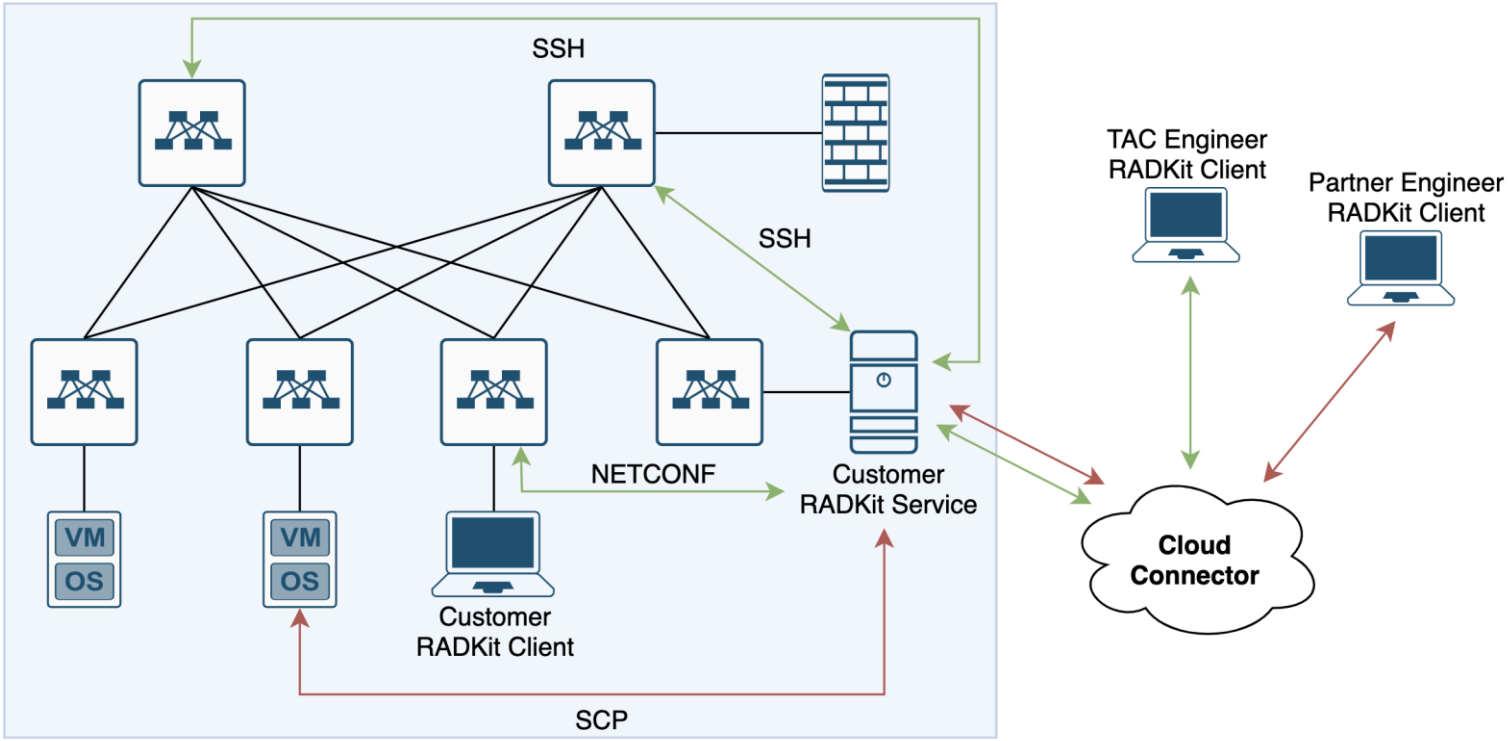
- Remote Automation Development Kit (SDK)
  - Set of ready-to-use Python modules and tools
- Allowing your network devices and services to be treated as objects
  - Scalable interactions with local or remote equipment
  - Improving monitoring/analysis of collected data
- Enhance NetOps activities
  - Compressing hours of manual approach to a matter of minutes
  - Allowing connections and operations on multiple different devices with use of one tool

RADKit components



RADKit Service	Secure gateway to the customer's network devices. Supports multiple management protocols
RADKit Client	Python-based application that enables programable way of troubleshooting and analysis via Service
RADKit Cloud	Connector between Service and local/remote Clients

RADKit architecture



RADKit capabilities

SSH/Telnet

SCP/SFTP

Uploading files to CX Drive

RADKit

Swagger

Parsing outputs with Genie

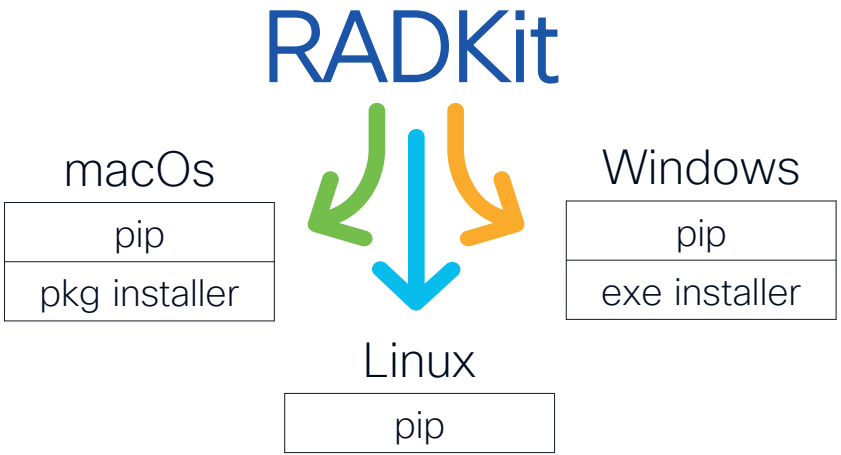
Port forwarding SOCKS5 proxy

NETCONF

Can be deployed without a need to communicate with Cloud

Installation:

It is possible to run **Service** in **Docker** container.



Download page:
https://radkit.cisco.com/downloads/release/

For python installations we strongly recommend setting up a virtual environment using Miniconda, venv or virtualenv.

RADKit Service

RADKit Service

Web UI

Logging service enables way of tracking changes

Bulk modifications

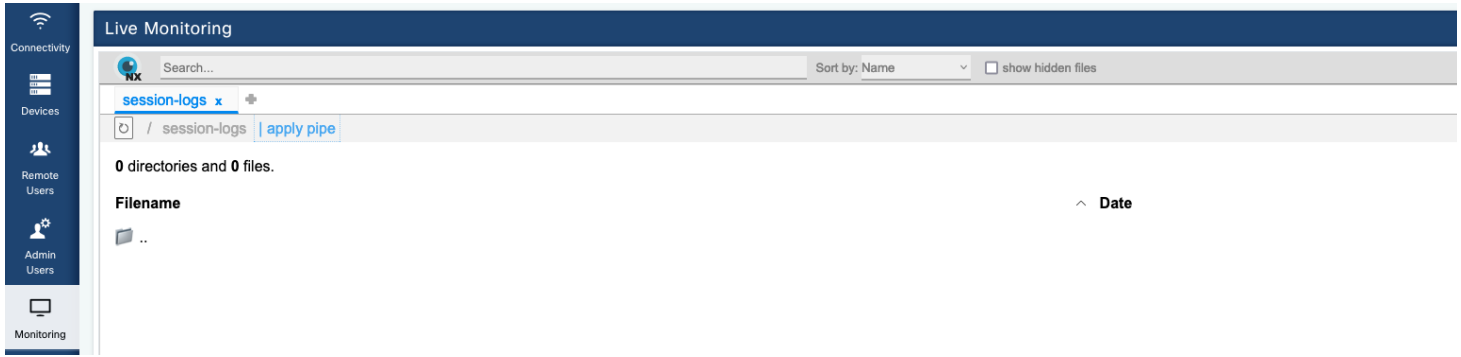
Enables control of who can access devices and how long

Full control of which protocol can be used and on which devices

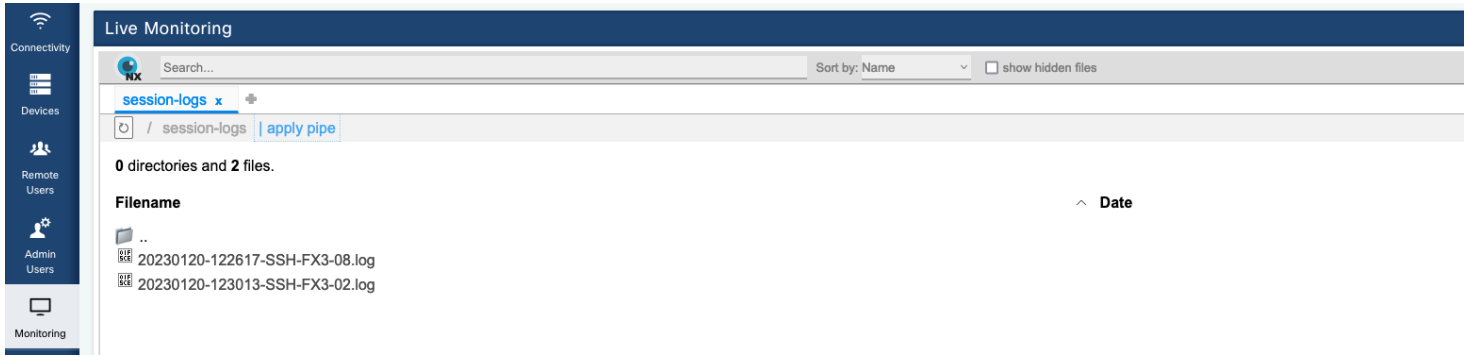
Tracking all of the activities

Client application logs	.radkit/logs/client/client.*
Client CLI history	.radkit/client/.history
Service application logs	.radkit/logs/service/service.*
Service session logs	.radkit/session_logs/service/

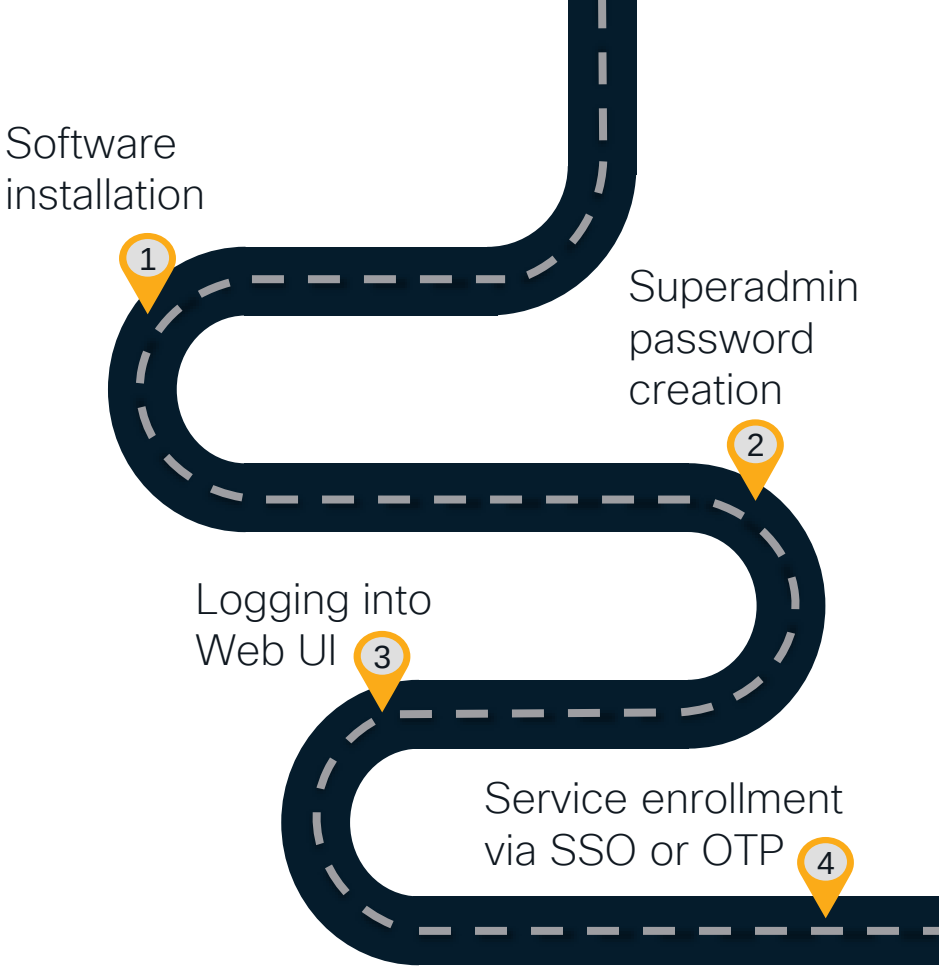
Logging services – demo slide




```
>>>
>>> var1 = service.inventory['fx3-02'].exec("show hostname").wait()
>>> var2 = service.inventory['fx3-08'].exec("show hostname").wait()
>>>
```



How to deploy RADKit Service?



Service deployment (with use of OTP)

Setting superadmin password

```
wkoziol$ radkit-service bootstrap
20:40:54:874Z | internal | RADKit Service [version='1.3.2']
Set up the superadmin user. You'll be asked to provide a superadmin password
Keep this password securely stored, as it will be impossible to recover it!
Superadmin password (for new setup): **********
Confirm: **********
20:41:17:386Z INFO | internal | Migrating DB [old_version='' new_version='0bc4b4bc847a']
20:41:17:521Z INFO | internal | Database has been upgraded
20:41:17:607Z INFO | internal | Creating local principal [username='superadmin']
```

Starting from 1.4.0 version it is recommended to use SSO instead of OTP to enroll Service. More info here: https://radkit.cisco.com/docs/pages/setup_service.html

OTP and service serial number

```
>>> client.grant_service_otp("wkoziol@cisco.com")
<radkit_client.client.ServiceEnrollInfo object {email='wkoziol@cisco.com', serial='wk62-e6z8-6c5o', otp='4691-6121-1773',
domain='PROD'} at 0x11405ed90>
------  --------------
email   wkoziol@cisco.com
serial  wk62-e6z8-6c5o
otp     4691-6121-1773
domain  PROD
```

Hidden slide

Service enrollment

```
wkoziol$ radkit-service --domain PROD enroll --email wkoziol@cisco.com --serial wk62-e6z8-6c5o --otp 4691-6121-177
```

Demo 1 – Service Overview

RADKit Client

RADKit Client

network-console provides simplified CLI for basic Client operations like interactive sessions, download, upload

SSO authentication

Python console

radkit-interactive provides ability to open SSH/Telnet session to the device just like any traditional terminal client

How to deploy RADKit Client?

Client deployment

Hidden slide

SSO Authentication

```
>> client = sso_login("wkoziol@cisco.com")
```

Connect to service

```
>> service = client.service("wk62-e6z8-6c5o")
15:40:24.793Z INFO  | internal | Connecting to forwarder [uri='wss://prod.radkit-cloud.cisco.com/forwarder-3/websocket/']
15:40:25.974Z INFO  | internal | Connection to forwarder successful [uri='wss://prod.radkit-cloud.cisco.com/forwarder-3/websocket/']
```

Inventory

```
>>> service.inventory
<radkit_client.device.DeviceDict object at 0x114e6bd60>

name          host          device_type  Terminal  Netconf  Swagger  HTTP  description  failed
-------------  ----------    -----------  --------  --------  --------  -----  ----------  -----
fx3-07        10.62.154.160  IOS          True      True     False    True                False
fx3-08        10.62.154.161  IOS          True      False    False    False               False
fx3-09        10.62.154.162  IOS          True      False    False    False               False
fx3-10        10.62.154.163  IOS          True      False    False    False               False
fx3-11        10.62.154.164  IOS          True      False    False    False               False
```

Demo 2 – Client

Basic operations within Client

Content:

- Command execution
- Terminal session to selected device
- SCP

Demo in PDF slides

Client overview – demo

```
>>> dev_1 = service.inventory.filter("name","fx3-02")
>>> dev_1.add(service.inventory['fx3-08'])
>>> dev_1
<radkit_client.device.DeviceDict object at 0x115151e50>
```

Can we use regex? ✔

```
reg_dev = service.inventory.filter("name","(fx3-).+")
```

```
name        host          device_type    Terminal   Netconf   Swagger   HTTP   description   failed
------ ------------- ------------- ---------- --------- --------- ------ ------------- --------
fx3-02  10.62.154.155  UNKNOWN        True       False     False     False                False
fx3-08  10.62.154.161  IOS            True       False     False     False                False
```

```
>>> out_1 = dev_1.exec("show clock").wait()
>>> print(out_1.result['fx3-02'].data)
FX3-02# show clock
Warning: No NTP peer/server configured. Time may be out of sync.
22:27:25.673 UTC Thu Jan 19 2023
Time source is NTP
FX3-02#
```

How to print all of the outputs?

```
>>> for dev in dev_1:
        print(out_1.result[dev].data)
```

Hidden slide

More information:
https://radkit.cisco.com/docs/pages/feature_exec.html#multiple-devices-single-command

Client overview – demo

```
>>> scp_box = service.inventory['fx3-02']
>>> scp = scp_box.scp_upload_from_file(remote_path="/", local_path="/Users/wkoziol/Downloads/file.txt")
>>> scp.show_progress()
0 kbps 100.0%
[==========================================================================================
==========================================================================================
==========================================================>]   9/  9 eta [00:00]
>>> scp
[TRANSFER_DONE] <radkit_client.terminal.connection.FileWriteConnection object at 0x11373b5b0>
----------------  ---------------------------------
serial            ns52-hkc8-jusv
device_name       fx3-02
chmod             644
size              9
remote_path       /file.txt
local_path        /Users/wkoziol/Downloads/file.txt
bytes_written     9
bytes_read        0
connection_result  connection succeeded
transfer_result    transfer completed
----------------  ---------------------------------
```

Hidden slide

Client overview – demo

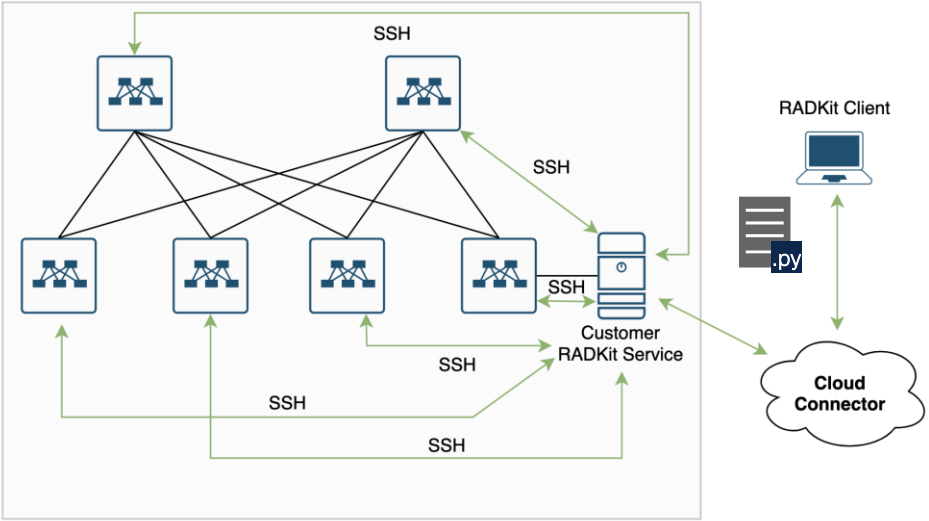
```
>>> service.inventory['fx3-02'].interactive()
22:59:27.895Z INFO  | internal | starting interactive session (will be closed when detached)

  Attaching to fx3-02 ...
    Type: ~. to detach.
          ~? for other shortcuts.
  When using nested SSH sessions, add an extra ~ per level of nesting.


Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Copyright (C) 2002-2021, Cisco and/or its affiliates.
All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under their own
licenses, such as open source.  This software is provided "as is," and unless
otherwise stated, there is no warranty, express or implied, including but not
limited to warranties of merchantability and fitness for a particular purpose.
Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or
GNU General Public License (GPL) version 3.0  or the GNU
Lesser General Public License (LGPL) Version 2.1 or
Lesser General Public License (LGPL) Version 2.0.
A copy of each such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://opensource.org/licenses/gpl-3.0.html and
http://www.opensource.org/licenses/lgpl-2.1.php and
http://www.gnu.org/licenses/old-licenses/library.txt.
FX3-02# show switchname
FX3-02
FX3-02# ~detached
22:59:42.736Z INFO  | internal | closing interactive session
>>>
```

Hidden slide

Demo 3 - Script

Scripting

Collecting and operating on data from multiple points of network



Demo task

- Collect "show system internal flash" from all devices in service
- Parse the output to JSON
- Find those devices where "/var/volatile/tmp" is over 50% utilization
- Print devices which are meeting above condition

Example syntax to run the script:

```
python <script name> --service "<serial number>" <additional arguments>
```

JSON output collected from Nexus switch

```
switch# show system internal flash | json-pretty
{
    "TABLE_flash": {
        "ROW_flash": [
            {
                "Mounted-on": "/var/volatile/tmp",
                "OneK-blocks": "614400",
                "used": "340",
                "Available": "614060",
                "Use-percent": "1",
                "Filesystem": "none"
            }
        ]
    }
}
```

This is the path we want to check

Amount of space utilized by tmp folder

Code review

Result of switch command executed from Client via exec():

```
>>> example_output = service.inventory['fx3-07'].exec("show clock | json | no-more").wait()

>>> print(example_output.result.data)
FX3-07# show clock | json | no-more
{"simple_time": "23:47:08.936 UTC Wed Jan 18 2023", "time_source": "NTP"}
FX3-07#
```

Only blue part of output is important for us. First and last lines should be removed in order to be able to translate it into python dictionary.

Code review

```
from radkit_client import Device, run_on_device_dict
from radkit_common import nglog
import re
import json

IC = nglog.LazyTag("NXOS Memory Checker", desc="Tag for NXOS Memory Checker")
nglog.basicConfig()

def get_commands(device : Device , * , path: str = "/bootflash", border_value: str = "5") -> None:
        device = device.filter("name","(fx3-).+")
        parsed_cmd = json_parser(["show system internal flash"],device)
        if len(path) > 0:
                space_check(device, parsed_cmd, path, border_value)

if __name__ == "__main__":
        run_on_device_dict(get_commands)
```

Code review

```
from radkit_client import Device, run_on_device_dict
from radkit_common import nglog
import re
import json

IC = nglog.LazyTag("NXOS Memory Checker", desc="Tag for NXOS Memory Checker")
nglog.basicConfig()

def get_commands(device : Device , * , path: str = "/bootflash", border_value: str = "5") -> None:
    parsed_cmd = json_parser(["show system internal flash"],device)
    if len(path) > 0:
        space_check(device, parsed_cmd, path, border_value)

if __name__ == "__main__":
    run_on_device_dict(get_commands)
```

Code review

```
def json_parser(commands,devices):

    pipes = " | json | no-more"
    cmds = [s + pipes for s in commands]

    request = devices.exec(cmds).wait()
    parsed_cmd_tmp = {}
    for dev in request.result:
        parsed_cmd_tmp[dev] = {}
        for cmd in request.result[dev]:
            if json_decoder(request.result[dev][cmd].data) is not None:
                tmp_json = json_decoder(request.result[dev][cmd].data)
                cmd = cmd.replace(" | json | no-more", "")
                parsed_cmd_tmp[dev][cmd] = {}
                parsed_cmd_tmp[dev][cmd] = tmp_json
            else:
                nglog.info("This command `" + cmd + "` has encountered error on device " + dev)
                cmd = cmd.replace(" | json | no-more", "")
                parsed_cmd_tmp[dev][cmd] = {}
                parsed_cmd_tmp[dev][cmd] = None
    return parsed_cmd_tmp
```

We execute ”show system internal flash” on all filtered devices from RADKit Service.

Since outputs from devices are in string format we need to translate them into dictionary.

```
def json_decoder(output):
    output = output[output.find('{'):]
    output = re.sub('([^}]*)$','',output)
    test_output = output.replace('\n','')
    try:
        output_json = json.loads(test_output)
    except ValueError:
        output_json = None
return output_json
```

Code review

```
from radkit_client import Device, run_on_device_dict
from radkit_common import nglog
import re
import json

lC = nglog.LazyTag("NXOS Memory Checker", desc="Tag for NXOS Memory Checker")
nglog.basicConfig()

def get_commands(device : Device , * , path: str = "/bootflash", border_value: str = "5") -> None:
        parsed_cmd = json_parser(["show system internal flash"],device)
        if len(path) > 0:
                space_check(device, parsed_cmd, path, border_value)

if __name__ == "__main__":
        run_on_device_dict(get_commands)
```

Code review

```
def space_check(device, parsed_cmd, path, border_value):
        devices = [item[0] for item in device.items()]

        nglog.info("Devices where checks are done: " + str(devices))

        for dev in devices:
            nglog.info("Device: " + dev)
            for item in parsed_cmd[dev]["show system internal flash"]["TABLE_flash"]["ROW_flash"]:
                if item["Mounted-on"] == path:
                    used_space = int(item["Use-percent"])
                    if used_space >= int(border_value):
                        nglog.info(item["Mounted-on"] + " | Used space: " + item["Use-percent"] + "%")
                        nglog.info("\n")
                    else:
                        nglog.info("Device is fine.")
                        nglog.info("\n")
```

https://github.com/loizok/RADKit_CL_AMS_23

What if my device does not return JSON?

Genie parser

RADKit - Genie

```
>> device = service.inventory['fx3-08']
>> output = device.exec('show version').wait()
>> dev_type = radkit_genie.fingerprint(device)
>> print(dev_type)
```

select device

execute command

Genie can recognize device type

```
{'fx3-08' : {'os' : 'nxos'}}
```

```
>> json_output = radkit_genie.parse(output,os = dev_type[device.name][ 'os' ])
>> print(json_output)
```

parse to JSON


```
{'fx3-08': {'show version': {'platform': {'name': 'Nexus', 'os': 'NX-OS', 'software': {'bios_version':
'01.08', 'system_version': '10.2(2) [Feature Release]', 'bios_compile_time': '05/06/2022',
'system_image_file': 'bootflash:///nxos64-cs.10.2.2.F.bin', 'system_compile_time': '12/14/2021
23:00:00 [12/15/2021 11:59:34]'}, 'hardware': {'model': 'Nexus9000 C93180YC-FX3', 'chassis':
'Nexus9000 C93180YC-FX3', 'slots': 'None', 'rp': 'None', 'cpu': 'Intel(R) Xeon(R) CPU D-1526 @
1.80GHz', 'memory': '32823016 kB', 'processor_board_id': 'FDO25250U2X', 'device_name': 'FX3-8',
'bootflash': '115343360 kB'}, 'kernel_uptime': {'days': 15, 'hours': 6, 'minutes': 14, 'seconds': 4},
'reason': 'Reset Requested by CLI command reload', 'system_version': '10.2(2)'}, '_exclude':
['seconds', 'minutes', 'hours', 'days', 'essor_board_id', 'reason']}}}
```

Do I need to add 100 of my devices manually to the service?

Adding multiple devices to service

```
from radkit_client import Device, run_on_device_dict, helpers, swagger

def get_commands(device: Device) -> None:
    r = open("Devices.csv").readlines()
    devs = []
    for l in r:
        l = l.strip()
        if l.startswith("#") == False:
            s = l.split(";")
            d = {
                "name":s[1],
                "host":s[2],
                "deviceType":s[3],
                "terminal":{"port":22,"connectionMethod":"SSH","username":s[4],"enableSet":False,"useInsecureAlgorithms":True,"enable":s[5],"password":s[5]}
            }
            devs.append(d)
device["radkit-service"].update_swagger().wait()

for d in devs:
        print(device["radkit-service"].swagger.paths['/devices'].post(json_data=d).wait().result.data)

if __name__ == "__main__":
    run_on_device_dict(get_commands)
```

Read CSV file.

Service swagger capabilities update.

Adding devices to service based on "devs" list content via POST call.

Script ideas:

1. Read CSV file and parse its content into list of python dictionaries.
2. Update RADKit Service Swagger capabilities.
3. Use POST operation to push device addition into service.

`Devices.csv` file format

	Name	IP	Type	User	Pass
1	fx3-02	1.1.1.1	IOS	admin	pass1

Hidden slide

Conclusions

Enables programatic way of managing your network.

Performs multiple operations at same time.

Contains multiple ready-to-use tools and modules.

Provides remote access to your devices.

https://radkit.cisco.com/

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

The bridge to possible

Thank you

cisco Live!

CISCO Live!

ALL IN