cisco Live!

GO BEYOND

#CiscoLiveAPJC

Automating Tenant Management using the Cisco Security Cloud Control APIs

A hands-free guide

Siddhu Warrier, Principal Engineer

DEVNET-1281

Cisco Webex App

https://ciscolive.ciscoevents.com/ciscolivebot/#DEVNET-1281

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 by the speaker until November 15, 2024.

Siddhu Warrier

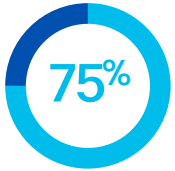
- Principal Engineer, Cisco Security
- Will bend your ear about:
  - APIs
  - Infrastructure as Code
  - CI/CD and automation
- Love
  - Cricket (IN and 🏴... sorry, AU)
  - Arsenal FC (this will be our year)

What is Security Cloud Control?

Cisco Security Cloud Control

Formerly Cisco Defense Orchestrator

Why the change?



% of security customers pursuing vendor consolidation

What is it?

Centralised security management for Cisco Firewall (ASA and FTD), Multicloud Defense, and Hypershield

- Enables real-time visibility into network traffic and security events, allowing security teams to monitor and respond to incidents promptly
- Supports security policy administration, auto-tuning, recommendations, and troubleshooting across diverse environments
- Is cloud-delivered, reducing total cost of ownership, simplifying scalability, and accelerating feature delivery

Where is it going?

Single, AI-native management experience for all Cisco security solutions

- Hosts individual products as microapps within platform
- Offers unified provisioning and access management (including RBAC, unified tenancy, and regional selection)
- All SBG products mapped to new left nav structure; cross-product navigation available via 9-dot menu
- Centralized Global Search, AI Assistant, Help and Documentation, Dashboard, Onboarding, and more across products

And we're available down under!

https://aus.manage.security.cisco.com/new

The Problem

From zero to customer success...

What do we need?

- Kick it!
  - Get a new customer set up with as few clicks as possible.
- Provision it!
  - Stand up the customer's security infrastructure, and configure it, with as little muss as possible.
- Monitor it!
  - Hit your Customer SLAs with proactive monitoring that plugs into your toolkit!
- Upgrade it!
  - New CVE? No problem!

The Solution?
The Cisco Security Cloud Control APIs

RESTful

```
curl --silent \
"https://edge.us.cdo.cisco.com/api/rest/v1/inventory/devices?q=name:*Hollis*" \
--header "Authorization: Bearer $PROD_TOKEN" \
| jq
```

Fully Documented

cisco DevNet | Documentation | Learn | Technologies | Community | Events | SIGN UP FREE | LOG IN

Documentation > All > Cisco Defense Orchestrator (CDO) > CDO API Documentation

Cisco Security Cloud Control API Documentation

Cisco Security Cloud Control (SCC) exposes a rich REST API. This document describes the semantics of the REST API. Backwards compatibility of the API is guaranteed using a versioning system.

What can you do with it?

- Manage your devices, services, and device managers
- Deploy changes to devices at scale
- Manage your objects
- Perform complex searches across SCC
- Execute queries across SCC and cloud-delivered Firewall Management Center (cdFMC)
- Monitor Remote Access Virtual Private Network (RA VPN) sessions and Multi-factor Authentication (MFA) events
- Monitor your changelog
- Execute commands across your entire fleet of devices
- Build your own dashboard as a Managed Service Provider

https://developer.cisco.com/docs/cisco-security-cloud-control

Developer Ready

Search projects

Help Sponsors Log in Register

cdo-sdk-python 1.0.179

```
pip install cdo-sdk-python
```

Latest version

Released: 2 minutes ago

CDO API

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Maintainers

cdo.ops

talhazi114

Project description

Cisco Defense Orchestrator (CDO) Python SDK

The CDO Python SDK facilitates automated interaction with the Cisco Defense Orchestrator (CDO) API, enabling developers to integrate Cisco's cloud-based security policy and device management into Python applications.

For more information and detailed documentation, visit the CDO Python SDK documentation.

Read more about CDO here.

Installation

Install the CDO Python SDK package with the following command:

```
pip install cdo-sdk-python
```

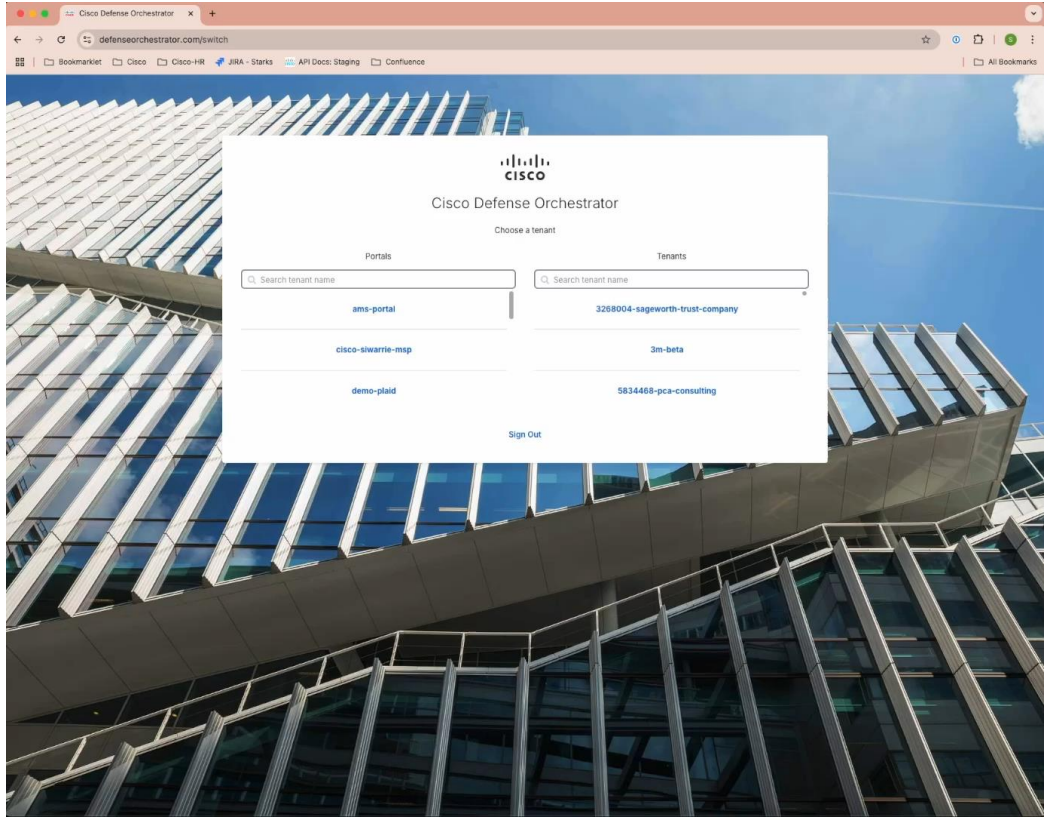
https://pypi.org/project/cdo-sdk-python

Getting Started

Step 1: Get an MSSP Portal

Contact Cisco TAC to get an MSSP Portal

Step 2: Generate an API Token



Step 3: Download the Postman Collection

- Download the Security Cloud Control Postman Collection and environment from https://devnetapps.cisco.com/docs/cisco-defense-orchestrator/postman-collection/
- Import your data into Postman: https://learning.postman.com/docs/getting-started/importing-and-exporting/importing-data/

Step 4: Set up your Python project

```
pip install cdo-sdk-python>=1.0.0,<2.0.0
```

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Create a Customer Tenant

Creating a Tenant: The API

- **POST /api/rest/v1/msp/tenants/create**
- Body:

```
{
  “tenantName”: “tenant-name”,
  “displayName”: “display—name”
}
```

- Authorization: Bearer <Token>
- Asynchronous Operation that returns a CdoTransaction object
- Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/create-security-cloud-control-tenant/

Creating a Tenant: Python Code

```python
with ApiClient(Configuration(host=base_url, access_token=api_token)) as api_client:
  msp_api: MSPApi = MSPApi(api_client)
  transaction: CdoTransaction = msp_api.create_tenant(
      MspCreateTenantInput(
          **{"tenant_name": tenant_name, "display_name": display_name}
      )
  )
```

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Waiting for Transactions: The API

- **GET /api/rest/v1/transactions/<uid>**
- Authorization: Bearer <Token>
- A CdoTransaction object with a field called cdoTransactionStatus
  - Poll until cdoTransactionStatus is DONE or ERROR
- Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/get-transaction/

Waiting for Transactions: Python Code

```python
transaction: CdoTransaction = self.transactions_api.get_transaction(
    transaction_uid
)
while transaction.cdo_transaction_status not in [
    CdoTransactionStatus.DONE,
    CdoTransactionStatus.ERROR,
]:
    time.sleep(5)
    transaction = self.transactions_api.get_transaction(transaction_uid)

if transaction.cdo_transaction_status == CdoTransactionStatus.ERROR:
    raise RuntimeError(
        f"Transaction {transaction_uid} failed: {transaction.transaction_details}"
    )
```

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Add users to MSP-managed Tenant

Add Users to MSP-managed Tenant: The API

- **POST /api/rest/v1/msp/tenants/{tenantUid}/users**
- Body:

```
{
  “tenantUid”: “<uuid-of-tenant>”,
  “users”: [
    {
      “username”: “user1@msp.com”,
      ”role”: “ROLE_ADMIN”,
      “apiOnlyUser”: false
    },…
    {
      “username”: “user2@customer.com”,
      ”role”: “ROLE_READ_ONLY”,
      “apiOnlyUser”: false
    }
  ]
}
```

- Asynchronous Operation that returns a CdoTransaction object
- Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/add-users-to-security-cloud-control-tenant-in-msp-portal/

Add Users to MSP-managed Tenant: Python Code

```python
with ApiClient(Configuration(host=base_url, access_token=api_token)) as api_client:
    msp_api = MSPApi(api_client)
    transaction: CdoTransaction = msp_api.add_users_to_tenant_in_msp_portal(
        msp_managed_tenant.uid, MspAddUsersToTenantInput(**{"users": users})
    )
```

And then wait for the transaction to finish, exactly as before!

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

What about federated IdPs?

- **POST /api/rest/v1/msp/tenants/{tenantUid}/users/groups**
- Body:

```
{
  “tenantUid”: “<uuid-of-tenant>”,
  “users”: [
    {
      “username”: “man-crush-pineapple@cisco.com”,
      ”role”: “ROLE_ADMIN”,
      “apiOnlyUser”: false
    },…
    {
      “username”: “man-crush-pineapple@cisco.com”,
      ”role”: “ROLE_ADMIN”,
      “apiOnlyUser”: false
    }
  ]
}
```

- Asynchronous Operation that returns a CdoTransaction object
- Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/add-active-directory-groups-to-security-cloud-control-tenant-in-msp-portal/

Provision cdFMC in MSP-managed Tenant

Provision cdFMC: The API

- **POST /api/rest/v1/msp/tenants/{tenantUid}/cdfmc**
- Body: {}
- Asynchronous Operation that returns a CdoTransaction object
- Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/provision-cdfmc-for-security-cloud-control-tenant-in-msp-portal/

Provision cdFMC: Python Code

```
transaction: CdoTransaction = (
    self.msp_api.provision_cd_fmc_for_tenant_in_msp_portal(
        msp_managed_tenant.uid
    )
)
```

And then wait for the transaction to finish, exactly as before!

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Configure default policies

Every MSP-managed tenant

- Should have a default access policy
- An access rule that blocks access to Gambling
- We will do this using the (cd)FMC APIs

cdFMC API == Security Cloud Control API

See https://developer.cisco.com/docs/cisco-security-cloud-control/proxy-request-to-cloud-delivered-fmc/

Code Step 1: Find the FMC Domain UUID

```
manager_page: DevicePage = self.inventory_api.get_device_managers(
    limit="1", offset="0", q="deviceType:CDFMC"
)
if len(manager_page.items) != 1:
    raise RuntimeError("CDFMC not found")
self.cdfmc_domain_uid = manager_page.items[0].fmc_domain_uid
```

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Code Step 2: Create an Access Policy

```
policy = CdFmcAccessPolicy(name="MSP Access Policy", default_action="BLOCK")
url = (
    f"{self.api_client.configuration.host}/v1/cdfmc/api/fmc_config/v1/domain/"
    f"{self.cdfmc_domain_uid}/policy/accesspolicies"
)
headers = {
    "Authorization": f"Bearer {self.api_client.configuration.access_token}",
    "Content-Type": "application/json",
}
response = requests.post(url, headers=headers, json=policy.__dict__)
response.raise_for_status()
return response.json()["id"]
```

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Code Step 3: Let’s Block Gambling!

```python
url = (f"{self.api_client.configuration.host}/v1/cdfmc/api/fmc_config/v1/domain/"
       f"{self.cdfmc_domain_uid}/policy/accesspolicies/{access_policy_uid}/accessrules")
headers = {
    "Authorization": f"Bearer {self.api_client.configuration.access_token}",
    "Content-Type": "application/json",
}
access_rule = CdFmcAccessRule(
    name="Block Gambling",
    action="BLOCK",
    enabled=True,
    urls=Urls(
        url_categories_with_reputation=[
            UrlCategoryWithReputation(
                reputation="TRUSTED_AND_UNKNOWN",
                category=UrlCategory(
                    name="Gambling",
                    id=gambling_category_id,
                ),
            )
        ]
    ),
    source_networks=SourceNetworks(
        objects=[
            NetworkObject(
                type="NetworkGroup",
                overridable=False,
                id=any_ipv4_obj_id,
                name="any-ipv4",
            )
        ]
    ),
)

response = requests.post(url, headers=headers, json=access_rule.to_dict())
response.raise_for_status()
```

Onboard Secure Firewall

Onboard FTD (using ZTP): The API

- **POST /api/rest/v1/inventory/devices/ftds/ztp**
- Body:

```
{
  “name”: “device-1”,
  ”serialNumber”: “XXXXX”,
  “fmcAccessPolicyUid”: “xxx-xxx-xxx-xxx”,
  “licenses”: [“BASE”, “CARRIER”],
  “adminPassword”: “xxxxx”
}
```

- Asynchronous Operation that returns a `CdoTransaction` object
- Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/onboard-ftd-device-using-zero-touch-provisioning/

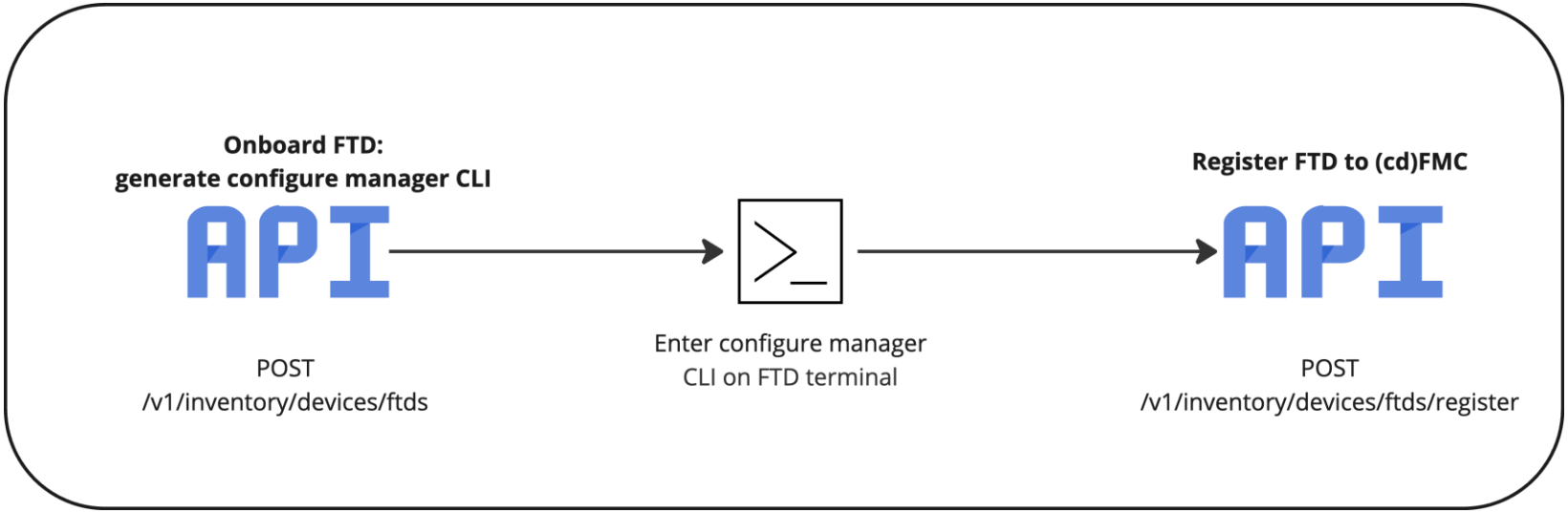
Onboard FTD using ZTP: Python Code

```python
ztp_onboarding_input = ZtpOnboardingInput(
    name="device-1",
    serial_number = "ABCDxxx1234",
    fmc_access_policy_uid = fmc_access_policy_uid,
    admin_password = "password"
)
transaction: CdoTransaction = (
    self.inventory_api.onboard_ftd_device_using_ztp(ztp_onboarding_input)
)
```

And then wait for the transaction to finish, exactly as before!

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Onboard FTD (non-ZTP): The API



- Docs:
  - https://developer.cisco.com/docs/cisco-security-cloud-control/onboard-ftd-device/
  - https://developer.cisco.com/docs/cisco-security-cloud-control/register-ftd-device-to-fmc/

Onboard FTD: Python Code

```
ftd_input = FtdCreateOrUpdateInput(
    name = "Alice-Springs",
    licenses = ["BASE", "MALWARE"],
    virtual=True,
    performance_tier = "FTDv10"
)
transaction: CdoTransaction = self.inventory_api.create_ftd_device(
    ftd_input
)
```

And then wait for the transaction to finish, exactly as before!

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Get CLI Key: Python Code

```python
finished_transaction: CdoTransaction = (
    self.transaction_service.wait_for_transaction_to_finish(
        transaction_uid=transaction.transaction_uid
    )
)
device: Device = self.inventory_api.get_device(
    device_uid=finished_transaction.entity_uid
)
```

```python
self.console.print(
    "Paste the following CLI key into the FTD terminal:"
    f"\n{'=' * 10}\n"
    f"{device.cd_fmc_info.cli_key}"
    f"\n{'=' * 10}"
)
```

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

Register FTD with Security Cloud Control/FMC: Python Code

```
transaction: CdoTransaction = (
    self.inventory_api.finish_onboarding_ftd_device(
        FtdRegistrationInput(ftd_uid=device.uid)
    )
)
```

And then wait for the transaction to finish, exactly as before!

Note: Execute this code after the `configure manager` CLI has been pasted into the FTD CLI

All of the code here can be found in https://github.com/siddhuwarrier/scc-automation-example

iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help Fri 8 Nov 19:34:59

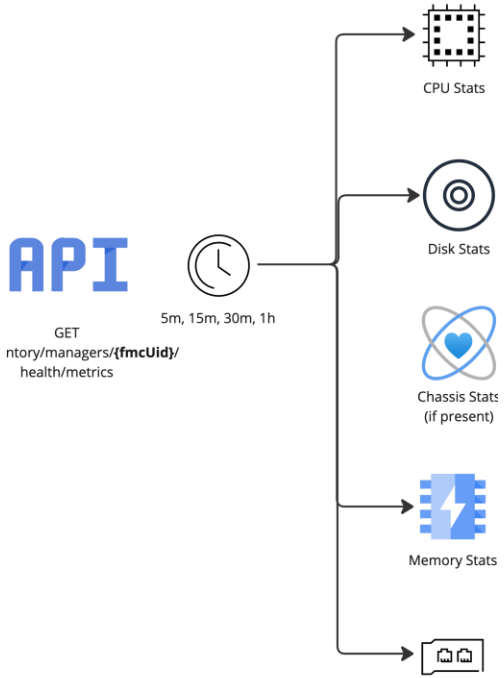
tmuxinator start mac

```
~/w/cdo/clapj24/mssp-tenant-automation-api  main ↑2 +8 !12 ❯ python provision_tenant.py --users-csv-file users.csv.sample --tenant-name cisco-live-demo-2024-11-07-1
322 --display-name "CL Melbourne Demo 07-11-1322" --provision-cdfmc yes
```

cdo-mac siwarrie 7:0 <:MicroservicesZ 4:Terraform 5:Infra 6:Deployments 7:clapj24* 8:cdo-public-api-docs-Z 19:34:58 08-Nov-24 Siddhus-MacBook-Pro-2.local

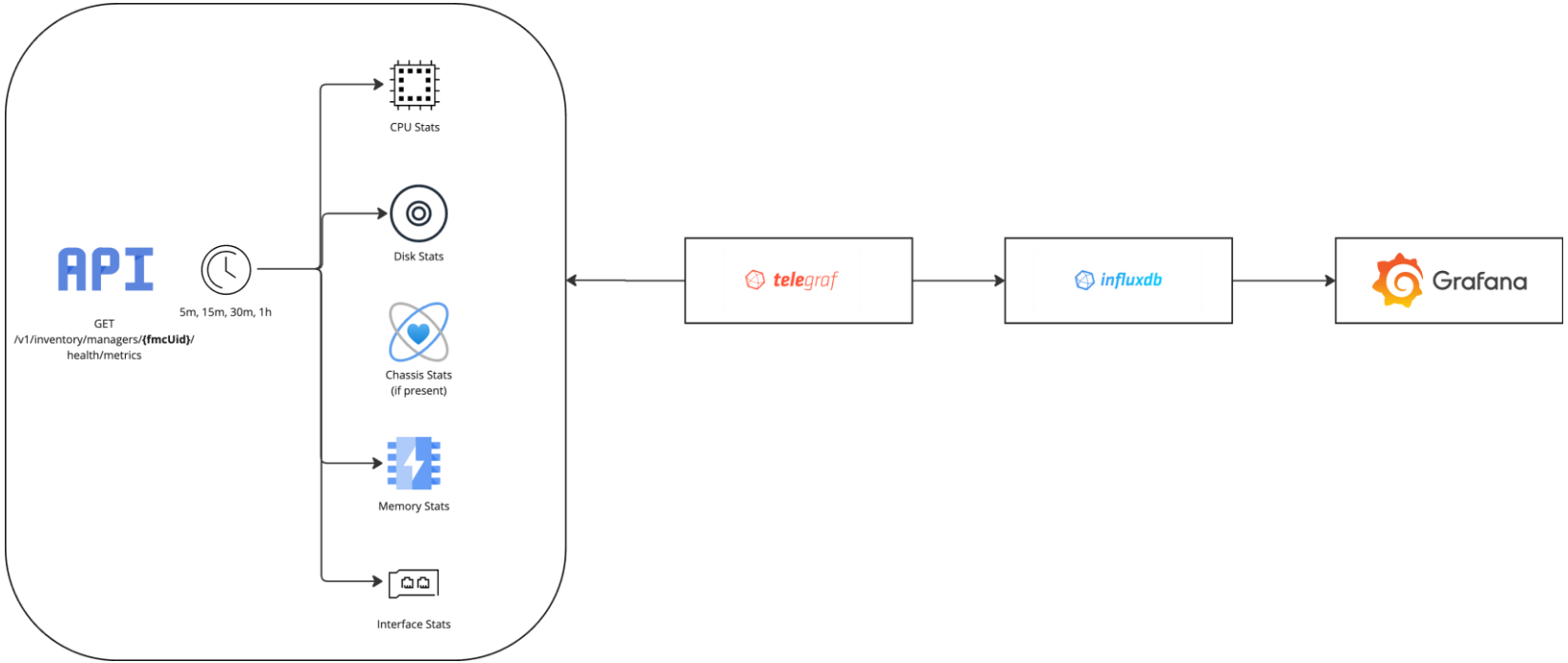
Monitoring

FTD Monitoring: The API



Docs: https://developer.cisco.com/docs/cisco-security-cloud-control/get-health-metrics-on-devices-managed-by-the-fmc-cdfmc-only/

FTD Monitoring: Integrate it into your tools



FTD Monitoring: An Example

Upgrades

A single API endpoint to upgrade your FTDs

Coming soon!

Continue your education

- Visit the Cisco Stand for related demos
- Book your one-on-one Meet the Expert meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me: siwarrie@cisco.com (email or Webex)

Complete Your Session Evaluations

Complete a minimum of 4 session surveys and the Overall Event Survey to claim a **Cisco Live T-Shirt**.

Complete your surveys in the **Cisco Live mobile app**.

Thank you