

Configure Cisco RADKit Integration in FMC

Contents

[Introduction](#)

[Background](#)

[Feature Description and Walkthrough](#)

[FMC REST APIs](#)

[Get Additional Details from the Devices](#)

[Cisco Support: RADKit Console](#)

[Upgrades and Backwards Compatibility](#)

[Troubleshooting](#)

[Overview of Diagnostics](#)

[RADKit Session Logs](#)

[Sample Problem with Troubleshooting Walkthrough](#)

[Telemetry](#)

[FAQ](#)

Introduction

This document describes the Cisco RADKit Integration in FMC feature added in 7.7 release.

Background

Problem Firewall Administrators Face

- The Remote Automation Development Kit (RADKit), developed by Cisco, is a network-wide orchestrator designed to provide users with the possibility to access in a secure manner and troubleshoot network devices. <https://radkit.cisco.com/>
- The Cisco Secure Firewall Management Center (FMC) manages and operates Secure Firewall Threat Defense (FTD) devices. A single FMC can manage several devices in various locations.
- While it is possible for users to install RADKit separately and onboard their FMC and FTDs into it, building the RADKit Service into the FMC and onboarding the FMC(s) and all the managed devices (FTDs) in an automated manner would be a better experience for end users.

Use Case

Some of the key capabilities from which the users could benefit after integrating RADKit in the FMC include:

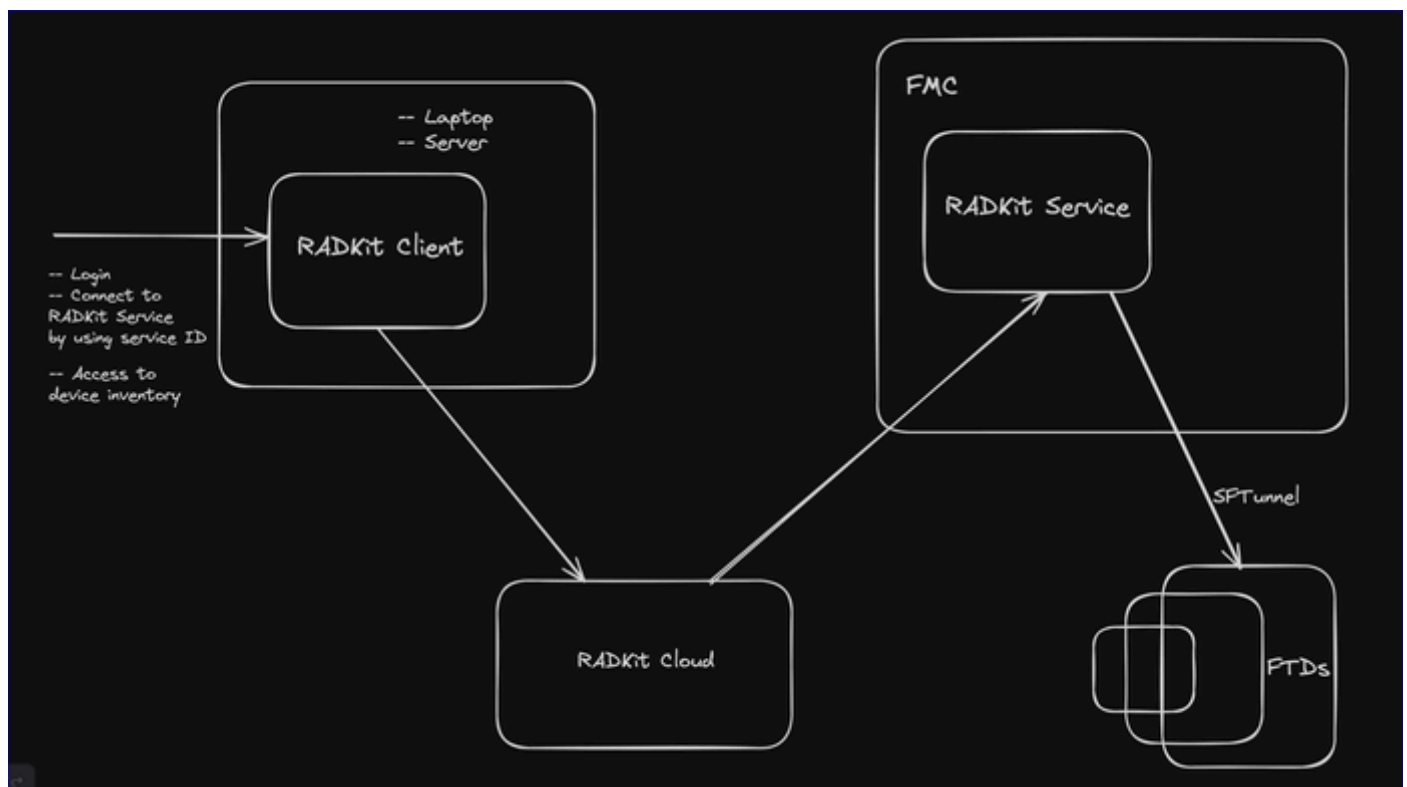
- Ability to access the FMCs/FTDs remotely from the RADKit client CLI.
- Ability to provide controlled access to the FMCs/FTDs to anyone that needs it (for example, a Cisco TAC engineer).
- Leverage automation capabilities for collecting data and diagnosing problems from the RADKit client (scripts that execute commands on multiple devices can be created and used from the RADKit client).

What's New - Solution

- Starting from Secure Firewall 7.7.0, the Remote Automation Development Kit (RADKit) Service is integrated in FMC.
- Users can enable or disable the RADKit service on demand, enroll it in the RADKit cloud, and create remote users' authorizations to access specific devices from the RADKit Client for a scheduled access duration.
 - Authorizations can be edited or revoked.
- There is also an option to provide devices sudo access for advanced troubleshooting.

RADKit Service Integration in FMC Diagram

This diagram shows how RADKit enables communication from user's (TAC engineer) RADKit client to production FTD devices:



Basics: Supported Platforms, Licensing

Applications and Managers

FTD		ASA	
FMC and FTD Platforms: All		Not supported	
FMC on 7.7.0 FMC REST API	Yes Yes	ASA CLI 9.23.1	No
FTD Supported Versions <i>(lowest version FMC on 7.7.0 can manage is 7.2)</i>	7.7.0 only	ASDM 7.23.1	No
Snort Support <i>(Snort 3 is the only Snort version supported in 7.7)</i>	Snort 3 Snort 2 <i>(only for devices on 7.2.x..7.6.x)</i>	CSM 4.30	No
FDM on 7.7.0	No		

Other Aspects of Support

Platforms	
FTD	
Licenses Required	No licensing requirements for this feature.
Works in Evaluation Mode	Yes
IP Addressing	IPv4 IPv6
Multi-instances supported?	Yes
Supported with HA'd devices	Yes
Supported with clustered devices?	Yes
Other (only routed mode transparent mode), etc.	No Special Notes
Internet access for the RADKit cloud enrollment required	Access to prod.radkit-cloud.cisco.com

Dependencies for Feature to Operate

- Minimum version is Secure Firewall 7.7.0.
- For connecting to the RADKit Service hosted within FMC, the RADKit Client needs to be installed from <https://radkit.cisco.com/downloads/release/> on support engineer's computer.
- The preferred version for the RADKit Client is 1.6.10 or higher.
- Older versions of RADKit Client could be used as the RADKit Service is backward compatible with older RADKit Client versions.

Feature Description and Walkthrough

Feature Overview

- The RADKit Service integration into FMC allows the device administrators to provide remote users (Cisco TAC engineers) access to specific FMC and FTD devices in their network for troubleshooting and automation purposes. RADKit is much more efficient for troubleshooting than screen sharing, it

does not require to control the user's computer, is a safer way to work on a network and nicely complements Webex.

- This makes a better experience for technical support as device administrators do not have to install and configure the RADKit Service separately. Also, this reduces the support time for Cisco TAC engineers in resolving support issues.

Configuration Steps: Overview

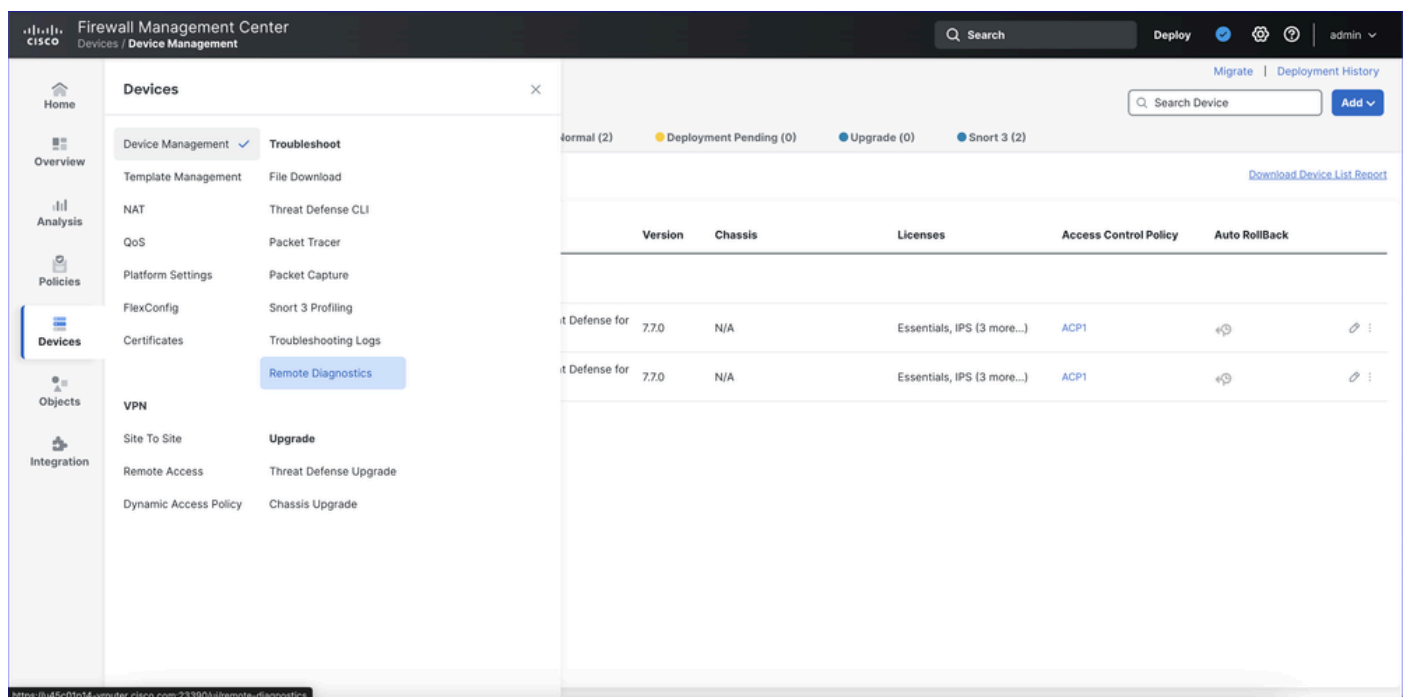
1. Device administrator (FMC admin user): Enable and enroll the RADKit service and configure authorizations on FMC GUI.

2. Cisco TAC/Cisco Support: Install RADKit Client on their computer, access, and troubleshoot the devices from RADKit Client.

FMC Admin User: Firewall Management Center Walkthrough

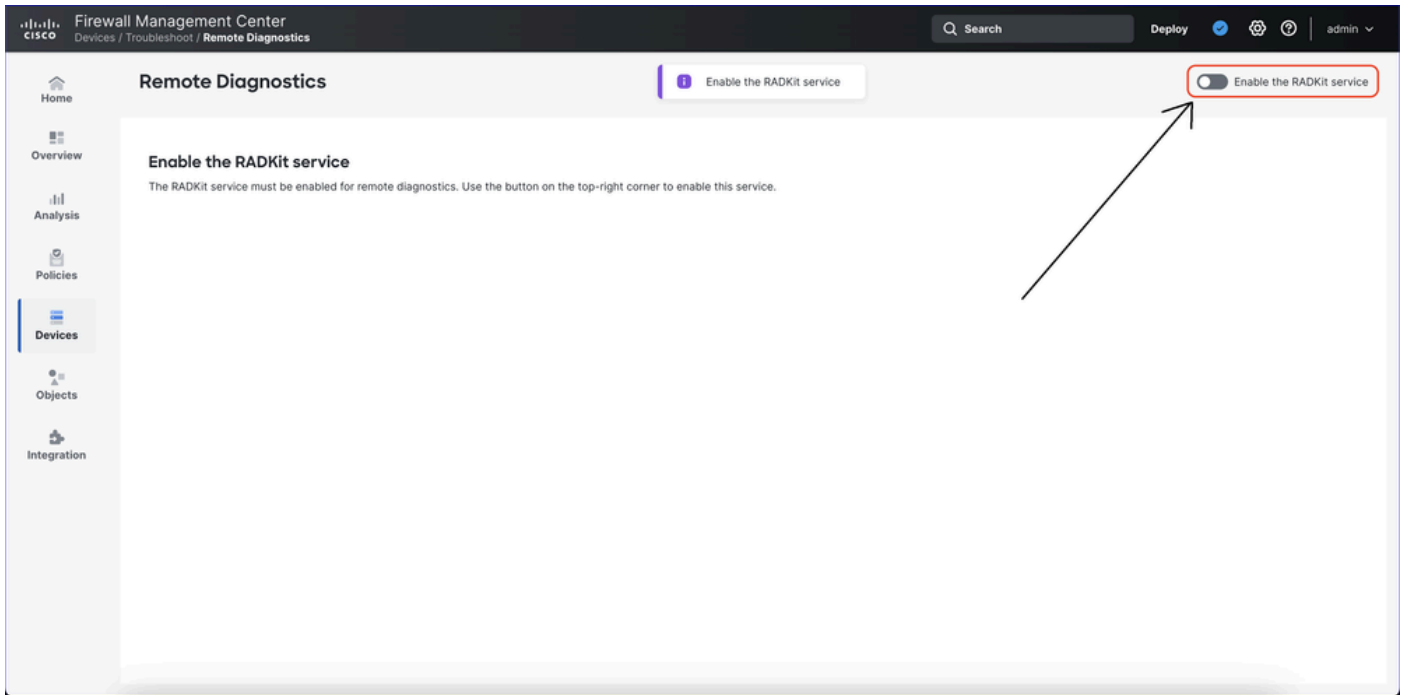
Remote Diagnostics Menu

- A new **"Remote Diagnostics"** menu item has been added for this feature under **Devices -> Troubleshoot**.
- Administrator, Network Admin, and Maintenance users have read/write permissions on the page.
- Security Analyst, Security Analyst (Read Only), and Security Approver users have read only permissions to the page.



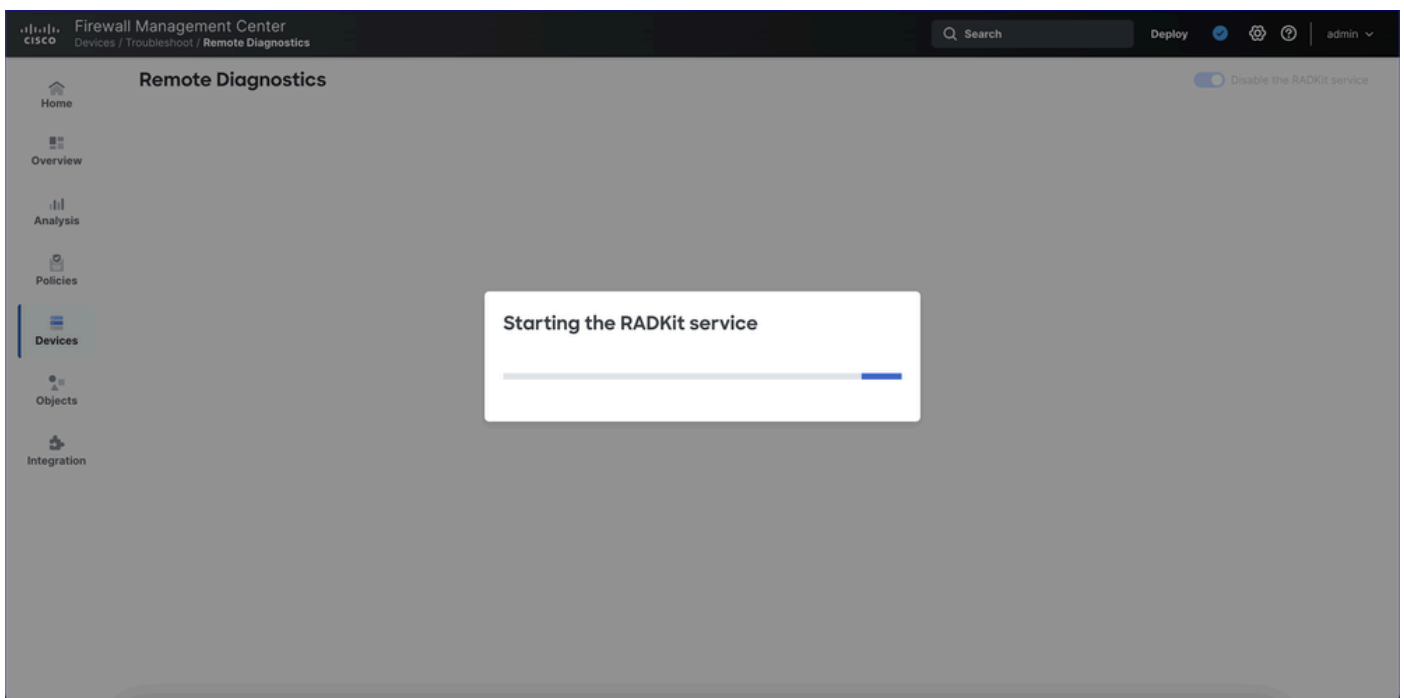
Initial Remote Diagnostics Page

This is the initial **Remote Diagnostics** page. The RADKit service can be enabled by toggling the **"Enable the RADKit service"** switch:



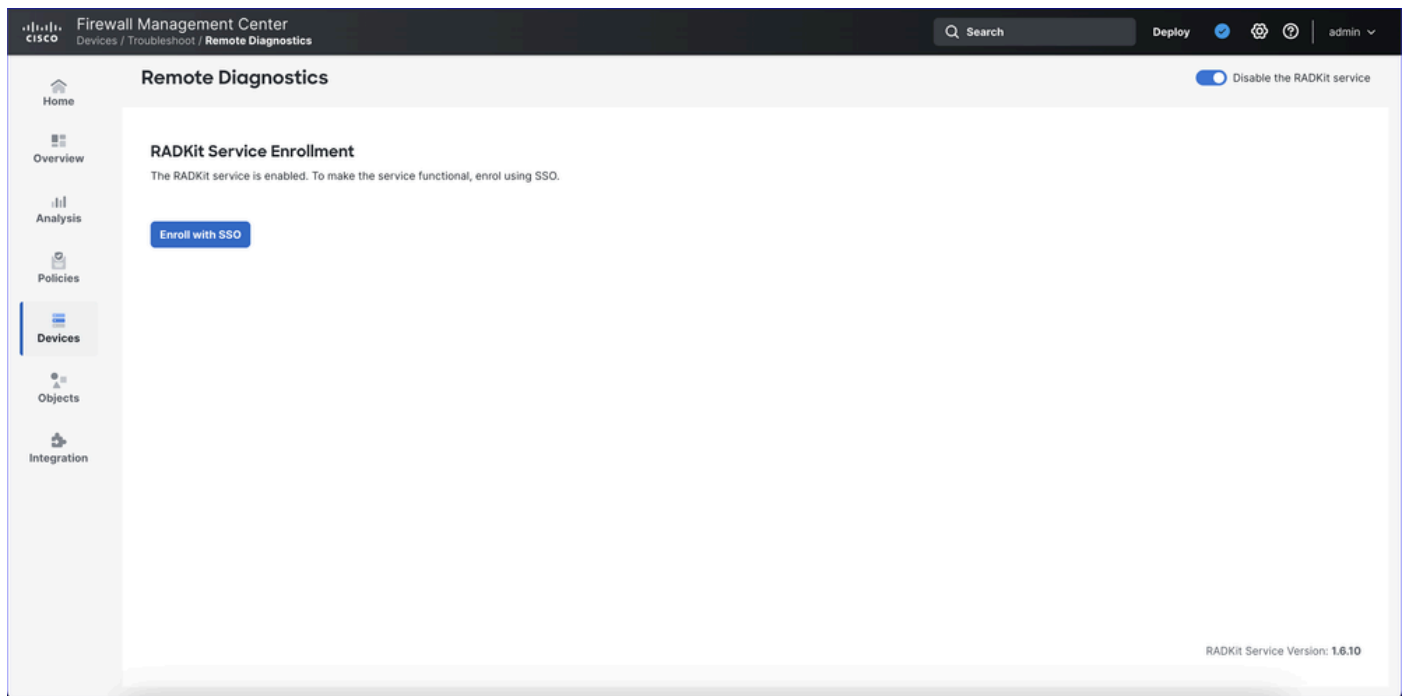
RADKit Service Starting

After enabling the RADKit service, a progress bar appears until the RADKit service is started:



RADKit Service Enabled

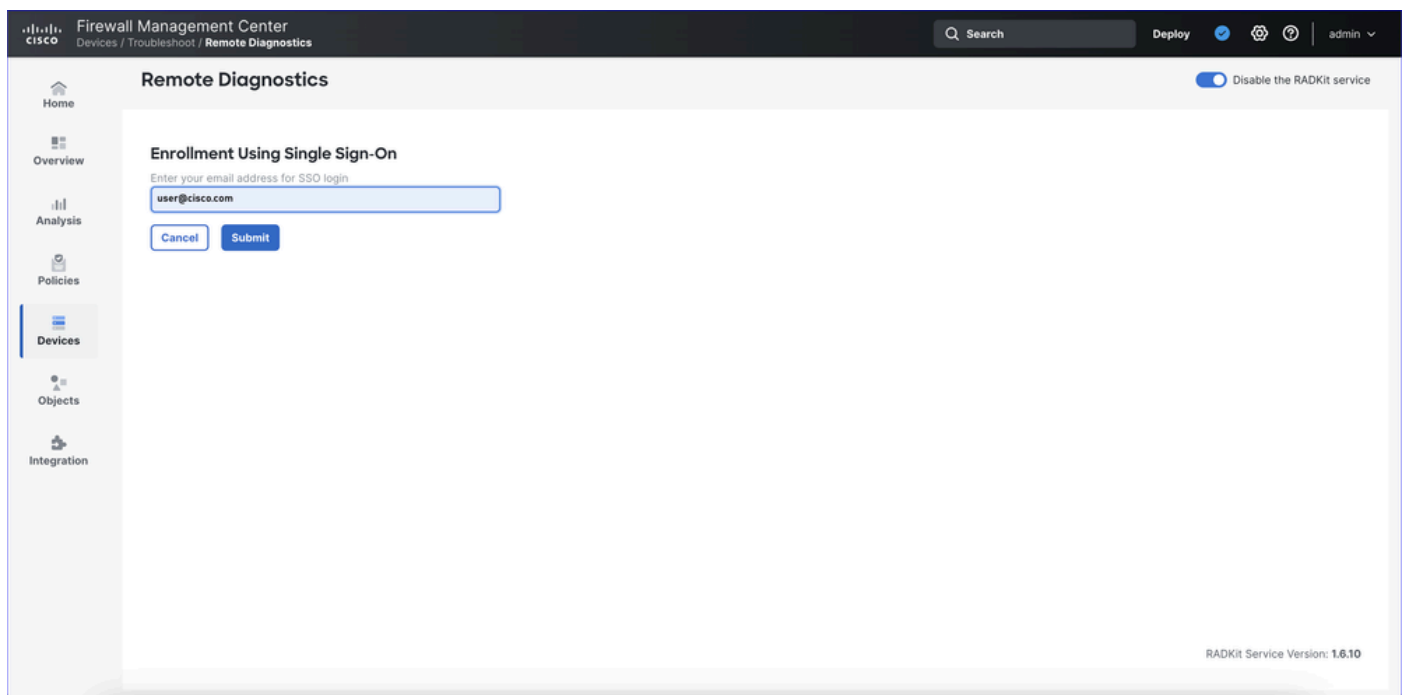
- When the RADKit service enabling process is completed, this page is shown:



Next step is enrollment in the RADKit cloud by clicking on “**Enroll with SSO**” button.

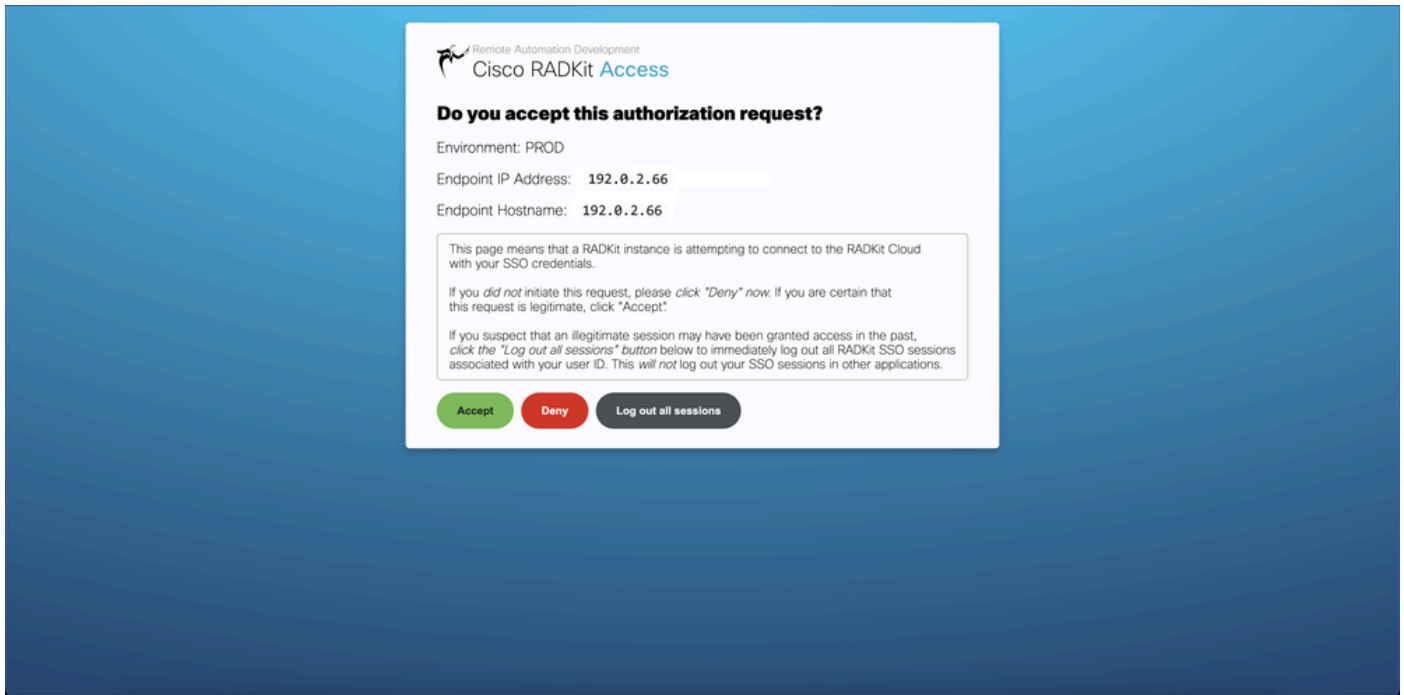
Enroll with SSO – Enter Email Address

Step 1 of the enrollment process consists of entering the user email address for the RADKit cloud enrollment:



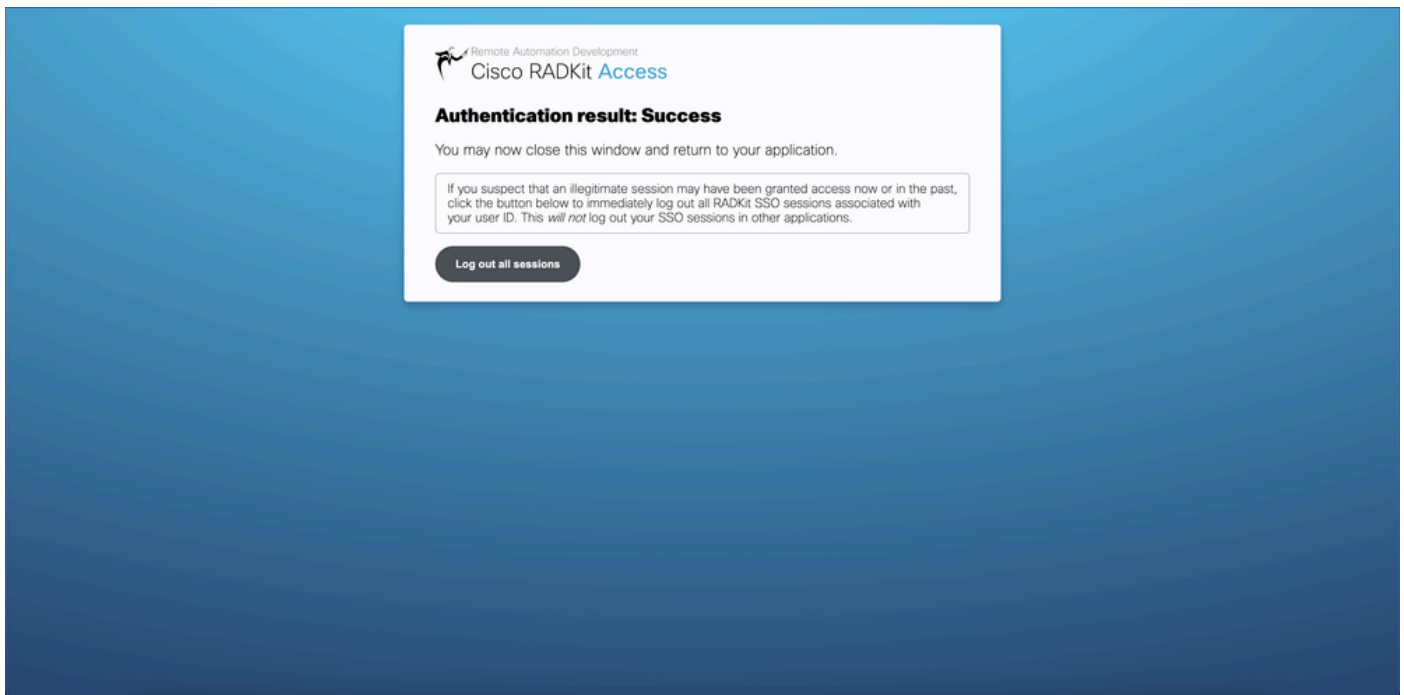
Enroll with SSO – Accept Authorization Request

A new browser tab (or window, depending on browser settings) opens. Click the **Accept** button.



Enroll with SSO – Authentication Successful

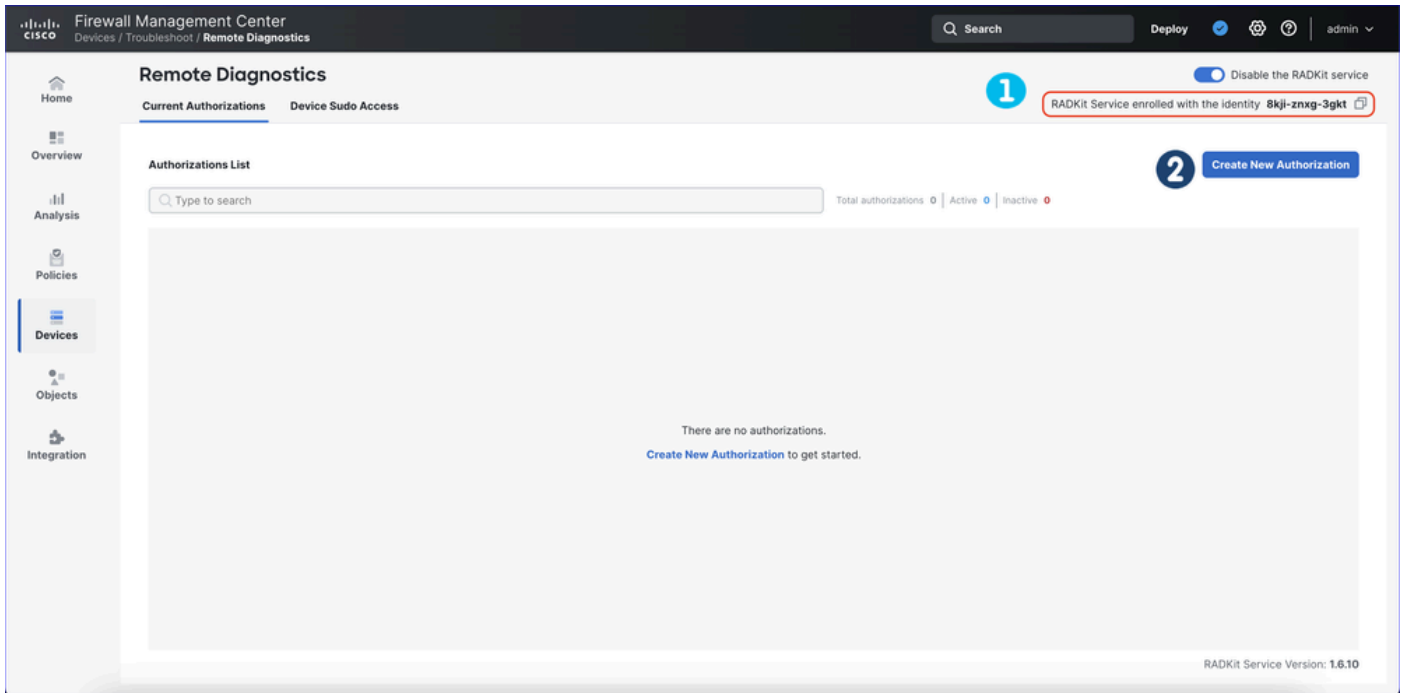
After successful authentication, the user can close the browser tab and return to the FMC **Remote Diagnostics** page.



RADKit Service Enrolled

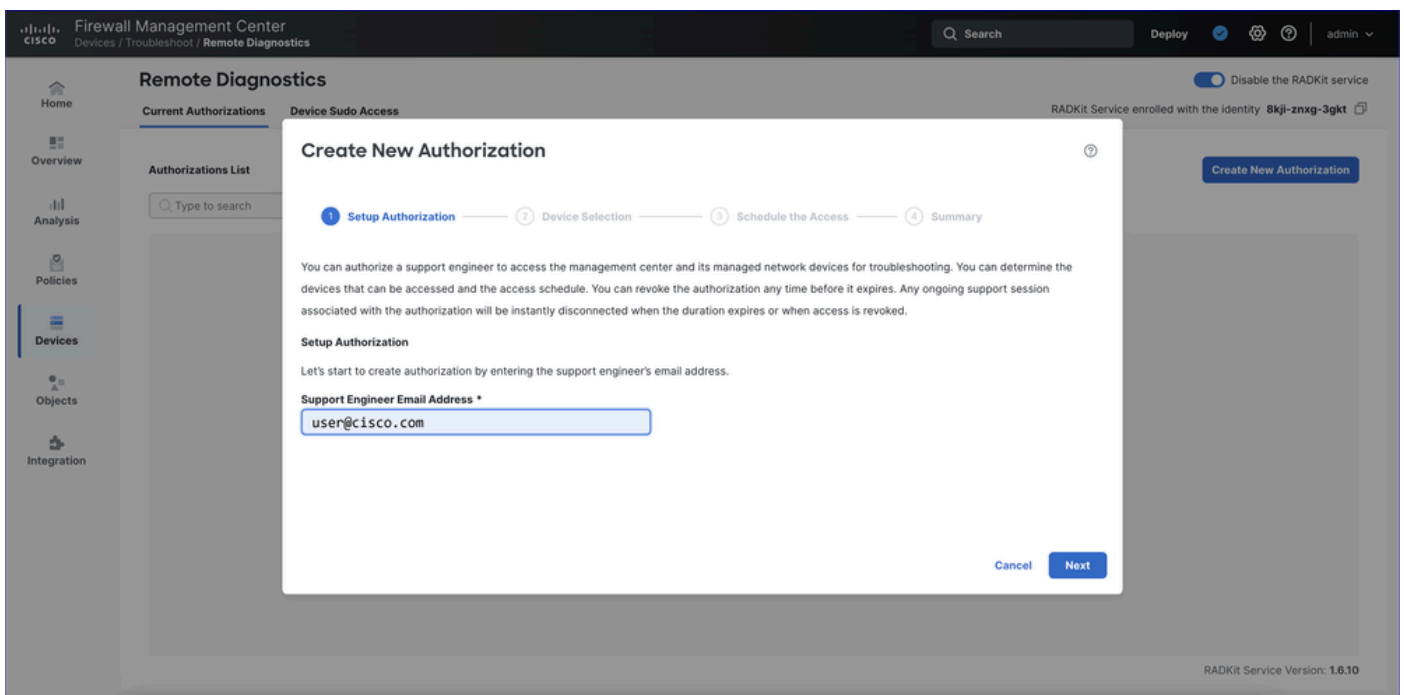
The RADKit service is enrolled with the specified service ID (in this example, the ID is 8kji-znxg-3gkt). The ID can be copied to clipboard. Give it to the Cisco TAC engineer so they can connect to the RADKit service from the RADKit client.

The next step is to create an authorization by clicking the “**Create New Authorization**” button:



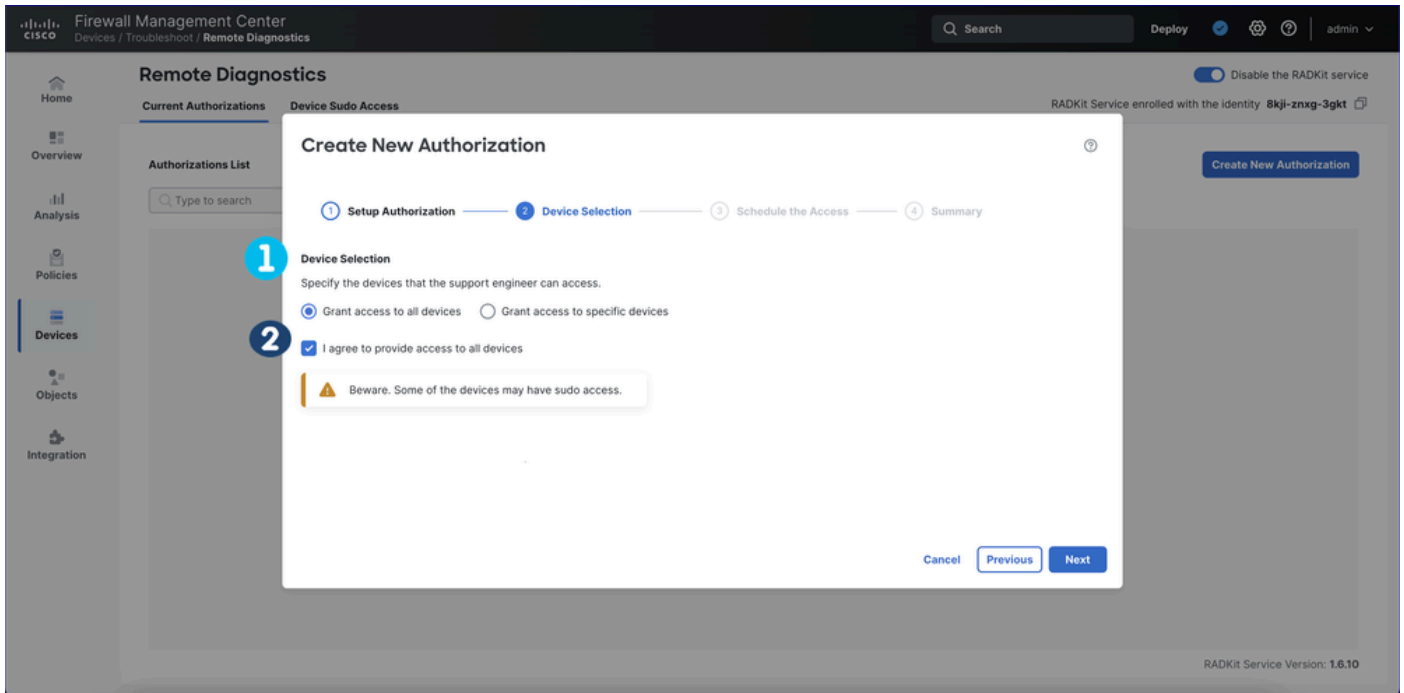
Create New Authorization: Step 1

- For creating a new authorization, the first step is to add the support engineer email address.
- There are four steps for creating a new authorization. Progress along the steps is shown at the top.



Create New Authorization: Step 2

- Step 1: Specify the devices that the support engineer can access. Or, as in this example, grant access to all devices.
- Step 2: Check the radio button for all or specific devices. For specific devices, FMC and/or FTD(s) can be picked. Note the warning that sudo access can provide to some devices in Device Sudo Access tab. The **Next** button is not enabled until the checkbox is checked.
- Sudo access is provided per device in the Device Sudo Access tab later (not during creating an authorization).

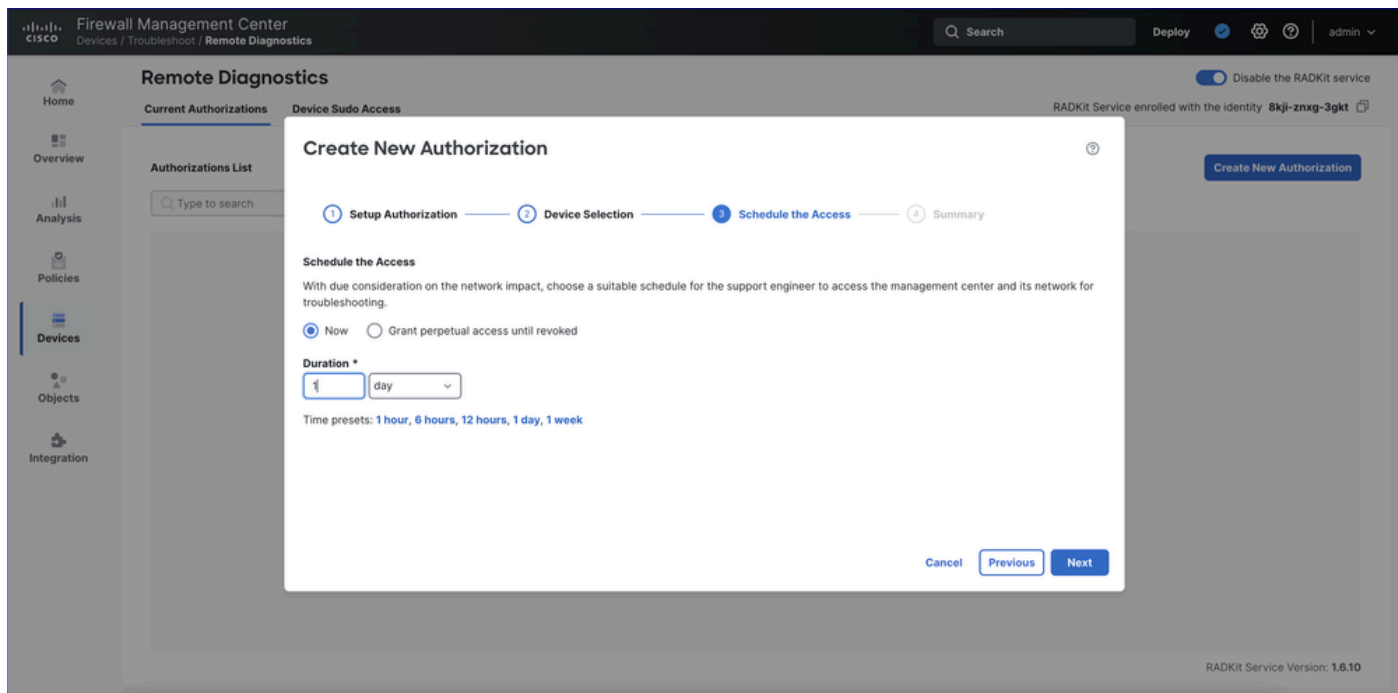


Notes About Picking Devices

- Only devices on a supported build (for example, in the initial release, only 7.7.0 devices) can be picked.
- Devices which are disabled and not reachable are not selectable. RADKit relies on sftunnel (TCP 8305) to access the devices.
 - If there is a sftunnel connectivity issue, it does not work, but it still is shown in the RADKit inventory.
 - If a device is powered off, it is not seen at all.
- If there are FMCs in an HA pair, only the Active/Primary can be added.
- The devices are added to the RADKit inventory when creating/editing an authorization. When devices are deregistered from FMC, they are removed from the **"inventory"** of devices.

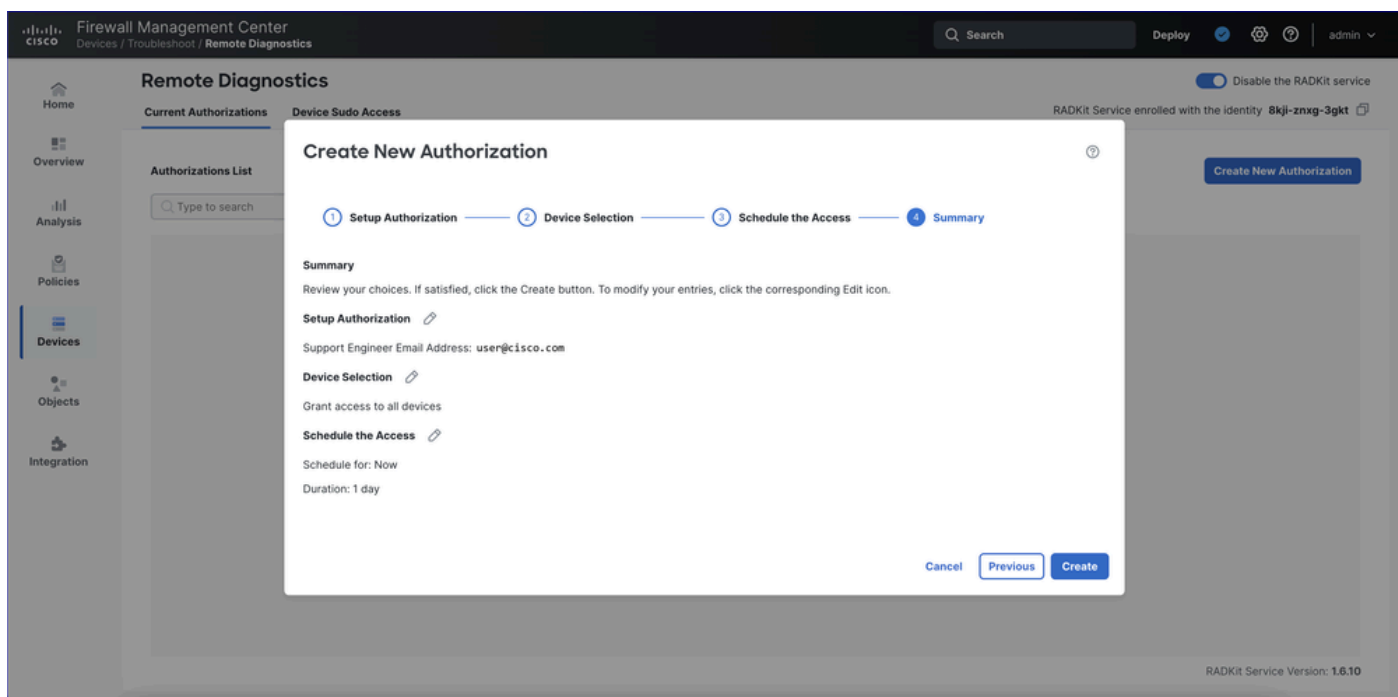
Create New Authorization: Step 3

- Step 3: Specify the duration for the support engineer access to devices.
- Pick **"Now"** and specify a duration, or,
- Pick **"Grant perpetual access until revoked"**.
- Default duration is 1 day. Any duration can be set; there are also some predefined duration values.



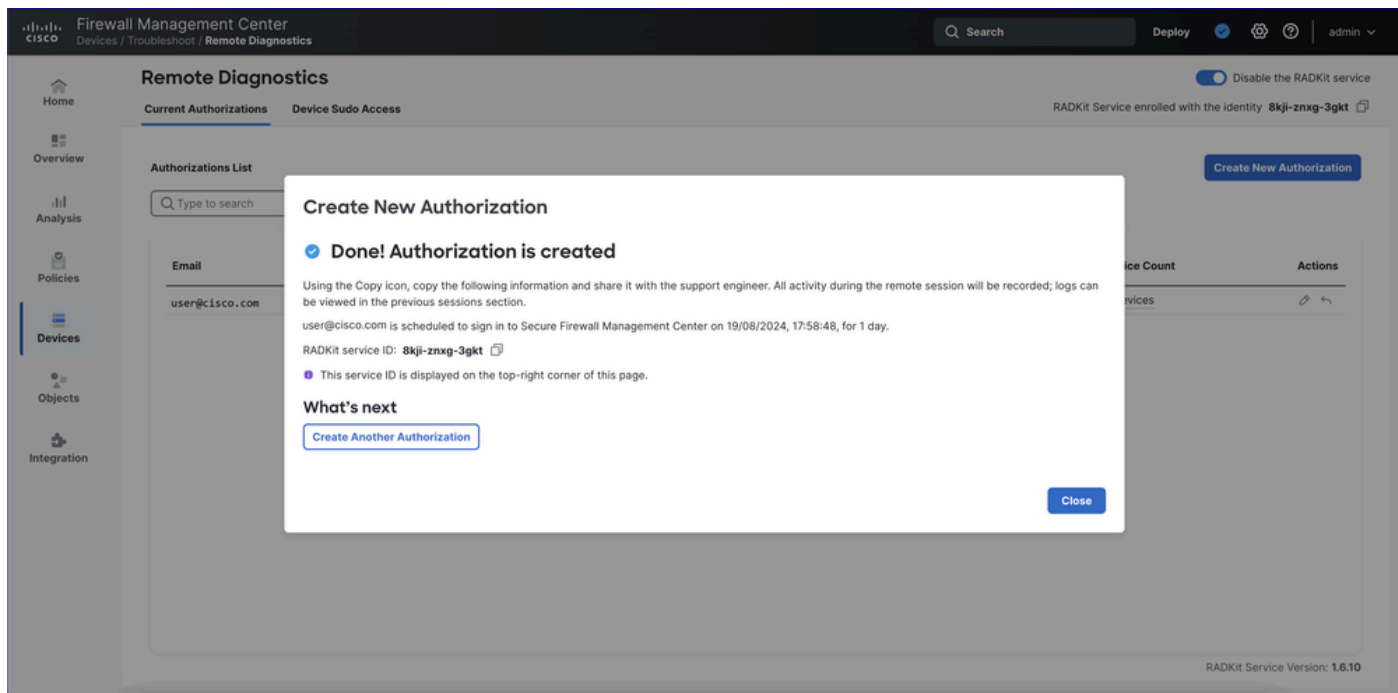
Create New Authorization Summary

The final step is the authorization summary. Here, a user can review and edit the configuration.



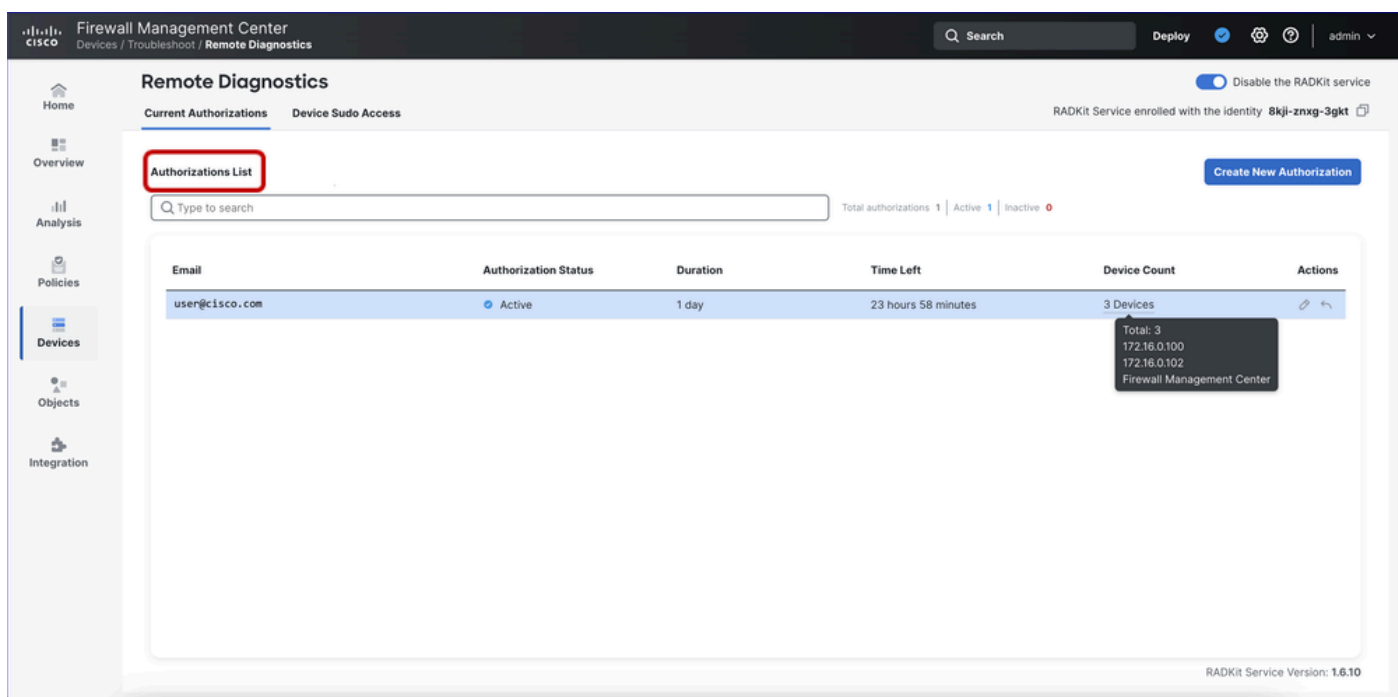
Create New Authorization Completed

A confirmation screen is displayed after authorization creation is completed:



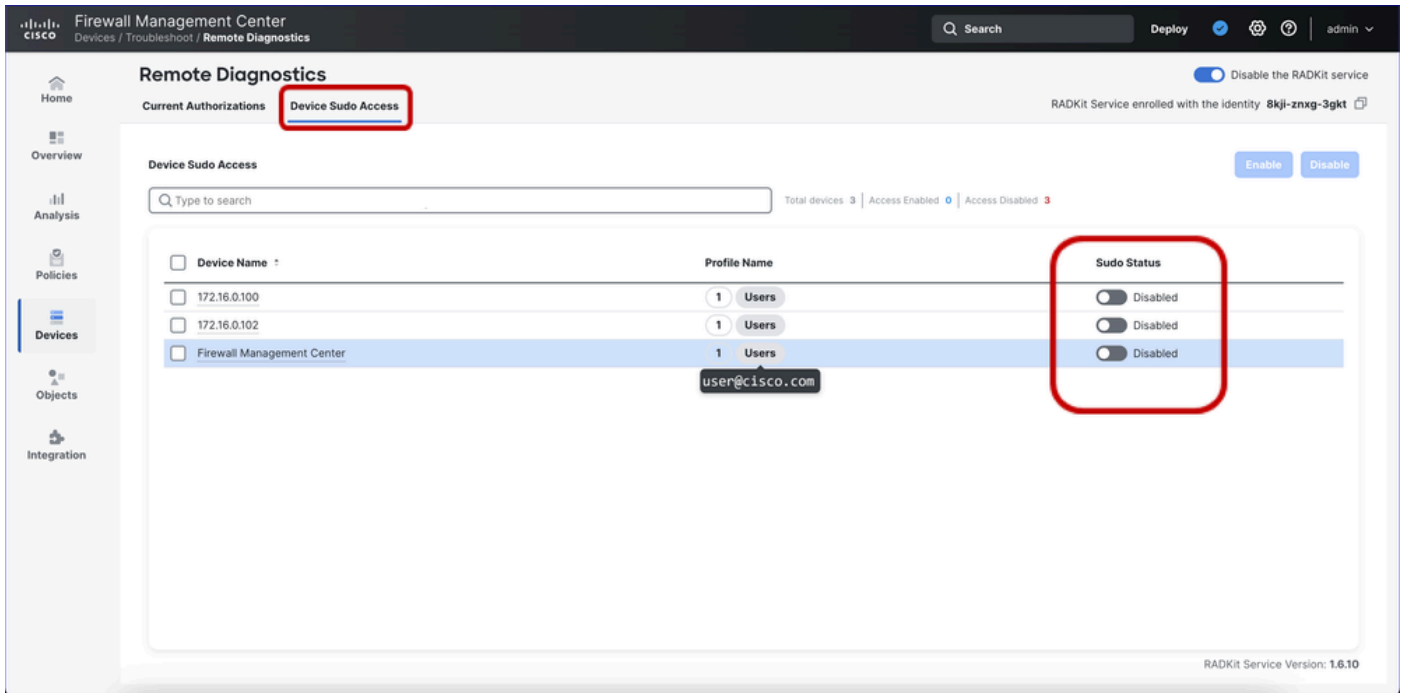
Current Authorizations List, Including Revoke

- The current authorizations list is displayed in the **Current Authorizations** tab.
- The Actions (far right column) are Revoke access and Edit authorization.



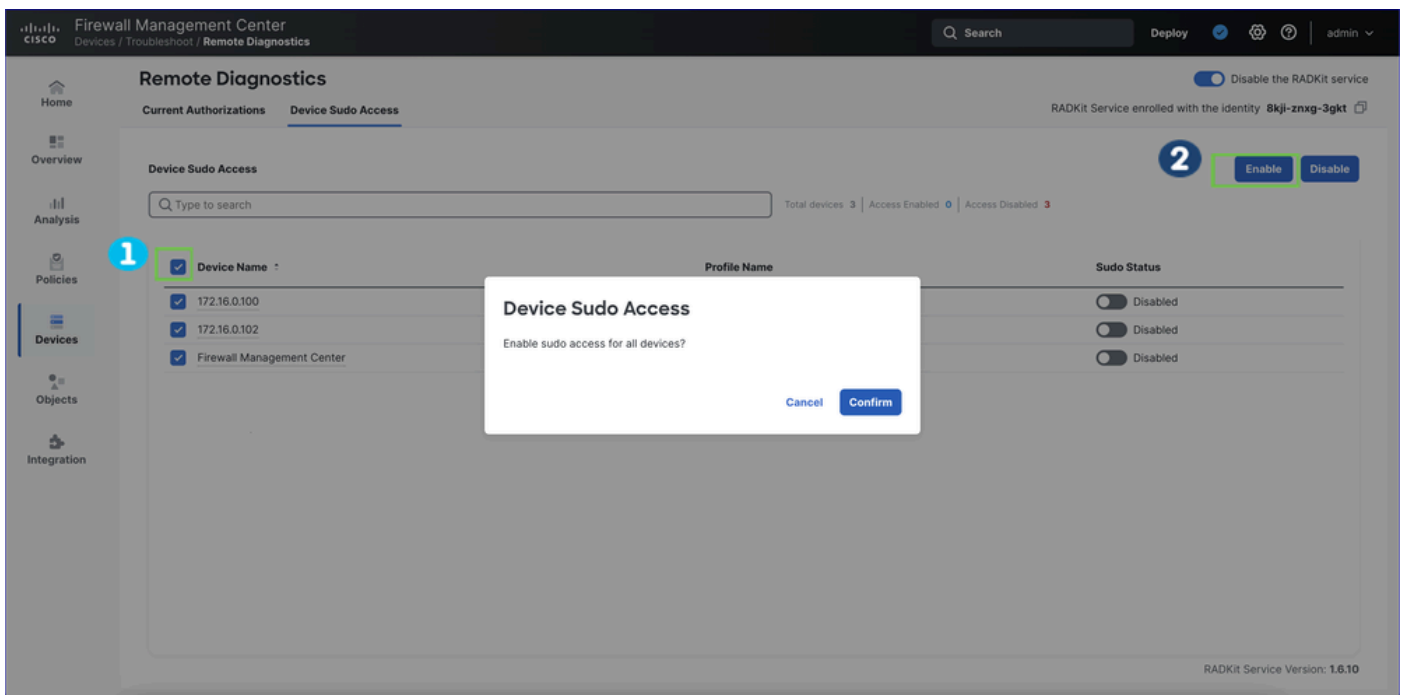
Device Sudo Access List

- The list of devices with sudo access settings is presented in the **Device Sudo Access** tab.
- Use the toggle in the right column to toggle sudo access on. It is off by default.
- Also, bulk enable/disable sudo access is available.



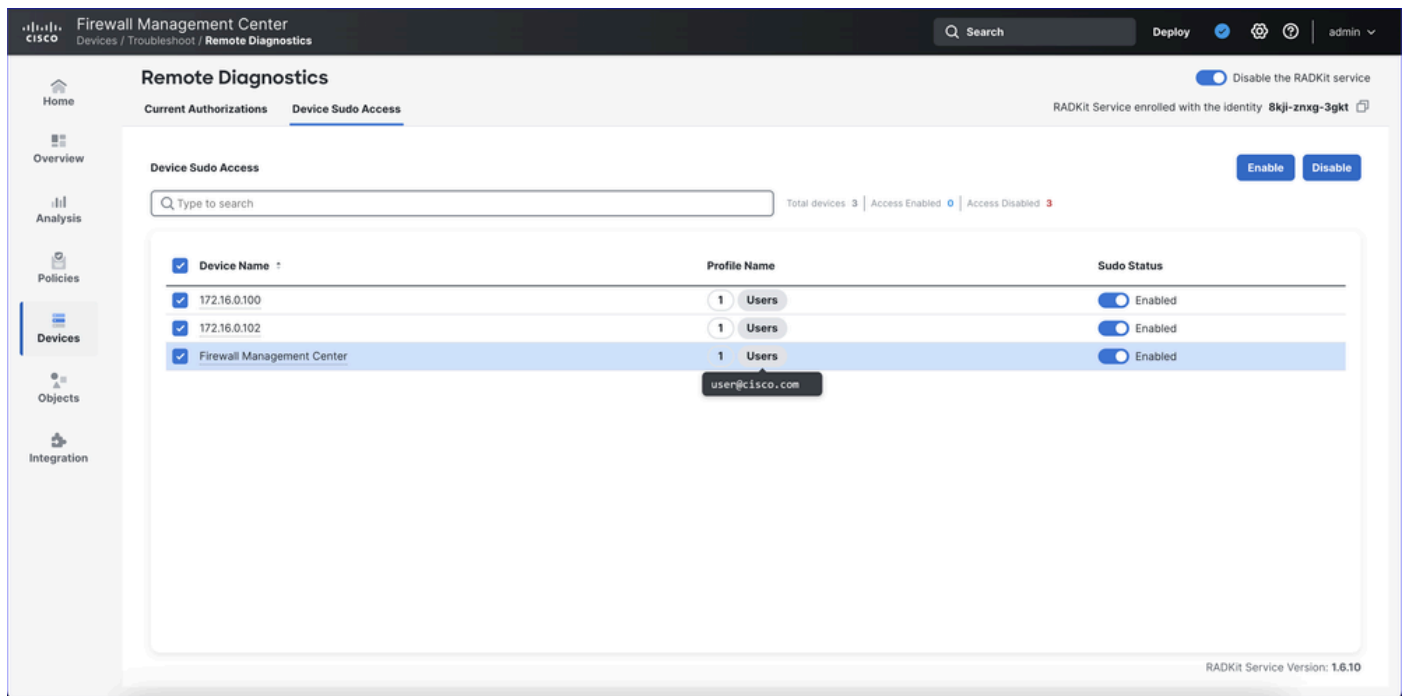
Confirm Enabling Device(s) Sudo Access

1. Sudo access can be enabled for all or only for some specific devices by selecting the devices then clicking on **"Enable"** button.
2. When enabling, a confirmation dialog appears and clicking **Confirm** is required.



Devices Sudo Access Enabled

- After enabling or disabling sudo access for a device, the **Sudo Status** column at the right of the page is updated.
- The support engineer is able to run **sudo su** on the device; this is passwordless. The support engineer does not need to have the root password.



Other Notes

- Only devices in the domain that the FMC user has access to are visible and can be authorized for remote access.
- If FMCs are in HA:
 - The RADKit Service can only be enabled on the Active/Primary.
 - The Secondary FMC cannot be added currently as a device to be accessed from RADKit client.
- Authorization can only be made for one support engineer at a time.
 - If you need another support engineer to have access, create another authorization for the additional engineer. The service ID would be the same.

FMC REST APIs

RADKit Service REST APIs

To support create and read operations on RADKit Service, these new URLs have been introduced:

- **GET: /api/fmc_troubleshoot/v1/domain/{domainUUID}/radkit/services**
 - Retrieves all the RADKit Service data from the FMC.
- **GET: /api/fmc_troubleshoot/v1/domain/{domainUUID}/radkit/services/{id}**
 - Fetch/retrieves the RADKit Service data from the specified ID.
- **POST: /api/fmc_troubleshoot/v1/domain/{domainUUID}/radkit/services**
 - Creates the RADKit Service on the FMC (enable/disable the service).

RADKit Service Model

The RADKit service model consists of:

- type
- id
- status

- isEnrolled
- serviceId
- version

```
{  
  "type": "RADKitService",  
  "id": "DummyContainerId",  
  "status": "RUNNING",  
  "isEnrolled": true,  
  "serviceId": "8kji-znxg-3gkt",  
  "version": "1.6.10"  
}
```

Cisco Support: RADKit Client Usage

Support Side: Install the RADKit Client

- To access the FMC/FTD(s), support needs to have the RADKit client installed.
 - The client works on Windows, Mac, and Linux operating systems.
- Support can have access to multiple devices from multiple user. Each RADKit authorization has its own "**inventory**" of devices.
 - For each user device inventory that support wants to access, the RADKit service ID is needed.
 - For a single inventory, access is possible both for the FMC and its managed FTDs from the RADKit client, as specified by the user when authorizing access.

Obtain and Install RADKit Client

The RADKit client can be installed locally from <https://radkit.cisco.com/downloads/release/> then launched from terminal with the command: **radkit-client**

Installers are available for Windows, MacOS, and Linux.

```
radkit-client - 147x40
15:07:59.886Z INFO | internal | CXD object created without authentication set, call '<this object>.authenticate()' to set authentication.

Example usage:
client = sso_login("<email_address>") # Open new client and authenticate with SSO
client = certificate_login("<email_address>") # OR authenticate with a certificate
client = access_token_login("<access_token>") # OR authenticate with an SSO Access Token
service = client.service("<serial>") # Then connect to a RADKit Service
service = start_integrated_service() # Immediately login to an integrated session
service = direct_login() # Establish cloud-less direct connection to service.
client.grant_service_otp() # Enroll a new service

>>> client = sso_login("user@cisco.com")

A browser window was opened to continue the authentication process. Please follow the instructions there.

Authentication result received.
>>> service = client.service("8kji-znxg-3gkt")
15:09:03.406Z INFO | internal | Connecting to forwarder [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-4/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-4/websocket/']
15:09:03.639Z INFO | internal | Connection to forwarder successful [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-4/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-4/websocket/']
15:09:03.727Z INFO | internal | Forwarder client created. [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-4/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-4/websocket/']
15:09:04.003Z INFO | internal | Connecting to forwarder [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-1/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-1/websocket/']
15:09:04.244Z INFO | internal | Connection to forwarder successful [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-1/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-1/websocket/']
15:09:04.332Z INFO | internal | Forwarder client created. [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-1/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-1/websocket/']
>>> service.inventory
<radkit_client.sync.device.DeviceDict object at 0x1154969a0>
name host device_type Terminal Netconf SNMP Swagger HTTP description failed
-----
172-16-0-100-1724078669 127.0.0.3 FTD True False False False False 172.16.0.100 False
172-16-0-102-1724078669 127.0.0.2 FTD True False False False False 172.16.0.102 False
firepower-1724078669 127.0.0.1 FMC True False False False False firepower False
Untouched inventory from service 8kji-znxg-3gkt.
>>> 
```

RADKit client screenshot with login commands (details on the next section).

RADKit Client Login Commands

- Use the email address the user entered during authorization in FMC.
- The RADKit client login and connecting to the specified service ID commands. The RADKit service ID, in this example 8abc-znxg-3abc, must match what the firewall administrator sees in FMC.

```
<#root>
```

```
>>>
```

```
client = sso_login("user@cisco.com")
```

A browser window was opened to continue the authentication process.

Please follow the instructions there.

Authentication result received.

```
>>>
```

```
service = client.service("8abc-znxg-3abc")
```

```
15:09:03.639Z INFO | internal | Connection to forwarder successful [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-4/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-4/websocket/']
15:09:03.727Z INFO | internal | Forwarder client created. [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-4/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-4/websocket/']
15:09:04.244Z INFO | internal | Connection to forwarder successful [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-1/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-1/websocket/']
15:09:04.332Z INFO | internal | Forwarder client created. [forwarder_base_url='wss://prod.radkit-cloud.cisco.com/forwarder-1/' uri='wss://prod.radkit-cloud.cisco.com/forwarder-1/websocket/']
```

RADKit Client Service Inventory Command

Command for listing the inventory that the remote user (Cisco TAC engineer) is authorised to access:

```
<#root>
```

```
>>>
```

```
service.inventory
```

```
<radkit_client.sync.device.DeviceDict object at 0x1154969a0>
```

name	host	device_type	Terminal	Netconf	SNMP	Swagger	HTTP	de
172-16-0-100-1724078669	127.0.0.3	FTD	True	False	False	False	False	17
172-16-0-102-1724078669	127.0.0.2	FTD	True	False	False	False	False	17
firepower-1724078669	127.0.0.1	FMC	True	False	False	False	False	fi

Untouched inventory from service 8kji-znxg-3gkt.

There is a filter command for the devices in the inventory (next section). Use the name in the left column to start an interactive session with the device (command on upcoming section).



Tip: If the inventory is outdated, you can update it using the command:

```
>>> service.update_inventory()
```

RADKit Client: Filter Devices

Command for filtering devices in the inventory:

```
<#root>
```

```
>>>
```

```
ftds = service.inventory.filter(attr='name',pattern='172-16-0')
```

```
>>>
```

```
ftds
```

```
<radkit_client.sync.device.DeviceDict object at 0x111a93130>
```

```
name host device_type Terminal Netconf SNMP Swagger HTTP description failed
```

172-16-0-100-1724078669	127.0.0.3	FTD	True	False	False	False	False	False	172.16.0.100	False
172-16-0-102-1724078669	127.0.0.2	FTD	True	False	False	False	False	False	172.16.0.102	False

2 device(s) from service 8kji-znxg-3gkt.

RADKit Client Device Interactive Session Command

Launching an interactive session for a device (in this case an FMC) with the name "firepower-1724078669" taken from the previous "service.inventory" command.:

```
<#root>
```

```
>>>
```

```
service.inventory["firepower-1724078669"].interactive()
```

```
08:56:10.829Z INFO | internal | Starting interactive session (will be closed when detached)
```

```
08:56:11.253Z INFO | internal | Session log initialized [filepath='/Users/use/.radkit/session_logs/client_
```

```
Attaching to firepower-1724078669 ...
```

```
Type: ~. to terminate.
```

```
~? for other shortcuts.
```

```
When using nested SSH sessions, add an extra ~ per level of nesting.
```

```
Warning: all sessions are logged. Never type passwords or other secrets, except at an echo-less password
```

```
Copyright 2004-2024, Cisco and/or its affiliates. All rights reserved.
```

```
Cisco is a registered trademark of Cisco Systems, Inc.
```

```
All other trademarks are property of their respective owners.
```

```
Cisco Firepower Extensible Operating System (FX-OS) v82.17.0 (build 170)
```

```
Cisco Secure Firewall Management Center for VMware v7.7.0 (build 1376)
```

RADKit Client Execute Commands on Devices

Execute the commands on devices!

```
<#root>
```

```
>>>
```

```
result = ftds.exec(['show version', 'show interface'])
```

```
>>>
```

```
>>>
```

```
result.status
```

```
<RequestStatus.SUCCESS: 'SUCCESS'>
```

```
>>>
```

```
>>>
```

```
result.result['172-16-0-100-1724078669']['show version'].data | print
```

```
> show version
-----[ firepower ]-----
Model : Cisco Secure Firewall Threat Defense for VMware (75) Version 7.7.0 (Build 1376)
UUID : 989b0f82-5e2c-11ef-838b-b695bab41ffa
LSP version : lsp-rel-20240815-1151
VDB version : 392
-----
```

Get Additional Details from the Devices

Considering this inventory:

```
<#root>
```

```
>>>
```

```
service.inventory
```

```
[READY] <radkit_client.sync.device.DeviceDict object at 0x192cdb77110>
```

name	host	device_type	Terminal	Netconf	SNMP	Swagger	HTTP	desc
10-62-184-69-1743156301	127.0.0.4	FTD	True	False	None	False	False	10.6
fmc1700-1-1742391113	127.0.0.1	FMC	True	False	None	False	False	FMC1
ftd3120-3-1743154081	127.0.0.2	FTD	True	False	None	False	False	FTD3
ftd3120-4-1743152281	127.0.0.3	FTD	True	False	None	False	False	FTD3

To get '**show version**' details from the FTD devices:

```
<#root>
```

```
>>>
```

```
command = "show version"
```

```
>>>
```

```
ftds = service.inventory.filter("device_type","FTD").exec(command).wait()
```

```
>>>
```

```
>>>
```

```
# Print the results
```

```
>>>
```

```
for key in ftds.result.keys():
```

```
...
```

```
print(key)
```

...

```
ftds.result.get(key).data | print
```

...

<- Press Enter twice

ftd3120-3-1743154081

> show version

-----[FTD3100-3]-----

Model : Cisco Secure Firewall 3120 Threat Defense (80) Version 7.7.0 (Build 89)

UUID : 123a456a-cccc-bbbb-aaaa-a123456abcde

LSP version : lsp-rel-20250327-1959

VDB version : 404

>

10-62-184-69-1743156301

> show version

-----[KSEC-FPR1010-10]-----

Model : Cisco Firepower 1010 Threat Defense (78) Version 7.7.0 (Build 89)

UUID : 123a456a-cccc-bbbb-aaaa-a123456abcde

LSP version : lsp-rel-20250327-1959

VDB version : 404

>

ftd3120-4-1743152281

> show version

-----[FTD3100-4]-----

Model : Cisco Secure Firewall 3120 Threat Defense (80) Version 7.7.0 (Build 89)

UUID : 123a456a-cccc-bbbb-aaaa-a123456abcde

LSP version : lsp-rel-20250327-1959

VDB version : 404

>

Alternative approach:

<#root>

>>> # Get the FTDs. This returns a DeviceDict object:

...

```
ftds = service.inventory.filter("device_type","FTD")
```

>>> # Access the dictionary of devices from the _async_object attribute

...

```
devices_obj = ftds.__dict__['_async_object']
```

>>> # Extract the 'name' from each AsyncDevice object

```

...

names = [device.name() for device in devices_obj.values()]

>>> # Get the 'show version' output from all FTD devices:
...

command = "show version"

...

show_ver_ftds = []

...

for name in names:

...

    ftd = service.inventory[name]

...

    req = ftd.exec(command)

...

    req.wait(30)

    # depending on the number of devices you might need to increase the timeout value
    ...

    show_ver_ftds.append(req.result.data)

>>> # Print the inventory device name + 'show version' output from each device:
...

for name, show_version in zip(names, show_ver_ftds):

...

    print(f"Inventory name: {name}")

...

    print(show_version[2:-2]) # Remove the leading '> ' and trailing ' \n>'

...

    print("\n")

```

```

Inventory name: ftd3120-3-1743154081
show version
-----[ FTD3100-3 ]-----
Model : Cisco Secure Firewall 3120 Threat Defense (80) Version 7.7.0 (Build 89)

```

```
UUID : 123a456a-cccc-bbbb-aaaa-a123456abcde
LSP version : lsp-rel-20250327-1959
VDB version : 404
-----
```

```
Inventory name: ftd3120-4-1743152281
show version
```

```
-----[ FTD3100-4 ]-----
Model : Cisco Secure Firewall 3120 Threat Defense (80) Version 7.7.0 (Build 89)
UUID : 123a456a-cccc-bbbb-aaaa-a123456abcde
LSP version : lsp-rel-20250327-1959
VDB version : 404
-----
```

```
Inventory name: 10-62-184-69-1743156301
show version
```

```
-----[ KSEC-FPR1010-10 ]-----
Model : Cisco Firepower 1010 Threat Defense (78) Version 7.7.0 (Build 89)
UUID : 123a456a-cccc-bbbb-aaaa-a123456abcde
LSP version : lsp-rel-20250327-1959
VDB version : 404
-----
```

Getting Files from Devices

- Through the RADKit client, a Cisco TAC engineer can SSH to devices and perform various operations, including generating troubleshoot files.

Cisco Support: RADKit Console

Using RADKit Network Console

- Alternatively to using the RADKit Client, a Cisco TAC support engineer could use the RADKit Network Console. The Network Console is part of the RADKit Client.
- The RADKit Network Console is a feature that provides a simple command line interface (CLI) for basic RADKit Client functions. It is meant to be used for quick connectivity to a RADKit Service instance, establishing interactive sessions and downloading/uploading files with no hassle and minimal training.
- Start the Network Console using the command line: **radkit-network-console**
- Check the RADKit documentation for more details.

Upgrades and Backwards Compatibility

Upgrading to 7.7 and from 7.7 upwards

- RADKit Service is added in Secure Firewall 7.7.0.
 - Devices upgraded to version 7.7.0+ have the required configuration for the RADKit Service.

Experience with Unsupported FTDs

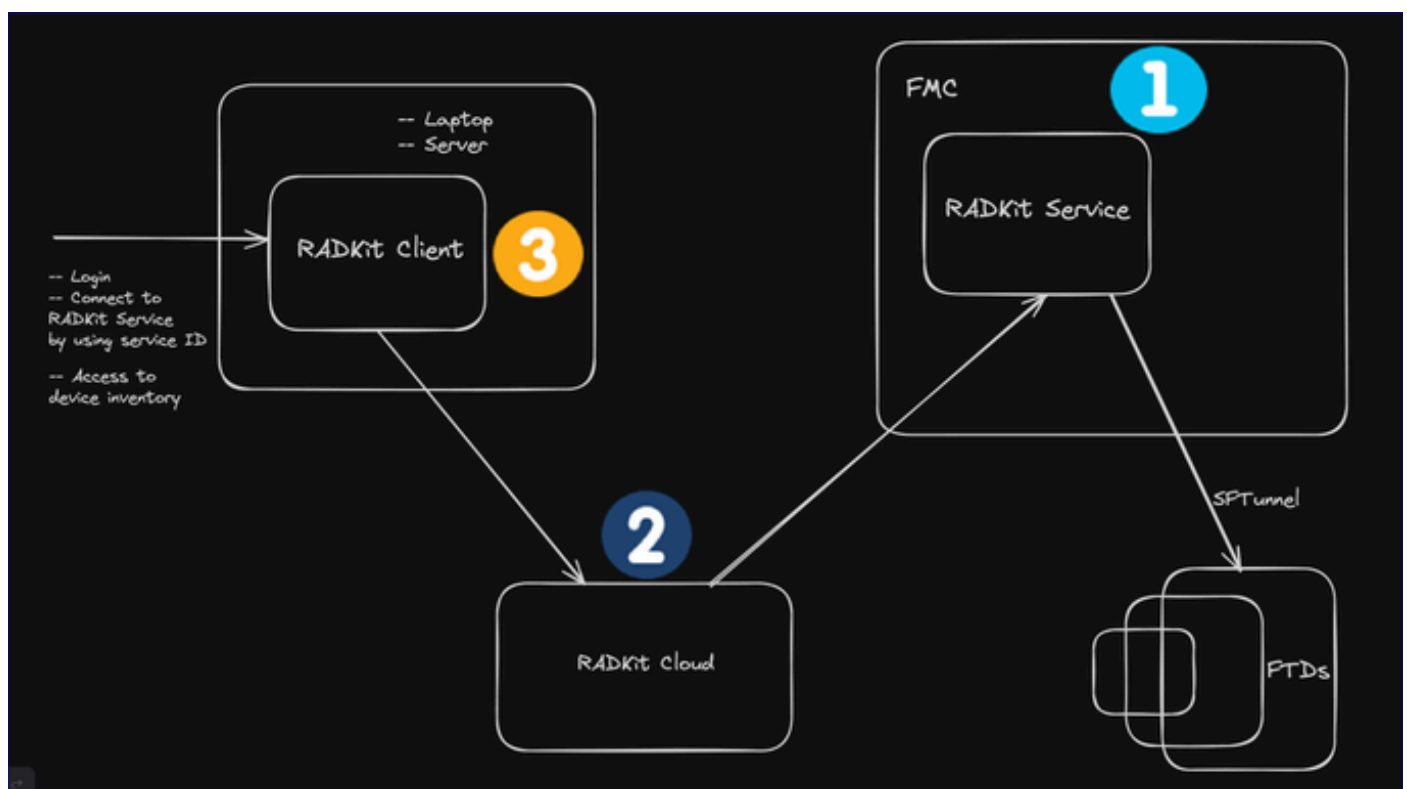
- FMCs and FTDs must have the minimum version 7.7.0 for this feature to work (FTDs with version lower than 7.7 cannot be added to a 7.7 FMC RADKit authorization).
- Registered FTDs which are not on 7.7.0 are not available for picking for enabling authorization.

Troubleshooting

Overview of Diagnostics

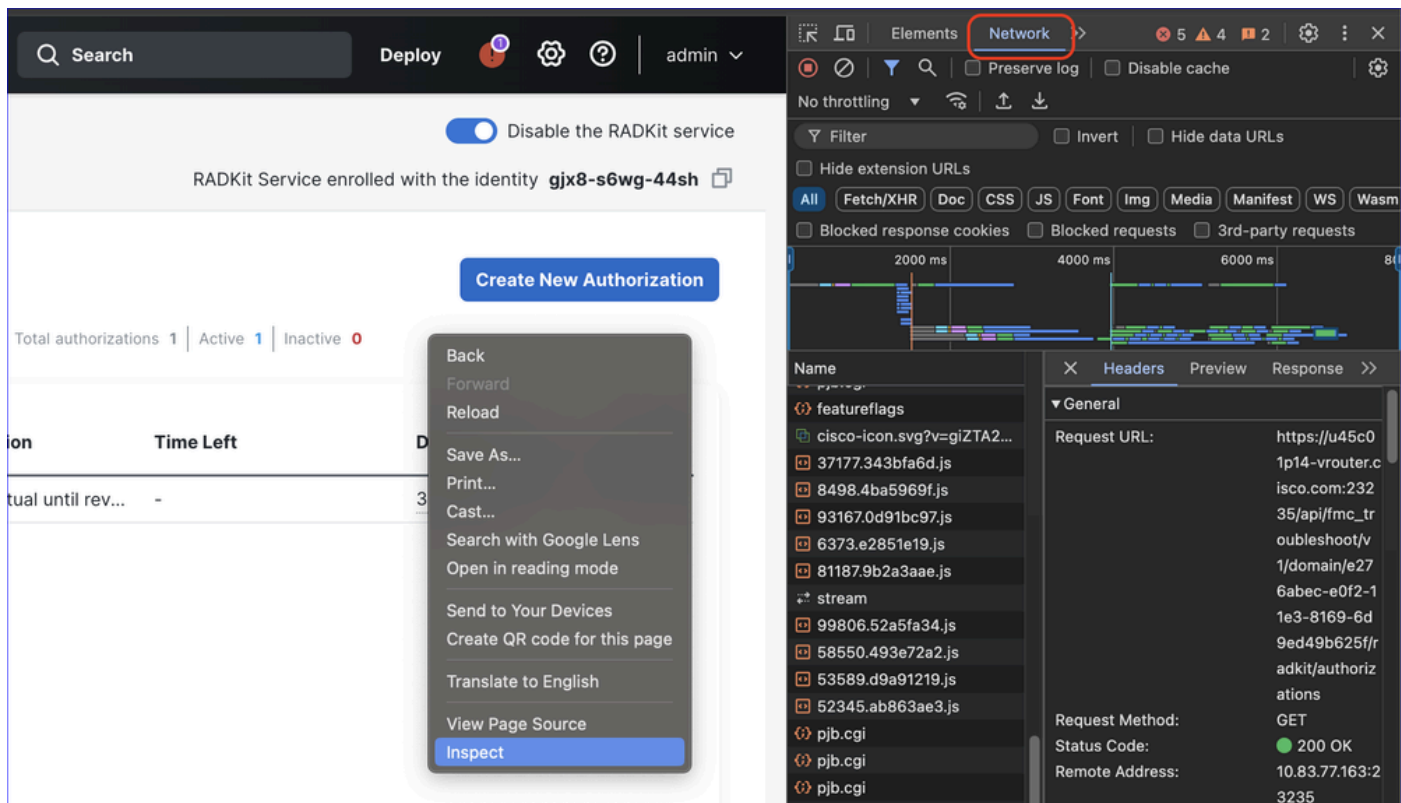
Troubleshooting Points

1. Use **Browser Development** tools and FMC logs to see what's happening in FMC.
2. For communication issues between RADKit Service on FMC, RADkit Cloud, and RADKit client, look in RADKit client logging.
3. RADKit Client.



How to Troubleshoot: Browser's Developer Tools

- The browser's **Developer Tools**, **Network** tab shows the API calls that were run on the page, this can be used when troubleshooting issues on FMC. This can be launched by right clicking on page then click on Inspect.
- Check the API call status code and response preview in the Network tab.



RADKit Service Go Middleware APIs

Go Middleware for the RADKit integration uses API calls which are not publicly available via the FMC API explorer. The Go Middleware APIs log is available in **/var/log/auth-daemon.log**. Functionality Go Middleware performs includes:

- Enroll the RADKit Service in the RADKit cloud with Single Sign On process.
- Fetch a list of all remote RADKit users' authorizations and associated devices.
- Fetch a specific remote RADKit user authorization and associated devices by using an email.
- Create a remote RADKit user authorization and grant permissions to access devices (all devices or a list of selected devices) for a specified timeframe.
- Modify a remote RADKit user authorization.
- Delete a remote RADKit user authorization.

Logs for Troubleshooting the RADKit Service

- General FMC logs: **pigtail** command from an FMC ssh session.
- Go Middleware APIs: **/var/log/auth-daemon.log**
- Logs that contain RADKit and auth-daemon processes data:

/var/log/process_stdout.log

/var/log/process_stderr.log

All these logs are included in FMC/FTD Troubleshoots.

- Internal RADKit service logs: **/var/lib/radkit/logs/service/**
- Logs for the operations performed from the RADKit Client on devices (FMCs and FTDs): **/var/lib/radkit/session_logs/service**

Logs to Submit to Cisco TAC

- Screenshots of errors.
- Description of problem.
- Steps to reproduce.
- Pigtail and **/var/log/auth-daemon.log** log extracts containing the errors.

Monitoring Access

Logging of who has been granted access for how long and who granted the access is in the FMC Audit logs.

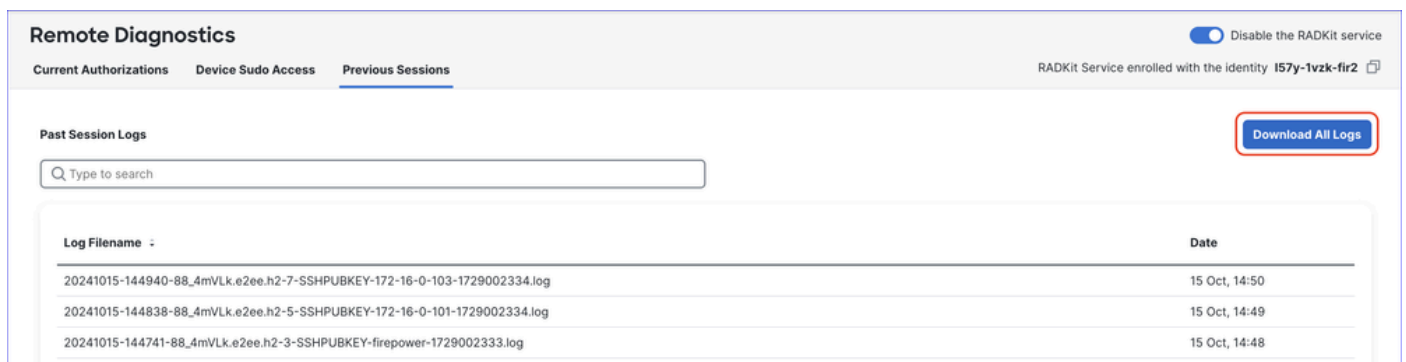
RADKit Session Logs

RADKit session logs for the operations performed from the RADKit Client on devices (FMCs and FTDs) are present on FMC at **/var/lib/radkit/session_logs/service:**

- The logs are from the RADKit service itself.
- These logs are included in a Troubleshoot bundle.
- The logs are accessible from UI as well (see the next section).
- There are multiple session log files; one per session.

RADKit Previous Sessions Logs

The RADKit sessions logs for the device operations performed from the RADKit client are available for download as an archive containing all the logs in the Previous Sessions tab by clicking on “**Download All Logs**” button.



Remote Diagnostics Disable the RADKit service

Current Authorizations Device Sudo Access **Previous Sessions** RADKit Service enrolled with the identity **i57y-1vzk-fir2**

Past Session Logs Download All Logs

Q Type to search

Log Filename	Date
20241015-144940-88_4mVlk.e2ee.h2-7-SSHPUBKEY-172-16-0-103-1729002334.log	15 Oct, 14:50
20241015-144838-88_4mVlk.e2ee.h2-5-SSHPUBKEY-172-16-0-101-1729002334.log	15 Oct, 14:49
20241015-144741-88_4mVlk.e2ee.h2-3-SSHPUBKEY-firepower-1729002333.log	15 Oct, 14:48

Sample Problem with Troubleshooting Walkthrough

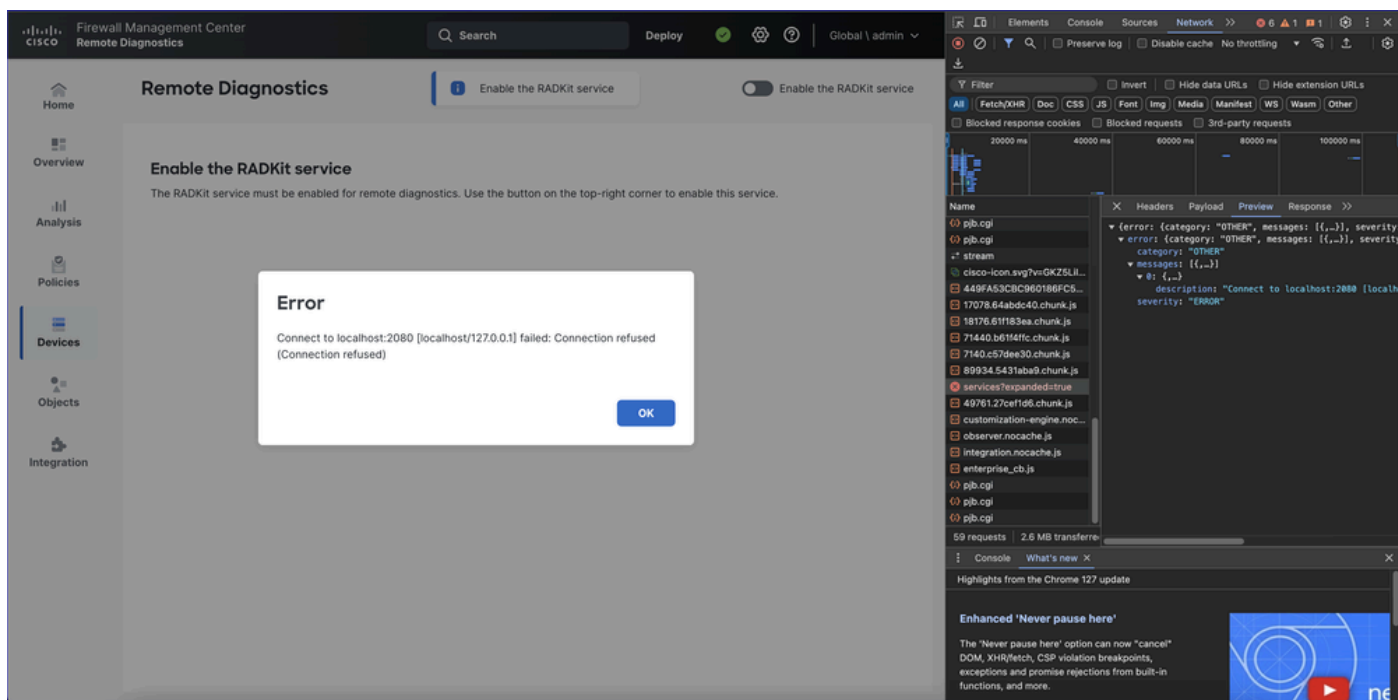
Troubleshooting Example

In case of an error like “Connect to localhost:2080 [localhost/127.0.0.1] failed: Connection refused (Connection refused)”, try restarting auth-daemon from an FMC SSH session:

```
<#root>
```

```
root@firepower:~$
```

```
sudo pmtool restartbyid auth-daemon
```

Telemetry

The telemetry output was added for this feature:

```
"remoteDiagnostics" : {
  "isRemoteDiagnosticsEnabled": 0 // 0 = false , 1 = true
}
```

FAQ

FAQs: Login and Enrollment

Q. Does the enrollment work with proxy if FMC doesn't have direct internet access?

A. Yes, if the proxy has access to prod.radkit-cloud.cisco.com which is used for the enrollment process.

Q. Can a user use their own IdP for this service?

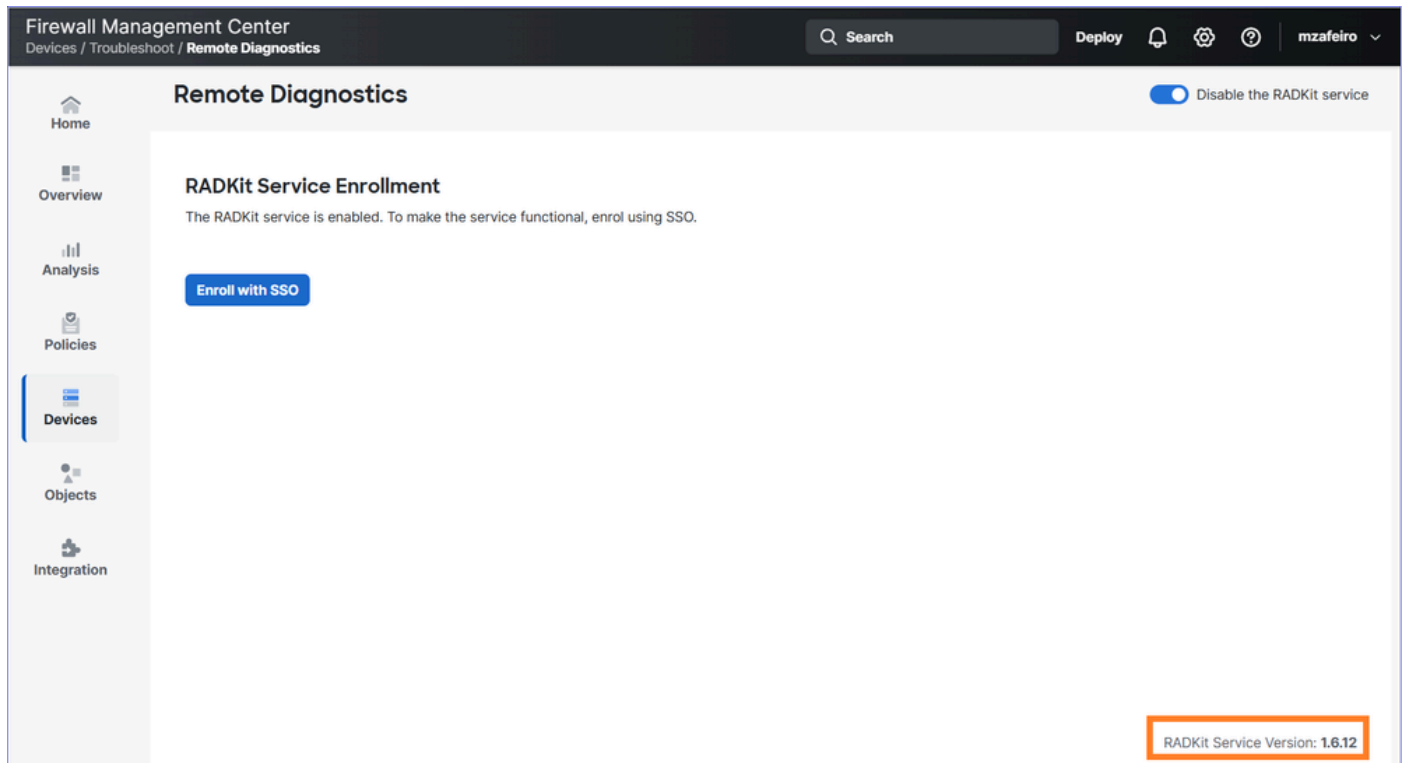
A. Only Cisco SSO is accepted on RADKit cloud. There is an option to associate your company account with a Cisco account, so that the RADKit service enrollment is possible with a non-Cisco email.

FAQs: RADKit Versions

Q. What version of RADkit is included in FMC in 7.7 release? How can we know which version of RADKit is included in FMC? Is this something that can be updated without an FMC upgrade?

A.

- The version of RADKit which comes with 7.7.0 is 1.6.12.
- The version of the RADKit service is displayed on the bottom of the FMC **Remote Diagnostics** page: "RADKit Service Version: 1.6.12."



- RADKit is bundled with FMC upgrade packages/hotfixes. Upgrading the RADKit service in FMC separately is not supported.

FAQs: Other

Q. Could external devices - not managed by the FMC – be included?

A. Only devices managed by the FMC can be added to the RADKit inventory and then can be accessed through an authorization.

Q. Is RADKit config backed up as part of FMC backup?

A.

- The configuration is not backed up as part of the FMC backup.
- It is not backed up because we anticipate that, generally, perpetual access won't be provided; the access is typically only be for a limited time.

References

Useful Links:

- [FMC Configuration Guide - RADKit](#)

- <https://radkit.cisco.com/>
- <https://radkit.cisco.com/docs/index.html>
- <https://radkit.cisco.com/downloads/release/>
- <https://github.com/Cisco-RADKit/Intro>