

Troubleshoot Actions on IMM Servers through Intersight API Requests

Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Background Information](#)

[API Requests from Intersight SaaS or Intersight Appliance Account](#)

[Overview Steps](#)

[Decommission/Recommission a Server](#)

[Unassign Server Profile](#)

[Remove Server](#)

[Reboot IMC of a Server](#)

[Troubleshoot Actions Through API Explorer in the Device Console](#)

[Reboot CIMC Management Controller of a Server](#)

[Reboot an I/O Module \(IOM\)](#)

[Related Information](#)

Introduction

This document describes API requests that can be useful at contention times when certain actions on servers cannot be performed through the UI.

Contributed by Luis Uribe Rojas, Cisco TAC Engineer and Justin Pierce, Technical Leader.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Intersight
- Unified Computing System (UCS) Servers
- Intersight Managed Mode (IMM)
- Application Programming Interface (API)

Components Used

The information in this document is based on these software and hardware versions:

- Cisco UCS 6454 Fabric Interconnect, firmware 4.2(1m)
- UCSB-B200-M5 blade server, firmware 4.2(1a)
- Intersight software as a service (SaaS)

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

Cisco Intersight provides a cloud-based RESTful API to manage Intersight-connected targets across multiple Data Centers. Cisco Intersight infrastructure services include the deployment, monitoring, management, and support for physical and virtual infrastructures.

In situations where certain actions on Intersight Managed Servers cannot be performed through the user interface (UI) of Intersight, either because options are grayed out or access to the UI is not available, API requests can be a useful alternative.


API Requests from Intersight SaaS or Intersight Appliance Account

Overview Steps


The next examples adhere to a consistent structure, although the specific parameters and values used can vary. This is a brief summary of the steps involved:

Log in to the Intersight account.

For a SaaS environment, in a browser, navigate to [API Reference](#) and log in with your account.

 **Note:** For an appliance environment, such as Intersight Connected Virtual Appliance (CVA) or Intersight Private Virtual Appliance (PVA) In a browser navigate to <https://<Appliance-Hostname>/apidocs/apirefs> and log in with the Appliance credentials.

1. Look for the API request that you need and use a GET call filtered with known field value(s), such as Serial Number, Server Profile, Server Name, Device Moid, and so on.
2. Use the **PATCH** call with the correspondent Action to perform the required task

 **Tip:** On Query Parameters, ensure to use the same exact letters for Key and Value Examples to avoid errors.

In the API Reference guide, it is useful to review the `Response Model` tab for the proper syntax and all the supported actions that can be used in the payload of a call. For example, from `/api/v1/compute/BladeIdentities/`, the supported `AdminAction` are `None`, `Decommission`, `Recommission`, `Reack`, `Remove`, and `Replace`. This model is used throughout this document.

The screenshot displays the Cisco Intersight Developer Center interface. On the left, the 'API Reference v1.0.11-11265' section shows a search for 'blade' and a list of endpoints for 'compute/BladeIdentities'. The main area shows the 'Response Model' for the PATCH endpoint `/api/v1/compute/BladeIdentities/{Moid}`. A red box highlights the `AdminAction` field, which is a string updated by UI/API to trigger specific action types. The actions listed are: 'None' (no operation), 'Decommission' (temporary removal), 'Recommission' (recommissioning), 'React' (re-acknowledgment), 'Remove' (permanent removal), and 'Replace' (replacement). Below this, the `AdminActionState` field is described as a string with states: 'Applying' (in progress), 'Applied' (completed), and 'Failed' (action failed). Other fields include `Identifier` (integer) and `Lifecycle` (string).

On the right, the 'REST Client' shows the PATCH request body with a highlighted `{Moid}` field, which is required. The 'Response Text' and 'Response Info' sections are also visible.

Decommission/Recommission a Server

In the Intersight API Reference document, look for the `compute/BladeIdentities` request, select the first GET call, and then enter the required Query Parameters.

This example uses these parameters:

Key	Value	Usage
\$filter	Serial Eq 'FLM2402001A'	To filter output to the server with the Serial Number provided.
\$select	Moid	To select the values to display from that object. Value displayed is the Server Moid.

Service: Intersight

API Reference v1.0.11-11360

compute/BladeIdentities

GET /api/v1/compute/BladeIdentities

Parameters: Response Model

\$filter (string) query

Filter criteria for the resources to return. A URI with a \$filter query option identifies a subset of the entries from the Collection of Entries. The subset is determined by selecting only the Entries that satisfy the predicate expression specified by the \$filter option. The expression language that is used in \$filter queries supports references to properties and literals. The literal values can be strings enclosed in single quotes, numbers and boolean values (true or false).

\$orderby (string) query

Determines what properties are used to sort the collection of resources.

\$stop (integer) query

Specifies the maximum number of resources to return in the response.

\$skip (integer) query

Specifies the number of resources to skip in the response.

\$select (string) query

Specifies a subset of properties to return.

\$expand (string) query

Specify additional attributes or related resources to return in addition to the primary resources.

REST Client

GET /api/v1/compute/BladeIdentities

+ Query Parameter 3

Key	Value
\$filter	Serial Eq_FLM2402001A
\$select	Moid

Send 200 Success

Response Text Response Info

```

1 {
2   "ObjectType": "compute.BladeIdentity.List",
3   "Results": [
4     {
5       "ClassId": "compute.BladeIdentity",
6       "Moid": "6424b4d66f62692d328bb7bd",
7       "ObjectType": "compute.BladeIdentity"
8     }
9   ]
10 }

```

Apply the PATCH call with the action required. This example uses:

```
{"AdminAction": "Decommission"}
```

Service: Intersight

API Reference v1.0.11-11360

compute/BladeIdentities

PATCH /api/v1/compute/BladeIdentities/{Moid}/{Moid}

Parameters: Request Model Response Model

Moid (string) path

The unique Moid identifier of a resource instance.

If-Match (string) header

For methods that apply server-side changes, and in particular for PUT, If-Match can be used to prevent the lost update problem. It can check if the modification of a resource that the user wants to upload will not override another change that has been done since the original resource was fetched. If the request cannot be fulfilled, the 412 (Precondition Failed) response is returned. When modifying a resource using POST or PUT, the If-Match header must be set to the value of the resource ModTime property after which no lost update problem should occur. For example, a client send a GET request to obtain a resource, which includes the ModTime property. The ModTime indicates the last time the resource was created or modified. The client then sends a POST or PUT request with the If-Match header set to the ModTime property of the resource as obtained in the GET request.

REST Client

PATCH /api/v1/compute/BladeIdentities/{Moid}

{Moid} *

6424b4d66f62692d328bb7bd

```

1 {"AdminAction": "Decommission"}

```


Send 200 Success

Response Text Response Info

```

1 {
2   "Moid": "6424b4d66f62692d328bb7bd",
3   "ObjectType": "compute.BladeIdentity",
4   "ClassId": "compute.BladeIdentity",
5   "CreateTime": "2023-03-20T21:59:50.321Z",
6   "ModTime": "2023-04-03T17:20:05.65772494Z",
7   "Tags": [],
8   "Owners": [
9     "5cc1f0de7564612d30190899",
10    "611184ec6f72612d33d4ff4"
11  ],
12  "SharedScope": "",
13  "AccountMoid": "5cc1f0de7564612d30190899",
14  "DomainGroupMoid": "5cc1f0de7564612d30190899",
15  "Ancestors": [],
16  "DisplayNames": {
17    "hierarchical": {
18      "chassis-1-server-2"
19    }
20  }
21 }

```

 **Tip:** If Recommission is required, use `{"AdminAction": "Recommission"}`.

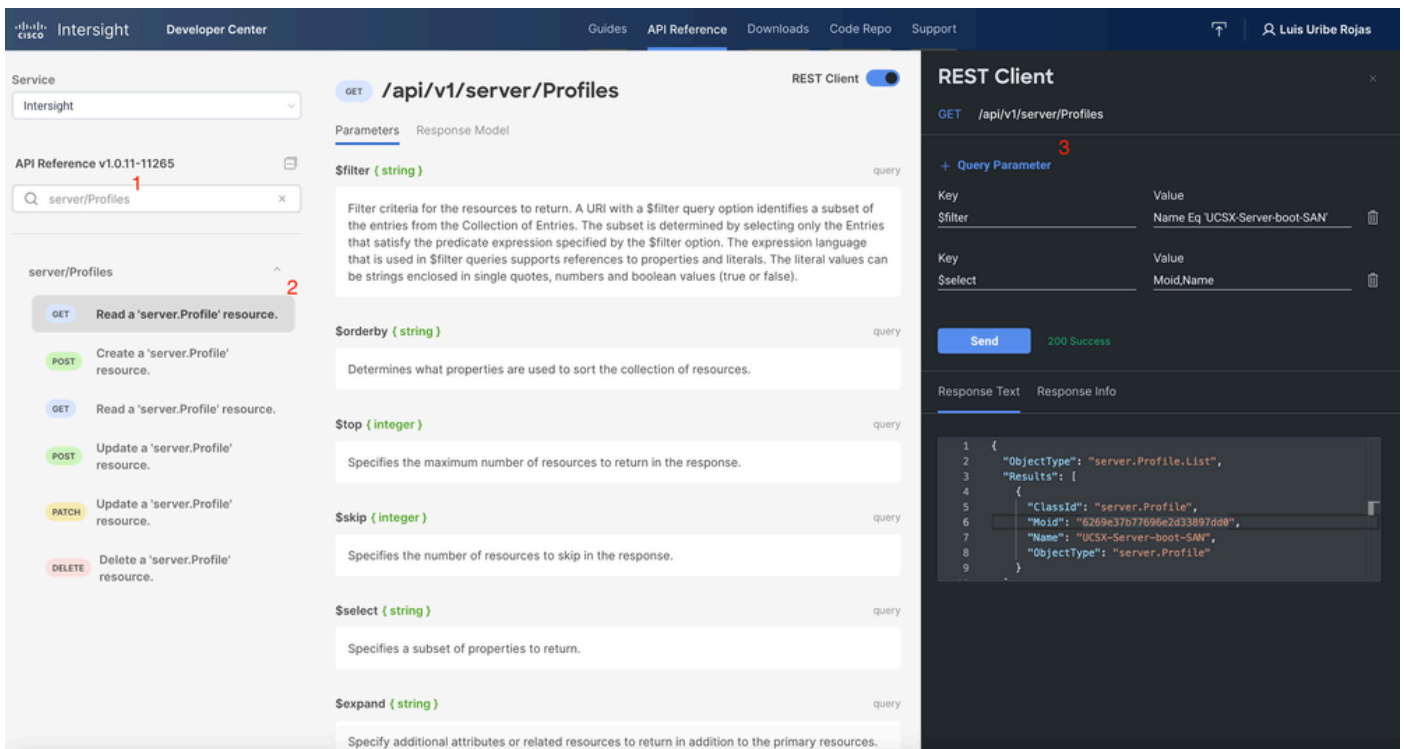
 **Tip:** Similar Actions can be issued for *Rack Integrated Servers* and *Chassis* in IMM. Look for `compute/RackUnitIdentities` and `equipment/ChassisIdentities` API calls.

Unassign Server Profile

Look for the *Server/Profiles* request and select the first **GET** call, then enter the required Query Parameters.

This example uses these parameters:

Key	Value	Usage
\$filter	Name Eq 'UCSX-Server-boot-SAN'	To filter output to server profile that has the name entered.
\$select	Moid,Name	To select the value(s) to display from that object. In this case



The screenshot shows the Cisco Intersight Developer Center interface. On the left, the API Reference for v1.0.11-11265 is displayed, with a search for 'server/Profiles' and a list of actions. The first action, 'Read a 'server.Profile' resource', is selected. The main area shows the GET endpoint `/api/v1/server/Profiles` with a REST Client toggle. The parameters section is expanded, showing the following query parameters:

- `$filter` (string): Filter criteria for the resources to return. A URI with a \$filter query option identifies a subset of the entries from the Collection of Entries. The subset is determined by selecting only the Entries that satisfy the predicate expression specified by the \$filter option. The expression language that is used in \$filter queries supports references to properties and literals. The literal values can be strings enclosed in single quotes, numbers and boolean values (true or false).
- `$orderby` (string): Determines what properties are used to sort the collection of resources.
- `$top` (integer): Specifies the maximum number of resources to return in the response.
- `$skip` (integer): Specifies the number of resources to skip in the response.
- `$select` (string): Specifies a subset of properties to return.
- `$expand` (string): Specify additional attributes or related resources to return in addition to the primary resources.

On the right, the REST Client interface shows the same endpoint and parameters. The 'Send' button has been clicked, resulting in a 200 Success response. The response text is displayed as follows:

```
1 {
2   "ObjectType": "server.Profile.List",
3   "Results": [
4     {
5       "ClassId": "server.Profile",
6       "Moid": "6269e37b77606e2d33997dd8",
7       "Name": "UCSX-Server-boot-SAN",
8       "ObjectType": "server.Profile"
9     }
10  ]
11 }
```

Apply the **PATCH** call with the action required. This example uses:

```
{"Action": "Unassign"}
```

The screenshot displays the Cisco Intersight Developer Center interface. On the left, the API Reference for the endpoint `/api/v1/server/Profiles/{Moid}` is shown, highlighting the `PATCH` method and the `If-Match` header. The `If-Match` header is described as a string used to prevent update conflicts. On the right, the REST Client shows a successful `PATCH` request to `/api/v1/server/Profiles/{Moid}` with a status of `200 Success`. The response body is a JSON object representing a server profile, including fields like `Moid`, `ObjectType`, `ClassId`, `CreateTime`, `ModTime`, `Tags`, `Owners`, `SharedScope`, `AccountMoid`, `DomainGroupMoid`, `Ancestors`, `DisplayNames`, and `hierarchical`.

Remove Server

In the Intersight API Reference document, look for *compute/BladeIdentities* request and select the first `GET` call, then enter the required Query Parameters.

This example uses these parameters:

Key	Value	Usage
\$filter	Serial Eq 'FLM2402001A'	To filter output to only server with Serial Number provided.
\$select	Moid	To select the values to display from that object. Value displayed is the Server Moid.

The screenshot displays the Intersight Developer Center API Reference for the endpoint `GET /api/v1/compute/BladeIdentities`. The left sidebar shows the navigation tree with `compute/BladeIdentities` selected. The main content area shows the endpoint details, including parameters like `$filter` (string), `Sorderby` (string), `Stop` (integer), and `Sskip` (integer). The REST Client panel on the right shows a successful 200 response with a JSON body:

```

1 {
2   "ObjectType": "compute.BladeIdentity.List",
3   "Results": [
4     {
5       "ClassId": "compute.BladeIdentity",
6       "Moid": "63c88bf66f62692d325b9643",
7       "ObjectType": "compute.BladeIdentity"
8     }
9   ]
10 }

```

Apply the **PATCH** call with the action required. This example uses:

```
{ "AdminAction": "Remove" }
```

Warning: This request results in the removal of the server from the Inventory. To add the server back into the inventory of the domain, a new discovery is required. This can be triggered through a physical reset of the server or by a chassis rediscovery task.

The screenshot displays the Intersight Developer Center API Reference for the endpoint `PATCH /api/v1/compute/BladeIdentities/{Moid}/{Moid}`. The left sidebar shows the navigation tree with `compute/BladeIdentities` selected. The main content area shows the endpoint details, including parameters like `Moid` (string) and `If-Match` (string). The REST Client panel on the right shows a successful 200 response with a JSON body:

```

1 { "AdminAction": "Remove" }
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

```

Reboot IMC of a Server

In the Intersight API Reference document, look for the `compute/ServerSettings` request, select the first GET call, and then enter the required Query Parameters.

This example uses these parameters:

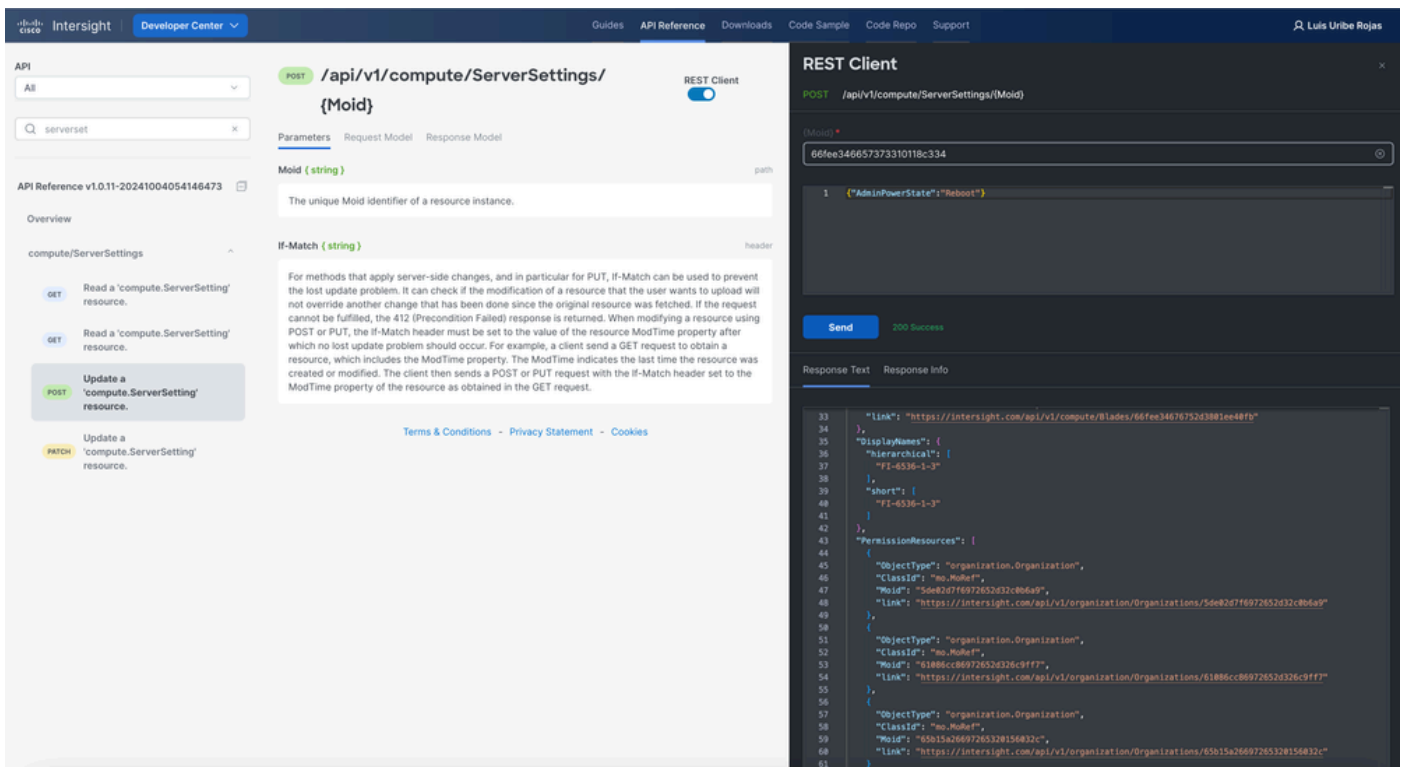
Key	Value	Usage
\$filter	Name Eq 'FI-6536-1-3'	To filter output to Blade that has the name entered.
\$select	Moid,Name	To select the value(s) to display from that object.

The screenshot displays the Intersight API Reference and REST Client interface. On the left, the API Reference shows the endpoint `/api/v1/compute/ServerSettings` with various query parameters listed, including `$filter`, `$orderby`, `$top`, `$skip`, `$select`, `$expand`, and `$apply`. The REST Client on the right shows the same endpoint with the query parameters `$filter` and `$select` entered, and a successful 200 response returned.

Apply the **PATCH** call with the action required. This example uses:

```
{"AdminPowerState": "Reboot"}
```

Warning: This request results in Reboot of the IMC traffic for the server in question, it is expected to see a Shallow discovery workflow after Management Controller is rebooted.



Troubleshoot Actions Through API Explorer in the Device Console

The Device Console allows you to monitor the health of your devices, and the status of their connection to Intersight. You can generate Tech Support bundles that contain diagnostic information to troubleshoot and analyze issues. In addition, the device console includes the ability to launch the API Explorer to perform Redfish™ based operations on servers.

In the event that connectivity is lost between Intersight or the Appliance, the API Explorer in the Device Console can also be used to perform some basic troubleshoot actions.

1) Open the Device console, navigate to one of the Fabric Interconnect IP addresses, and select the Inventory tab.

2) Navigate to the specific device that needs to be troubleshoot, select the three dots to the right of it and select Launch API Explorer. The API Explorer is launched only for that device, and no others.

Reboot CIMC Management Controller of a Server

Launch the API Explorer for the Server:

Name	Health	Status	PID	Serial	User Label
F340-24-21-IMM-1-1-3	Healthy	Active	UCSB-B480-M5	FLM224403QB	qam-imm-1
F340-24-21-IMM-1-1-5	Healthy	Active	UCSB-B200-M5	FCH21427BPW	imm-test-user-laber-1
F340-24-21-IMM-1-1-6	Healthy	Active	UCSB-B200-M5	FLM23390B0R	-
F340-24-21-IMM-1-2-1	Healthy	Decommissioned	UCSB-B200-M6	FCH24387E7G	-
F340-24-21-IMM-1-2-5	Healthy	Active	UCSB-B200-M5	FCH22457G25	-
F340-24-21-IMM-1-3-1	Healthy	Active	UCSX-210C-M6	FCH251372LZ	-
F340-24-21-IMM-1-3-3	Healthy	Active	UCSX-210C-M6	FCH244572F0	-

Type **CIMC** in {ManagerID} and apply a **POST** *Managers/{ManagerId}/Actions/Manager.Reset* call and add the type of reset.

This example uses:

```
{"ResetType": "ForceRestart"}
```

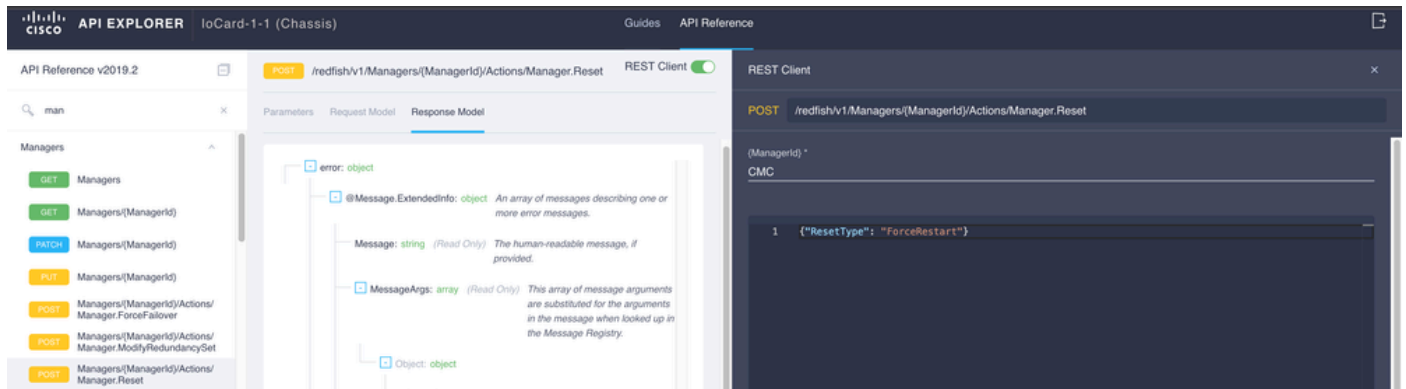
Reboot an I/O Module (IOM)

Launch the API Explorer of the IOM:

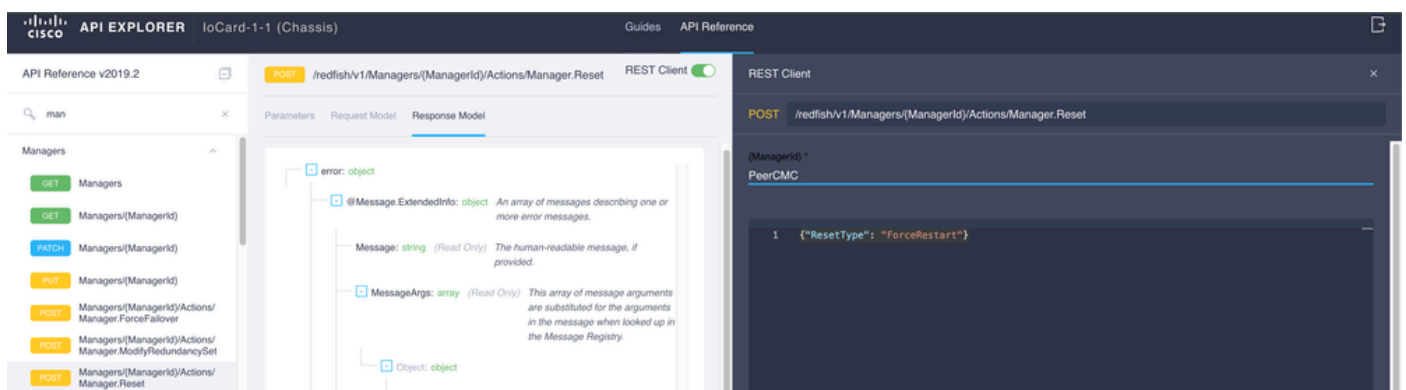
Name	ID	Status	Model	Serial
F340-24-21-IMM-1-1	chassis-1	Active	UCSB-5108-AC2	FOX1813GDWM
F340-24-21-IMM-1-2	chassis-2	Active	N20-C6508	FOX1611HC
F340-24-21-IMM-1-3	chassis-3	Active	UCSX-9508	FOX2503P

Type **CMC** in {ManagerID} and apply a **POST** *Managers/{ManagerId}/Actions/Manager.Reset* call and add the Reset Type. This example uses:

```
{"ResetType": "ForceRestart"}
```



To reboot a peer IOM in the {ManagerID} field, enter **PeerCMC** with the same call as before.



Related Information

[Intersight API Overview](#)

[Device Console Overview](#)