Python API

[Python API について（About the Python API）]

Python は簡単に習得できる、強力なプログラミング言語です。効率的で高水準なデータ構造を持ち、オブジェクト指向プログラミングに対してシンプルで効果的なアプローチを取っています。Python の構文と動的な型指定は、インタープリタ型という特長と相まって、スクリプティングと高速アプリケーション開発のための理想的な言語にしています。Python の Web サイト、www.python.org には、サードパーティが無償で提供している多数の Python モジュール、プログラム、ツールのディストリビューションとそれらへのリンク、さらに追加のドキュメンテーションが掲載されています。

Python インタープリタは、標準および Cisco Python モジュールとともに Cisco MDS NX-OS コマンドラインインターフェイス（CLI）で使用できます。インタラクティブモードと非インタラクティブ（スクリプト）モードの両方がサポートされています。これにより、繰り返しタスクを実行するよう MDS デバイスをプログラムで制御できます。これを活用できる例としては、PowerOn Auto Provisioning（POAP）スクリプトと Embedded Event Manager（EEM）アクションがあります。また、SAN 分析機能のオーバーレイ CLI で分析データをフォーマットするためにも使用されます。

サポートされるバージョン

表 1 : MDS プラットフォームの Python バージョン履歴 （2 ページ） に、Cisco MDS スイッチでの Python のマイルストーンを示します。各プラットフォームでサポートされているバージョンとデフォルトのバージョンがいつ変更されたかが表示されます。

表 *1: MDS* プラットフォームの *Python* バージョン履歴

リリース	Cisco NX-OS MDS プラットフォーム							
	9132T	**9148S**	**9148T**	**9220i**	**9250i**	**9396S**	**9396T**	**9700**
6.2(29)	—	—	—	—	—	—	—	2.7
8.3(1)	2.7.8	—	2.7.8	—	—	—	2.7.8	2.7.8
8.4（2）	2.7.8* 3.7.3	—	2.7.8* 3.7.3	—	—	—	2.7.8* 3.7.3	2.7.8* 3.7.3
8.5(1)	2.7.8* 3.7.3	—	2.7.8* 3.7.3	2.7.8* 3.7.3	—	—	2.7.8* 3.7.3	2.7.8* 3.7.3
9.2(1)	2.7.8* 3.7.3	—	2.7.8* 3.7.3	2.7.8* 3.7.3	—	—	2.7.8* 3.7.3	2.7.8* 3.7.3
9.2（2）	3.7.3*	—	3.7.3*	3.7.3*	—	—	3.7.3*	3.7.3*

* はデフォルトの Python バージョンを示しています。

（注） Python API は、Cisco MDS 9148S、Cisco MDS 9250i、Cisco MDS 9396S などの Cisco MDS 16 Gbps ファブリック スイッチではサポートされていません。

Python の使用

ここでは、Python スクリプトの作成と実行の方法について説明します。

Cisco Python パッケージ

Cisco MDS NX-OS は、インターフェイス、VLAN、ルートなど、多くのコア ネットワーク デバイス モジュールへのアクセスを可能にする Cisco Python パッケージを提供します。**help()** コマンドを入力すると、Cisco Python パッケージの詳細を表示できます。モジュール内のクラスとメソッドに関する追加情報を取得するには、特定のモジュールに対して help コマンドを実行します。たとえば、**help**（*cisco.interface*） は、cisco.interface モジュールのプロパティを表示します。

次の例は、Cisco python パッケージに関する情報を表示する方法を示します。

```
>>> import cisco
>>> help(cisco)
Help on package cisco:

NAME
    cisco
```

```
DESCRIPTION
    ################################################################################
    #
    #      File:   cli.py
    #      Name:
    #
    #      Description:
    #
    # Copyright (c) 2015-2017, 2019-2020 by cisco Systems, Inc.
    # All rights reserved.
    #
    ################################################################################

PACKAGE CONTENTS
    acl
    bgp
    buffer_depth_monitor
    check_port_discards
    cisco_secret
    feature
    history
    interface
    ipaddress
    key
    line_parser
    mac_address_table
    nxcli
    ospf
    routemap
    routes
    section_parser
    ssh
    system
    tacacs
    transfer
    vlan
    vrf

CLASSES
    builtins.object
```

CLI コマンド API の使用

Python プログラミング言語は、CLI コマンドを実行できる 3 つの API を使用します。API は Python CLI モジュールから利用できます。

これらの API については、次の表で説明します。**from cli import *** コマンドを使用して API を有効にする必要があります。これらの API の引数は CLI コマンドの文字列です。Python インタープリタ経由で CLI コマンドを実行するには、次の API のいずれかの引数文字列として CLI コマンドを入力します。

表 *2: CLI* コマンド *API*

API	説明
cli() 例： `string = cli ("cli-command")`	制御文字/特殊文字を含む CLI コマンドの未処理の出力を返します。 （注） インタラクティブな Python インタープリタは、制御文字/特殊文字を「エスケープ」して出力します。改行は「\n」として出力されるため、結果が読みにくくなる場合があります。**clip()** API は、判読性が高い結果を出力します。
clid() 例： `json_string = clid ("cli-command")`	XML をサポートする CLI コマンドの場合、この API は JSON 出力を返します。 （注） XML が使用されていない場合は、例外がスローされます。 この API は、show コマンドの出力の検索時に使用すると便利な場合があります。

clip()
例：
`clip ("cli-command")`
	CLI コマンドの出力を直接 stdout に出力し、Python には何も返されません。
	（注）
	`clip ("cli-command")`
	と同等です（is equivalent to）
	`r=cli("cli-command")`
	`print r`

2 つ以上のコマンドを個別に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続しません。

次の例では、最初のコマンドの状態が 2 番目のコマンドで持続しないため、2 番目のコマンドが失敗します。

```
>>> cli("conf t")
>>> cli("interface fc4/1")
```

2 つ以上のコマンドを同時に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続します。

次の例では、2 番目と 3 番目のコマンドの状態が持続するため、2 番目のコマンドは成功しています。

```
>>> cli("conf t ; interface fc4/1 ; shut")
```

（注） 例に示すように、コマンドは「;」で区切られます。（; は、単一のブランク文字で囲む必要があります。）

CLI からの Python インタープリタの呼び出し

次に、CLI から Python を呼び出す方法の例を示します：

（注）
- Python インタープリタのプロンプトは「>>>」または「…」で表示されます。
- Cisco MDS NX-OS リリース 7.3(x) 以降のリリースでは、Cisco MDS 9700 シリーズ スイッチの特権 EXEC モードでのみ Python インタープリタを呼び出すことができます。
- Cisco MDS NX-OS リリース 9.2(2) 以降では、python コマンドは Python 3.0 を実行します。

次に、CLI から Python2 .7.5 を呼び出す方法の例を示します：

```
switch# python

Python 2.7.5 (default, Jun  3 2016, 03:57:06)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from cli import *
>>> cli("show clock")
'Time source is NTP\n06:32:20.023 UTC Tue Jan 31 2017\n'
>>> exit()
```

次に、CLI から Python3 を呼び出す方法の例を示します：

```
switch# python3

Python 3.7.3 (default, Aug 26 2019, 23:20:10)
[GCC 4.6.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cli import *
>>> cli("show clock")
'Time source is NTP\n07:46:50.923 UTC Tue Apr 28 2020\n'
```

表示フォーマット

次に、Python API を使用したさまざまな表示フォーマットを示します：

例 1：

```
>>> from cli import *
>>> cli("conf ; interface fc1/1")
''
clip('where detail')
  mode:
  username:            admin
```

```
>>>
```

例 2 :

```
>>> from cli import *
>>> cli("conf ; interface fc1/1")
''
>>> cli('where detail')
'  mode:                \n  username:            admin\n'
>>>
```

例 3 :

```
>>> from cli import *
>>> cli("conf ; interface fc1/1")
''
>>> r = cli('where detail') ; print(r)
  mode:
  username:            admin

>>>
```

例 4 :

```
>>> from cli import *
>>> import json
>>> out=json.loads(clid('show version'))
>>> for k in out.keys():
...    print("%30s = %s" % (k, out[k]))
...
                    header_str = Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Documents: http://www.cisco.com/en/US/products/ps9372/tsd_products_support_series_home.html
Copyright (c) 2002-2022, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under
license. Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php
                   bios_ver_str = 2.12.0
              kickstart_ver_str = 9.2(2)
                    sys_ver_str = 9.2(2)
                bios_cmpl_time = 05/26/2021
                kick_file_name = bootflash:///m9700-sf4ek9-kickstart-mz.9.2.2.bin
                kick_cmpl_time = 1/1/2022 12:00:00
                   kick_tmstmp = 01/20/2022 22:42:00
                isan_file_name = bootflash:///m9700-sf4ek9-mz.9.2.2.bin
                isan_cmpl_time = 1/1/2022 12:00:00
                   isan_tmstmp = 01/21/2022 00:12:45
                    chassis_id = MDS 9706 (6 Slot) Chassis
                     module_id = Supervisor Module-4
                      cpu_name = Intel(R) Xeon(R) CPU D-1548 @ 2.00GHz
                        memory = 14146356
                      mem_type = kB
                  proc_board_id = JAE22320AXJ
                     host_name = sw184-9706
                bootflash_size = 3932160
                    slot0_size = 0
                kern_uptm_days = 5
```

```
                 kern_uptm_hrs = 21
                kern_uptm_mins = 39
                kern_uptm_secs = 27
                      rr_usecs = 699796
                      rr_ctime = Tue Jan 25 10:40:02 2022
                     rr_reason = Reset Requested by CLI command reload
                    rr_sys_ver = 9.2(2)
                    rr_service = None
                  manufacturer = Cisco Systems, Inc.
>>>
```

非インタラクティブ Python

Python スクリプト名を引数として Python CLI コマンドで使用することで、Python スクリプトを非インタラクティブ モードで実行できます。Python スクリプトは、ブートフラッシュまたは揮発性スキームの下に配置する必要があります。Python CLI コマンドでは、Python スクリプトで最大 32 のコマンド ライン引数を使用できます。

Cisco MDS 9000 デバイスは、Python スクリプトを実行するためのソース CLI コマンドもサポートしています。`bootflash:scripts` ディレクトリは、ソース CLI コマンドのデフォルトのスクリプト ディレクトリです。

（注） Python3 インタープリタを使用して Python スクリプトを実行するには、スクリプトの最初の行に #!/isan/bin/python3 または #!/usr/bin/env python3 のような python3 文字列が含まれていることを確認します。

次に、スクリプトとそれを実行する方法の例を示します。

```
switch# show file bootflash:flashCheck.py
#!/bin/env python
import re
import json
import cli
import syslog

threshold = 60
ignore_paths = ['bootflash']

allmodules = json.loads(cli.clid("show module"))['TABLE_modinfo']['ROW_modinfo']
if type(allmodules) is dict:
        allmodules = [allmodules]
for eachmodule in allmodules:
        mod = eachmodule['mod']
       modtype = eachmodule['modtype']
        cmd = "slot " + str(mod) + " show system internal flash"
        if 'Supervisor' in modtype:
                s = "Supervisor(Module " + str(mod) + ")"
                regex_to_match =
r'(?P<mnton>\S+)\s+(?P<onekblks>\d+)\s+(?P<used>\d+)\s+(?P<avail>\d+)\s+(?P<useper>\d+)\s+(?P<fs>\S+)'

        elif 'Sup' in modtype:
            cmd = " show system internal flash"
            s = "Sup and LC - Module 1"
            regex_to_match =
r'(?P<mnton>\S+)\s+(?P<onekblks>\d+)\s+(?P<used>\d+)\s+(?P<avail>\d+)\s+(?P<useper>\d+)\s+(?P<fs>\S+)'
```

```
        else:
                s = "Module " + str(mod)
                regex_to_match =
r'(?P<fs>\S+)\s+(?P<onekblks>\d+)\s+(?P<used>\d+)\s+(?P<avail>\d+)\s+(?P<useper>\d+)%\s+(?P<mnton>\S+)'


        out = cli.cli(cmd)
        alllines = out.splitlines()
        for eachline in alllines:
                match = re.search(regex_to_match,eachline)
                if match:
                        grps = match.groupdict()
                        #print(grps)
                       sysstr = "Flash usage exceeds threshold({}%) on {} - Filesystem:
 {}, Mounted-On: {}, Usage:
{}%".format(str(threshold),s,grps['fs'],grps['mnton'],grps['useper'])
                        for each_ignore_path in ignore_paths:
                                if each_ignore_path in grps['mnton']:
                                        break
                                else:
                                        if int(grps['useper']) > threshold:
                                                syslog.syslog(2,sysstr)
```

Embedded Event Manager でのスクリプトの実行

Cisco MDS 9000 デバイスのEmbedded Event Manager（EEM：組み込みイベント マネージャ）のポリシーは、Python スクリプトをサポートします。

次の例は、EEM アクションとして Python スクリプトを実行する方法を示しています。

- アクション コマンドを使用することで、EEM アプレットに Python スクリプトを含めることができます。

```
switch# show running-config eem

!Command: show running-config eem
!Time: Sun May  1 14:40:07 2011

version 6.1(2)I2(1)
event manager applet a1
  event cli match "show clock"
  action 1 cli python bootflash:pydate.py
  action 2 event-default
```

- **show file** *logflash*:*event_archive_1* コマンドを実行して、ログ ファイル内のイベントによってトリガーされたアクションを検索できます。

```
switch# show file logflash:event_archive_1 | last 33

eem_event_time:05/01/2011,19:40:28 event_type:cli event_id:8 slot:active(1)
vdc:1 severity:minor applets:a1
eem_param_info:command = "exshow clock"
Starting with policy a1
Python

2011-05-01 19:40:28.644891
Executing the following commands succeeded:
          python bootflash:pydate.py
```

```
PC_VSH_CMD_TLV(7679) with q
```

Cisco MDS NX-OS のセキュリティと Python

Cisco MDS NX-OS 情報技術は、ソフトウェアの Cisco MDS NX-OS サンドボックス レイヤおよび CLI ロールベース アクセス コントロール（RBAC）によって保護されます。

Cisco MDS NX-OS network-admin または dev-ops ロールに関連付けられているすべてのユーザーは、特権ユーザーです。カスタム ロールで Python へのアクセスが許可されているユーザーは、非特権ユーザーと見なされます。非特権ユーザーは、ファイル システム、ゲスト シェル、Bash コマンドなどの Cisco MDS NX-OS 情報技術へのアクセスが制限されています。特権ユーザーは、Cisco MDS NX-OS のすべての情報技術へのより大きなアクセス権を持ちます。

セキュリティとユーザー権限の例

次の例は、特権ユーザーがコマンドを実行する方法を示しています：

```
switch# python
Python 2.7.5 (default, Oct  8 2013, 23:59:43)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('whoami')
admin
0
>>> f=open('/tmp/test','w')
>>> f.write('hello from python')
>>> f.close()
>>> r=open('/tmp/test','r')
>>> print r.read()
hello from python
>>> r.close()
```

次の例は、アクセスを拒否されている非特権ユーザーを示しています：

```
switch# python
Python 2.7.5 (default, Oct  8 2013, 23:59:43)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('whoami')
system(whoami): rejected!
-1
>>> f=open('/tmp/test','r')
Permission denied. Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IOError: [Errno 13] Permission denied: '/tmp/test'
>>>
```

RBAC は、ログイン ユーザー権限に基づいて CLI アクセスを制御します。ログイン ユーザーの ID は、CLI シェルまたは Bash から呼び出される Python に与えられます。Python は、Python から呼び出されたサブプロセスにログイン ユーザーの ID を渡します。

以下は、特権ユーザーの例です：

```
>>> from cli import *
>>> cli('show clock')
'11:28:53.845 AM UTC Sun May 08 2011\n'
>>> cli('configure terminal ; vrf context myvrf')
''
>>> clip('show running-config l3vm')

!Command: show running-config l3vm
!Time: Sun May  8 11:29:40 2011

version 6.1(2)I2(1)

interface Ethernet1/48
  vrf member blue

interface mgmt0
  vrf member management
vrf context blue
vrf context management
vrf context myvrf
```

以下は、非特権ユーザーの例です：

```
>>> from cli import *
>>> cli('show clock')
'11:18:47.482 AM UTC Sun May 08 2011\n'
>>> cli('configure terminal ; vrf context myvrf2')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/isan/python/scripts/cli.py", line 20, in cli
    raise cmd_exec_error(msg)
errors.cmd_exec_error: '% Permission denied for the role\n\nCmd exec error.\n'
```

次の例は、RBAC 構成を示しています：

```
switch# show user-account
user:admin
        this user account has no expiry date
        roles:network-admin
user:pyuser
        this user account has no expiry date
        roles:network-operator python-role
switch# show role name python-role
```

スケジューラでスクリプトを実行する例

次の例は、スケジューラ機能を使用してスクリプトを実行する Python スクリプトを示しています。

```
#!/bin/env python
from cli import *
from nxos import *
import os

switchname = cli("show switchname")
try:
    user = os.environ['USER']
except:
    user = "No user"
    pass
```

```
msg = user + " ran " + __file__ + " on : " + switchname
print msg
py_syslog(1, msg)
# Save this script in bootflash:///scripts
```

```
switch# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# feature scheduler
switch(config)# scheduler job name testplan
switch(config-job)# python bootflash:///scripts/testplan.py
switch(config-job)# exit
switch(config)# scheduler schedule name testplan
switch(config-schedule)# job name testplan
switch(config-schedule)# time start now repeat 0:0:4
Schedule starts from Mon Mar 14 16:40:03 2011
switch(config-schedule)# end
switch# term mon
2011 Mar 14 16:38:03 switch %VSHD-5-VSHD_SYSLOG_CONFIG_I: Configured from vty by admin
on 10.19.68.246@pts/2
switch# show scheduler schedule
Schedule Name       : testplan
------------------------------
User Name           : admin
Schedule Type       : Run every 0 Days 0 Hrs 4 Mins
Start Time          : Mon Mar 14 16:40:03 2011
Last Execution Time : Yet to be executed
-----------------------------------------------
     Job Name            Last Execution Status
-----------------------------------------------
    testplan                              -NA-
==============================================================================
switch#
switch# 2011 Mar 14 16:40:04 switch  %USER-1-SYSTEM_MSG: No user ran
/bootflash/scripts/testplan.py on : switch   - nxpython
2011 Mar 14 16:44:04 switch last message repeated 1 time
switch#
```

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。