Python API

Python プログラマビリティは、Python API をサポートしています。

Python の使用

Cisco Python モジュール

シスコが提供する Python モジュールでは、EXEC および設定コマンドを実行するアクセス権が提供されます。 **help()** コマンドを入力すると、Cisco Python モジュールの詳細が表示されます。 **help()** コマンドは Cisco CLI モジュールのプロパティを表示します。

次の例は、Cisco Python モジュールに関する情報を示します。

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> >>> from cli import cli,clip,configure,configurep, execute, executep
>>> help(configure)
Help on function configure in module cli:

configure(configuration)
Apply a configuration (set of Cisco IOS CLI config-mode commands) to the device
and return a list of results.

configuration = '''interface gigabitEthernet 0/0
no shutdown'''

# push it through the Cisco IOS CLI.
try:
results = cli.configure(configuration)
print "Success!"
except CLIConfigurationError as e:
print "Failed configurations:"
for failure in e.failed:
print failure
```

```
Args:
configuration (str or iterable): Configuration commands, separated by newlines.

Returns:
list(ConfigResult): A list of results, one for each line.

Raises:
CLISyntaxError: If there is a syntax error in the configuration.
```

```
>>> help(configurep)
Help on function configurep in module cli:

configurep(configuration)
Apply a configuration (set of Cisco IOS CLI config-mode commands) to the device
and prints the result.

configuration = '''interface gigabitEthernet 0/0
no shutdown'''

# push it through the Cisco IOS CLI.
configurep(configuration)

Args:
configuration (str or iterable): Configuration commands, separated by newlines.
```

```
>>> help(execute)
Help on function execute in module cli:

execute(command)
Execute Cisco IOS CLI exec-mode command and return the result.

command_output = execute("show version")

Args:
command (str): The exec-mode command to run.

Returns:
str: The output of the command.

Raises:
CLISyntaxError: If there is a syntax error in the command.
```

```
>>> help(executep)
Help on function executep in module cli:

executep(command)
Execute Cisco IOS CLI exec-mode command and print the result.

executep("show version")

Args:
command (str): The exec-mode command to run.
```

```
>>> help(cli)
Help on function cli in module cli:

cli(command)
    Execute Cisco IOS CLI command(s) and return the result.

    A single command or a delimited batch of commands may be run. The
    delimiter is a space and a semicolon, " ;". Configuration commands must be
    in fully qualified form.
```

```
    output = cli("show version")
    output = cli("show version ; show ip interface brief")
    output = cli("configure terminal ; interface gigabitEthernet 0/0 ; no shutdown")

    Args:
        command (str): The exec or config CLI command(s) to be run.

    Returns:
        string: CLI output for show commands and an empty string for
            configuration commands.

    Raises:
        errors.cli_syntax_error: if the command is not valid.
        errors.cli_exec_error: if the execution of command is not successful.
```

```
>>> help(clip)
Help on function clip in module cli:

clip(command)
    Execute Cisco IOS CLI command(s) and print the result.

    A single command or a delimited batch of commands may be run. The
    delimiter is a space and a semicolon, " ;". Configuration commands must be
    in fully qualified form.

    clip("show version")
    clip("show version ; show ip interface brief")
    clip("configure terminal ; interface gigabitEthernet 0/0 ; no shutdown")

    Args:
        command (str): The exec or config CLI command(s) to be run.
```

IOS CLI コマンドを実行するための Cisco Python モジュール

（注） Python を実行するには、ゲスト シェルが有効である必要があります。詳細については、「ゲスト シェル」の章を参照してください。

Python プログラミング言語は CLI コマンドを実行できる 6 つの関数を使用します。これらの関数は、Python CLI モジュールから利用できます。これらの関数を使用するには、**import cli** コマンドを実行します。これらの関数が機能するには、**ip http server** コマンドが有効になっている必要があります。

これらの関数の引数は 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 6 つの関数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。

- **cli.cli(command)**：この関数は IOS コマンドを引数として取り、IOS パーサーからコマンドを実行し、結果のテキストを返します。このコマンドの形式が正しくない場合、Python の例外が発生します。次に、**cli.cli(command)** 関数の出力例を示します。

```
>>> import cli
>>> cli.clip('configure terminal; interface loopback 10; ip address
```

```
 10.10.10.10 255.255.255.255')
*Mar 13 18:39:48.518: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback10,
changed state to up
>>> cli.clip('show clock')
'\n*18:11:53.989 UTC Mon Mar 13 2017\n'
>>> output=cli.cli('show clock')
>>> print(output)
*18:12:04.705 UTC Mon Mar 13 2017
```

- **cli.clip(command)**：この関数は **cli.cli(command)** 関数と機能はまったく同じです。ただし結果のテキストを（返すのではなく）stdout に出力する点が異なります。次に、**cli.clip(command)** 関数の出力例を示します。

```
>>> cli
>>> cli.clip('configure terminal; interface loopback 11; ip address
 10.11.11.11 255.255.255.255')
*Mar 13 18:42:35.954: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback11,
changed state to up
*Mar 13 18:42:35.954: %LINK-3-UPDOWN: Interface Loopback11, changed state to up
>>> cli.clip('show clock')
*18:13:35.313 UTC Mon Mar 13 2017
>>> output=cli.clip('show clock')
*18:19:26.824 UTC Mon Mar 13 2017
>>> print (output)
None
```

- **cli.execute(command)**：この関数は単一の EXEC コマンドを実行して出力を返します。ただし結果のテキストは出力しません。このコマンドの一部としてセミコロンまたは改行を使用することは許可されません。この関数を複数回実行するには、for-loop が指定された Python リストを使用します。次に、 **cli.execute(command)**

  関数の出力例を示します。

```
>>> cli.execute("show clock")
'15:11:20.816 UTC Thu Jun 8 2017'
>>>
>>> cli.execute('show clock'; 'show ip interface brief')
  File "<stdin>", line 1
    cli.execute('show clock'; 'show ip interface brief')
                            ^
SyntaxError: invalid syntax
>>>
```

- **cli.executep(command)**：この関数は単一のコマンドを実行して、結果のテキストを（返すのではなく）stdout に出力します。次に、**cli.executep(command)** 関数の出力例を示します。

```
>>> cli.executep('show clock')
*18:46:28.796 UTC Mon Mar 13 2017
>>> output=cli.executep('show clock')
*18:46:36.399 UTC Mon Mar 13 2017
>>> print(output)
```

```
None
```

- **cli.configure(command)**：この関数は、コマンドで使用できる設定によりデバイスを設定します。これは次に示すように、コマンドとその結果が含まれる名前付きタプルのリストを返します。

```
[Think: result = (bool(success), original_command, error_information)]
```

コマンド パラメータは複数行に入力することができ、**show running-config** コマンドの出力に表示されているのと同じ形式にすることができます。次に、**cli.configure(command)** 関数の出力例を示します。

```
>>>cli.configure(["interface GigabitEthernet1/0/7", "no shutdown",
"end"])
[ConfigResult(success=True, command='interface GigabitEthernet1/0/7',
line=1, output='', notes=None), ConfigResult(success=True, command='no shutdown',
line=2, output='', notes=None), ConfigResult(success=True, command='end',
line=3, output='', notes=None)]
```

- **cli.configurep(command)**：この関数は **cli.configure(command)** 関数と機能はまったく同じです。ただし結果のテキストを（返すのではなく）stdout に出力する点が異なります。次に、**cli.configurep(command)** 関数の出力例を示します。

```
>>> cli.configurep(["interface GigabitEthernet1/0/7", "no shutdown",
 "end"])
Line 1 SUCCESS: interface GigabitEthernet1/0/7
Line 2 SUCCESS: no shut
Line 3 SUCCESS: end
```

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。