CLI 出力の検索とフィルタリング

Cisco IOS CLI には、大量のコマンド出力を検索したり、出力をフィルタリングして不要な情報を除外するための手段が提供されています。これらの機能は、一般に大量のデータが表示される、**show** コマンドと **more** コマンドで使用できます。

（注） **Show** コマンドと **more** コマンドは、常にユーザー EXEC モードまたは特権 EXEC モードで実行します。

画面に表示される内容を超えて出力が続く場合、Cisco IOS CLI では --More-- プロンプトが表示されます。Return キーを押すことで次の行が表示され、スペースキーを押すことで次の画面が表示されます。CLI ストリング検索機能を使用すると、--More-- プロンプトからの出力を検索またはフィルタリングできます。

機能情報の確認

Cisco Feature Navigator を使用すると、プラットフォーム、Cisco IOS ソフトウェア イメージ、および Catalyst OS ソフトウェア イメージの各サポート情報を検索できます。Cisco Feature Navigator には、http://www.cisco.com/go/cfn からアクセスします。Cisco.com のアカウントは必要ありません。

正規表現について

正規表現は、CLI ストリング検索機能によって、**show** コマンドまたは **more** コマンドの出力と照合されるパターン（句、数値、またはより複雑なパターン）です。正規表現では、大文字と小文字が区別され、複雑な一致要件を指定することが可能です。単純な正規表現には、Serial、

misses、138 などのエントリが含まれます。複雑な正規表現としては、00210... 、(is)、[Oo]utput などがあります。

正規表現は、単一文字パターンか複数文字パターンです。つまり、正規表現は、コマンド出力中の同じ 1 文字に一致する 1 つの文字か、コマンド出力中の同じ複数の文字に一致する複数の文字です。コマンド出力中のパターンをストリングと呼びます。この項では、単一文字パターンと複数文字パターンの作成について説明します。また、量指定子、選択、位置指定、カッコを使用したより複雑な正規表現についても説明します。

単一文字パターン

最も単純な正規表現は、コマンド出力内の同じ1つの文字と一致する単一文字です。任意の文字（A ～ Z、a ～ z）または数字（0 ～ 9）を 1 文字のパターンとして使用できます。また、その他のキーボード文字（「!」や「~」など）も 1 文字のパターンとして使用できますが、一部のキーボード文字は正規表現では特別な意味を持ちます。次の表に、特別な意味を持つキーボード文字の一覧を示します。

表 *1:* 特別な意味を持つ文字

文字	特別な意味
.	スペースを含む任意の単一文字と一致します。
*	0 個以上のパターンのシーケンスと一致します。
+	1 個以上のパターンのシーケンスと一致します。
?	0 または 1 回のパターンと一致します。
^	ストリングの先頭と一致します。
$	ストリングの末尾と一致します。
_（アンダースコア）	カンマ（,）、左波カッコ（{）、右波カッコ（}）、左カッコ（(）、右カッコ（)）、ストリングの先頭、ストリングの末尾、またはスペースと一致します。

これらの特殊文字を単一文字パターンとして使用するときは、各文字の前にバックスラッシュ（\）を置いて特別な意味を除外してください。次の例は、それぞれドル記号、アンダースコア、プラス記号に一致する単一文字パターン マッチングの例です。

\$ _ \+

単一文字パターンを範囲指定して、コマンド出力とのマッチングを行うことができます。たとえば、文字 a、e、i、o、u のいずれかを含むストリングに一致する正規表現を作成できます。パターンマッチングが成功するためには、これらの文字のいずれかだけがストリング中に存在する必要があります。1 文字のパターンの範囲を指定するには、1 文字のパターンを角カッコ（**[]**）で囲みます。たとえば、**[aeiou]** は小文字アルファベットの 5 つの母音のうちの任意の 1

文字と一致しますが、**[abcdABCD]** は小文字または大文字アルファベットの最初の 4 つの文字のうちの任意の 1 文字と一致します。

ダッシュ（-）で区切って範囲の終点だけを入力することにより範囲を簡略化することができます。上の範囲は次のように単純化されます。

[a-dA-D]

ダッシュを範囲内の単一文字パターンとして追加するには、ダッシュをもう 1 つ追加し、その前にバックスラッシュを入力します。

[a-dA-D\-]

次に示すように、右角カッコ（]）を、範囲内の単一文字パターンとして追加することもできます。

[a-dA-D\-\]]

上の例は、大文字または小文字のアルファベットの最初の 4 文字、ダッシュ、右角カッコのいずれかに一致します。

範囲の先頭にキャレット（^）を追加することで、範囲の一致を反転させることができます。次の例は、その中の文字以外の文字に一致します。

[^a-dqsv]

次の例は、右角カッコ（]）または文字 d 以外のすべてと一致します。

[^\]d]

複数文字のパターン

正規表現を作成するとき、複数の文字を含むパターンを指定することもできます。複数文字正規表現は、文字、数字、特別な意味のないキーボード文字を組み合わせて作成します。たとえば、a4% は複数文字の正規表現です。文字をそのとおりに解釈することを指示するには、特別な意味のあるキーボード文字の前にバックスラッシュを挿入します。

複数文字パターンでは、順序が大切です。**a4%** という正規表現は、a という文字のあとに 4 が続き、そのあとに % 記号が続く文字と一致します。ストリングの中に a4% という文字がその順序で含まれていないと、パターン マッチングは失敗します。複数文字の正規表現 **a.** では、ピリオド文字の特別な意味を使用し、文字 a の後に任意の 1 文字が続く文字列と一致します。この例では、ab、a!、または a2 というストリングはすべてこの正規表現と一致します。

ピリオド文字の特別な意味を無効にするには、その前にバックスラッシュを挿入します。たとえば、表現 **a\.** がコマンド構文で使用されている場合、ストリング a. だけが一致します。

すべての文字、すべての数字、すべてのキーボード文字、文字と数字とその他のキーボード文字の組み合わせを含む複数文字正規表現を作成できます。たとえば、**telebit3107v32bis** は有効な正規表現です。

量指定子

Cisco IOS ソフトウェアに対して、指定した正規表現の複数の出現に一致させることを指示するため、より複雑な正規表現を作成できます。そのためには、単一文字パターンおよび複数文

字パターンとともに、いくつかの特殊文字を使用します。次の表は、「複数」の正規表現を示す特殊文字の一覧を示します。

表 *2:* 量指定子として使用される特殊文字

文字	Description
*	0 以上の単一文字パターンまたは複数文字パターンと一致します。
+	1 以上の単一文字パターンまたは複数文字パターンと一致します。
?	1 以上の単一文字パターンまたは複数文字パターンの 0 回または 1 回の出現と一致します。

次の例は、空文字を含む文字 a の任意の回数の出現と一致します。

a*

次のパターンでは、ストリングが一致するためには、文字 a が少なくとも 1 文字含まれていることが必要です。

a+

次のパターンは、ストリング bb または bab と一致します。

ba?b

次のストリングは、任意の数のアスタリスク（*）と一致します。

******

複数文字パターンとともに量指定子を使用するには、パターンをカッコで囲みます。次の例で、パターンは複数文字ストリング ab の任意の回数の出現と一致します。

(ab)*

より複雑な例として、次のパターンは、英数字のペアの 1 つ以上のインスタンスに一致しますが、空文字には一致しません（つまり、空のストリングは一致しません）。

([A-Za-z][0-9])+

量指定子（*、+、または?）を使用した一致の順序は、最長構造優先です。ネストした構造は、外側から内側に一致します。連結された構造は、構造の左側から一致します。そのため、この正規表現は A9b3 に一致しますが、9Ab3 には一致しません。これは、英字が数字の前に指定されているためです。

代替

選択を使用すると、ストリングに対して一致する代替パターンを指定できます。選択肢は垂直線（|）で区切ります。代替パターンのうちの 1 つがストリングに一致します。たとえば、正規表現 **codex|telebit** は、ストリング codex またはストリング telebit に一致しますが、codex と telebit の両方には一致しません。

位置指定

Cisco IOS ソフトウェアに対し、ストリングの先頭または末尾に対して正規表現パターンを一致させることを指示できます。つまり、ストリングの先頭または末尾に特定のパターンが含まれていることを指定できます。ストリングの一部に対してこれらの正規表現を「位置指定」するには、次の表に示す特殊文字を使用します。

表 *3*：位置指定に用いられる特殊文字

文字	Description
^	ストリングの先頭と一致します。
$	ストリングの末尾と一致します。

たとえば、正規表現 **^con** は con で始まる任意のストリングと一致し、**$sole** は sole で終わる任意のストリングと一致します。

^ 記号は、ストリングの先頭を示すのに加えて、角カッコの中で使用された場合に論理的な「not」を示すものとして使用できます。たとえば、正規表現 **[^abcd]** は、a、b、c、または d 以外の任意の単一文字に一致する範囲を示します。

これらの位置指定文字は、特殊文字アンダースコア（_）とともに使用します。アンダースコアは、ストリングの先頭（^）、ストリングの末尾（$）、カッコ（( )）、スペース（ ）、波カッコ（{}）、カンマ（,）、アンダースコア（_）に一致します。アンダースコア文字を使用すると、パターンがストリング中のいずれかの場所に存在することを指定できます。たとえば、**_1300_** は、ストリング中のいずれかの場所に 1300 がある任意のストリングに一致します。ストリング 1300 の前後にスペース、波カッコ、カンマ、アンダースコアのいずれかがあってもかまいません。そのため、{1300_ は正規表現 **_1300_** に一致しますが、21300 や 13000 は一致しません。

アンダースコア文字を使用することで、長い正規表現リストを置き換えることができます。たとえば、**^1300()()1300${1300,,1300,{1300},1300,(1300** と指定する代わりに、**_1300_** と指定できます。

後方参照のためのカッコ

「繰り返し指定」のセクションに示したように、複数文字正規表現をカッコで囲み、パターンの出現を繰り返すことができます。また、単一文字パターンまたは複数文字パターンをカッコで囲み、Cisco IOS ソフトウェアに対して、正規表現の別の場所で使用するためにパターンを覚えておくことを指示できます。

前のパターンを後方参照する正規表現を作成するには、カッコを使用して特定のパターンの記憶を指示し、バックスラッシュ（\）の後に数字を使用して記憶したパターンを再利用します。数字は、正規表現パターン内のカッコの出現を指定します。正規表現内に複数のパターンがある場合、\1 は最初に記憶したパターンを示し、\2 は 2 番目に記憶したパターンとなり、以下同様となります。

次の正規表現では、後方参照のためにカッコを使用しています。

a(.)bc(.)\1\2

この正規表現は、後に任意の文字（文字番号 1 とする）が続き、その後に bc が続き、その後に任意の文字（文字番号 2 とする）が続き、そのまた後に文字番号 1 が再び続き、最後に文字番号が続く文字 a と一致します。2 が続くストリングに一致します。そのため、この正規表現は aZbcTZT に一致します。ソフトウェアは、文字番号 1 が Z であり、文字番号 2 が T であることを記憶し、正規表現の後半で Z と T を再度使用します。

show コマンドの検索とフィルタリング

show コマンドの出力を検索するには、特権 EXEC モードで次のコマンドを使用します。

コマンド	目的
`Router# show any-command | begin regular-expression`	**show** コマンドのフィルタリングされていない出力を、正規表現を含む最初の行で開始します。

（注） Cisco IOS のマニュアルでは、縦線を、一般に構文の選択肢を示すために使用します。しかし、**show** コマンドと **more** コマンドの出力を検索するには、パイプ文字（縦線）を入力する必要があります。このセクションでは、パイプを入力する必要があることを示すために、太字（|）で表します。

show コマンドの出力をフィルタリングするには、特権 EXEC モードで次のコマンドのいずれかを使用します。

コマンド	目的
`Router# show any-command | exclude regular-expression`	正規表現を含まない出力行を表示します。
`Router# show any-command | include regular-expression`	正規表現を含む出力行を表示します。

ほとんどのシステムで、Ctrl+Z キーの組み合わせを使用して、いつでも出力を中断し特権 EXEC モードに戻ることができます。たとえば、**showrunning-config|beginhostname** コマンドを使用して、実行コンフィギュレーションファイルの、ホスト名の設定を含む行から表示を開始できます。次に、関心のある情報の最後まで確認し終えたら、Ctrl+Z を使用します。

（注） 感嘆符（!）またはセミコロン（;）が続く文字は、コメントとして扱われ、コマンドでは無視されます。

more コマンドの検索とフィルタリング

more コマンドは、**show** コマンドと同様に検索できます（**more** コマンドは、**show** コマンドと同じ機能を実行します）。**more** コマンドの出力を検索するには、ユーザー EXEC モードで次のコマンドを使用します。

コマンド	目的
`Router# more` *any-command* `| begin` *regular-expression*	**more** コマンドのフィルタリングされていない出力を、正規表現を含む最初の行で開始します。

more コマンドは、**show** コマンドと同様にフィルタリングできます。**more** コマンドの出力をフィルタリングするには、ユーザー EXEC モードで次のコマンドのいずれかを使用します。

コマンド	目的
`Router# more` *any-command* `| exclude` *regular-expression*	正規表現を含まない出力行を表示します。
`Router# more` *any-command* `| include` *regular-expression*	正規表現を含む出力行を表示します。

--More-- プロンプトからの検索およびフィルタリング

--More-- プロンプトから出力を検索できます。**show** コマンドまたは **more** コマンドの出力を --More-- プロンプトから検索するには、ユーザー EXEC モードで次のコマンドを使用します。

コマンド	目的
`-More-` `/` *regular-expression*	フィルタリングされていない出力を、正規表現を含む最初の行で開始します。

--More-- プロンプトから出力をフィルタリングできます。ただし、各コマンドに対して 1 つのフィルタだけを指定できます。フィルタは、**show** コマンドまたは **more** コマンドの出力が終了するか、出力を中断（Ctrl+Z または Ctrl+6 を使用します）するまで継続されます。そのため、元のコマンドか前の --More-- プロンプトですでにフィルタを指定してある場合、--More-- プロンプトで別のフィルタを追加できません。

（注） 検索とフィルタリングは異なる機能です。**begin** キーワードを使用してコマンド出力を検索し、同時に --More-- プロンプトでフィルタを指定することはできます。

--More-- プロンプトで **show** コマンドまたは **more** コマンドの出力をフィルタリングするには、ユーザー EXEC モードで次のコマンドのいずれかを使用します。

コマンド	目的
-More- **|** *regular-expression*	正規表現を含まない出力行を表示します。
-More- **+** *regular-expression*	正規表現を含む出力行を表示します。

CLI 出力の検索とフィルタリングの例

次に、**morenvram:startup-config|begin** 特権 EXEC モード コマンドの部分的な出力例を示します。これは、正規表現を含む最初の行で、フィルタリングされていない出力が開始されています。--More-- プロンプトで、正規表現 ip を含む出力行を除外するためのフィルタを指定します。

```
Router# more nvram:startup-config | begin ip
ip subnet-zero
ip domain-name cisco.com
ip name-server 192.168.48.48
ip name-server 172.16.2.132
!
isdn switch-type primary-5ess
.
.
.
interface Ethernet1
 ip address 10.5.5.99 10.255.255.0
 --More--
-ip
filtering...
 media-type 10BaseT
!
interface Serial0:23
 encapsulation frame-relay
 no keepalive
 dialer string 4001
 dialer-group 1
 isdn switch-type primary-5ess
 no fair-queue
```

次に、**morenvram:startup-config|include** コマンドの出力例の一部を示します。正規表現 ip を含む行だけが表示されています。

```
Router# more nvram:startup-config | include ip
ip subnet-zero
ip domain-name cisco.com
ip name-server 1192.168.48.48
ip name-server 172.16.2.132
```

次に、**morenvram:startup-config|exclude** コマンドの出力例の一部を示します。正規表現 service を含む行が除外されています。--More-- プロンプトで、正規表現 Dialer1 をフィルタとして指定します。このフィルタを指定することにより、Dialer1 を含む最初の行で出力が再開されます。

```
Router# more nvram:startup-config | exclude service
!
version 12.2
!
hostname router
!
boot system flash
no logging buffered
!
ip subnet-zero
ip domain-name cisco.com
.
.
.
--More--
/Dialer1
filtering...
interface Dialer1
 no ip address
 no ip directed-broadcast
 dialer in-band
 no cdp enable
```

次に、出力の検索が指定された、**showinterface** コマンドの部分的な出力例を示します。パイプの後でキーワード **beginEthernet** を使用することで、正規表現 Ethernet を含む最初の行でフィルタリングされていない出力が開始されます。--More-- プロンプトで、正規表現 Serial を含む行だけを表示するフィルタを指定します。

```
Router# show interface | begin Ethernet
Ethernet0 is up, line protocol is up
Hardware is Lance, address is 0060.837c.6399 (bia 0060.837c.6399)
  Description: ip address is 172.1.2.14 255.255.255.0
  Internet address is 172.1.2.14/24
.
.
.
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
--More--
+Serial
filtering...
Serial1 is up, line protocol is up
Serial2 is up, line protocol is up
Serial3 is up, line protocol is down
Serial4 is down, line protocol is down
Serial5 is up, line protocol is up
```

```
Serial6 is up, line protocol is up
Serial7 is up, line protocol is up
```

次に、**showbuffers|exclude** コマンドの出力例の一部を示します。正規表現 ip を含む行が除外されています。--More-- プロンプトで、フィルタされていない出力を、Serial0 を含む最初の行から続行するための検索を指定します。

```
Router# show buffers | exclude 0 misses
Buffer elements:
     398 in free list (500 max allowed)
Public buffer pools:
Small buffers, 104 bytes (total 50, permanent 50):
     50 in free list (20 min, 150 max allowed)
     551 hits, 3 misses, 0 trims, 0 created
Big buffers, 1524 bytes (total 50, permanent 50):
     49 in free list (5 min, 150 max allowed)
Very Big buffers, 4520 bytes (total 10, permanent 10):
.
.
.
Huge buffers, 18024 bytes (total 0 permanent 0):
     0 in free list (0 min, 4 max allowed)
--More--
/Serial0
filtering...
Serial0 buffers, 1543 bytes (total 64, permanent 64):
     16 in free list (0 min, 64 max allowed)
     48 hits, 0 fallbacks
```

次に、**showinterface|include** コマンドの出力例の一部を示します。パイプ（|）の後で **include(is)** キーワードを使用することにより、正規表現 (is) が含まれる行だけが表示されます。カッコにより、is の前後にスペースが含まれることが指定されます。カッコを使用することで、is の前後にスペースを含む行だけが出力に含まれます（「disconnect」などの文字は検索から除外されます）。

```
router# show interface | include ( is )
ATM0 is administratively down, line protocol is down
  Hardware is ATMizer BX-50
Dialer1 is up (spoofing), line protocol is up (spoofing)
  Hardware is Unknown
  DTR is pulsed for 1 seconds on reset
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0060.837c.6399 (bia 0060.837c.6399)
  Internet address is 172.21.53.199/24
Ethernet1 is up, line protocol is up
  Hardware is Lance, address is 0060.837c.639c (bia 0060.837c.639c)
  Internet address is 10.5.5.99/24
Serial0:0 is down, line protocol is down
  Hardware is DSX1
.
.
.
 --More--
```

--More-- プロンプトで、Serial0:13 を含む最初の行でフィルタリングされた出力を続行する検索を指定します。

```
/Serial0:13
filtering...
```

```
Serial0:13 is down, line protocol is down
  Hardware is DSX1
  Internet address is 10.0.0.2/8
     0 output errors, 0 collisions, 2 interface resets
  Timeslot(s) Used:14, Transmitter delay is 0 flag
```

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。