Condition match modes

Note: This feature is currently in beta and is available only to early adopters.

All Compliance Condition types support three match modes:

- **Loose**
- **Sequential**
- **Strict**

Match mode determines how the order and consecutiveness of input lines in the source affect condition evaluation.

Mode	Order matters	Additional content allowed between matches
Loose	No	Yes
Sequential	Yes	Yes
Strict	Yes	No

Reference Source:

The examples throughout this article use the following device configuration:

```
hostname R1
!
interface Ethernet1
 description Uplink
 ip address 10.0.0.1/24
 no shutdown
!
interface Ethernet2
 description LAN
 ip address 192.168.1.1/24
 no shutdown
!
interface Loopback0
 description Management Loopback
 ip address 10.255.255.1 255.255.255.255
!
router ospf
 router-id 1.1.1.1
 network 10.0.0.0/24 area 0
!
```

Note: The Compliance engine can validate MCP automation outputs in addition to device configurations. To validate an automation output, select Source Mass Config Push Result. The same condition types and matching principles described in this article apply when evaluating automation output.

Text matching condition types:

Loose Mode

Loose mode verifies that all input lines exist somewhere in the source. Their order is ignored, and any content may appear between matching lines in the source.

Example Input 1:

```
description LAN
hostname R1
```

Result: Condition successful

Although the order of the input lines differs from the source, both lines are present.

Example Input 2:

```
hostname R1
 description WAN
```

Result: Condition failed

The line *description WAN* does not exist in the source.

Sequential Mode

Sequential mode requires all input lines to appear in the source in the same order as entered. Additional text/lines may appear between them.

Example Input 1:

```
interface Ethernet1
 ip address 10.0.0.1/24
```

Result: Condition successful

Both input lines appear in the source in the correct order. The "description Uplink" line between them is ignored.

Example Input 2:

```
ip address 10.0.0.1/24
interface Ethernet1
```

Result: Condition failed

The input lines appear in the source in reverse order.

Strict Mode

Strict mode requires input lines to appear consecutively in the source. No additional text/lines may exist between the matching lines in the source.

Example Input 1:

```
interface Ethernet1
 description Uplink
 ip address 10.0.0.1/24
```

Result: Condition successful

All three lines appear consecutively in the source.

Example Input 2:

```
interface Ethernet1
 ip address 10.0.0.1/24
```

Result: Condition failed

The source contains the line "description Uplink" between the two input lines. Strict mode therefore fails.

Regex matching condition types:

The same three match modes apply when using regex matching condition types. Each regular expression in the input is evaluated against lines in the source, and the match mode governs whether the matches must occur in order and whether intervening unmatched lines are permitted.

Loose Mode

```
(?m)^router ospf$
(?m)^hostname R[0-9]+$
```

This verifies that the device follows the *R<number>* hostname convention and has OSPF enabled. The two matches are independent, so their order does not matter.

Sequential Mode

```
(?m)^router ospf$
(?m)^ network [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/([0-9]+) area [0-9]+$
```

This verifies that an OSPF configuration section is followed by an OSPF network statement. Other OSPF configuration, such as a router-id, may appear between the matches.

Strict Mode

```
(?m)^interface Loopback[0-9]+$
(?m)^ description Management Loopback$
(?m)^ ip address 10\.255\.255\.[0-9]+ 255\.255\.255\.255$
```

This verifies that a Loopback interface with a variable interface number has the required "Management Loopback" description and uses a /32 address from the 10.255.255.0/24 management subnet.

Choosing the Right Match Mode

Use case	Recommended mode
Verify several independent configuration lines exist in the source	Loose
Verify the presence and order of configuration commands in the source	Sequential
Verify the presence of a configuration block	Strict

Common use cases:

TODO