

## show dhcp count

---

**Syntax** To display counts of DHCP client bindings and interfaces on the specified subnet:

```
show dhcp count [ local | external | relay-proxy ] [ subnetAddress ] [filter ]
```

To display counts of DHCP client bindings and interfaces for the specified IP prefix:

```
show dhcp count [ local | external | relay-proxy ] [ subnetAddress ] ip-prefix ipPrefix [filter ]
```

To display counts of DHCP client bindings and interfaces for the specified interface string:

```
show dhcp count [ local | external | relay-proxy ] [ subnetAddress ] interface string [filter ]
```

To display counts of DHCP client bindings and interfaces without a lower-layer interface:

```
show dhcp count [ local | external | relay-proxy ] [ subnetAddress ] no-interface [filter ]
```

To display counts of DHCP client bindings and interfaces for the specified agent-circuit-id suboption (suboption 1) string of the DHCP relay agent information option (option 82):

```
show dhcp count [ local | external | relay-proxy ] [ subnetAddress ] circuit-id string [filter ]
```

To display counts of DHCP client bindings and interfaces for the specified agent-remote-id suboption (suboption 2) string of the DHCP relay agent information option (option 82):

```
show dhcp count [ local | external | relay-proxy ] [ subnetAddress ] remote-id string [filter ]
```

**Release Information** Command introduced in JUNOS Release 9.3.0.

**Description** Displays counts of DHCP client bindings and interfaces.

- Options**
- *local*—Specifies DHCP local server client bindings that meet the display criteria
  - *external*—Specifies DHCP external server client bindings that meet the display criteria
  - *relay-proxy*—Specifies DHCP relay proxy client bindings that meet the display criteria
  - *subnetAddress*—IP address of the subnet on which the DHCP clients reside
  - *ipPrefix*—IP prefix (address and subnetwork mask) of the DHCP clients; for example, 10.10.10.0/24

- `no-interface`—Specifies DHCP clients without a lower-layer interface; use this keyword to display count information for DHCP client bindings configured over dynamic interfaces for which the lower-layer interface has been shut down
- `filter`—See Filtering show Commands
- `string`—Regular expression string that represents the interface, circuit ID, or remote ID to be matched; you must enclose elements containing a space within double quotes (“*one element*”)

Each element is either a literal string, a metacharacter, or a combination. You can remove the special meaning of a metacharacter by preceding it with a backslash (\). Regular expressions support the following metacharacters:

- `^` Matches the beginning of the input string. Alternatively, when used as the first character within brackets—`[^ ]`—matches any number except the ones specified within the brackets.
- `$` Matches the end of the input string
- `.` (period) Matches any single character, including white space
- `*` Matches 0 or more sequences of the immediately previous character or pattern.
- `+` Matches 1 or more sequences of the immediately previous character or pattern
- `?` Matches 0 or 1 sequence of the immediately previous character or pattern
- `()` Specifies patterns for multiple use when followed by one of the multiplier metacharacters: asterisk `*`, plus sign `+`, or question mark `?`
- `[ ]` Matches any enclosed character; specifies a range of single characters
- `-` (hyphen) Used within brackets to specify a range of AS or community numbers
- `_` (underscore) Matches a `^`, a `$`, a comma, a space, a `{`, or a `}`. Placed on either side of a string to specify a literal and disallow substring matching. Numerals enclosed by underscores can be preceded or followed by any of the characters listed above
- `|` Matches characters on either side of the metacharacter; logical OR

You must specify the interface string as a regular expression without spaces; for example, `fastEthernet1.1/100` or `fastEthernet.*100`

The following rules apply for representing nonprintable character sequences in the circuit ID string or the remote ID string:

- To represent the binary sequence 0d 0a (hex), use the string '\\r\\n'. This consists of four ASCII characters: 5c for \\, 72 for r, 5c for \\, and 6e for n.

For example, to match the sequence 74 65 73 74 0d 0a 6f 6e 65 (hex), use the string 'test\\r\\n\\none'. In this string, 74 is represented by t, 65 is represented by e, 73 is represented by s, 74 is represented by t, 0d 0a is represented by \\r\\n, 6f is represented by o, 6e is represented by n, and 65 is represented by e.

- To represent the binary sequence 0d 00 (hex), use the string '\\r'. This consists of two ASCII characters: 5c for \\, and 72 for r.
- To represent the binary sequence 0a 00 (hex), use the string '\\n'. This consists of two ASCII characters: 5c for \\, and 6e for n.

For example, to match the sequence 74 65 73 74 0a 00 6f 6e 65 (hex), use the string 'test\\n\\none'. In this string, 74 is represented by t, 65 is represented by e, 73 is represented by s, 74 is represented by t, 0a 00 is represented by \\n, 0a is represented by \\n, 6f is represented by o, 6e is represented by n, and 65 is represented by e.

- To represent all other cases, use the string '\\xab', where ab is a hex code of the byte. For example, to represent byte 3A, use '\\x3a'. This consists of four ASCII characters: 5c for \\, 78 for x, 33 for 3, and 61 for a.

As another example, to match the sequence 74 65 73 74 f3 6f 6e 65 (hex), use the string 'test\\xf3\\none'. In this string, 74 is represented by t, 65 is represented by e, 73 is represented by s, 74 is represented by t, byte F3 is represented by \\xf3, 6f is represented by o, 6e is represented by n, and 65 is represented by e.

**Mode** Privileged Exec

