

Chapter 10

Using Shortcuts, Wildcards, and Regular Expressions

This chapter provides information on how to use keyboard shortcuts, wildcards, and other advanced techniques to save time when entering commands and configuration statements.

Topics include:

- Moving Around and Editing the Command Line on page 154
- Wildcard Characters in Interface Names on page 155
- Using Global Replace in a Configuration on page 155
- Using Regular Expressions to Delete Related Configuration Items on page 161

Moving Around and Editing the Command Line

In the CLI, you can use keyboard sequences to move around on a command line and edit the command line. You can also use keyboard sequences to scroll through a list of recently executed commands. Table 17 lists some of the CLI keyboard sequences. They are the same as those used in Emacs.

Table 17: CLI Keyboard Sequences (1 of 2)

Category	Action	Keyboard Sequence
Move the Cursor	Move the cursor back one character.	Ctrl+b
	Move the cursor back one word.	Esc+b or Alt+b
	Move the cursor forward one character.	Ctrl+f
	Move the cursor forward one word.	Esc+f or Alt+f
	Move the cursor to the beginning of the command line.	Ctrl+a
	Move the cursor to the end of the command line.	Ctrl+e
Delete Characters	Delete the character before the cursor.	Ctrl+h, Delete, or Backspace
	Delete the character at the cursor.	Ctrl+d
	Delete all characters from the cursor to the end of the command line.	Ctrl+k
	Delete all characters on the command line.	Ctrl+u or Ctrl+x
	Delete the word before the cursor.	Ctrl+w, Esc+Backspace, or Alt+Backspace
	Delete the word after the cursor.	Esc+d or Alt+d
Insert Recently Deleted Text	Insert the most recently deleted text at the cursor.	Ctrl+y
Redraw the Screen	Redraw the current line.	Ctrl+l
Display Previous Command Lines	Scroll backward through the list of recently executed commands.	Ctrl+p
	Scroll forward through the list of recently executed commands.	Ctrl+n
	Search the CLI history in reverse order for lines matching the search string.	Ctrl+r
	Search the CLI history by typing some text at the prompt, followed by the keyboard sequence. The CLI attempts to expand the text into the most recent word in the history for which the text is a prefix.	Esc+/ [text]

Table 17: CLI Keyboard Sequences (2 of 2)

Category	Action	Keyboard Sequence
Display Previous Command Words	Scroll backward through the list of recently entered words in a command line.	Esc+. or Alt+.
Repeat Keyboard Sequences	Specify the number of times to execute a keyboard sequence. <i>number</i> can be from 1 through 9.	Esc+ <i>number sequence</i> or Alt+ <i>number sequence</i>

Wildcard Characters in Interface Names

You can use wildcard characters in operational commands to specify groups of interface names without having to type each name individually. Table 18 lists the available wildcard characters. You must enclose all wildcard characters except the asterisk (*) in quotation marks (“ ”).

Table 18: Wildcard Characters for Specifying Interface Names

Wildcard Character	Description
* (asterisk)	Match any string of characters in that position in the interface name. For example, <code>so*</code> matches all SONET/SDH interfaces.
“ <code>[character <character...>]</code> ”	Match one or more individual characters in that position in the interface name. For example, “ <code>so-[03]</code> ” matches all SONET/SDH interfaces in slots 0 and 3.
“ <code>[!character <character...>]</code> ”	Match all characters except the ones included in the brackets. For example, “ <code>so-[!03]</code> ” matches all SONET/SDH interfaces except those in slots 0 and 3.
“ <code>[character1-character2]</code> ”	Match a range of characters. For example, <code>so-[0-3]</code> matches all SONET/SDH interfaces in slots 0, 1, 2, and 3.
“ <code>[!character1-character2]</code> ”	Match all characters that are not in the specified range of characters. For example, <code>so-[!0-3]</code> matches all SONET/SDH interfaces in slots 4, 5, 6, and 7.

Using Global Replace in a Configuration

To make global changes to variables and identifiers in a configuration, use the `replace` configuration mode command. This command replaces a pattern in a configuration with another pattern. For example, you can use this command to find and replace all occurrences of an interface name when a PIC is moved to another slot in the router.

```
user@host> replace pattern pattern1 with pattern2 <upto n>
```

pattern *pattern1* is a text string or regular expression that defines the identifiers and values you want to replace in the configuration.

pattern2 is a text string or regular expression that replaces the identifiers and values located with *pattern1*.

Juniper Networks uses standard UNIX-style regular expression syntax (as defined in POSIX 1003.2). If the regular expression contains spaces, operators, or wildcard characters, enclose the expression in quotation marks. Greedy qualifiers (match as much as possible) are supported. Lazy qualifiers (match as little as possible) are not.

upto *n* specifies the number of objects replaced. The value of *n* controls the total number of objects that are replaced in the configuration (not the total number of times the pattern occurs). Objects at the same hierarchy level (siblings) are replaced first. Multiple occurrences of a pattern within a given object are considered a single replacement. For example, if a configuration contains a **010101** text string, the following command:

```
replace pattern 01 with pattern 02 upto 2
```

replaces **010101** with **020202** (instead of **020201**). Replacement of **010101** with **020202** is considered a single replacement (*n* = 1), not three separate replacements (*n* = 3).

If you do not specify an **upto** option, all identifiers and values in the configuration that match *pattern1* are replaced.

The **replace** command is available in configuration mode at any hierarchy level. All matches are case-sensitive.

Table 19 shows some common regular expressions you can use with the **replace** command. Table 20 provides some examples of pattern replacement.

Table 19: Common Regular Expressions

Operator	Function
	Indicates that a match can be one of the two terms on either side of the pipe.
^	Used at the beginning of an expression, denotes where a match should begin.
\$	Used at the end of an expression, denotes that a term must be matched exactly up to the point of the \$ character.
[]	Specifies a range of letters or digits to match. To separate the start and end of a range, use a hyphen (-).
()	Specifies a group of terms to match. Stored as numbered variables. Use for back references as \1 \2 ... \9.
*	0 or more terms.
+	One or more terms.
.	Any character except for a space " ".
\	A backslash escapes special characters to suppress their special meaning. For example, \. matches . (period symbol).
\n	Back reference. Matches the <i>n</i> th group.
&	Back reference. Matches the entire match.

Table 20: Replacement Examples

Command	Result
replace pattern myrouter with router1	Match: myrouter Result: router1
replace pattern "192.168\.(.*)/24" with "10.2.1/28"	Match: 192.168.3.4/24 Result: 10.2.3.4/28
replace pattern "1.\1" with "abc&def"	Match: 1.1 Result: abc1.1def
replace pattern 1.1 with "abc&def"	Match: 1#1 Result: abc&def

Example 1: Using Global Replace in a Configuration

Replace an interface name in a configuration:

```
[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-0/0/0 {
        hello-interval 5;
      }
    }
  }
}
[edit]
user@host# replace so-0/0/0 with so-1/1/0

[edit]
user@host# show
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-1/1/0 {
        hello-interval 5;
      }
    }
  }
}
```

Example 2: Using Global Replace in a Configuration

Use the `\n` back reference to replace a pattern:

```
[edit]
user@host# show interfaces
xe-0/0/0 {
    unit 0;
}
fe-3/0/1 {
    vlan-tagging;
    unit 0 {
        description "inet6 configuration. IP: 2000::c0a8::1bf5";
        vlan-id 100;
        family inet {
            address 17.10.1.1/24;
        }
        family inet6 {
            address 2000::c0a8:1bf5/3;
        }
    }
}
```

```
[edit]
user@host# replace pattern "(.*)1bf5" with "\11bf5"
```

```
[edit]
user@host# show interfaces
xe-0/0/0 {
    unit 0;
}
fe-3/0/1 {
    vlan-tagging;
    unit 0 {
        description "inet6 configuration. IP: 2000::c0a8:1bf5";
        vlan-id 100;
        family inet {
            address 17.10.1.1/24;
        }
        family inet6 {
            address 2000::c0a8:1bf4/3;
        }
    }
}
```

The pattern `2000::c0a8::1bf5` is replaced with `2000::c0a8:1bf5`.

Example 3: Using Global Replace in a Configuration

Consider the hierarchy shown in Figure 23 on page 160. The text string `010101` appears in three places (description sections of `xe-0/0/0`, `xe-0/0/0.0`, and `fe-0/0/1`). These three instances are three objects.

Use the `upto` option to perform a replacement:

```

user@host# show interfaces
xe-0/0/0 {
description "mkt 010101"; #1st instance in the hierarchy
unit 0 {
description "mkt 010101"; #3rd instance in the hierarchy (child of the 1st
instance)
}
}
fe-0/0/1 {
description "mkt 010101"; #2nd instance in the hierarchy (sibling of the 1st
instance)
unit 0 {
family inet {
address 200.200.20.2/24;
}
}
}
[edit]
user@host# replace pattern 01 with 02 upto 2
[edit]
user@host# commit
commit complete

```

An `upto 2` option in the `replace` command converts `01` to `02` for two object instances. The objects under the main interfaces `xe-0/0/0` and `fe-0/0/1` will be replaced first (since these are siblings in the hierarchy level). Because of the `upto 2` restriction, the `replace` command replaces patterns in the first and second instance in the hierarchy (siblings), but not the third instance (child of the first instance).

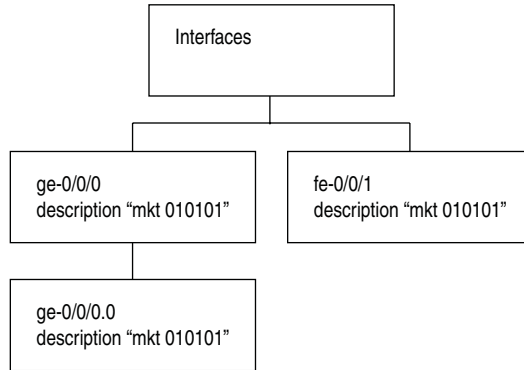
```

[edit]
user@host# show interfaces
xe-0/0/0 {
description "mkt 020202"; #1st instance in the hierarchy
unit 0 {
description "mkt 010101"; #3rd instance in the hierarchy (child of the 1st
instance)
}
}
fe-0/0/1 {
description "mkt 020202"; #2nd instance in the hierarchy (sibling of the 1st
instance)
unit 0 {
family inet {
address 200.200.20.2/24;
}
}
}

```

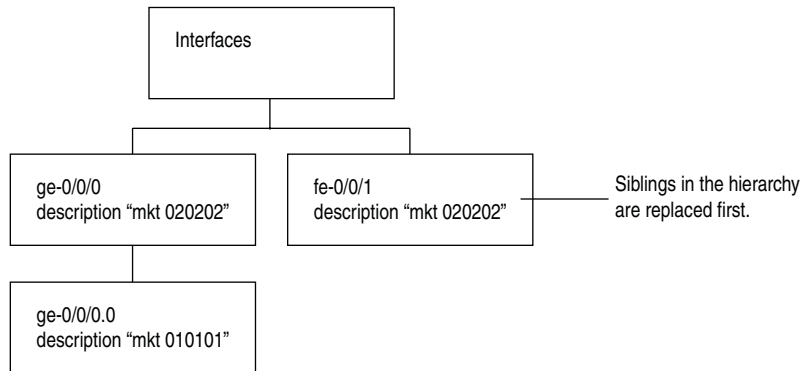
Figure 23: Replacement by Object

Current Configuration:



user@host # **replace pattern 01 with pattern 02 upto 2**

Resulting Configuration:



901728

Using Regular Expressions to Delete Related Configuration Items

You can delete related configuration items simultaneously, such as channelized interfaces or static routes, by using a single command and regular expressions. Deleting a statement or an identifier effectively “unconfigures” the functionality associated with that statement or identifier, returning that functionality to its default condition.

You can only delete several parts of the configuration where you normally put multiple items; for example, interfaces. However, you cannot delete “groups” of different items; for example:

```

user@host# show system services
ftp;
rlogin;
rsh;
ssh {
    root-login allow;
}
telnet;

[edit]
user@host# wildcard delete system services *
                                     ^
syntax error.

```

When you delete a statement, the statement and all its subordinate statements and identifiers are removed from the configuration.

To delete related configuration items, issue the `wildcard` configuration mode command with the `delete` option and specify the statement path, the items to be summarized with a regular expression, and the regular expression.

```

user@host# wildcard delete <statement-path> <identifier> <regular-expression>

```



NOTE: When you use the `wildcard` command to delete related configuration items, the regular expression must be the final statement.

If the JUNOS software matches more than eight related items, the CLI displays only the first eight items.

Example: Deleting Interfaces from the Configuration

Delete multiple T1 interfaces in the range from t1-0/0/0:0 through t1-0/0/0:23:

```
user@host# wildcard delete interfaces t1-0/0/0:. *
matched: t1-0/0/0:0
matched: t1-0/0/0:1
matched: t1-0/0/0:2
Delete 3 objects? [yes,no] (no) no
```

Example: Deleting Routes from the Configuration

Delete static routes in the range from 172.0.0.0 to 172.255.0.0:

```
user@host# wildcard delete routing-options static route 172.*
matched: 172.16.0.0/12
matched: 172.16.14.0/24
matched: 172.16.100.0/24
matched: 172.16.128.0/19
matched: 172.16.160.0/24
matched: 172.17.12.0/23
matched: 172.17.24.0/23
matched: 172.17.28.0/23
...
Delete 13 objects? [yes,no] (no)
```