

Junos® OS Evolved

User Access and Authentication Administration Guide for Junos OS Evolved

Published
2022-09-13

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Junos® OS Evolved User Access and Authentication Administration Guide for Junos OS Evolved
Copyright © 2022 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

[About This Guide | ix](#)

1

[Login Classes and Login Settings](#)

[Login Classes Overview | 2](#)

[Login Classes Overview | 2](#)

[Example: Create Login Classes with Specific Privileges | 8](#)

[Login Settings | 9](#)

[Display a System Login Announcement or Message | 9](#)

[Display System Alarms Upon Login | 11](#)

[Configure Login Tips | 12](#)

[Configure the Timeout Value for Idle Login Sessions | 13](#)

[Login Retry Options | 14](#)

2

[User Accounts](#)

[User Accounts | 18](#)

[User Accounts Overview | 18](#)

[Example: Configure New User Accounts | 20](#)

[Requirements | 20](#)

[Overview | 20](#)

[Configuration | 21](#)

[Verification | 24](#)

[Configure User Accounts in a Configuration Group | 24](#)

[Administrative Roles | 28](#)

[How to Design Administrative Roles | 28](#)

[Example: How to Configure Administrative Roles | 30](#)

[Requirements | 31](#)

[Overview | 31](#)

[Configuration | 31](#)

Verification | 38

How to Configure a Local Administrator Account | 40

User Access Privileges | 41

Access Privilege Levels Overview | 41

Example: Configure User Permissions with Access Privilege Levels | 47

Requirements | 47

Overview | 47

Configuration | 48

Verification | 49

Regular Expressions to Allow and Deny Operational Mode Commands, Configuration Statements, and Hierarchies | 51

How to Define Access Privileges with allow-configuration and deny-configuration Statements | 72

Example: Use Additive Logic with Regular Expressions to Specify Access Privileges | 75

Requirements | 75

Overview | 75

Configuration | 76

Examples | 77

Example: Configure User Permissions with Access Privileges for Operational Mode Commands | 79

Requirements | 80

Overview and Topology | 80

Configuration | 81

Verification | 87

Example: Configure User Permissions with Access Privileges for Configuration Statements and Hierarchies | 91

Requirements | 91

Overview and Topology | 91

Configuration | 92

Verification | 98

3

Passwords for User Access

Root Password | 102

Configure the Root Password | 102

Example: Configure a Plain-Text Password for Root Logins | 104

Requirements | 104

Overview | 105

Configuration | 105

Verification | 106

Recover a Root Password | 107

How to Recover the Root Password for Junos OS Evolved | 107

How to Connect to the Serial Port | 108

How to Recover the Root Password | 109

Plain-Text Passwords | 111

Change the Requirements for Plain-Text Passwords | 111

How to Change the Requirements for Plain-Text Passwords | 112

Overview | 112

Configuration | 112

4

User Authentication

User Authentication Overview | 116

User Authentication Methods | 116

Configure Local User Template Accounts for User Authentication | 117

Configure Remote User Template Accounts for User Authentication | 119

Example: Create Template Accounts | 119

Requirements | 120

Overview | 120

Configuration | 120

Verification | 122

What Are Remote Authentication Servers? | 123

Authentication Order for LDAPS, RADIUS, TACACS+, and Local Password | 124

Authentication Order Overview | 125

Configure the Authentication Order for LDAPS, RADIUS, TACACS+ and Local Password Authentication | 131

Example: Configure Authentication Order | 133

- Requirements | 133
- Overview | 134
- Configuration | 134
- Verification | 136

RADIUS Authentication | 137

Configure RADIUS Server Authentication | 137

- Why Use RADIUS | 138
- Configure RADIUS Server Details | 138
- Configure RADIUS to Use the Management Instance | 142

Example: Configure a RADIUS Server for System Authentication | 143

- Requirements | 143
- Overview | 143
- Configuration | 144
- Verification | 146

Juniper Networks Vendor-Specific RADIUS and LDAP Attributes | 147

Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands | 152

Understanding RADIUS Accounting | 156

Configure RADIUS System Accounting | 157

- Configure Auditing of User Events on a RADIUS Server | 158

TACACS+ Authentication | 161

Configure TACACS+ Authentication | 162

- Configure TACACS+ Server Details | 162
- Configure TACACS+ to Use the Management Instance | 167
- Configure the Same Authentication Service for Multiple TACACS+ Servers | 167
- Configure Juniper Networks Vendor-Specific TACACS+ Attributes | 168

Configure Periodic Refresh of the TACACS+ Authorization Profile | 169

Example: Configure a TACACS+ Server for System Authentication | 170

- Requirements | 170
- Overview | 171
- Configuration | 171
- Verification | 173

Juniper Networks Vendor-Specific TACACS+ Attributes | 174

Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands | 176

Configuring TACACS+ System Accounting | 181

Configure TACACS+ Server Accounting | 182

Authentication for Routing Protocols | 186

Authentication Methods for Routing Protocols | 186

Example: Configure the Authentication Key for BGP and IS-IS Routing Protocols | 187

Configure the Authentication Key Update Mechanism for Routing Protocols | 190

Configure Authentication Key Updates | 190

Configure BGP and LDP for Authentication Key Updates | 191

5

Remote Access Management

Remote Access Overview | 194

System Services Overview | 194

Configure Telnet Service for Remote Access to a Router or Switch | 195

Configure FTP Service for Remote Access to the Router or Switch | 195

Configure Finger Service for Remote Access to the Router | 196

Configure SSH Service for Remote Access to the Router or Switch | 196

Configure the Root Login Through SSH | 198

Configure Incoming SFTP Connections | 198

Configure the SSH Protocol Version | 199

Configure the Client Alive Mechanism | 199

Configure the SSH Fingerprint Hash Algorithm | 200

The telnet Command | 200

The ssh Command | 201

Configure SSH Host Keys for Secure Copying of Data | 202

Configure SSH Known Hosts | 202

Configure Support for SCP File Transfer | 203

Update SSH Host Key Information | 204

Configure the SSH Service to Support Legacy Cryptography | 205

Configure Outbound SSH Service | 207

- Send the Public SSH Host Key to the Outbound SSH Client | 208
- Configure Keepalive Messages for Outbound SSH Connections | 209
- Configure a New Outbound SSH Connection | 210
- Configure the Outbound SSH Client to Accept NETCONF as an Available Service | 210
- Configure Outbound SSH Clients | 210
- Configure Routing Instances for Outbound SSH Clients | 211

Configure NETCONF-Over-SSH Connections on a Specified TCP Port | 211

Configuration Guidelines for Securing Console Port Access | 212

- Secure the Console Port | 212

6

Device Discovery

Device Discovery Using LLDP | 215

Understanding LLDP | 215

Configuring LLDP (CLI Procedure) | 216

- Enable LLDP on Interfaces | 217
- Adjust LLDP Advertisement Settings | 217
- Adjust SNMP Notification Settings of LLDP Changes | 218
- Specify a Management Address for the LLDP Management TLV | 219
- Specify a Management Interface for the LLDP Management TLV | 220
- Configure LLDP Power Negotiation | 220
- Disable LLDP TLVs | 221

7

Operational Commands

request security uefi revoke | 224

About This Guide

Junos OS Evolved enables you to configure user access and authentication features at the `[edit system]` hierarchy level of the CLI. Essential user access features include login classes, user accounts, access privilege levels, and user authentication methods. Use the topics on this page to configure essential user access features for your system.

1

CHAPTER

Login Classes and Login Settings

[Login Classes Overview](#) | 2

[Login Settings](#) | 9

Login Classes Overview

IN THIS SECTION

- [Login Classes Overview | 2](#)
- [Example: Create Login Classes with Specific Privileges | 8](#)

Junos OS Evolved login classes define the access privileges, permissions for using CLI commands and statements, and session idle time for the users assigned to that class. You (the system administrator) can apply a login class to an individual user account, thereby assigning certain privileges and permissions to the user.

Login Classes Overview

IN THIS SECTION

- [Permission Bits | 3](#)
- [Deny or Allow Individual Commands and Statement Hierarchies | 7](#)

All users who can log in to a device running Junos OS Evolved must be in a login class. Each login class defines the following:

- Access privileges that users have when they log in to the network device
- Commands that users can and cannot execute
- Configuration statements that users can and cannot view or modify
- Amount of time a login session can be idle before the system disconnects the user

You can define any number of login classes. However, you only assign one login class to an individual user account.

Junos OS Evolved includes predefined login classes, which are listed in [Table 1 on page 3](#). You cannot modify the predefined login classes.

Table 1: Predefined System Login Classes

Login Class	Permission Flag Set
operator	clear, network, reset, trace, and view
read-only	view
superuser or super-user	all
unauthorized	None

NOTE:

- You cannot modify a predefined login class name. If you issue the set command on a predefined class name, the device appends `-local` to the login class name and issues the following warning:

warning: '`<class-name>`' is a predefined class name; changing to '`<class-name>-local`'

- You cannot issue the rename or the copy command on a predefined login class. Doing so results in the following error message:

error: target '`<class-name>`' is a predefined class

Permission Bits

Each top-level CLI command and each *configuration statement* has an access privilege level associated with it. Users can execute only those commands and configure and view only those statements for which they have access privileges. Each login class defines one or more permission bits that determine the access privileges.

Two forms for the permissions control whether a user can view or modify the individual parts of the configuration:

- "Plain" form—Provides read-only capability for that permission type. An example is `interface`.
- `-control` form—Provides read and write capability for that permission type. An example is `interface-control`.

[Table 2 on page 4](#) outlines the permission flags and associated access privileges.

Table 2: Login Class Permission Flags

Permission Flag	Description
<code>access</code>	Can view the access configuration in operational mode or configuration mode.
<code>access-control</code>	Can view and configure access information at the <code>[edit access]</code> hierarchy level.
<code>admin</code>	Can view user account information in operational mode or configuration mode.
<code>admin-control</code>	Can view user account information and configure it at the <code>[edit system]</code> hierarchy level.
<code>all</code>	Can access all operational mode commands and configuration mode commands. Can modify the configuration in all the configuration hierarchy levels.
<code>clear</code>	Can clear (delete) information that the device learns from the network and stores in various network databases (using the <code>clear</code> commands).
<code>configure</code>	Can enter configuration mode (using the <code>configure</code> command) and commit configurations (using the <code>commit</code> command).
<code>control</code>	Can perform all control-level operations—all operations configured with the <code>-control</code> permission flags.
<code>field</code>	Can view field debug commands. Reserved for debugging support.
<code>firewall</code>	Can view the <i>firewall filter</i> configuration in operational mode or configuration mode.

Table 2: Login Class Permission Flags *(Continued)*

Permission Flag	Description
firewall-control	Can view and configure firewall filter information at the [edit firewall] hierarchy level.
floppy	Can read from and write to the removable media.
flow-tap	Can view the flow-tap configuration in operational mode or configuration mode.
flow-tap-control	Can view and configure flow-tap information at the [edit services flow-tap] hierarchy level.
flow-tap-operation	<p>Can make flow-tap requests to the router or switch. For example, a Dynamic Tasking Control Protocol (DTCP) client must have flow-tap-operation permission to authenticate itself to Junos OS Evolved as an administrative user.</p> <p>NOTE: The flow-tap-operation option is not included in the all-control permissions flag.</p>
idp-profiler-operation	Can view profiler data.
interface	Can view the interface configuration in operational mode and configuration mode.
interface-control	<p>Can view chassis, <i>class of service</i> (CoS), groups, forwarding options, and interfaces configuration information. Can modify the configuration at the following hierarchy levels:</p> <ul style="list-style-type: none"> • [edit chassis] • [edit class-of-service] • [edit groups] • [edit forwarding-options] • [edit interfaces]

Table 2: Login Class Permission Flags (Continued)

Permission Flag	Description
<code>maintenance</code>	Can perform system maintenance, including starting a local shell on the device and becoming the superuser in the shell (using the <code>su root</code> command) and halting and rebooting the device (using the <code>request system</code> commands).
<code>network</code>	Can access the network by using the <code>ping</code> , <code>ssh</code> , <code>telnet</code> , and <code>traceroute</code> commands.
<code>pgcp-session-mirroring</code>	Can view the pgcp session mirroring configuration.
<code>pgcp-session-mirroring-control</code>	Can modify the pgcp session mirroring configuration.
<code>reset</code>	Can restart software processes by using the <code>restart</code> command.
<code>rollback</code>	Can use the <code>rollback</code> command to return to a previously committed configuration.
<code>routing</code>	Can view general routing, routing protocol, and routing policy configuration information in configuration mode and operational mode.
<code>routing-control</code>	Can view and configure general routing at the <code>[edit routing-options]</code> hierarchy level, routing protocols at the <code>[edit protocols]</code> hierarchy level, and routing policy information at the <code>[edit policy-options]</code> hierarchy level.
<code>secret</code>	Can view passwords and other authentication keys in the configuration.
<code>secret-control</code>	Can view and modify passwords and other authentication keys in the configuration.
<code>security</code>	Can view security configuration information in operational mode and configuration mode.
<code>security-control</code>	Can view and configure security information at the <code>[edit security]</code> hierarchy level.

Table 2: Login Class Permission Flags (Continued)

Permission Flag	Description
shell	Can start a local shell on the router or switch by using the <code>start shell</code> command.
snmp	Can view Simple Network Management Protocol (SNMP) configuration information in operational mode or configuration mode.
snmp-control	Can view and modify SNMP configuration information at the <code>[edit snmp]</code> hierarchy level.
system	Can view system-level information in operational mode or configuration mode.
system-control	Can view and modify system-level configuration information at the <code>[edit system]</code> hierarchy level.
trace	Can view trace file settings and configure trace file properties.
trace-control	Can modify trace file settings and configure trace file properties.
view	Can use various commands to display current system-wide, routing table, and protocol-specific values and statistics. Cannot view the secret configuration.
view-configuration	<p>Can view all of the configuration excluding secrets, system scripts, and event options.</p> <p>NOTE: Only users with the <code>maintenance</code> permission can view <code>commit script</code>, <code>op script</code>, or <code>event script</code> configuration.</p>

Deny or Allow Individual Commands and Statement Hierarchies

By default, all top-level CLI commands and statements have associated access privilege levels. Users can execute only those commands and view and configure only those statements for which they have access privileges. For each login class, you can explicitly deny or allow users the use of operational mode commands and configuration mode commands and configuration statement hierarchies that are otherwise allowed or denied by a permission bit.

Example: Create Login Classes with Specific Privileges

You define login classes to assign certain permissions or restrictions to groups of users, ensuring that sensitive commands are only accessible to the appropriate users. By default, Juniper Networks devices have four types of login classes with preset permissions: operator, read-only, superuser or super-user, and unauthorized.

You can create custom login classes to define different combinations of permissions that are not found in the default login classes. The following example shows three custom login classes, each with specific privileges and inactivity timers. Inactivity timers help protect network security by disconnecting a user from the network if the user is inactive for too long. Disconnecting the user prevents potential security risks that result when a user leaves an unattended account logged in to a switch or router. The permissions and inactivity timers shown here are only examples; you should customize the values to your organization.

The three login classes and their privileges are as follows. All three login classes use the same inactivity timer of 5 minutes.

- observation—Can only view statistics and the configuration
- operation—Can view and modify the configuration
- engineering—Unlimited access and control

```
[edit]
system {
  login {
    class observation {
      idle-timeout 5;
      permissions [ view ];
    }
    class operation {
      idle-timeout 5;
      permissions [ admin clear configure interface interface-control network
reset routing routing-control snmp snmp-control trace-control
firewall-control rollback ];
    }
    class engineering {
      idle-timeout 5;
      permissions all;
    }
  }
}
```

Login Settings

IN THIS SECTION

- [Display a System Login Announcement or Message | 9](#)
- [Display System Alarms Upon Login | 11](#)
- [Configure Login Tips | 12](#)
- [Configure the Timeout Value for Idle Login Sessions | 13](#)
- [Login Retry Options | 14](#)

Junos OS Evolved enables you to define various settings for users when they log in to a device. You (the system administrator) can configure:

- Messages or announcements to display before or after login
- Whether to display system alarms upon login
- Login tips
- Timeout values for idle sessions
- Whether to lock a user account after a number of failed authentication attempts

Display a System Login Announcement or Message

Sometimes you want to make announcements only to authorized users after they log in to a device. For example, you might want to announce an upcoming maintenance event. At other times, it might be appropriate to display a message, such as a security warning, to any user that connects to the device.

By default, Junos OS Evolved does not display any login message or announcement. You can configure the device to display a login message or announcement by including the `message` statement or the `announcement` statement at the `[edit system login]` hierarchy level. Whereas the device displays a login *message* after a user connects to the device but before the user logs in, it displays an *announcement* only after the user successfully logs in to the device.

You can format the message or announcement text using the following special characters. If the text contains spaces, enclose it in quotation marks:

- \n—New line
- \t—Horizontal tab
- \'—Single quotation mark
- \"—Double quotation mark
- \\—Backslash

To configure an announcement that only authorized users can see and a message that any user can see:

1. Include the announcement statement and the message statement at the [edit system login] hierarchy level.

```
[edit system login]
user@host# set announcement text
user@host# set message text
```

For example:

```
system {
  login {
    announcement "\tJuly 27th 1:00 AM to 8:00\n\nPlanned Network Maintenance\n
\nAFFECTED LOCATIONS: Sunnyvale\n\nPLANNED ACTIVITY: Upgrade all 6200 switch firmware to
the Enterprise TAC recommended firmware version\n\nPURPOSE: This activity will help to
minimize the impact of unplanned power outages as well as address known issues within our
currently installed firmware version(s)\n\nWHAT TO EXPECT: During the maintenance window for
your site, the office network will not be available.\n\n";
    message "\n\tAcme Router Lab\n\n\tUNAUTHORIZED USE OF THIS ROUTER\n\tIS STRICTLY
PROHIBITED!\n\n\tPlease contact \'jsmith@example.com\' to gain\n\taccess to this equipment if
you need authorization.\n\n"
  }
}
```

2. Commit the configuration.

```
[edit system login]
user@host# commit
```

3. Connect to the device to verify the presence of the new message.

The preceding configuration example displays the following login message after the user connects to the device. The example displays the announcement after the user logs in:

```
server% ssh host
```

```
Acme Router Lab
```

```
UNAUTHORIZED USE OF THIS ROUTER  
IS STRICTLY PROHIBITED!
```

```
Please contact 'jsmith@example.com' to gain  
access to this equipment if you need authorization.
```

```
Password:
```

```
July 27th 1:00 AM to 8:00
```

```
Planned Network Maintenance
```

```
AFFECTED LOCATIONS: Sunnyvale
```

```
PLANNED ACTIVITY: Upgrade all 6200 switch firmware to the Enterprise TAC recommended firmware  
version
```

```
PURPOSE: This activity will help to minimize the impact of unplanned power outages as well as  
address known issues within our currently installed firmware version(s)
```

```
WHAT TO EXPECT: During the maintenance window for your site, the office network will not be  
available.
```

Display System Alarms Upon Login

You can configure Juniper Networks devices to execute the `show system alarms` command whenever a user in a given login class logs in to the device.

To display alarms whenever a user in a specific login class logs in to the device:

1. Configure the login-alarms statement for the appropriate login class.

```
[edit system login class class-name]
user@host# set login-alarms
```

For example, to display alarms whenever a user in the `admin` login class logs in to the device:

```
[edit system login class admin]
user@host# set login-alarms
```

2. Commit the configuration.

```
[edit system login class class-name]
user@host# commit
```

When a user in the given login class logs in to the device, the device displays the current alarms.

```
$ ssh user@host.example.com
Password:
--- JUNOS 21.1R2.6-EVO Linux (none) 4.8.28-WR2.2.1_standard-g3999f55 #1 SMP PREEMPT Fri Jun 4
00:19:58 PDT 2021 x86_64 x86_64 x86_64 GNU/Linux

2 alarms currently active
Alarm time          Class  Description
2021-07-22 15:00:14 PDT  Minor  port-1/0/0: Optics does not support configured speed
2021-07-22 15:00:14 PDT  Minor  port-1/0/1: Optics does not support configured speed
```

Configure Login Tips

You can configure the Junos OS Evolved CLI to display a tip whenever a user in the given login class logs in to the device. The device does not display tips by default.

To enable tips:

1. Configure the login-tip statement at the [edit system login class *class-name*] hierarchy level.

```
[edit system login class class-name]  
user@host# set login-tip
```

2. Commit the configuration.

```
[edit system login class class-name]  
user@host# commit
```

When you configure the login-tip statement, the device displays a tip to any user in the specified class who logs in to the device.

```
$ ssh user@host.example.com  
Password:  
  
JUNOS tip:  
In configuration mode, the [edit] banner displays the current location  
in the configuration hierarchy.  
  
user@host>
```

Configure the Timeout Value for Idle Login Sessions

An idle login session is one in which the CLI displays the operational mode or configuration mode prompt but there is no input from the keyboard. By default, a login session remains established until a user logs out of the device, even if that session is idle. To close idle sessions automatically, you must configure a time limit for each login class. If a session established by a user in that class remains idle for the configured time limit, the session automatically closes. Automatically closing idle login sessions helps to prevent malicious users from gaining access to the device and performing operations with an authorized user account.

You can configure an idle timeout only for user-defined classes. You cannot configure this option for the system predefined classes: operator, read-only, super-user or superuser, and unauthorized.

To define the timeout value for idle login sessions:

1. Specify the number of minutes that a session can be idle before the system automatically closes the session.

```
[edit system login class class-name]  
user@host# set idle-timeout minutes
```

For example, to automatically disconnect idle sessions of users in the `admin` class after fifteen minutes:

```
[edit system login class admin]  
user@host# set idle-timeout 15
```

2. Commit the configuration.

```
[edit system login class class-name]  
user@host# commit
```

If you configure a timeout value, the CLI displays messages similar to the following when timing out an idle user. The CLI starts displaying these messages 5 minutes before disconnecting the user.

```
user@host> Session will be closed in 5 minutes if there is no activity.  
Warning: session will be closed in 1 minute if there is no activity  
Warning: session will be closed in 10 seconds if there is no activity  
Idle timeout exceeded: closing session
```

If you configure a timeout value, the session closes after the specified time elapses, except in the following cases:

- The user is running the `ssh` or `telnet` command.
- The user is logged into the local UNIX shell.
- The user is monitoring interfaces using the `monitor interface` or the `monitor traffic` command.

Login Retry Options

You can configure login retry options on Juniper Network devices to protect the devices from malicious users. You can configure the following options:

- The number of times a user can enter invalid login credentials before the system closes the connection.
- Whether and for how long to lock a user account after the user reaches the threshold of failed authentication attempts.

Limiting the login attempts and locking the user account help to protect the device from malicious users attempting to access the system by guessing the password of an authorized user account. You can unlock the user account or define a time period for the user account to remain locked.

You configure login retry options at the `[edit system login retry-options]` hierarchy level. Junos OS Evolved allows three unsuccessful login attempts before the device disconnects the user. You cannot modify the default threshold for failed login attempts.

The `lockout-period` statement instructs the device to lock the user account for the specified amount of time if the user reaches the threshold of unsuccessful login attempts. The lock prevents the user from performing activities that require authentication, until the lockout time period has elapsed or a system administrator manually clears the lock. Any existing locks are ignored when the user attempts to log in from the local console.

To configure login retry options:

1. Configure the number of minutes that the user account remains locked after a user reaches the threshold of failed login attempts.

```
[edit system login retry-options]
user@host# set lockout-period minutes
```

For example, to lock a user account for 120 minutes after a user reaches the threshold of failed login attempts:

```
[edit system login retry-options]
user@host# set lockout-period 120
```

2. Commit the configuration.

```
[edit system login retry-options]
user@host# commit
```


2

CHAPTER

User Accounts

User Accounts | 18

Administrative Roles | 28

User Access Privileges | 41

User Accounts

IN THIS SECTION

- [User Accounts Overview | 18](#)
- [Example: Configure New User Accounts | 20](#)
- [Configure User Accounts in a Configuration Group | 24](#)

Junos OS Evolved enables you (the system administrator) to create accounts for router, switch, and security users. All users belong to one of the system login classes.

You create user accounts so that users can access a router, switch, or security device. All users must have a predefined user account before they can log in to the device. You create user accounts and then define the login name and identifying information for each user account.

User Accounts Overview

User accounts provide one way for users to access a device. For each account, you define the user's login name, password, and any additional user information. After you have created an account, the software creates a home directory for the user.

An account for the user `root` is always present in the configuration. You can configure the password for `root` using the `root-authentication` statement.

While it is common to use remote authentication servers to centrally store information about users, it is also good practice to configure at least one non-root user on each device. This way, you can still access the device if its connection to the remote authentication server is disrupted. This non-root user usually has a generic name such as `admin`.

For each user account, you can define the following:

- **Username (Required):** Name that identifies the user. It must be unique. Avoid using spaces, colons, or commas in the username. The username can include up to 64 characters.
- **User's full name: (Optional)** If the full name contains spaces, enclose it in quotation marks. Avoid the use of colons or commas.

- **User identifier (UID):** (Optional) Numeric identifier that is associated with the user account name. The UID is assigned automatically when you commit the configuration, so you do not need to set it manually. However, if you choose to configure the UID manually, use a unique value in the range from 100 through 64,000.
- **User's access privilege:** (Required) One of the login classes you defined in the class statement at the [edit system login] hierarchy or one of the default login classes.
- **Authentication method or methods and passwords for device access** (Required): You can use a SSH key, a Message Digest 5 (MD5) password, or a plain-text password that Junos OS Evolved encrypts using MD5-style encryption before entering it in the password database. For each method, you can specify the user's password. If you configure the plain-text-password option, you receive a prompt to enter and confirm the password:

```
[edit system login user username]
user@host# set authentication plain-text-password
New password: type password here
Retype new password: retry password here
```

To create valid plain-text passwords, make sure that they:

- Contain between 6 and 128 characters.
- Include most character classes (uppercase letters, lowercase letters, numbers, punctuation marks, and other special characters) but do not include control characters.
- Contain at least one change of case or character class.

For SSH authentication, you can copy the contents of an SSH key file into the configuration. You can also configure SSH key information directly. Use the load-key-file statement to load an SSH key file that was generated previously, (for example, by using ssh-keygen). The load-key-file argument is the path to the file location and name. The load-key-file statement loads RSA (SSH version 1 and SSH version 2) public keys. The contents of the SSH key file are copied into the configuration immediately after you configure the load-key-file statement.

Avoid using the following Transport Layer Security (TLS) version and cipher suite (RSA host key) combinations, which will fail:

With RSA host keys:

- TLS_1.0@DHE-RSA-AES128-SHA
- TLS_1.0@DHE-RSA-AES256-SHA

For each user account and for root logins, you can configure more than one public RSA key for user authentication. When a user logs in using a user account or as root, the configured public keys are referenced to determine whether the private key matches any of the user accounts.

To view the SSH key entries, use the configuration mode `show` command. For example:

```
[edit system login user boojum]
user@host# set authentication load-key-file my-host:.ssh/id_rsa.pub
.file.19692          |          0 KB |   0.3 kB/s | ETA: 00:00:00 | 100%
[edit system login user boojum]
user@host# show
authentication {
    ssh-rsa "$ABC123"; # SECRET-DATA
}
```

Example: Configure New User Accounts

IN THIS SECTION

- [Requirements | 20](#)
- [Overview | 20](#)
- [Configuration | 21](#)
- [Verification | 24](#)

This example shows how to configure new user accounts.

Requirements

You do not need any special configurations before using this feature.

Overview

You can add new user accounts to the device's local database. For each account, you (the system administrator) define a login name and password for the user and specify a login class for access privileges. The login password must meet the following criteria:

- The password must be at least six characters long.

- You can include most character classes in the password (alphabetic, numeric, and special characters), but not control characters.
- The password must contain at least one change of case or character class.

In this example, you create a login class named `operator-and-boot` and allow it to reboot the device. You can define any number of login classes. Then, allow the `operator-and-boot` login class to use commands defined in the following bits:

- `clear`
- `network`
- `reset`
- `trace`
- `view permission`

Next, create user accounts to enable access to the device. Set the username as `randomuser` and the login class as `superuser`. Finally, define the encrypted password for the user.

Configuration

IN THIS SECTION

- [Procedure | 21](#)

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` in configuration mode.

```
set system login class operator-and-boot allow-commands "request system reboot"
set system login class operator-and-boot permissions [clear network reset trace view]
set system login user randomuser class superuser authentication encrypted-password $1$ABC123
```

Step-by-Step Procedure

To configure new users:

1. Set the name of the login class and allow the use of the reboot command.

```
[edit system login]
user@host# set class operator-and-boot allow-commands "request system reboot"
```

2. Set the permission bits for the login class.

```
[edit system login]
user@host# set class operator-and-boot permissions [clear network reset trace view]
```

3. Set the username, login class, and encrypted password for the user.

```
[edit system login]
user@host# set userrandomuser class superuser authentication encrypted-password $1$ABC123
```

Results

In configuration mode, confirm your configuration by entering the `show system login` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show system login
    class operator-and-boot {
    permissions [ clear network reset trace view ];
    allow-commands "request system reboot";
    }
user randomuser {
    class superuser;
    authentication {
    encrypted-password "$1$ABC123";
    }
}
```

The following example shows how to create accounts for four users. It also shows how to create an account for the template user `remote`. All users use one of the default system login classes.

```
[edit]
system {
  login {
    user philip {
      full-name "Philip of Macedonia";
      uid 1001;
      class super-user;
      authentication {
        encrypted-password "$ABC123";
      }
    }
    user alexander {
      full-name "Alexander the Great";
      uid 1002;
      class operator;
      authentication {
        encrypted-password "$ABC123";
      }
    }
    user darius {
      full-name "Darius King of Persia";
      uid 1003;
      class operator;
      authentication {
        ssh-rsa "1024 37 12341234@ecbatana.per";
      }
    }
    user anonymous {
      class unauthorized;
    }
    user remote {
      full-name "All remote users";
      uid 9999;
      class read-only;
    }
  }
}
```

After you configure the device, enter `commit` in configuration mode.

Verification

IN THIS SECTION

- [Verify the New Users Configuration | 24](#)

Confirm that the configuration is working properly.

Verify the New Users Configuration

Purpose

Verify that the new users are configured.

Action

Log in to the device with the new user account or accounts and password to confirm that you have access.

Configure User Accounts in a Configuration Group

To make it easier to configure the same user accounts on multiple devices, configure the accounts inside of a configuration group. The examples shown here are in a configuration group called `global`. Using a configuration group for your user accounts is optional.

To create a user account:

1. Add a new user, using the user's assigned account login name.

```
[edit groups global]  
user@host# edit system login user username
```

2. (Optional) Configure a descriptive name for the account.

If the name includes spaces, enclose the entire name in quotation marks.

```
[edit groups global system login user user-name]
user@host# set full-name complete-name
```

For example:

```
user@host# show groups
global {
    system {
        login {
            user admin {
                full-name "general administrator";
            }
        }
    }
}
```

3. (Optional) Set the user identifier (UID) for the account.

As with UNIX systems, the UID enforces user permissions and file access. If you do not set the UID, the software assigns one for you. The format of the UID is a number between 100 and 64,000.

```
[edit groups global system login user user-name]
user@host# set uid uid-value
```

For example:

```
user@host# show groups
global {
    system {
        login {
            user admin {
                uid 9999;
            }
        }
    }
}
```

4. Assign the user to a login class.

You can define your own login classes or assign one of the predefined login classes.

The predefined login classes are as follows:

- super-user—all permissions
- operator—clear, network, reset, trace, and view permissions
- read-only—view permissions
- unauthorized—no permissions

```
[edit groups global system login user user-name]
user@host# set class class-name
```

For example:

```
user@host# show groups
global {
  system {
    login {
      user admin {
        class super-user;
      }
    }
  }
}
```

5. Use one of the following methods to configure the user password:

- To enter a clear-text password that the system encrypts for you, use the following command to set the user password:

```
[edit groups global system login user user-name]
user@host# set authentication plain-text-password
New Password: type password here
Retype new password: retype password here
```

As you enter the password in plain text, the software encrypts it. You do not need to configure the software to encrypt the password. Plain-text passwords are hidden and marked as ## SECRET-DATA in the configuration.

- To enter a password that is encrypted, use the following command to set the user password:



CAUTION: Do not use the encrypted-password option unless the password is *already* encrypted and you are entering the encrypted version of the password.

If you accidentally configure the encrypted-password option with a plain-text password or with blank quotation marks (" "), you will not be able to log in to the device as this user.

```
[edit groups global system login user user-name]
user@host# set authentication encrypted-password "password"
```

- To load previously generated public keys from a named file at a specified URL location, use the following command:

```
[edit groups global system login user user-name]
user@host# set authentication load-key-file URL filename
```

- To enter an SSH public string, use the following command:

```
[edit groups global system login user user-name]
user@host# set authentication (ssh-ecdsa | ssh-ed25519 | ssh-rsa) authorized-key
```

6. At the top level of the configuration, apply the configuration group.

If you use a configuration group, you must apply it for it to take effect.

```
[edit]
user@host# set apply-groups global
```

7. Commit the configuration.

```
user@host# commit
```

8. To verify the configuration, log out and log back in as the new user.

Administrative Roles

IN THIS SECTION

- [How to Design Administrative Roles | 28](#)
- [Example: How to Configure Administrative Roles | 30](#)
- [How to Configure a Local Administrator Account | 40](#)

Junos OS Evolved enables you to define a system user to act as a specific kind of administrator for the system. You can assign an administrative role to a user by configuring a login class to have the administrative role attributes. You can assign one of the role attributes such as audit-officer, crypto-officer, security-officer, ids-officer to an administrative user.

How to Design Administrative Roles

A system user can be a member of a class that allows the user to act as a specific kind of administrator for the system. Requiring a specific role to view or modify an item restricts the extent of information a user can obtain from the system. It also limits how much of the system is open to modification or observation by a user. You (the system administrator) should use the following guidelines when you are designing administrative roles:

- Do not allow any user to log in to the system as root.
- Restrict each user to the smallest set of privileges needed to perform the user's duties.
- Do not allow any user to belong to a login class containing the `shell` permission flag. The `shell` permission flag allows users to run the `start shell` command from the CLI.
- Allow users to have rollback permissions. Rollback permissions allow users to undo an action performed by an administrator but does not allow them to commit the changes.

You can assign an administrative role to a user by configuring a login class to have the privileges required for the role. You can configure each class to allow or deny access to configuration statements and commands by name. These restrictions override and take precedence over any permission flags also configured in the class. You can assign one of the following role attributes to an administrative user:

- **Crypto-administrator**—Allows the user to configure and monitor cryptographic data.

- **Security-administrator**—Allows the user to configure and monitor security data.
- **Audit-administrator**—Allows the user to configure and monitor audit data.
- **IDS-administrator**—Allows the user to monitor and clear the intrusion detection service (IDS) security logs.

Each role can perform the following specific management functions:

- **Cryptographic Administrator**
 - Configures the cryptographic self-test.
 - Modifies the cryptographic security data parameters.
- **Audit Administrator**
 - Configures and deletes the audit review search-and-sort feature.
 - Searches and sorts audit records.
 - Configures search and sort parameters.
 - Manually deletes audit logs.
- **Security Administrator**
 - Invokes, determines, and modifies the cryptographic self-test behavior.
 - Enables, disables, determines, and modifies the audit analysis and audit selection functions, and configures the device to automatically delete audit logs.
 - Enables or disables security alarms.
 - Specifies limits for quotas on Transport Layer connections.
 - Specifies the limits, network identifiers, and time periods for quotas on controlled connection-oriented resources.
 - Specifies the network addresses permitted to use Internet Control Message Protocol (ICMP) or Address Resolution Protocol (ARP).
 - Configures the time and date used in time stamps.
 - Queries, modifies, deletes, and creates the information flow or access control rules and attributes for the unauthenticated information flow security function policy (SFP), the authenticated information flow security function policy, the unauthenticated device services, and the discretionary access control policy.

- Specifies initial values that override default values when object information is created under unauthenticated information flow SFP, the authenticated information flow SFP, the unauthenticated target of evaluation (TOE) services, and the discretionary access control policy.
- Creates, deletes, or modifies the rules that control the address from which management sessions can be established.
- Specifies and revokes security attributes associated with the users, subjects, and objects.
- Specifies the percentage of audit storage capacity at which the device alerts administrators.
- Handles authentication failures and modifies the number of failed authentication attempts through SSH or from the CLI that can occur before progressive throttling is enforced for further authentication attempts and before the connection is dropped.
- Manages basic network configuration of the device.
- **IDS Administrator**—Specifies IDS security alarms, intrusion alarms, audit selections, and audit data.

You must set the security-role attribute in the classes created for these administrative roles. This attribute restricts which users can show and clear the security logs, actions that cannot be performed through configuration alone.

For example, you must set the security-role attribute in the `ids-admin` class created for the IDS administrator role if you want to restrict clearing and showing IDS logs to the IDS administrator role. Likewise, you must set the security-role to one of the other admin values to restrict that class from being able to clear and show non-IDS logs only.

NOTE: When a user deletes an existing configuration, the configuration statements under the hierarchy level of the deleted configuration (the child objects that the user does not have permission to modify) remain in the device.

Example: How to Configure Administrative Roles

IN THIS SECTION

- [Requirements | 31](#)
- [Overview | 31](#)
- [Configuration | 31](#)

This example shows how to configure individual administrative roles for a distinct, unique set of privileges apart from all other administrative roles.

Requirements

No action beyond device initialization is required before configuring this feature.

Overview

This example illustrates how to configure four admin user roles:



- audit-officer of the class audit-admin
- crypto-officer of the class crypto-admin
- security-officer of the class security-admin
- ids-officer of the class ids-admin

When a security-admin class is configured, the privileges for creating administrators are revoked from the user who created the security-admin class. Creation of new users and logins is at the discretion of the security-officer.

In this example, you create the four administrative user roles shown in the preceding list (audit admin, crypto admin, security admin, and ids admin). For each role, you assign relevant permission flags for the role. You then allow or deny access to configuration statements and commands by name for each administrative role. These specific restrictions take precedence over the permission flags configured in the class. For example, only the crypto-admin can run the `request system set-encryption-key` command, which requires having the security permission flag to access it. Only the security-admin can include the `system time-zone` statement in the configuration, which requires having the system-control permission flag.

Configuration

IN THIS SECTION

-  [Procedure | 32](#)
-  [Results | 36](#)

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` in configuration mode.

```
set system login class audit-admin permissions security
set system login class audit-admin permissions trace
set system login class audit-admin permissions maintenance
set system login class audit-admin allow-commands "^clear (log|security log)"
set system login class audit-admin deny-commands "^clear (security alarms|system login lockout)|^file (copy|delete|rename)|^request (security|system set-encryption-key)|^rollback|^set date|^show security (alarms|dynamic-policies|match-policies|policies)|^start shell";
set system login class audit-admin security-role audit-administrator
set system login class crypto-admin permissions admin-control
set system login class crypto-admin permissions configure
set system login class crypto-admin permissions maintenance
set system login class crypto-admin permissions security-control
set system login class crypto-admin permissions system-control
set system login class crypto-admin permissions trace
set system login class crypto-admin allow-commands "^request system set-encryption-key"
set system login class crypto-admin deny-commands "^clear (log|security alarms|security log|system login lockout)|^file (copy|delete|rename)|^rollback|^set date|^show security (alarms|dynamic-policies|match-policies|policies)|^start shell"
set system login class crypto-admin allow-configuration-regexps ["security (ike|ipsec) (policy|proposal)" "security ipsec ^vpn$ .* manual (authentication|encryption|protocol|spi)" "system fips self-test after-key-generation"]
set system login class crypto-admin security-role crypto-administrator
set system login class security-admin permissions all
set system login class security-admin deny-commands "^clear (log|security log)|^(clear|show) security alarms alarm-type idp|^request (security|system set-encryption-key)|^rollback|^start shell"
set system login class security-admin deny-configuration-regexps ["security alarms potential-violation idp" "security (ike|ipsec) (policy|proposal)" "security ipsec ^vpn$ .* manual (authentication|encryption|protocol|spi)" "security log cache" "security log exclude .* event-id IDP_.*" "system fips self-test after-key-generation"]
set system login class security-admin security-role security-administrator
set system login class ids-admin permissions configure
set system login class ids-admin permissions security-control
set system login class ids-admin permissions trace
```

```

set system login class ids-admin permissions maintenance
set system login class ids-admin allow-configuration-regexps ["security alarms potential-
violation idp" "security log exclude .* event-id IDP_.*"]
set system login class ids-admin deny-commands "^clear log|^(clear|show) security alarms (alarm-
id|all|newer-than|older-than|process|severity)|^(clear|show) security alarms alarm-type
(authentication|cryptographic-self-test|decryption-failures|encryption-failures|ike-phase1-
failures|ike-phase2-failures|key-generation-self-test|non-cryptographic-self-test|policy|replay-
attacks)|^file (copy|delete|rename)|^request (security|system set-encryption-key)|^rollback|^set
date|^show security (dynamic-policies|match-policies|policies)|^start shell"
set system login class ids-admin deny-configuration-regexps ["security alarms potential-
violation (authentication|cryptographic-self-test|decryption-failures|encryption-failures|ike-
phase1-failures|ike-phase2-failures|key-generation-self-test|non-cryptographic-self-test|policy|
replay-attacks)"]
set system login class ids-admin security-role ids-admin
set system login user audit-officer class audit-admin
set system login user crypto-officer class crypto-admin
set system login user security-officer class security-admin
set system login user ids-officer class ids-admin
set system login user audit-officer authentication plain-text-password
set system login user crypto-officer authentication plain-text-password
set system login user security-officer authentication plain-text-password
set system login user ids-officer authentication plain-text-password

```

Step-by-Step Procedure

To configure administrative roles:

1. Create the audit-admin login class.

```

[edit]
user@host# edit system login class audit-admin
[edit system login class audit-admin]
user@host# set permissions security
user@host# set permissions trace
user@host# set permissions maintenance

```

2. Configure the audit-admin login class restrictions.

```

[edit system login class audit-admin]
user@host# set allow-commands "^clear (log|security log)"
user@host# set deny-commands "^clear (security alarms|system login logout)|^file (copy|

```

```
delete|rename)|^request (security|system set-encryption-key)|^rollback|^set date|^show
security (alarms|dynamic-policies|match-policies|policies)|^start shell"
user@host# set security-role audit-administrator
```

3. Create the crypto-admin login class.

```
[edit]
user@host# edit system login class crypto-admin
[edit system login class crypto-admin]
user@host# set permissions admin-control
user@host# set permissions configure
user@host# set permissions maintenance
user@host# set permissions security-control
user@host# set permissions system-control
user@host# set permissions trace
```

4. Configure the crypto-admin login class restrictions.

```
[edit system login class crypto-admin]
user@host# set allow-commands "^request system set-encryption-key"
user@host# set deny-commands "^clear (log|security alarms|security log|system login
lockout)|^file (copy|delete|rename)|^rollback|^set date|^show security (alarms|dynamic-
policies|match-policies|policies)|^start shell"
user@host# set allow-configuration-regexps ["security (ike|ipsec) (policy|proposal)"
"security ipsec ^vpn$ .* manual (authentication|encryption|protocol|spi)" "system fips self-
test after-key-generation"]
user@host# set security-role crypto-administrator
```

5. Create the security-admin login class.

```
[edit]
user@host# edit system login class security-admin
[edit system login class security-admin]
user@host# set permissions all
```

6. Configure the security-admin login class restrictions.

```
[edit system login class security-admin]
user@host# set deny-commands "^clear (log|security log)|^(clear|show) security alarms alarm-
```

```

type idp|^request (security|system set-encryption-key)|^rollback|^start shell"
user@host# set deny-configuration-regexps ["security alarms potential-violation idp"
"security (ike|ipsec) (policy|proposal)" "security ipsec ^vpn$ .* manual (authentication|
encryption|protocol|spi)" "security log cache" "security log exclude .* event-id IDP_.*"
"system fips self-test after-key- generation"]
user@host# set security-role security-administrator

```

7. Create the ids-admin login class.

```

[edit]
user@host# edit system login class ids-admin
[edit system login class ids-admin]
user@host# set permissions configure
user@host# set permissions maintenance
user@host# set permissions security-control
user@host# set permissions trace

```

8. Configure the ids-admin login class restrictions.

```

[edit system login class ids-admin]
user@host# set allow-configuration-regexps ["security alarms potential-violation idp"
"security log exclude .* event-id IDP_.*"
user@host# set deny-commands "^clear log|^(clear|show) security alarms (alarm-id|all|newer-
than|older-than|process|severity)|^(clear|show) security alarms alarm-type (authentication|
cryptographic-self-test|decryption-failures|encryption-failures|ike-phase1-failures|ike-
phase2-failures|key-generation-self-test|non-cryptographic-self-test|policy|replay-attacks)|
^file (copy|delete|rename)|^request (security|system set-encryption-key)|^rollback|^set
date|^show security (dynamic-policies|match-policies|policies)|^start shell"
user@host# set deny-configuration-regexps ["security alarms potential-violation
(authentication|cryptographic-self-test|decryption-failures|encryption-failures|ike-phase1-
failures|ike-phase2-failures|key-generation-self-test|non-cryptographic-self-test|policy|
replay-attacks)"]
user@host# set security-role ids-administrator

```

9. Assign users to the roles.

```

[edit]
user@host# edit system login
[edit system login]
user@host# set user audit-officer class audit-admin

```

```

user@host# set user crypto-officer class crypto-admin
user@host# set user security-officer class security-admin
user@host# set user ids-officer class ids-admin

```

10. Configure passwords for the users.

```

[edit system login]
user@host# set user audit-officer authentication plain-text-password
user@host# set user crypto-officer authentication plain-text-password
user@host# set user security-officer authentication plain-text-password
user@host# set user ids-officer authentication plain-text-password

```

Results

In configuration mode, confirm your configuration by entering the **show system** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```

[edit]
user@host# show system
system {
  login {
    class audit-admin {
      permissions [ maintenance security trace ];
      allow-commands "^clear (log|security log)";
      deny-commands "^clear (security alarms|system login lockout)|^file (copy|delete|
rename)|^request (security|system set-encryption-key)|^rollback|^set date|^show security (alarms|
dynamic-policies|match-policies|policies)|^start shell";
      security-role audit-administrator;
    }
    class crypto-admin {
      permissions [ admin-control configure maintenance security-control system-control
trace ];
      allow-commands "^request (system set-encryption-key)";
      deny-commands "^clear (log|security alarms|security log|system login lockout)|^file
(copy|delete|rename)|^rollback|^set date|^show security (alarms|dynamic-policies|match-policies|
policies)|^start shell";
      allow-configuration-regexps [ "security (ike|ipsec) (policy|proposal)" "security
ipsec ^vpn$ .* manual (authentication|encryption|protocol|spi)" "system fips self-test after-key-
generation" ];

```

```

        security-role crypto-administrator;
    }
    class security-admin {
        permissions [all];
        deny-commands "^clear (log|security log)|^(clear|show) security alarms alarm-type
idp|^request (security|system set-encryption-key)|^rollback|^start shell";
        deny-configuration-regexps [ "security alarms potential-violation idp" "security
(ike|ipsec) (policy|proposal)" "security ipsec ^vpn$ .* manual (authentication|encryption|
protocol|spi)" "security log exclude .* event-id IDP_.*" "system fips self-test after-key-
generation" ];
        security-role security-administrator;
    }
    class ids-admin {
        permissions [ configure maintenance security-control trace ];
        deny-commands "^clear log|^(clear|show) security alarms (alarm-id|all|newer-than|
older-than|process|severity)|^(clear|show) security alarms alarm-type
(authentication | cryptographic-self-test | decryption-failures | encryption-failures
| ike-phase1-failures | ike-phase2-failures|key-generation-self-test |
non-cryptographic-self-test |policy | replay-attacks) | ^file (copy|delete|rename)
|^request (security|system set-encryption-key) | ^rollback |
^set date | ^show security (dynamic-policies|match-policies|policies) |^start shell";
        allow-configuration-regexps [ "security alarms potential-violation idp" "security
log exclude .* event-id IDP_.*" ];
        deny-configuration-regexps "security alarms potential-violation (authentication|
cryptographic-self-test|decryption-
failures|encryption-failures|ike-phase1-failures|ike-phase2-failures|
key-generation-self-test|non-cryptographic-self-test|policy|replay-attacks)"
        security-role ids-administrator;
    }
    user audit-officer {
        class audit-admin;
        authentication {
            encrypted-password "$1$ABC123"; ## SECRET-DATA
        }
    }
    user crypto-officer {
        class crypto-admin;
        authentication {
            encrypted-password "$1$ABC123."; ## SECRET-DATA
        }
    }
    user security-officer {
        class security-admin;

```

```

        authentication {
            encrypted-password "$1$ABC123."; ##SECRET-DATA
        }
    }
    user ids-officer {
        class ids-admin;
        authentication {
            encrypted-password "$1$ABC123/"; ## SECRET-DATA
        }
    }
}

```

After you configure the device, enter **commit** in configuration mode.

Verification

IN THIS SECTION

- [Verify the Login Permissions | 38](#)

Confirm that the configuration is working properly.

Verify the Login Permissions

Purpose

Verify the login permissions for the current user.

Action

In operational mode, enter the `show cli authorization` command to verify the user's login permissions.

```

user@host> show cli authorization
Current user: 'example' class 'super-user'
Permissions:
  admin      -- Can view user accounts
  admin-control-- Can modify user accounts
  clear      -- Can clear learned network info

```

```

configure  -- Can enter configuration mode
control    -- Can modify any config
edit       -- Can edit full files
field      -- Can use field debug commands
floppy     -- Can read and write the floppy
interface  -- Can view interface configuration
interface-control-- Can modify interface configuration
network    -- Can access the network
reset      -- Can reset/restart interfaces and daemons
routing    -- Can view routing configuration
routing-control-- Can modify routing configuration
shell      -- Can start a local shell
snmp       -- Can view SNMP configuration
snmp-control-- Can modify SNMP configuration
system     -- Can view system configuration
system-control-- Can modify system configuration
trace      -- Can view trace file settings
trace-control-- Can modify trace file settings
view       -- Can view current values and statistics
maintenance -- Can become the super-user
firewall   -- Can view firewall configuration
firewall-control-- Can modify firewall configuration
secret     -- Can view secret statements
secret-control-- Can modify secret statements
rollback   -- Can rollback to previous configurations
security   -- Can view security configuration
security-control-- Can modify security configuration
access     -- Can view access configuration
access-control-- Can modify access configuration
view-configuration-- Can view all configuration (not including secrets)
flow-tap   -- Can view flow-tap configuration
flow-tap-control-- Can modify flow-tap configuration
idp-profiler-operation-- Can Profiler data
pgcp-session-mirroring-- Can view pgcp session mirroring configuration
pgcp-session-mirroring-control-- Can modify pgcp session mirroring configuration
tion
storage    -- Can view fibre channel storage protocol configuration
storage-control-- Can modify fibre channel storage protocol configuration
all-control -- Can modify any configuration
Individual command authorization:
Allow regular expression: none
Deny regular expression: none

```



```
Allow configuration regular expression: none
Deny configuration regular expression: none
```

This output summarizes the login permissions.

How to Configure a Local Administrator Account

Superuser privileges give a user permission to use any command on the router and are generally reserved for a select few users such as system administrators. You (the system administrator) need to protect the local administrator account with a password to prevent unauthorized users from gaining access to superuser commands. These superuser commands can be used to alter the system configuration. Users with RADIUS authentication should also configure a local password. If the RADIUS server does not respond, the login process reverts to local password authentication on the local administrator account.

The following example shows how to configure a password-protected local administration account called `admin` with superuser privileges:

```
[edit]
system {
  login {
    user admin {
      uid 1000;
      class superuser;
      authentication {
        encrypted-password "<PASSWORD>"; ## SECRET-DATA
      }
    }
  }
}
```

User Access Privileges

IN THIS SECTION

- [Access Privilege Levels Overview | 41](#)
- [Example: Configure User Permissions with Access Privilege Levels | 47](#)
- [Regular Expressions to Allow and Deny Operational Mode Commands, Configuration Statements, and Hierarchies | 51](#)
- [How to Define Access Privileges with allow-configuration and deny-configuration Statements | 72](#)
- [Example: Use Additive Logic with Regular Expressions to Specify Access Privileges | 75](#)
- [Example: Configure User Permissions with Access Privileges for Operational Mode Commands | 79](#)
- [Example: Configure User Permissions with Access Privileges for Configuration Statements and Hierarchies | 91](#)

You (the system administrator) grant users access or permissions to commands and configuration hierarchy levels and statements. Users can execute only those commands and view and configure only those statements for which they have access privileges. You can also use extended regular expressions to specify which operational mode commands, configuration statements, and hierarchies are allowed or denied for users. This practice prevents unauthorized users from executing sensitive commands or configuring statements that could cause damage to the network.

Access Privilege Levels Overview

IN THIS SECTION

- [Login Class Permission Flags | 42](#)
- [Allow and Deny Individual Commands and Statement Hierarchies for Login Classes | 46](#)

Each top-level CLI command and *configuration statement* has an associated access privilege level. Users can execute only those commands and configure and view only those statements for which they have access privileges. One or more permission flags define the access privileges for each login class.

For each login class, you can also explicitly allow or deny the use of operational mode and configuration mode commands and statement hierarchies that would otherwise be allowed or denied by a privilege level specified in the `permissions` statement.

Login Class Permission Flags

You use permission flags to grant a user access to operational mode commands and configuration hierarchy levels and statements. You configure permission flags for the user's login class at the `[edit system login class]` hierarchy level. When you specify a certain permission flag, the user gains access to the commands and to the configuration hierarchy levels and statements that correspond to that flag. To grant access to all commands and configuration statements, use the `all` permissions flag.

NOTE: Each command listed represents that command and all subcommands with that command as a prefix. Each *configuration statement* listed represents the top of the configuration hierarchy to which that flag grants access.

The `permissions` statement specifies one or more of the permission flags listed in [Table 3 on page 43](#). Permission flags are not cumulative. For each class you must list all the permission flags needed, including `view` to display information and `configure` to enter configuration mode. Two forms of permissions control a user's access to the individual parts of the configuration:

- "Plain" form—Provides read-only capability for that permission type. An example is `interface`.
- `-control` form—Provides read and write capability for that permission type. An example is `interface-control`.

For permission flags that grant access to configuration hierarchy levels and statements, the plain form flags grant read-only privilege to that configuration. For example, the `interface` permission flag grants read-only access to the `[edit interfaces]` hierarchy level. The `-control` form of the flag grants read-write access to that configuration. For example, the `interface-control` flag grants read-write access to the `[edit interfaces]` hierarchy level.

[Table 3 on page 43](#) lists the login class permission flags that you can configure by including the `permissions` statement at the `[edit system login class class-name]` hierarchy level.

The permission flags grant a specific set of access privileges. Each permission flag is listed with the operational mode or configuration mode commands and configuration hierarchy levels and statements for which that flag grants access.

Table 3: Login Class Permission Flags

Permission Flag	Description
access	Can view the access configuration in operational mode or configuration mode.
access-control	Can view and configure access information at the [edit access] hierarchy level.
admin	Can view user account information in operational mode or configuration mode.
admin-control	Can view user account information and configure it at the [edit system] hierarchy level.
all	Can access all operational mode commands and configuration mode commands. Can modify the configuration in all the configuration hierarchy levels.
clear	Can clear (delete) information that the device learns from the network and stores in various network databases (using the clear commands).
configure	Can enter configuration mode (using the configure command) and commit configurations (using the commit command).
control	Can perform all control-level operations—all operations configured with the -control permission flags.
field	Can view field debug commands. Reserved for debugging support.
firewall	Can view the <i>firewall filter</i> configuration in operational mode or configuration mode.
firewall-control	Can view and configure firewall filter information at the [edit firewall] hierarchy level.
floppy	Can read from and write to the removable media.
flow-tap	Can view the flow-tap configuration in operational mode or configuration mode.

Table 3: Login Class Permission Flags (Continued)

Permission Flag	Description
flow-tap-control	Can view and configure flow-tap information at the [edit services flow-tap] hierarchy level.
flow-tap-operation	<p>Can make flow-tap requests to the router or switch. For example, a Dynamic Tasking Control Protocol (DTCP) client must have flow-tap-operation permission to authenticate itself to Junos OS Evolved as an administrative user.</p> <p>NOTE: The flow-tap-operation option is not included in the all-control permissions flag.</p>
idp-profiler-operation	Can view profiler data.
interface	Can view the interface configuration in operational mode and configuration mode.
interface-control	<p>Can view chassis, <i>class of service</i> (CoS), groups, forwarding options, and interfaces configuration information. Can modify the configuration at the following hierarchy levels:</p> <ul style="list-style-type: none"> • [edit chassis] • [edit class-of-service] • [edit groups] • [edit forwarding-options] • [edit interfaces]
maintenance	Can perform system maintenance, including starting a local shell on the device and becoming the superuser in the shell (using the su root command) and halting and rebooting the device (using the request system commands).
network	Can access the network by using the ping, ssh, telnet, and traceroute commands.
pgcp-session-mirroring	Can view the pgcp session mirroring configuration.

Table 3: Login Class Permission Flags (Continued)

Permission Flag	Description
pgcp-session-mirroring-control	Can modify the pgcp session mirroring configuration.
reset	Can restart software processes by using the restart command.
rollback	Can use the rollback command to return to a previously committed configuration.
routing	Can view general routing, routing protocol, and routing policy configuration information in configuration mode and operational mode.
routing-control	Can view and configure general routing at the [edit routing-options] hierarchy level, routing protocols at the [edit protocols] hierarchy level, and routing policy information at the [edit policy-options] hierarchy level.
secret	Can view passwords and other authentication keys in the configuration.
secret-control	Can view and modify passwords and other authentication keys in the configuration.
security	Can view security configuration information in operational mode and configuration mode.
security-control	Can view and configure security information at the [edit security] hierarchy level.
shell	Can start a local shell on the router or switch by using the start shell command.
snmp	Can view Simple Network Management Protocol (SNMP) configuration information in operational mode or configuration mode.
snmp-control	Can view and modify SNMP configuration information at the [edit snmp] hierarchy level.

Table 3: Login Class Permission Flags (Continued)

Permission Flag	Description
system	Can view system-level information in operational mode or configuration mode.
system-control	Can view and modify system-level configuration information at the [edit system] hierarchy level.
trace	Can view trace file settings and configure trace file properties.
trace-control	Can modify trace file settings and configure trace file properties.
view	Can use various commands to display current system-wide, routing table, and protocol-specific values and statistics. Cannot view the secret configuration.
view-configuration	Can view all of the configuration excluding secrets, system scripts, and event options. NOTE: Only users with the maintenance permission can view commit script, op script, or event script configuration.

Allow and Deny Individual Commands and Statement Hierarchies for Login Classes

By default, all top-level CLI commands and configuration hierarchy levels have associated access privilege levels. Users can execute only those commands and view and configure only those statements for which they have access privileges. For each login class, you can explicitly allow and deny the use of operational mode and configuration mode commands and statement hierarchies that would otherwise be allowed or denied by a privilege level specified in the `permissions` statement.

Permission flags grant a user access to operational mode and configuration mode commands and to configuration hierarchy levels and statements. By specifying a specific permission flag on the user's login class at the [edit system login class] hierarchy level, you grant the user access to the corresponding commands and configuration hierarchy levels and statements. To grant access to all commands and configuration statements, use the `all` permissions flag.

You can explicitly allow or deny the use of commands and statements by configuring the `allow-commands`, `deny-commands`, `allow-configuration`, and `deny-configuration` statements for a login class. In the statements, you use extended regular expressions to define which commands and statements to allow or deny for users assigned to the class.

Example: Configure User Permissions with Access Privilege Levels

IN THIS SECTION

- Requirements | 47
- Overview | 47
- Configuration | 48
- Verification | 49

This example configures the user permissions for a login class. You configure user permissions for a login class to prevent users from performing unauthorized network actions. Users can execute only those commands and view and modify only those statements for which they have access privileges. This constraint prevents unauthorized users from executing sensitive commands or configuring statements that could cause damage to the network.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

Each top-level CLI command and each configuration statement has an access privilege level associated with it. When you configure a login class, you can explicitly allow or deny the use of operational mode and configuration mode commands and configuration statements. Users can execute only those commands and view and configure only those statements for which they have access privileges.

You define the access privileges for each login class by specifying one or more permission flags in the `permissions` statement. Permission flags grant a user access to commands, statements, and hierarchies. Permission flags are not cumulative. For each login class you must list all the permission flags needed, including `view` to display information and `configure` to enter configuration mode. By specifying a specific permission flag on the user's login class, you grant the user access to the corresponding commands, statements, and hierarchies. To grant access to all commands and configuration statements, use the `all` permissions flag. The permission flags provide read-only ("plain" form) and read and write (form that ends in `-control`) capability for a permission type.

NOTE: The `all` login class permission bits take precedence over extended regular expressions when a user issues a `rollback` command with the `rollback` permission flag enabled.

To configure user access privilege levels for a login class, include the `permissions` statement at the `[edit system login class class-name]` hierarchy level, followed by the permission flags. Configure multiple permissions as a space-separated list enclosed in square brackets:

```
[edit system login]
user@host# set class class-name permissions permission-flag
user@host# set class class-name permissions [flag1 flag2 flag3]
```

TIP: To view the available permissions, use the CLI's context-sensitive help and type a question mark (?) after the `permissions` statement:

```
[edit system login]
user@host# set class class-name permissions ?
```

Configuration

IN THIS SECTION

- [Configure User Permissions with Access Privilege Levels | 48](#)
- [Results | 49](#)

This example configures the `snmp-admin` login class. Users in this login class can configure and view SNMP parameters only.

Configure User Permissions with Access Privilege Levels

Step-by-Step Procedure

To configure access privileges for the login class:

1. Configure the `snmp-admin` login class with the `configure`, `snmp`, and `snmp-control` permission flags.

```
[edit system login]
user@host# set class snmp-admin permissions [configure snmp snmp-control]
```

The configured permission flags provide both read (snmp) and read-and-write (snmp-control) capability for SNMP, and this is the only allowed access privilege for this login class. All other access privileges are denied.

2. Create the user accounts that are assigned to the `snmp-admin` login class.

```
[edit system login]
user@host# set user snmpuser class snmp-admin authentication plain-text-password
New password:
Retype new password:
```

Results

In configuration mode, confirm your configuration by entering the `show system login` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show system login
class snmp-admin {
    permissions [ configure snmp snmp-control ];
}
user snmpuser {
    class snmp-admin;
    authentication {
        encrypted-password "$ABC123"; ## SECRET-DATA
    }
}
```

After configuring the device, enter `commit` in configuration mode.

Verification

IN THIS SECTION

- [Verify SNMP Configuration | 50](#)
- [Verify non-SNMP Configuration | 50](#)

Log in using a username assigned to the new login class, and confirm that the configuration is working properly.

Verify SNMP Configuration

Purpose

Verify that a user in the `snmp-admin` login class can configure SNMP.

Action

In configuration mode, configure SNMP statements at the `[edit snmp]` hierarchy level.

```
[edit snmp]
user@host# set name device1
user@host# set description switch1
user@host# set location Lab1
user@host# set contact example.com
user@host# commit
```

Meaning

The user in the `snmp-admin` login class is able to configure SNMP parameters. The user can configure these parameters because the permission flags specified for this class include both `snmp` (read capabilities) and `snmp-control` (read and write capabilities) permission bits.

Verify non-SNMP Configuration

Purpose

Verify that a user in the `snmp-admin` login class cannot modify non-SNMP configuration statements.

Action

In configuration mode, attempt to configure any non-SNMP statement, such as a statement in the interfaces hierarchy.

```
[edit]
user@host# edit interfaces
Syntax error, expecting <statement> or <identifier>.
```

Meaning

The user in the `snmp-admin` login class is not able to configure the `[edit interfaces]` hierarchy because the permission flags specified for this class do not allow it. In this case, the CLI issues an error message.

Regular Expressions to Allow and Deny Operational Mode Commands, Configuration Statements, and Hierarchies

IN THIS SECTION

- [Understanding the Allow and Deny Statements | 52](#)
- [Understanding the Allow and Deny Statement Syntax | 53](#)
- [Understanding the Allow and Deny Statement Precedence and Matching | 55](#)
- [Understanding the Allow and Deny Statement Rules | 56](#)
- [Understanding Differences for the *-regexps Statements | 57](#)
- [Using Regular Expressions on Remote Authorization Servers | 59](#)
- [Specify Regular Expressions | 63](#)
- [Regular Expressions Operators | 65](#)
- [Regular Expression Examples | 69](#)

This topic contains the following sections:

Understanding the Allow and Deny Statements

Each top-level CLI command and configuration statement hierarchy has an access privilege level associated with it. Each login class can explicitly allow or deny the use of operational mode and configuration mode commands and configuration hierarchies and statements that would otherwise be allowed or denied by a privilege level. Users can execute only those commands and view and configure only those statements for which they have access privileges.

The access privileges for each login class are defined by one or more permission flags specified in the `permissions` statement at the `[edit system login class class-name]` hierarchy level. In addition, you can allow or deny the use of specific commands and configuration hierarchies by defining extended regular expressions. You can specify the regular expressions by configuring the following statements for a login class:

- `allow-commands` and `deny-commands`—Allow or deny access to operational mode and configuration mode commands.
- `allow-configuration` and `deny-configuration`—Allow or deny access to specific configuration hierarchies.

NOTE: These statements perform slower matching, with more flexibility, especially in wildcard matching. However, it can take a very long time to evaluate all of the possible statements if a great number of full-path regular expressions or wildcard expressions are configured, possibly negatively affecting performance.

- `allow-commands-regexps` and `deny-commands-regexps`—Allow or deny access to particular commands using strings of regular expressions.
- `allow-configuration-regexps` and `deny-configuration-regexps`—Allow or deny access to specific configuration hierarchies using strings of regular expressions.

NOTE: If your existing configurations use the `allow/deny-commands` or `allow/deny-configuration` statements, using the same configuration options with the `allow/deny-commands-regexps` or `allow/deny-configuration-regexps` statements might not produce the same results. The search and match methods differ in the two forms of these statements.

Explicitly allowing commands and configuration statement hierarchies using the `allow/deny-*` statements adds to the permissions that the `permissions` statement already defines. Likewise, explicitly denying commands and configuration statement hierarchies using the `allow/deny-*` statements removes permissions that the `permissions` statement already defines.

For example, in the following configuration, the `configure` permission enables users in the login class to enter configuration mode. Additionally, the `allow-configuration` expression allows users to modify the configuration at the `[edit system services]` hierarchy level and commit it.

```
[edit system login class test]
user@host# set permissions configure allow-configuration "system services"
```

Similarly, in the following configuration, the login class user can perform all operations that the `all` permissions flag allows, except that the user cannot view or modify the configuration at the `[edit system services]` hierarchy level:

```
[edit system login class test]
user@host# set permissions all deny-configuration "system services"
```

Understanding the Allow and Deny Statement Syntax

You can configure an `allow/deny-*` statement only once in each login class. When you configure a statement:

- You can configure as many regular expressions as needed.
- Regular expressions are not case-sensitive

The `allow/deny-commands` statements are mutually exclusive with the `allow/deny-commands-regexps` statements, and the `allow/deny-configuration` statements are mutually exclusive with the `allow/deny-configuration-regexps` statements. For example, you cannot configure both `allow-configuration` and `allow-configuration-regexps` in the same login class.

To define access privileges to commands, specify extended regular expressions using the `allow-commands` and `deny-commands` statements. Enclose each complete standalone expression in parentheses (), and use the pipe (|) symbol to separate the expressions. Do not use spaces between regular expressions that are connected with the pipe symbol. The complete expression is enclosed in double quotation marks.

```
allow-commands "(cmd1)|(cmd2)|(cmdn)"
allow-configuration "(config1)|(config2)|(confign)"
```

For example:

```
[edit system login class test]
user@host# set allow-commands "(ping .*)|(traceroute .*)|(show .*)|(configure .*)|(edit)|(exit)|
(commit)|(rollback .*)"
```

You must use anchors when specifying complex regular expressions with the `allow-commands` statement. For example:

```
[edit system login]
user@host# set class test allow-commands "(^monitor)|(^ping)|(^show)|(^exit)"
```

To define access privileges to parts of the configuration hierarchy, specify extended regular expressions in the `allow-configuration` and `deny-configuration` statements. Enclose the full paths in parentheses (), and use the pipe (|) symbol to separate the expressions. Do not use spaces between regular expressions that are connected with the pipe symbol. The complete expression is enclosed in double quotation marks.

```
allow-configuration "(config1)|(config2)|(confign)"
```

For example:

```
[edit system login class test]
user@host# set deny-configuration "(system login class)|(system services)"
```

When specifying extended regular expressions using the `allow/deny-commands-regexps` or `allow/deny-configuration-regexps` statements, enclose each expression within quotation marks (" "), and separate the expressions using a space. Enclose multiple expressions in square brackets []. For example:

```
[edit system login class test]
user@host# set allow-configuration-regexps ["interfaces .* description .*" "interfaces .*
unit .* description .*" "interfaces .* unit .* family inet address .*" "interfaces.* disable"]
```

Modifiers such as `set`, `log`, and `count` are not supported within the regular expression string to be matched. If you use a modifier, then nothing is matched.

Correct configuration:

```
[edit system login class test]
user@host# set deny-commands protocols
```

Incorrect configuration:

```
[edit system login class test]
user@host# set deny-commands "set protocols"
```

Understanding the Allow and Deny Statement Precedence and Matching

By default, the `allow-commands` and `allow-configuration` regular expressions take precedence over `deny-commands` and `deny-configuration` expressions. Thus, if you configure the same command for both the `allow-commands` and `deny-commands` statements, then the `allow` operation takes precedence over the `deny` operation. Similarly, if you configure the same statement for both the `allow-configuration` and `deny-configuration` statements, then the `allow` operation takes precedence over the `deny` operation.

For instance, the following configuration allows a user in the `test` login class to install software using the `request system software add` command, even though the `deny-commands` statement includes the same command:

```
[edit system login class test]
user@host# set allow-commands "request system software add"
user@host# set deny-commands "request system software add"
```

Similarly, the following configuration allows a user in the `test` login class to view and modify the `[edit system services]` configuration hierarchy, even though the `deny-configuration` statement includes the same hierarchy:

```
[edit system login class test]
user@host# set allow-configuration "system services"
user@host# set deny-configuration "system services"
```

If the `allow-commands` and `deny-commands` statements have two different variants of a command, the longest match is always executed. The following configuration allows a user in the `test` login class to execute the

`commit synchronize` command but not the `commit` command. This is because `commit synchronize` is the longest match between `commit` and `commit synchronize`, and it is specified for `allow-commands`.

```
[edit system login class test]
user@host# set allow-commands "commit synchronize"
user@host# set deny-commands commit
```

The following configuration allows a user in the test login class to execute the `commit` command but not the `commit synchronize` command. This is because `commit synchronize` is the longest match between `commit` and `commit synchronize`, and it is specified for `deny-commands`.

```
[edit system login class test]
user@host# set allow-commands commit
user@host# set deny-commands "commit synchronize"
```

In contrast to the other statements, the default behavior for the `*-regexps` statements is that the `deny-commands-regexps` and `deny-configuration-regexps` regular expressions take precedence over `allow-commands-regexps` and `allow-configuration-regexps` expressions. You can configure the `regex-additive-logic` statement at the `[edit system]` hierarchy level to force the `allow-configuration-regexps` regular expressions to take precedence over the `deny-configuration-regexps` statements. Configuring the statement enables you to deny configuration hierarchies at a higher level and then only allow the user access to specific sub-hierarchies.

Understanding the Allow and Deny Statement Rules

The `allow/deny-commands`, `allow/deny-configuration`, `allow/deny-commands-regexps`, and `allow/deny-configuration-regexps` statements take precedence over the login class permissions. When you configure these statements, the following rules apply:

- Regular expressions for `allow-commands` and `deny-commands` statements can also include the `commit`, `load`, `rollback`, `save`, `status`, and `update` commands.
- The all login class permission bits take precedence over extended regular expressions when a user issues the `rollback` command with the `rollback` permission flag enabled.
- Users cannot issue the `load override` command when specifying an extended regular expression. Users can only issue the `merge`, `replace`, and `patch` configuration commands.
- You can use the `*` wildcard character when denoting regular expressions. However, you must use it as part of a regular expression. You cannot use `[*]` or `[.*]` as the only expression. Additionally, you cannot configure the `allow-configuration` statement with an expression such as `(interfaces (description (|.*)`), because this evaluates to `allow-configuration .*`.

Understanding Differences for the *-regexps Statements

This section outlines the differences between the allow/deny-configuration statements and the allow/deny-configuration-regexps statements.

The allow/deny-configuration-regexps statements split up the regular expression into tokens and match each piece against each part of the specified configuration's full path, whereas the allow/deny-configuration statements match against the full string. For allow/deny-configuration-regexps statements, you configure a set of strings in which each string is a regular expression, with spaces between the terms of the string. This syntax provides very fast matching but offers less flexibility. For specifying wildcard expressions, you must set up wildcards for each token of the space-delimited string you want to match, which makes it more difficult to use wildcard expressions for these statements.

For example:

- **Regular expression matching one token using allow-configuration-regexps**

This example shows that `options` is the only matched expression against the first token of the statement.

```
[edit system]
login {
  class test {
    permissions configure;
    allow-configuration-regexps .*options;
  }
}
```

The preceding configuration matches the following statements:

- set policy-**options** condition *condition* dynamic-db
- set routing-**options** static route *static-route* next-hop *next-hop*
- set event-**options** generate-event *event* time-interval *seconds*

The preceding configuration does not match the following statements:

- system host-name host-**options**
- interfaces *interface-name* description **options**

- **Regular expression matching three tokens using allow-configuration-regexps**

This example shows that ssh is the only matched expression against the third token of the statement.

```
[edit system]
login {
  class test {
    permissions configure;
    allow-configuration-regexps ".* .* .*ssh";
  }
}
```

In the preceding example, the three tokens include .*, .*, and .*ssh, respectively.

The preceding configuration matches the following statements:

- system host-name hostname-**ssh**
- system services **ssh**
- system services outbound-**ssh**

The preceding configuration does not match the following statement:

- interfaces *interface-name* description **ssh**

It is easier to use the deny-configuration statement to restrict configuration access than to use the deny-configuration-regexps statement. [Table 4 on page 58](#) illustrates the use of both the deny-configuration and deny-configuration-regexps statements in different configurations to achieve the same result of restricting access to a particular configuration.

Table 4: Restricting Configuration Access Using deny-configuration and deny-configuration-regexps Statements

Configuration Denied	Using: deny-configuration	Using: deny-configuration-regexps	Result

xnm-ssl	<pre>[edit system] login { class test { permissions configure; allow-configuration .*; deny-configuration .*xnm-ssl; } }</pre>	<pre>[edit system] login { class test { permissions configure; allow-configuration .*; deny-configuration-regexps ".* .* .*-ssl"; } }</pre>	<p>The following configuration statement is denied:</p> <ul style="list-style-type: none"> • system services xnm-ssl
ssh	<pre>[edit system] login { class test { permissions configure; allow-configuration .*; deny-configuration ".*ssh"; } }</pre>	<pre>[edit system] login { class test { permissions configure; allow-configuration .*; deny-configuration-regexps ".*ssh"; deny-configuration-regexps ".* .*ssh"; deny-configuration-regexps ".* .* .*ssh"; } }</pre>	<p>The following configuration statements are denied:</p> <ul style="list-style-type: none"> • system host-name hostname-ssh • system services ssh • system services outbound-ssh • security ssh-known-host

Although the allow/deny-configuration statements are also useful when you want a simple configuration, the allow/deny-configuration-regexps statements provide better performance and overcome the ambiguity that existed when combining expressions in the allow/deny-configuration statements.

Using Regular Expressions on Remote Authorization Servers

You can use extended regular expressions to specify which operational mode and configuration mode commands and configuration statements and hierarchies are allowed or denied for certain users. You specify these regular expressions locally in the allow/deny-commands, allow/deny-configuration, allow/deny-commands-regexps and allow/deny-configuration-regexps statements at the [edit system login class *class-name*] hierarchy level. You specify these regular expressions remotely by specifying Juniper Networks vendor-specific TACACS+ or RADIUS attributes in your authorization server's configuration. When you

configure authorization parameters both locally and remotely, the device merges the regular expressions received during TACACS+ or RADIUS authorization with any regular expressions defined on the local device.

When specifying multiple regular expressions in a local configuration using the `allow-commands`, `deny-commands`, `allow-configuration`, or `deny-configuration` statements, you configure regular expressions within parentheses and separate them using the pipe symbol. You enclose the complete expression in double quotation marks. For example, you can specify multiple `allow-commands` parameters with the following syntax:

```
allow-commands "(cmd1)|(cmd2)|(cmdn)"
```

The RADIUS authorization server uses the following attributes and syntax:

```
Juniper-Allow-Commands += "(cmd1)|(cmd2)|(cmd3)",
Juniper-Deny-Commands += "(cmd1)|(cmd2)",
Juniper-Allow-Configuration += "(config1)|(config2)",
Juniper-Deny-Configuration += "(config1)|(config2)",
```

The TACACS+ authorization server uses the following attributes and syntax:

```
allow-commands = "(cmd1)|(cmd2)|(cmdn)"
deny-commands = "(cmd1)|(cmd2)|(cmdn)"
allow-configuration = "(config1)|(config2)|(confign)"
deny-configuration = "(config1)|(config2)|(confign)"
```

When specifying multiple regular expressions in a local configuration using the `allow-commands-regexps`, `deny-commands-regexps`, `allow-configuration-regexps`, or `deny-configuration-regexps` statements, you configure regular expressions within double quotation marks and separate them using the space operator. You enclose the complete expression in square brackets. For example, you can specify multiple `allow-commands` parameters with the following syntax:

```
allow-commands-regexps [ "cmd1" "cmd2" "cmdn" ]
```

The RADIUS authorization server uses the following attributes and syntax:

```
Juniper-Allow-Configuration-Regexps += "(config1)|(config2)|(confign)",
Juniper-Deny-Configuration-Regexps += "(config1)|(config2)|(confign)",
```

The TACACS+ authorization server uses the following attributes and syntax:

```
allow-commands-regexps = "(cmd1)|(cmd2)|(cmdn)"
deny-commands-regexps = "(cmd1)|(cmd2)|(cmdn)"
allow-configuration-regexps = "(config1)|(config2)|(confign)"
deny-configuration-regexps = "(config1)|(config2)|(confign)"
```

RADIUS and TACACS+ servers also support a simplified syntax where you specify each individual expression on a separate line. For example, the RADIUS server simplified syntax is:

```
Juniper-Allow-Commands += "cmd1",
Juniper-Allow-Commands += "cmd2",
Juniper-Allow-Commands += "cmdn",
```

Similarly, the TACACS+ server simplified syntax is:

```
allow-commands-regexps1 = "cmd1"
allow-commands-regexps2 = "cmd2"
allow-commands-regexpsn = "cmdn"
```

[Table 5 on page 62](#) differentiates the local authorization configuration and the TACACS+ server authorization configuration using regular expressions.

Table 5: Sample Local and Remote Authorization Configuration Using Regular Expressions

Local Configuration	Remote TACACS+ Configuration
<pre>login { class local { permissions configure; allow-commands "(ping .*) (traceroute .*) (show .*) (configure .*) (edit) (exit) (commit) (rollback .*)"; deny-commands .*; allow-configuration "(interfaces .* unit 0 family ethernet-switching vlan mem.* .*) (interfaces .* native.* .*) (interfaces .* unit 0 family ethernet- switching interface-mo.* .*) (interfaces .* unit .*) (interfaces .* disable) (interfaces .* description .*) (vlans .* vlan-.* .*)" deny-configuration .*; } }</pre>	<pre>user = remote { login = username service = junos-exec { allow-commands1 = "ping .*" allow-commands2 = "traceroute .*" allow-commands3 = "show .*" allow-commands4 = "configure" allow-commands5 = "edit" allow-commands6 = "exit" allow-commands7 = "commit" allow-commands8 = ".*xml-mode" allow-commands9 = ".*netconf.*" allow-commands10 = ".*need-trailer" allow-commands11 = "rollback.*" allow-commands12 = "junoscript" deny-commands1 = ".*" allow-configuration1 = "interfaces .* unit 0 family ethernet- switching vlan mem.* .*" allow-configuration2 = "interfaces .* native.* .*" allow-configuration3 = "interfaces .* unit 0 family ethernet- switching interface-mo.* .*" allow-configuration4 = "interfaces .* unit .*" allow-configuration5 = "interfaces .* disable" allow-configuration6 = "interfaces .* description .*" allow-configuration7 = "interfaces .*" allow-configuration8 = "vlans .* vlan-.* .*" deny-configuration1 = ".*" local-user-name = <i>local-username</i> user-permissions = "configure" } }</pre>

NOTE:

- You need to explicitly allow access to the NETCONF mode, either locally or remotely, by issuing the following three commands: `xml-mode`, `netconf`, and `need-trailer`.

- When you use the `deny-configuration = ".*"` statement, you must allow all the desired configurations using the `allow-configuration` statement. However, this configuration can affect the allowed regular expressions buffer limit for the `allow-configuration` statement. If this limit is exceeded, the allowed configuration might not work.

Specify Regular Expressions



WARNING: When you specify regular expressions for commands and configuration statements, pay close attention to the following examples. A regular expression with invalid syntax might not produce the desired results, even if the configuration is committed without any error.

You should specify regular expressions for commands and configuration statements in the same manner as executing the complete command or statement. [Table 6 on page 64](#) lists the regular expressions for configuring access privileges for the `[edit interfaces]` and `[edit vlans]` statement hierarchies.

Table 6: Specify Regular Expressions

Statement	Regular Expression	Configuration Notes
<p>[edit interfaces]</p> <p>The set command for interfaces is executed as follows:</p> <p>[edit] user@host# set interfaces <i>interface-name</i> unit <i>interface-unit-number</i></p>	<p>The set interfaces statement is incomplete by itself and requires the unit option to execute the statement.</p> <p>As a result, the regular expression required for denying the set interfaces configuration must specify the entire executable string with the .* operator in place of statement variables:</p> <p>[edit system login class <i>class-name</i>] user@host# set permissions configure user@host# set deny-configuration "interfaces .* unit ."</p>	<ul style="list-style-type: none"> The .* operator denotes everything from the specified point onward for that particular command or statement. In this example, it denotes any interface name with any unit value. Specifying only the deny-configuration "interfaces .*" statement is incorrect and does not deny access to the interfaces configuration for the specified login class. Other valid options can be included in the regular expression. For example: <p>[edit system login class <i>class-name</i>] user@host# set permissions configure user@host# set deny-configuration "interfaces .* description ."</p> <p>[edit system login class <i>class-name</i>] user@host# set permissions configure user@host# set allow-configuration-regexps ["interfaces .* description ." "interfaces .* unit .* description ." "interfaces .* unit .* family inet address ." "interfaces.* disable"]</p> <p>[edit system login class <i>class-name</i>] user@host# set permissions configure user@host# set allow-configuration "interfaces .* unit 0 family ethernet-switching vlan mem.* ."</p> <p>Note: The mem.* regular expression in this example is used when multiple</p>

Table 6: Specify Regular Expressions (*Continued*)

Statement	Regular Expression	Configuration Notes
		<p>strings starting with the <i>mem</i> keyword are expected to be included in the specified regular expression.</p> <p>When only one member string is expected to be included, the member <i>.*</i> regular expression is used.</p>
<p>[edit vlans]</p> <p>The set command for VLANs is executed as follows:</p> <p>[edit] user@host# set vlans <i>vlan-name</i> <i>vlan-id</i> <i>vlan-id</i></p>	<p>Here, the set vlans statement is incomplete by itself, and requires the <i>vlan-id</i> option to execute the statement.</p> <p>As a result, the regular expression required for allowing the set vlans configuration must specify the entire executable string with the <i>.*</i> operator in place of statement variables:</p> <pre>[edit system login class <i>class-name</i>] user@host# set permissions configure user@host# set allow-configuration "vlans .* <i>vlan-id</i> .*"</pre>	<ul style="list-style-type: none"> The <i>.*</i> operator denotes everything from the specified point onward for that particular command or statement. In this example, it denotes any VLAN name with any VLAN ID. Other valid options under the [edit vlans] statement hierarchy can be included in the regular expression. For example: <pre>[edit system login class <i>class-name</i>] user@host# set permissions configure user@host# set allow-configuration- regexps ["vlans .* <i>vlan-id</i> .*" "vlans .* <i>vlan-id</i> .* description .*" "vlans .* <i>vlan-id</i> .* filter .*"]</pre>

Regular Expressions Operators

Table 7 on page 66 lists common regular expression operators that you can use for allowing or denying operational and configuration modes.

Command regular expressions implement the extended (modern) regular expressions, as defined in POSIX 1003.2.

Table 7: Common Regular Expression Operators

Operator	Match	Example
	One of two or more terms separated by the pipe. Each term must be a complete standalone expression enclosed in parentheses (), with no spaces between the pipe and the adjacent parentheses.	<pre>[edit system login class test] user@host# set permissions configure user@host# set allow-commands "(ping) (traceroute) (show system alarms) (show system software)" user@host# set deny-configuration "(access) (access-profile) (accounting-options) (applications) (apply-groups) (bridge-domains) (chassis) (class-of-service)"</pre> <p>With the preceding configuration, the users assigned to the test login class have operational mode access restricted to only the commands specified in the allow-commands statement. They also have access to configuration mode, excluding the hierarchy levels specified in the deny-configuration statement.</p>
^	At the beginning of an expression, used to denote where the command begins, where there might be some ambiguity.	<pre>[edit system login class test] user@host# set permissions interface user@host# set permissions interface-control user@host# set allow-commands " (^show) (log interfaces policer)) (^monitor)"</pre> <p>With the preceding configuration, the users assigned to the test login class have access to viewing and configuring the interface configuration. The allow-commands statement grants access to commands that begin with the show and monitor keywords.</p> <p>For the first filter, the commands specified include the show log, show interfaces, and show policer commands. The second filter specifies all commands starting with the monitor keyword, such as the monitor interfaces or the monitor traffic commands.</p>

Table 7: Common Regular Expression Operators *(Continued)*

Operator	Match	Example
\$	Character at the end of a command. Used to denote a command that must be matched exactly up to that point.	<pre>[edit system login class test] user@host# set permissions interface user@host# set allow-commands "(show interfaces\$)"</pre> <p>With the preceding configuration, the users assigned to the test login class can view the interfaces configuration in configuration mode. The users can also view the interface configuration with the show configuration operational mode command. However, the regular expression specified in the allow-commands statement restricts the users to execute only the show interfaces command and denies access to the command extensions such as show interfaces detail or show interfaces extensive.</p>
[]	Range of letters or digits. To separate the start and end of a range, use a hyphen (-).	<pre>[edit system login class test] user@host# set permissions clear user@host# set permissions configure user@host# set permissions network user@host# set permissions trace user@host# set permissions view user@host# set allow-configuration-regexps ["interfaces [gx]e-.* unit [0-9]* description .*"]</pre> <p>With the preceding configuration, the users assigned to the test login class have operator-level user permissions. These users also have access to configure interfaces within the specified range of interface name and unit number (0 through 9).</p>

Table 7: Common Regular Expression Operators *(Continued)*

Operator	Match	Example
()	A group of commands indicating a complete, standalone expression to be evaluated. The result is then evaluated as part of the overall expression. Parentheses must be used in conjunction with pipe operators, as explained.	<pre>[edit system login class test] user@host# set permissions all user@host# set allow-commands "(clear) (configure)" user@host# deny-commands "(mtrace) (start) (delete)"</pre> <p>With the above configuration, users assigned to the test login class have superuser-level permissions and have access to the commands specified in the allow-commands statement.</p>
*	Zero or more terms.	<pre>[edit system login class test] user@host# set permissions configure user@host# set deny-configuration "(system login class m*)"</pre> <p>With the above configuration, users assigned to the test login class whose login username begins with <code>m</code> are denied configuration access.</p>
+	One or more terms.	<pre>[edit system login class test] user@host# set permissions configure user@host# set deny-configuration "(system login class m+)"</pre> <p>With the above configuration, users assigned to the test login class whose login username begins with <code>m</code> are denied configuration access.</p>

Table 7: Common Regular Expression Operators (*Continued*)

Operator	Match	Example
.	Any character except for a space " ".	<pre>[edit system login class test] user@host# set permissions configure user@host# set deny-configuration "(system login class m.)"</pre> <p>With the above configuration, users assigned to the test login class whose login username begins with <code>m</code> are denied configuration access.</p>
.*	Everything from the specified point onward.	<pre>[edit system login class test] user@host# set permissions configure user@host# set deny-configuration "(system login class m .*)"</pre> <p>With the above configuration, users assigned to the test login class whose login username begins with <code>m</code> are denied configuration access.</p> <p>Similarly, the deny-configuration <code>"protocols .*"</code> statement denies all configuration access under the <code>[edit protocols]</code> hierarchy level.</p> <p>NOTE:</p> <ul style="list-style-type: none"> • The <code>*</code>, <code>+</code>, and <code>.</code> operations can be achieved by using <code>.*</code>. • The deny-commands <code>.*</code> and deny-configuration <code>.*</code> statements deny access to all operational mode commands and configuration hierarchies, respectively.

NOTE: The `!` regular expression operator is not supported.

Regular Expression Examples

Table 8 on page 70 lists the regular expressions used to allow configuration options under two configuration hierarchies—`[edit system ntp server]` and `[edit protocols rip]`—as an example for specifying regular expressions.

NOTE: Table 8 on page 70 does not provide a comprehensive list of all regular expressions and keywords for all configuration statements and hierarchies. The regular expressions listed in the

table are validated only for the [edit system ntp server] and [edit protocols rip] statement hierarchies.

Table 8: Regular Expressions Examples

Statement Hierarchy	Regular Expressions	Allowed Configuration	Denied Configuration
[edit system ntp server]			
key <i>key-number</i>	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps ["system ntp server .*" "system ntp server .* key .*"] set deny-configuration-regexps ["system ntp server .* version .*" "system ntp server .* prefer"]</pre>	<ul style="list-style-type: none"> server IP server IP and key 	<ul style="list-style-type: none"> version prefer
version <i>version-number</i>	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps ["system ntp server .*" "system ntp server .* version .*"] set deny-configuration-regexps ["system ntp server .* key .*" "system ntp server .* prefer"]</pre>	<ul style="list-style-type: none"> server IP server IP and version 	<ul style="list-style-type: none"> key prefer
prefer	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps ["system ntp server .*" "system ntp server .* prefer"]; set deny-configuration-regexps ["system ntp server .* key .*" "system ntp server .* version .*"]</pre>	<ul style="list-style-type: none"> server IP server IP and prefer 	<ul style="list-style-type: none"> key version

Table 8: Regular Expressions Examples (*Continued*)

Statement Hierarchy	Regular Expressions	Allowed Configuration	Denied Configuration
[edit protocols rip]			
message-size <i>message-size</i>	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps "protocols rip message-size .*" set deny-configuration-regexps ["protocols rip metric-in .*" "protocols rip route-timeout .*" "protocols rip update-interval .*"]</pre>	<ul style="list-style-type: none"> • message-size 	<ul style="list-style-type: none"> • metric-in • route-timeout • update-interval
metric-in <i>metric-in</i>	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps "protocols rip metric-in .*" set deny-configuration-regexps ["protocols rip message-size .*" "protocols rip route-timeout .*" "protocols rip update-interval .*"]</pre>	<ul style="list-style-type: none"> • metric-in 	<ul style="list-style-type: none"> • message-size • route-timeout • update-interval
route-timeout <i>route-timeout</i>	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps "protocols rip route-timeout .*" set deny-configuration-regexps ["protocols rip metric-in .*" "protocols rip message-size .*" "protocols rip update-interval .*"]</pre>	<ul style="list-style-type: none"> • route-timeout 	<ul style="list-style-type: none"> • message-size • metric-in • update-interval

Table 8: Regular Expressions Examples (*Continued*)

Statement Hierarchy	Regular Expressions	Allowed Configuration	Denied Configuration
update-interval <i>update-interval</i>	<pre>[edit system login class test] set permissions configure set allow-configuration-regexps "protocols rip update-interval .*" set deny-configuration-regexps ["protocols rip metric-in .*" "protocols rip route-timeout .*" "protocols rip message-size .*"]</pre>	<ul style="list-style-type: none"> • update-interval 	<ul style="list-style-type: none"> • message-size • metric-in • route-timeout

How to Define Access Privileges with allow-configuration and deny-configuration Statements

You can define access privileges for configuration statement hierarchies by using a combination of the following types of statements:

- permission flags
- allow-configuration and deny-configuration statements

The permission flags define the larger boundaries of what a person or login class can access and control. The allow-configuration and deny-configuration statements contain one or more regular expressions that allow or deny specific configuration hierarchies and statements. The allow-configuration and deny-configuration statements take precedence over permission flags and give the administrator finer control over exactly what hierarchies and statements the user can view and configure.

This topic explains how to define access privileges using allow-configuration and deny-configuration statements by showing examples of login class configurations that use these statements. Examples 1 through 3 create login classes that allow users access to all commands and statements except those defined in the deny-configuration statement.

Notice that *permission bit* and *permission flag* are used interchangeably.

Example 1

To create a login class that allows the user to execute all commands and configure everything except telnet parameters:

1. Set the user's login class permissions to all.

```
[edit system login]
user@host# set class all-except-telnet permissions all
```

2. Include the following deny-configuration statement.

```
[edit system login class all-except-telnet]
user@host# set deny-configuration "system services telnet"
```

Example 2

To create a login class that allows the user to execute all commands and configure everything except statements within any login class whose name begins with "m":

1. Set the user's login class permissions to all.

```
[edit system login]
user@host# set class all-except-login-class-m permissions all
```

2. Include the following deny-configuration statement.

```
[edit system login class all-except-login-class-m]
user@host# set deny-configuration "system login class m.*"
```

Example 3

To create a login class that allows the user to execute all commands and configure everything except the [edit system login class] or [edit system services] hierarchy levels:

1. Set the user's login class permissions to all.

```
[edit system login]
user@host# set class all-except-login-class-or-system-services permissions all
```

2. Include the following deny-configuration statement:

```
[edit system login class all-except-login-class-or-system-services]
user@host# set deny-configuration "(system login class) | (system services)"
```

The following examples show how to use the allow-configuration and deny-configuration statements to determine permissions inverse to each other for the [edit system services] hierarchy level.

Example 4

To create a login class that allows the user to have full configuration privileges only at the [edit system services] hierarchy level:

1. Set the user's login class permissions to configure.

```
[edit system login]
user@host# set class configure-only-system-services permissions configure
```

2. Include the following allow-configuration statement:

```
[edit system login class configure-only-system-services]
user@host# set allow-configuration "system services"
```

Example 5

To create a login class that allows the user full permissions for all commands and all configuration hierarchies except the [edit system services] hierarchy level:

1. Set the user's login class permissions to all.

```
[edit system login]
user@host# set class all-except-system-services permissions all
```

2. Include the following deny-configuration statement.

```
[edit system login class all-except-system-services]
user@host# set deny-configuration "system services"
```

Example: Use Additive Logic with Regular Expressions to Specify Access Privileges

IN THIS SECTION

- [Requirements | 75](#)
- [Overview | 75](#)
- [Configuration | 76](#)
- [Examples | 77](#)

This example shows how to use additive logic when using regular expressions to set up configuration access privileges.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

You can define regular expressions to control who can make changes to the configuration and what they can change. These regular expressions indicate specific configuration hierarchies that users in a login class are permitted to access. For example, you can define regular expressions that allow users to modify a group of routing instances and define regular expressions that prevent the users from making changes to any other routing instances or to other configuration levels. You define the regular expressions by configuring the `allow-configuration-regexps` and `deny-configuration-regexps` statements for a login class.

By default, the `deny-configuration-regexps` statement takes precedence over the `allow-configuration-regexps` statement. If a configuration hierarchy appears in a `deny-configuration-regexps` statement for a login class, it is not visible to the users in that class, regardless of the contents of the `allow-configuration-regexps` statement. If a configuration hierarchy does not appear in a `deny-configuration-regexps` statement, it is visible to the users in that class if it appears in an `allow-configuration-regexps` statement.

You can change this default behavior by enabling additive logic for the `*-configuration-regexps` statements. When you enable additive logic, the `allow-configuration-regexps` statement takes precedence over the `deny-configuration-regexps` statement.

Thus, if the `deny-configuration-regexps` statement denies access to all configuration hierarchies at a given level (protocols `.*`) but the `allow-configuration-regexps` statement allows access to one sub-hierarchy (protocols `bgp .*`), then by default the device denies access to the hierarchies for users in that login class

because the `deny-configuration-regexps` statement takes precedence. However, if you enable additive logic, the device allows access to the specified sub-hierarchy for users in that login class because the `allow-configuration-regexps` takes precedence in this case.

Configuration

IN THIS SECTION

- [Step-by-Step Procedure | 76](#)

Step-by-Step Procedure

To enable additive logic to explicitly allow users in a given login class access to one or more individual configuration hierarchies:

1. Include the `deny-configuration-regexps` statement, and explicitly deny access to configuration hierarchies.

```
[edit system login class class-name]
user@host# set deny-configuration-regexps ["regular expression 1" "regular expression 2"
"regular expression 3"]
```

For example:

```
[edit system login class class-name]
user@host# set deny-configuration-regexps "protocols .*"
```

2. Include the `allow-configuration-regexps` statement, and define regular expressions for the specific hierarchies to allow.

```
[edit system login class class-name]
user@host# set allow-configuration-regexps ["regular expression 1" "regular expression 2"
"regular expression 3"]
```

For example:

```
[edit system login class class-name]
user@host# set allow-configuration-regexps ["protocols bgp .*" "protocols ospf .*"]
```

3. Enable additive logic for the allow-configuration-regexps and deny-configuration-regexps regular expressions.

```
[edit system]
user@host# set regex-additive-logic
```

4. Assign the login class to one or more users.

```
[edit system login]
user@host# set user username class class-name
```

5. Commit your changes.

Users assigned to this login class have access to the configuration hierarchies included in the allow-configuration-regexps statement but do not have access to the other hierarchies specified in the deny-configuration-regexps statement.

NOTE: When you configure the regex-additive-logic statement, the behavior change applies to all allow-configuration-regexps and deny-configuration-regexps statements present in all login classes. If you enable additive logic, you should evaluate existing statements for any impact, and update the regular expressions in those statements as appropriate.

Examples

IN THIS SECTION

- [Use Regular Expressions with Additive Logic | 78](#)

Use Regular Expressions with Additive Logic

Purpose

This section provides examples of regular expressions that use additive logic to give you ideas for creating configurations appropriate for your system.

Allow Specific Routing Instances

The following example login class includes a regular expression that allows configuration of routing instances whose names start with CUST-VRF-; for example, CUST-VRF-1, CUST-VRF-25, CUST-VRF-100, and so on. The example also includes a regular expression that prevents the configuration of any routing instances.

```
[edit system login class class-name]
user@host# set permissions [configure routing-control view view-configuration]
user@host# set deny-configuration-regexps "routing-instances .*"
user@host# set allow-configuration-regexps "routing-instances CUST-VRF-.* ."
```

By default, the deny-configuration-regexps statement takes precedence, and the previous configuration prevents the users in the login class from configuring any routing instances, regardless of the name.

However if you configure the following statement, the allow-configuration-regexps statement takes precedence. Thus, the users can configure routing instances whose names start with CUST-VRF-, but the users cannot configure any other routing instances.

```
[edit system]
user@host# set regex-additive-logic
```

Allow BGP Peer Configuration Only

The following example login class includes regular expressions that prevent configuration at the [edit protocols] hierarchy level but allow configuration of BGP peers:

```
[edit system login class class-name]
user@host# set permissions [configure routing-control view view-configuration]
user@host# set deny-configuration-regexps "protocols .*"
user@host# set allow-configuration-regexps "protocols bgp group *"
```

By default, the previous configuration prevents the users in the login class from making changes to any hierarchies under [edit protocols].

However, if you configure the following statement, the users in the login class can make changes to BGP peers, but the users cannot configure other protocols or other BGP statements outside of the allowed hierarchy level.

```
[edit system]
user@host# set regex-additive-logic
```

Verification

To verify that you have set the access privileges correctly:

1. Configure a login class and commit the changes.
2. Assign the login class to a *username*.
3. Log in as the *username* assigned with the new login class.
4. Attempt to configure the hierarchy levels that are allowed.
 - You should be able to configure statements in hierarchy levels that have been allowed.
 - Hierarchy levels that are denied should not be visible.
 - Any allowed or denied expressions should take precedence over any permissions granted with the permissions statement.

Example: Configure User Permissions with Access Privileges for Operational Mode Commands

IN THIS SECTION

- [Requirements | 80](#)
- [Overview and Topology | 80](#)
- [Configuration | 81](#)
- [Verification | 87](#)

This example shows how to configure custom login classes and assign access privileges for operational mode commands. Users in the login class can execute only the commands for which they have access.

This prevents unauthorized users from executing sensitive commands that could cause damage to the network.

Requirements

This example uses the following hardware and software components:

- One Juniper Networks device
- One TACACS+ (or RADIUS) server

Before you begin, establish a TCP connection between the device and the TACACS+ server. In the case of the RADIUS server, establish a UDP connection between the device and the RADIUS server.

Overview and Topology

Figure 1 on page 80 illustrates a simple topology, where Router R1 is a Juniper Networks device and has a TCP connection established with a TACACS+ server.

Figure 1: Topology



This example configures R1 with three customized login classes: Class1, Class2, and Class3. Each class defines access privileges for the user by configuring the `permissions` statement and by defining extended regular expressions using the `allow-commands` and `deny-commands` statements.

The purpose of each login class is as follows:

- **Class1**—Defines access privileges for the user with the `allow-commands` statement only. This login class provides operator-level user permissions and authorization for rebooting the device.
- **Class2**—Defines access privileges for the user with the `deny-commands` statement only. This login class provides operator-level user permissions and denies access to set commands.
- **Class3**—Defines access privileges for the user with both the `allow-commands` and `deny-commands` statements. This login class provides superuser-level user permissions and authorization for accessing interfaces and viewing device information. It also denies access to the `edit` and `configure` commands.

Router R1 has three different users, User1, User2, and User3 assigned to the Class1, Class2, and Class3 login classes, respectively .

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 81](#)
- [Configure Authentication Parameters for Router R1 | 82](#)
- [Configure Access Privileges with the allow-commands Statement \(Class1\) | 83](#)
- [Configure Access Privileges with the deny-commands Statement \(Class2\) | 84](#)
- [Configure Access Privileges with Both the allow-commands and deny-commands Statements \(Class3\) | 84](#)
- [Results | 85](#)

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` in configuration mode.

R1

```
set system authentication-order tacplus
set system authentication-order radius
set system authentication-order password
set system radius-server 10.209.1.66 secret "$ABC123"
set system tacplus-server 10.209.1.66 secret "$ABC123"
set system radius-options enhanced-accounting
set system tacplus-options enhanced-accounting
set system accounting events login
set system accounting events change-log
set system accounting events interactive-commands
set system accounting destination tacplus server 10.209.1.66 secret "$ABC123"
set system login class Class1 permissions clear
set system login class Class1 permissions network
set system login class Class1 permissions reset
set system login class Class1 permissions trace
set system login class Class1 permissions view
set system login class Class1 allow-commands "request system reboot"
set system login class Class2 permissions clear
```

```

set system login class Class2 permissions network
set system login class Class2 permissions reset
set system login class Class2 permissions trace
set system login class Class2 permissions view
set system login class Class2 deny-commands set
set system login class Class3 permissions all
set system login class Class3 allow-commands configure
set system login class Class3 deny-commands .*
set system login user User1 uid 2001
set system login user User1 class Class1
set system login user User1 authentication encrypted-password "$ABC123"
set system login user User2 uid 2002
set system login user User2 class Class2
set system login user User2 authentication encrypted-password "$ABC123"
set system login user User3 uid 2003
set system login user User3 class Class3
set system login user User3 authentication encrypted-password "$ABC123"
set system syslog file messages any any

```

Configure Authentication Parameters for Router R1

Step-by-Step Procedure

To configure Router R1 authentication:

1. Configure the order in which R1 attempts to authenticate the user. In this example, TACACS+ server authentication is first, followed by RADIUS server authentication, and then the local password.

```

[edit system]
user@R1# set authentication-order tacplus
user@R1# set authentication-order radius
user@R1# set authentication-order password

```

2. Configure the TACACS+ server.

```

[edit system]
user@R1# set tacplus-server 10.209.1.66 secret "$ABC123"
user@R1# set tacplus-options enhanced-accounting
user@R1# set accounting destination tacplus server 10.209.1.66 secret "$ABC123"

```

3. Configure the RADIUS server.

```
[edit system]
user@R1# set radius-server 10.209.1.66 secret "$ABC123"
user@R1# set radius-options enhanced-accounting
```

4. Configure R1 accounting parameters.

```
[edit system]
user@R1# set accounting events login
user@R1# set accounting events change-log
user@R1# set accounting events interactive-commands
```

Configure Access Privileges with the allow-commands Statement (Class1)

Step-by-Step Procedure

To specify regular expressions using the `allow-commands` statement:

1. Configure the Class1 login class and assign operator-level user permissions.

```
[edit system login]
user@R1# set class Class1 permissions [clear network reset trace view]
```

2. Configure the `allow-commands` regular expression to enable users in the class to reboot the device.

```
[edit system login]
user@R1# set class Class1 allow-commands "request system reboot"
```

3. Configure the user account for the Class1 login class.

```
[edit system login]
user@R1# set user User1 uid 2001
user@R1# set user User1 class Class1
user@R1# set user User1 authentication encrypted-password "$ABC123"
```

Configure Access Privileges with the deny-commands Statement (Class2)

Step-by-Step Procedure

To specify regular expressions using the deny-commands statement:

1. Configure the Class2 login class and assign operator-level user permissions.

```
[edit system login]
user@R1# set class Class1 permissions [clear network reset trace view]
```

2. Configure the deny-commands regular expression to prevent users in the class from executing set commands.

```
[edit system login]
user@R1# set class Class1 deny-commands "set"
```

3. Configure the user account for the Class2 login class.

```
[edit system login]
user@R1# set user User2 uid 2002
user@R1# set user User2 class Class2
user@R1# set user User2 authentication encrypted-password "$ABC123"
```

Configure Access Privileges with Both the allow-commands and deny-commands Statements (Class3)

Step-by-Step Procedure

To specify regular expressions using both the allow-commands and deny-commands statements:

1. Configure the Class3 login class and assign superuser-level permissions.

```
[edit system login]
user@R1# set class Class3 permissions all
```

2. Configure the `deny-commands` regular expression to prevent users in the class from executing any commands.

```
[edit system login]
user@R1# set class Class3 deny-commands ".*"
```

3. Configure the `allow-commands` regular expression to allow users to enter configuration mode.

```
[edit system login]
user@R1# set class Class3 allow-commands configure
```

4. Configure the user account for the Class3 login class.

```
[edit system login]
user@R1# set user User3 uid 2003
user@R1# set user User3 class Class3
user@R1# set user User3 authentication encrypted-password "$ABC123"
```

Results

In configuration mode, confirm your configuration by entering the `show system` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show system
authentication-order [ tacplus radius password ];
radius-server {
    10.209.1.66 secret "$ABC123";
}
tacplus-server {
    10.209.1.66 secret "$ABC123";
}
radius-options {
    enhanced-accounting;
}
tacplus-options {
    enhanced-accounting;
}
accounting {
```

```

events [ login change-log interactive-commands ];
destination {
    tacplus {
        server {
            10.209.1.66 secret "$ABC123";
        }
    }
}
login {
    class Class1 {
        permissions [ clear network reset trace view ];
        allow-commands "request system reboot";
    }
    class Class2 {
        permissions [ clear network reset trace view ];
        deny-commands set;
    }
    class Class3 {
        permissions all;
        allow-commands configure;
        deny-commands .*;
    }
    user User1 {
        uid 2001;
        class Class1;
        authentication {
            encrypted-password "$ABC123";
        }
    }
    user User2 {
        uid 2002;
        class Class2;
        authentication {
            encrypted-password "$ABC123";
        }
    }
    user User3 {
        uid 2003;
        class Class3;
        authentication {
            encrypted-password "$ABC123";
        }
    }
}

```

```

    }
}
syslog {
    file messages {
        any any;
    }
}

```

Verification

IN THIS SECTION

- [Verifying the Class1 Configuration | 87](#)
- [Verifying the Class2 Configuration | 88](#)
- [Verifying Class3 Configuration | 90](#)

Log in as the username assigned with the new login class, and confirm that the configuration is working properly.

Verifying the Class1 Configuration

Purpose

Verify that the permissions and commands allowed in the Class1 login class are working.

Action

In operational mode, run the `show system users` command.

```

User1@R1> show system users
12:39PM up 6 days, 23 mins, 6 users, load averages: 0.00, 0.01, 0.00
USER   TTY   FROM                               LOGIN@  IDLE  WHAT
User1  p0    abc.example.net 12:34AM 12:04 cli
User2  p1    abc.example.net 12:36AM 12:02 -cli (cli)
User3  p2    abc.example.net 10:41AM 11 -cli (cli)

```


In operational mode, run the `request system reboot` command.

```
User1@R1> request system ?  
Possible completions:  
  reboot                Reboot the system
```

Meaning

The Class1 login class to which User1 is assigned has operator-level user permissions and allows users in the class to execute the `request system reboot` command.

The predefined operator login class has the following permission flags specified:

- **clear**—Can use `clear` commands to clear (delete) information that the device learns from the network and stores in various network databases.
- **network**—Can access the network by using the `ping`, `ssh`, `telnet`, and `traceroute` commands.
- **reset**—Can restart software processes by using the `restart` command.
- **trace**—Can view trace file settings and configure trace file properties.
- **view**—Can use various commands to display current system-wide, routing table, and protocol-specific values and statistics. Cannot view the secret configuration.

For the Class1 login class, in addition to the above-mentioned user permissions, User1 can execute the `request system reboot` command. The first output displays the view permissions as an operator, and the second output shows that the only `request system` command that User1 can execute as an operator is the `request system reboot` command.

Verifying the Class2 Configuration

Purpose

Verify that the permissions and commands allowed for the Class2 login class are working.

Action

In operational mode, run the `ping` command.

```
User2@R1> ping 10.209.1.66  
ping 10.209.1.66
```

```

PING 10.209.1.66 (10.209.1.66): 56 data bytes
64 bytes from 10.209.1.66: icmp_seq=0 ttl=52 time=212.521 ms
64 bytes from 10.209.1.66: icmp_seq=1 ttl=52 time=212.844 ms
64 bytes from 10.209.1.66: icmp_seq=2 ttl=52 time=211.304 ms
64 bytes from 10.209.1.66: icmp_seq=3 ttl=52 time=210.963 ms
^C
--- 10.209.1.66 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 210.963/211.908/212.844/0.792 ms

```

From the CLI prompt, check the available commands.

```

User2@R1> ?
Possible completions:
  clear          Clear information in the system
  file           Perform file operations
  help           Provide help information
  load           Load information from file
  monitor        Show real-time debugging information
  mtrace         Trace multicast path from source to receiver
  op             Invoke an operation script
  ping           Ping remote target
  quit           Exit the management session
  request        Make system-level requests
  restart        Restart software process
  save           Save information to file
  show           Show system information
  ssh            Start secure shell on another host
  start          Start shell
  telnet         Telnet to another host
  test           Perform diagnostic debugging
  traceroute     Trace route to remote host

```

From the CLI prompt, execute any set command.

```

User2@R1> set
      ^
unknown command.

```

Meaning

The Class2 login class to which User2 is assigned has operator-level user permissions and denies access to all set commands.

The permission flags specified for the predefined operator login class are the same as those specified for Class1.

Verifying Class3 Configuration

Purpose

Verify that the permissions and commands allowed for the Class3 login class are working.

Action

In operational mode, check the available commands.

```
User3@R1> ?  
Possible completions:  
  configure          Manipulate software configuration information
```

Enter configuration mode.

```
User3@R1> configure  
Entering configuration mode  
  
[edit]  
User3@R1#
```

Meaning

The Class3 login class to which User3 is assigned has superuser (all) permissions, but this class only allows users to execute the `configure` command. The class denies access to all other operational mode commands. Because the regular expressions specified in the `allow/deny-commands` statements take precedence over the user permissions, User3 on R1 has access only to configuration mode and is denied access to all other operational mode commands.

Example: Configure User Permissions with Access Privileges for Configuration Statements and Hierarchies

IN THIS SECTION

- Requirements | 91
- Overview and Topology | 91
- Configuration | 92
- Verification | 98

This example shows how to configure custom login classes and assign access privileges to specific configuration hierarchies. Users in the login class can view and modify only those configuration statements and hierarchies to which they have access. This prevents unauthorized users from modifying device configurations that could cause damage to the network.

Requirements

This example uses the following hardware and software components:

- One Juniper Networks device
- One TACACS+ (or RADIUS) server

Before you begin, establish a TCP connection between the device and the TACACS+ server. In the case of the RADIUS server, establish a UDP connection between the device and the RADIUS server.

Overview and Topology

Figure 2 on page 91 illustrates a simple topology, where Router R1 is a Juniper Networks device and has a TCP connection established with a TACACS+ server.

Figure 2: Topology



This example configures R1 with two customized login classes: Class1 and Class2. Each class defines access privileges for the user by configuring the permissions statement and by defining extended regular expressions using the allow-configuration, deny-configuration, allow-configuration-regexps, and deny-configuration-regexps statements.

The purpose of each login class is as follows:

- **Class1**—Defines access privileges for the user with the allow-configuration and deny-configuration statements. This login class provides access to configure the [edit interfaces] hierarchy only and denies all other access on the device. To do this, the user permissions include configure to provide configuration access. In addition, the allow-configuration statement allows access to the interfaces configuration, and the deny-configuration statement denies access to all other configuration hierarchies. Because the allow statement takes precedence over the deny statement, the users assigned to the Class1 login class can access only the [edit interfaces] hierarchy level.
- **Class2**—Defines access privileges for the user with the allow-configuration-regexps and deny-configuration-regexps statements. This login class provides superuser-level user permissions and explicitly allows configuration under multiple hierarchy levels for interfaces. It also denies access to the [edit system] and [edit protocols] hierarchy levels.

Router R1 has two users, User1 and User2, assigned to the Class1 and Class2 login classes, respectively.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 92](#)
- [Configure Authentication Parameters for Router R1 | 93](#)
- [Configure Access Privileges with the allow-configuration and deny-configuration Statements \(Class1\) | 94](#)
- [Configure Access Privileges with the allow-configuration-regexps and deny-configuration-regexps Statements \(Class2\) | 95](#)
- [Results | 96](#)

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter commit in configuration mode.

R1

```

set system authentication-order tacplus
set system authentication-order radius
set system authentication-order password
set system radius-server 10.209.1.66 secret "$ABC123"
set system tacplus-server 10.209.1.66 secret "$ABC123"
set system radius-options enhanced-accounting
set system tacplus-options enhanced-accounting
set system accounting events login
set system accounting events change-log
set system accounting events interactive-commands
set system accounting destination tacplus server 10.209.1.66 secret "$ABC123"
set system login class Class1 permissions configure
set system login class Class1 allow-configuration "interfaces .* unit .*"
set system login class Class1 deny-configuration .*
set system login class Class2 permissions all
set system login class Class2 allow-configuration-regexps [ "interfaces .* description .*"
"interfaces .* unit .* description .*" "interfaces .* unit .* family inet address .*"
"interfaces.* disable" ]
set system login class Class2 deny-configuration-regexps [ "system" "protocols" ]
set system login user User1 uid 2004
set system login user User1 class Class1
set system login user User1 authentication encrypted-password "$ABC123"
set system login user User2 uid 2006
set system login user User2 class Class2
set system login user User2 authentication encrypted-password "$ABC123"
set system syslog file messages any any

```

Configure Authentication Parameters for Router R1

Step-by-Step Procedure

To configure Router R1 authentication:

1. Configure the order in which R1 attempts to authenticate the user. In this example, TACACS+ server authentication is first, followed by RADIUS server authentication, and then the local password.

```

[edit system]
user@R1# set authentication-order tacplus

```

```
user@R1# set authentication-order radius
user@R1# set authentication-order password
```

2. Configure the TACACS+ server.

```
[edit system]
user@R1# set tacplus-server 10.209.1.66 secret "$ABC123"
user@R1# set tacplus-options enhanced-accounting
user@R1# set accounting destination tacplus server 10.209.1.66 secret "$ABC123"
```

3. Configure the RADIUS server.

```
[edit system]
user@R1# set radius-server 10.209.1.66 secret "$ABC123"
user@R1# set radius-options enhanced-accounting
```

4. Configure the R1 accounting parameters.

```
[edit system]
user@R1# set accounting events login
user@R1# set accounting events change-log
user@R1# set accounting events interactive-commands
```

Configure Access Privileges with the allow-configuration and deny-configuration Statements (Class1)

Step-by-Step Procedure

To specify regular expressions using the allow-configuration and deny-configuration statements:

1. Configure the Class1 login class with configure permissions.

```
[edit system login]
user@R1# set class Class1 permissions configure
```

2. Configure the allow-configuration regular expression to allow users in the class to view and modify part of the [edit interfaces] hierarchy level.

```
[edit system login]
user@R1# set class Class1 allow-configuration "interfaces .* unit ."
```

3. Configure the deny-configuration regular expression to deny access to all configuration hierarchies.

```
[edit system login]
user@R1# set class Class1 deny-configuration .*
```

4. Configure the user account for the Class1 login class.

```
[edit system login]
user@R1# set user User1 uid 2004
user@R1# set user User1 class Class1
user@R1# set user User1 authentication encrypted-password "$ABC123"
```

Configure Access Privileges with the allow-configuration-regexps and deny-configuration-regexps Statements (Class2)

Step-by-Step Procedure

To specify regular expressions using the allow-configuration-regexps and deny-configuration-regexps statements:

1. Configure the Class2 login class and assign superuser (all) permissions.

```
[edit system login]
user@R1# set class Class2 permissions all
```

2. Configure the allow-configuration-regexps regular expression to allow users in the class to access multiple hierarchies under the [edit interfaces] hierarchy level.

```
[edit system login]
user@R1# set class Class2 allow-configuration-regexps [ "interfaces .* description ."
```



```
"interfaces .* unit .* description .*" "interfaces .* unit .* family inet address .*"
"interfaces.* disable" ]
```

3. Configure the deny-configuration-regexps regular expression to prevent users in the class from viewing or modifying the configuration at the [edit system] and [edit protocols] hierarchy levels.

```
[edit system login]
user@R1# set class Class2 deny-configuration-regexps [ "system" "protocols" ]
```

4. Configure the user account for the Class2 login class.

```
[edit system login]
user@R1# set user User2 uid 2006
user@R1# set user User2 class Class2
user@R1# set user User2 authentication encrypted-password "$ABC123"
```

Results

In configuration mode, confirm your configuration by entering the `show system` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show system
authentication-order [ tacplus radius password ];
radius-server {
    10.209.1.66 secret "$ABC123";
}
tacplus-server {
    10.209.1.66 secret "$ABC123";
}
radius-options {
    enhanced-accounting;
}
tacplus-options {
    enhanced-accounting;
}
accounting {
    events [ login change-log interactive-commands ];
    destination {
```

```

        tacplus {
            server {
                10.209.1.66 secret "$ABC123";
            }
        }
    }
}

login {
    class Class1 {
        permissions configure;
        allow-configuration "interfaces .* unit .*";
        deny-configuration .*;
    }
    class Class2 {
        permissions all;
        allow-configuration-regexps [ "interfaces .* description .*" "interfaces .* unit .*
description .*" "interfaces .* unit .* family inet address .*" "interfaces.* disable" ];
        deny-configuration-regexps [ "system" "protocols" ];
    }
    user User1 {
        uid 2001;
        class Class1;
        authentication {
            encrypted-password "$ABC123";
        }
    }
    user User2 {
        uid 2002;
        class Class2;
        authentication {
            encrypted-password "$ABC123";
        }
    }
}

syslog {
    file messages {
        any any;
    }
}

```

Verification

IN THIS SECTION

- [Verify the Class1 Configuration | 98](#)
- [Verify the Class2 Configuration | 99](#)

Log in as the username assigned with the new login class, and confirm that the configuration is working properly.

Verify the Class1 Configuration

Purpose

Verify that the permissions allowed in the Class1 login class are working.

Action

In operational mode, check the available commands.

```
User1@R1> ?
Possible completions:
clear          Clear information in the system
configure      Manipulate software configuration information
file           Perform file operations
help           Provide help information
load           Load information from file
op             Invoke an operation script
quit           Exit the management session
request        Make system-level requests
save           Save information to file
set            Set CLI properties, date/time, craft interface message
start          Start shell
test           Perform diagnostic debugging
```

In configuration mode, check the available configuration permissions.

```
User1@R1# edit ?
Possible completions:
> interfaces          Interface configuration
```

Meaning

User1 has configure user permissions, as seen in the first output. Additionally, in configuration mode, User1 has access to the interfaces hierarchy level, but only that hierarchy level, as seen in the second output.

Verify the Class2 Configuration

Purpose

Verify that the Class2 configuration is working as expected.

Action

In configuration mode, access the interfaces configuration.

```
[edit interfaces]
User2@R1# set ?
Possible completions:
  <interface-name> Interface name
+ apply-groups      Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
  ge-0/0/3          Interface name
> interface-range   Interface ranges configuration
> interface-set     Logical interface set configuration
> traceoptions      Interface trace options
```

In configuration mode, access the system and protocols configuration hierarchies.

```
User2@R1# edit system
      ^
Syntax error, expecting <statement> or <identifier>.
User2@R1# edit protocols
```

^

Syntax error, expecting <statement> or <identifier>.

Meaning

User2 has permissions to configure interfaces on R1, but the user does not have permission to view or modify the [edit system] or [edit protocols] hierarchy levels.

3

CHAPTER

Passwords for User Access

[Root Password | 102](#)

[Recover a Root Password | 107](#)

[Plain-Text Passwords | 111](#)

Root Password

IN THIS SECTION

- [Configure the Root Password | 102](#)
- [Example: Configure a Plain-Text Password for Root Logins | 104](#)

When the device is powered on for the first time, it is ready to be configured. Initially, you log in as the user *root* with no password. You must configure a plain-text password for the root-level user (whose username is *root*) the first time you modify and commit the configuration. Configuring a plain-text password is one way to protect access to the root level by unauthorized users. If you forget the root password for the device, you can use the password recovery procedure to reset the root password.

Configure the Root Password

When you power on the router or switch, it is ready to be configured. Initially, you log in as the user *root* with no password. The root directory is the entry point to all other folders and files on that device. As a result, access to the root directory is restricted by default to a predefined user account known as the *root user*. The root user (also referred to as *superuser*) has unrestricted access and full permissions within the system. The expression “log in as root” is commonly used when an action requires the user to log in to the device as the root user.

NOTE: If you configure a blank password using the encrypted-password statement at the [edit system root-authentication] hierarchy level for root authentication, you can commit a configuration. You *cannot*, however, log in as the root user and gain root level access to the router or switch.

After you log in, you should configure the root (superuser) password by including the root-authentication statement at the [edit system] hierarchy level and configuring one of the password options:

```
[edit system]
root-authentication {
    (encrypted-password "password" | plain-text-password);
    load-key-file URL filename;
```

```
ssh-ecdsa "public-key" <from hostname>;
ssh-rsa "public-key" <from hostname>;
}
```

If you configure the `plain-text-password` option, you are prompted to enter and confirm the password:

```
[edit system]
user@host# set root-authentication plain-text-password
New password: type password here
Retype new password: retype password here
```

The default requirements for plain-text passwords are:

- The password must be between 6 and 128 characters long.
- You can include most character classes in a password (uppercase letters, lowercase letters, numbers, punctuation marks, and other special characters). Control characters are not recommended.
- Valid passwords must contain at least one uppercase letter or one lowercase letter, or one character class.

If you use the `encrypted-password` option, then a null-password (empty) is not permitted. You must configure a password whose number of characters range from 1 through 128 characters and enclose the password in quotation marks.

You can use the `load-key-file URL filename` statement to load an SSH key file that was previously generated using `ssh-keygen`. The `URL filename` option is the path to the file's location and name. When using this option, the contents of the key file are copied into the configuration immediately after entering the `load-key-file URL` statement. This command loads RSA (SSH version 1 and SSH version 2) and DSA (SSH version 2) public keys.

Optionally, you can use the `ssh-ecdsa` or `ssh-rsa` statements to directly configure SSH RSA and ECDSA keys to authenticate root logins. You can configure more than one public key for SSH authentication of root logins as well as for user accounts. When a user logs in as root, the device determines whether the private key matches any of the configured public keys.

```
[edit system]
user@host# set root-authentication load-key-file my-host:.ssh/id_rsa.pub
.file.19692          |          0 KB |    0.3 kB/s | ETA: 00:00:00 | 100%
```


In configuration mode, you can confirm your SSH key entries by entering the `show` command. It should look similar to the following output:

```
[edit system]
user@host# show
root-authentication {
    ssh-rsa "$ABC123"; ## SECRET-DATA
}
```

Example: Configure a Plain-Text Password for Root Logins

IN THIS SECTION

- [Requirements | 104](#)
- [Overview | 105](#)
- [Configuration | 105](#)
- [Verification | 106](#)

This example shows how to configure a plain-text password for the root-level user (the username is *root*). Configuring a plain-text password is one way to prevent unauthorized users from accessing the root level. You must prevent unauthorized users from gaining access to superuser commands that can be used to alter your system configuration.

Requirements

No special configuration beyond device initialization is required before configuring this example.

The default requirements for a plain-text password are as follows:

- Must be from 6 up to 128 characters long.
- Can include most character classes (uppercase letters, lowercase letters, numbers, punctuation marks, and other special characters). Control characters are not recommended.
- Must contain at least one change of case or character class.

Overview

When you power on the router, it is ready to be configured. Initially, you log in as the root-level user with no password. To set the root password, you have several options. This example shows how to enter a plain-text password that the device then encrypts for you.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 105](#)
- [Configure a Plain-Text Password for User Root | 105](#)
- [Results | 106](#)

CLI Quick Configuration

To quickly configure this example, copy the following command and paste it into the window. When prompted, type the new password, and then when prompted, retype it.

```
set system root-authentication plain-text-password
```

Configure a Plain-Text Password for User Root

Step-by-Step Procedure

To configure a plain-text password for the root-level user:

1. Type the `set` command for the plain-text password and press Enter.

```
[edit]
user@host# set system root-authentication plain-text-password
New password:
```

2. Type the new password next to the New password prompt and press Enter.

```
New password: new-password
Retype new password:
```

3. Retype the same password next to the Retype new password prompt and press Enter.

```
New password: new-password
Retype new password: new-password
```

Results

In configuration mode, confirm your configuration by using the `show system` command. It should look something like this:

```
[edit]
user@host# show system
root-authentication {
    encrypted-password "$ABC123"; ## SECRET-DATA
}
```

If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

After you have confirmed that the configuration is correct, enter `commit` in configuration mode.

Verification

IN THIS SECTION

- [Verify the Configuration of a Plain-Text Password for User Root | 106](#)

Verify the Configuration of a Plain-Text Password for User Root

Purpose

Verify the configuration of a plain-text password for the root-level user.

Action

In operational mode, confirm your configuration by entering the `show configuration system` command.

```
user@host> show configuration system
root-authentication {
    encrypted-password "$ABC123"; ## SECRET-DATA
}
```

Meaning

If you use a plain-text password, the device automatically encrypts the password as soon as you configure it. You do not have to configure the device to encrypt the password, as in some other systems. Plain-text passwords are hidden and marked as `## SECRET-DATA` in the configuration. When a user views the configuration, the user sees only the encrypted string, not the unencrypted password.

Recover a Root Password

IN THIS SECTION

- [How to Recover the Root Password for Junos OS Evolved | 107](#)

If you forget the root password, you can use the password recovery procedure to reset the root password.

How to Recover the Root Password for Junos OS Evolved

IN THIS SECTION

- [How to Connect to the Serial Port | 108](#)

This procedure resets the root password without resetting the device configuration to the factory default configuration. Only the root password is reset. This procedure does not affect the device state or any other function.



Video: [How to Recover the Root Password in Junos OS Evolved](#)

How to Connect to the Serial Port

The first task in the password reset operation is to connect to the serial port of the device.

To connect to the serial port:

1. Power off the router by pressing the power button on the front panel.
2. Turn off the power to the management device (usually a computer) that you will use to access the CLI.
3. Plug one end of the Ethernet rollover cable supplied with the router into the RJ-45 to DB-9 serial port adapter supplied with the router.
4. Plug the RJ-45 to DB-9 serial port adapter into the serial port on the management device.
5. Connect the other end of the Ethernet rollover cable to the console port on the router.
6. Turn on the power to the management device.
7. On the management device, start your asynchronous terminal emulation application (such as Microsoft Windows Hyperterminal), and select the appropriate COM port to use (for example, COM1).
8. Configure the port settings as follows:
 - Bits per second: 9600
 - Data bits: 8
 - Parity: None
 - Stop bits: 1
 - Flow control: None
9. Power on the router by pressing the power button on the front panel.

Verify that the POWER LED on the front panel turns green. The terminal emulation screen on your management device displays the router's boot sequence.

How to Recover the Root Password

The password reset operation is triggered early in the boot process. You reset the password in the shell.

To recover the root password:

1. Do a hard reboot of the Routing Engine (that is, shut down the OS, power off the device, and then power on the device).

On the terminal, you will see the following screen:

```
GNU GRUB  version 2.02~beta3
```

```
+-----+
|*Primary ptx-fixed-19.1-16          |
| Primary [Recover password]         |
| Primary-Rollback ptx-fixed-19.1-15 |
| Primary-Rollback [Recover password]|
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
+-----+
```

```
Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands
before booting or `c' for a command-line.
```

2. Use the arrow keys to scroll down to the Primary [Recover password] option, and press Enter.

```
Disk boot ...
IMA is 0
Loading kernel ...ok.
Loading initrd ...ok.
Booting ...
[ 0.791088] Failed to find cpu0 device node
Processing /dev/sda2 for mount on /soft ...[checking]..ok [mounting]..done
Processing /dev/sda5 for mount on /data ...[checking]..ok [mounting]..done
Processing /dev/sda6 for mount on /data/config ...[checking]..ok [mounting]..done
```

```

Processing /dev/sda7 for mount on /data/var ...[checking]..ok [mounting]..done
mkswap: /dev/sda3: warning: wiping old swap signature.
Setting up swapspace version 1, size = 4 GiB (4294963200 bytes)
no label, UUID=7d905478-773b-41e5-8f1d-8166ec03f93a
Processing /dev/sda1 for mount on /boot ...[checking]..ok [mounting]..done
Done with local filesystems setup.
Mounting version junos-linux-install-ptx-x86-64-16.2I20180502181332_evo-builder...done.
System is running with minimal count of software versions
modprobe: FATAL: Module jnx_cbd_fpga_ptx21k not found in directory /lib/modules/4.8.24-
WR2.2.1_standard
Installing kexec kernel...Cannot get kernel page_offset_base symbol address
done
Password recovery in progress. Please enter new password.

```

3. Enter the new password, and then retype the new password and press Enter.

```

New password:
Retype new password:
passwd: password updated successfully
Password recovery done

Welcome to Linux!

```

The reboot proceeds until the login prompt appears.

```

[ OK ] Started Serial Getty on ttyS0.
[ OK ] Reached target Login Prompts.
[ OK ] Started Vsftpd ftp daemon.
[ OK ] Started Network Time Service.
[ OK ] Started strongSwan IPsec IKEv1/IKEv2 daemon using swanctl.
[ OK ] Started Arp filtering arptables.
[ OK ] Started Management Ethernet Interface Manager Service.
[ OK ] Started OFP on RE.
      Starting MGD initialization of schema and database on RE...

Juniper Linux Distribution 2.2.1 re0 ttyS0

re0 login:

```

4. At the prompt, log in as root using new password.

You will see a shell prompt.

```
Last login: Mon May  7 13:09:08 PDT 2018 from xxxx
--- JUNOS builder Linux re0 4.8.24-WR2.2.1_standard #1 SMP PREEMPT Mon Apr 9 13:21:32 PDT
2018 x86_64 x86_64 x86_64 GNU/Linux
root@RE0:~#
```

5. To start the CLI, enter `cli`.

Plain-Text Passwords

IN THIS SECTION

- [Change the Requirements for Plain-Text Passwords | 111](#)
- [How to Change the Requirements for Plain-Text Passwords | 112](#)

Change the Requirements for Plain-Text Passwords

To change the requirements for plain-text passwords, include the `password` statement at the `[edit system login]` hierarchy level:

```
[edit system login]
password {
    change-type (set-transitions | character-set);
    format (sha1 | sha256 | sha512);
    maximum-length length;
    maximum-lifetime days
    minimum-changes number;
    minimum-character-changes number
    minimum-length length;
    minimum-lifetime days
    minimum-lower-cases number;
    minimum-numeric number;
    minimum-reuse number
```



```
minimum-punctuations number;  
minimum-upper-cases number;  
}
```

NOTE: These statements apply to plain-text passwords only, not encrypted passwords.

How to Change the Requirements for Plain-Text Passwords

IN THIS SECTION

- [Overview | 112](#)
- [Configuration | 112](#)

This example shows how to set various maximum and minimum requirements for plain-text passwords to increase password strength.

Overview

You can use a variety of requirements to strengthen plain-text passwords for greater security. Many possible configurations exist at the [edit system login password] hierarchy level that allow you to require users to create plain-text passwords conforming to a particular set of requirements. These requirements may include such things as password length, number of changes, type of characters, numbers, or letter case.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 113](#)
- [Configure the Requirements for Plain-Text Passwords | 113](#)
- [Results | 114](#)

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level.

```
set system login password minimum-length 12
set system login password maximum-length 22
set system login password minimum-numeric 1
set system login password minimum-upper-cases 1
set system login password minimum-lower-cases 1
set system login password minimum-punctuations 1
```

Configure the Requirements for Plain-Text Passwords

Step-by-Step Procedure

This example configures password requirements that require the user to create a password with at least 12 characters but no more than 22 characters. The password requirements also specify at least one lowercase letter and one uppercase letter, at least one punctuation character, and at least one numeric character.

1. Enter configuration mode and navigate to the [edit system login password] hierarchy level.

```
user@host> configure
[edit]
user@host# edit system login password
```

2. Set a minimum length requirement of 12 characters and a maximum length requirement of 22 characters for user passwords.

```
[edit system login password]
user@host# set minimum-length 12
[edit system login password]
user@host# set maximum-length 22
```

3. Require users to set a password that has at least one lowercase letter and at least one uppercase letter.

```
[edit system login password]
user@host# set minimum-lower-cases 1
[edit system login password]
user@host# set minimum-upper-cases 1
```

4. Require users to set a password that has at least one punctuation character and at least one numeric character.

```
[edit system login password]
user@host# set minimum-punctuations 1
[edit system login password]
user@host# set minimum-numeric 1
```

Results

In configuration mode, confirm your configuration by entering the `show` command at the `[edit system login password]` hierarchy level..

```
[edit system login password]
user@host# show
minimum-length 12;
maximum-length 22;
minimum-numeric 1;
minimum-upper-cases 1;
minimum-lower-cases 1;
minimum-punctuations 1;
```

If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

After you have confirmed that the configuration is correct, enter `commit` in configuration mode.

4

CHAPTER

User Authentication

User Authentication Overview | 116

Authentication Order for LDAPS, RADIUS, TACACS+, and Local Password | 124

RADIUS Authentication | 137

TACACS+ Authentication | 161

Authentication for Routing Protocols | 186

User Authentication Overview

IN THIS SECTION

- [User Authentication Methods | 116](#)
- [Configure Local User Template Accounts for User Authentication | 117](#)
- [Configure Remote User Template Accounts for User Authentication | 119](#)
- [Example: Create Template Accounts | 119](#)
- [What Are Remote Authentication Servers? | 123](#)

Junos OS Evolved supports different authentication methods that you (the network administrator) use to control user access to the network. These methods include local password authentication, RADIUS, and TACACS+. You use one of these authentication methods to validate users and devices that attempt to access the router or switch using SSH and Telnet. Authentication prevents unauthorized devices and users from gaining access to your LAN.

User Authentication Methods

Junos OS Evolved supports three methods of user authentication: local password authentication, RADIUS, and TACACS+.

With local password authentication, you configure a password for each user allowed to log in to the router or switch.

RADIUS and TACACS+ are authentication methods for validating users who attempt to access the router or switch using any of the login methods. They are distributed client/server systems—the RADIUS and TACACS+ clients run on the router or switch, and the server runs on a remote network system.

You can configure the router or switch to be a RADIUS or TACACS+ client, or both. You can also configure authentication passwords in the Junos OS Evolved configuration file. You can prioritize the methods to configure the order in which the software tries the different authentication methods when verifying user access.

Configure Local User Template Accounts for User Authentication

You use local user template accounts to assign different login classes, and thus grant different permissions, to users who are authenticated through a remote authentication server. Each template can define a different set of permissions appropriate for the users assigned to that template. You define the templates locally on the router or switch, and the TACACS+ and RADIUS authentication servers reference the templates. When an authenticated user is assigned to a template account, the CLI username is the login name, but the user inherits privileges, file ownership, and effective user ID from the template account.

When you configure local user templates and a user logs in, Junos OS Evolved issues a request to the authentication server to authenticate the user's login name. If the user is authenticated, the server returns the local username to Junos OS Evolved (`local-user-name` for TACACS+ and `Juniper-Local-User-Name` for RADIUS). Junos OS Evolved then determines whether a local username is specified for that login name, and if so, Junos OS Evolved assigns the user to that local user template. If a local user template does not exist for the authenticated user, the router or switch defaults to the `remote` template, if configured.

To configure a local user template, define the template username at the `[edit system login]` hierarchy level. Assign a class to specify the privileges you want to grant to the local users to whom the template applies:

```
[edit system login]
user local-username {
    full-name "Local user account";
    uid uid-value;
    class class-name;
}
```

To assign a user to the local user template, configure the remote authentication server with appropriate parameter (`local-user-name` for TACACS+, and `Juniper-Local-User-Name` for RADIUS), and specify the username defined for the local user template. To configure different access privileges for users who share the local user template account, you can use vendor-specific attributes in the authentication server configuration file to allow or deny specific commands and configuration hierarchies for a user.

This example configures the `sales` and `engineering` user templates on the local device. The TACACS+ server configuration file then assigns users to specific templates.

```
[edit]
system {
    login {
        user sales {
```

```

        uid 6003;
        class sales-class;
    }
    user engineering {
        uid 6004;
        class super-user;
    }
}

```

When the users Simon and Rob are authenticated, the router or switch applies the sales local user template. When login users Harold and Jim are authenticated, the router or switch applies the engineering local user template.

```

user = simon {
    ...
    service = junos-exec {
        local-user-name = sales
        allow-commands = "configure"
        deny-commands = "shutdown"
    }
}
user = rob {
    ...
    service = junos-exec {
        local-user-name = sales
        allow-commands = "(request system) | (show rip neighbor)"
        deny-commands = "clear"
    }
}
user = harold {
    ...
    service = junos-exec {
        local-user-name = engineering
        allow-commands = "monitor | help | show | ping | traceroute"
        deny-commands = "configure"
    }
}
user = jim {
    ...
    service = junos-exec {
        local-user-name = engineering

```

```

        allow-commands = "show bgp neighbor"
        deny-commands = "telnet | ssh"
    }
}

```

Configure Remote User Template Accounts for User Authentication

The network device can map remotely-authenticated users to a locally defined user account or user template account, which determines authorization. The `remote` template account is a special user template. By default, Junos OS Evolved assigns remotely-authenticated users to the `remote` template account, if configured, when:

- The authenticated user does not have a user account configured on the local device.
- The remote authentication server either does not assign the user to a local user template, or the template that the server assigns is not configured on the local device.

To configure the `remote` template account, include the `user remote` statement at the `[edit system login]` hierarchy level, and specify the login class for users assigned to the `remote` template:

```

[edit system login]
user remote {
    full-name "remote users";
    uid uid-value;
    class class-name;
}

```

To configure different access privileges for users who share the `remote` template account, you can use vendor-specific attributes in the authentication server configuration file to allow or deny specific commands and configuration hierarchies for a user.

Example: Create Template Accounts

IN THIS SECTION

● [Requirements](#) | 120

- [Overview | 120](#)
- [Configuration | 120](#)
- [Verification | 122](#)

This example shows how to create template accounts.

Requirements

No special configuration beyond device initialization is required before configuring this feature.

Overview

You can create template accounts that are shared by a set of users when you are using RADIUS or TACACS+ authentication. When an authenticated user is assigned to a template account, the CLI username is the login name, but the user inherits privileges, file ownership, and effective user ID from the template account.

By default, Junos OS Evolved assigns remotely-authenticated users to the `remote` template account when:

- The authenticated user does not have a user account configured on the local device.
- The remote authentication server either does not assign the user to a local user template, or the template that the server assigns is not configured on the local device.

In this example, you create the `remote` template account and set the username to `remote` and the login class for the user as `operator`. The device assigns the `remote` template to users who are authenticated by RADIUS or TACACS+ but who do not have a local user account or belong to a different local template account.

You then create a local template account and set the username as `admin` and the login class as `superuser`. You use local template accounts when you need to assign remotely authenticated users to different login classes. Thus, each template can grant a different set of permissions appropriate for the users assigned to that user template.

Configuration

IN THIS SECTION

- [Create a Remote Template Account | 121](#)
- [Create a Local Template Account | 121](#)

Create a Remote Template Account

Step-by-Step Procedure

To create the remote template account:

- Set the username and the login class for the remote user.

```
[edit system login]
user@host# set user remote class operator
```

Results

In configuration mode, confirm your configuration by entering the `show system login` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show system login
    user remote {
        class operator;
    }
```

After you configure the device, enter `commit` in configuration mode.

Create a Local Template Account

Step-by-Step Procedure

To create a local template account:

1. Set the username and the login class for the user template.

```
[edit system login]
user@host# set user admin class superuser
```

Results

In configuration mode, confirm your configuration by entering the `show system login` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show system login
    user admin {
        class super-user;
    }
```

After you configure the device, enter `commit` in configuration mode.

NOTE: To completely set up RADIUS or TACACS+ authentication, you must configure at least one RADIUS or TACACS+ server and specify a system authentication order. For more information, see the following tasks:

- Configure a RADIUS server. See ["Example: Configure a RADIUS Server for System Authentication" on page 143](#).
- Configure a TACACS+ server. See ["Example: Configure a TACACS+ Server for System Authentication" on page 170](#).
- Configure system authentication order. See ["Example: Configure Authentication Order" on page 133](#).

Verification

IN THIS SECTION

- [Verify the Template Accounts Creation | 123](#)

Confirm that the configuration is working properly.

Verify the Template Accounts Creation

Purpose

Verify that the template accounts have been created.

Action

In operational mode, enter the `show system login` command.

What Are Remote Authentication Servers?

You probably already use a remote authentication server (or servers) in your network. Using these servers is a best practice, because they allow you to create a consistent set of user accounts centrally for all devices in your network. Managing user accounts is much easier when you use remote authentication servers to implement an authentication, authorization, and accountability (AAA) solution in your network.

Most enterprises use one or more of three basic remote authentication methods: LDAPS, RADIUS, and TACACS+. Junos OS Evolved supports RADIUS and TACACS+, and you can configure Junos OS Evolved to query either type of remote authentication server. The idea behind a RADIUS or TACACS+ server is simple: Each acts as a central authentication server that routers, switches, security devices, and servers can use to authenticate users as they attempt to access these systems. Think of the advantages that a central user directory offers for authentication auditing and access control in a client/server model. The RADIUS and TACACS+ authentication methods offer comparable advantages for your network infrastructure.

Using a central server has multiple advantages over the alternative of creating local users on each device, a time-consuming and error-prone task. A central authentication system also simplifies the use of one-time password systems such as SecureID, which offer protection against password sniffing and password replay attacks. In such attacks, someone can use a captured password to pose as a system administrator.

- **RADIUS**—You should use RADIUS when your priorities are interoperability and performance.
 - **Interoperability**—RADIUS is more interoperable than TACACS+, primarily because of the proprietary nature of TACACS+. While TACACS+ supports more protocols, RADIUS is universally supported.
 - **Performance**—RADIUS is much lighter on your routers and switches than TACACS+. For this reason, network engineers generally prefer RADIUS over TACACS+.

- TACACS+—You should use TACACS+ when your priorities are security and flexibility.
- Security—TACACS+ is more secure than RADIUS. Not only is the full session encrypted, but authorization and authentication are done separately to prevent anyone from trying to force their way into your network.
- Flexibility—Transmission Control Protocol (TCP) is a more flexible transport protocol than UDP. You can do more with TCP in more advanced networks. In addition, TACACS+ supports more of the enterprise protocols, such as NetBIOS.

Authentication Order for LDAPS, RADIUS, TACACS+, and Local Password

IN THIS SECTION

- [Authentication Order Overview | 125](#)
- [Configure the Authentication Order for LDAPS, RADIUS, TACACS+ and Local Password Authentication | 131](#)
- [Example: Configure Authentication Order | 133](#)

Junos OS Evolved supports different authentication methods, including local password authentication, RADIUS, and TACACS+ to control access to the network.

NOTE: Junos OS Evolved does not support LDAPS.

When you configure a device to support multiple authentication methods, you can prioritize the order in which the device tries the different methods. This topic discusses how the authentication order works and how to configure it on a device.

Authentication Order Overview

IN THIS SECTION

- [Using Remote Authentication | 126](#)
- [How to Use Local Password Authentication | 126](#)
- [Order of Authentication Attempts | 127](#)

You (the network administrator) can configure the `authentication-order` statement to prioritize the order in which Junos OS Evolved tries different authentication methods to verify user access to a router or switch. If you do not set an authentication order, by default, Junos OS Evolved verifies users based on their configured local passwords.

If the authentication order includes LDAPS, RADIUS, or TACACS+ servers, but the servers do not respond to a request, Junos OS Evolved always defaults to trying local password authentication as a last resort.

If the authentication order includes LDAPS, RADIUS, or TACACS+ servers, but the servers reject the request, the handling of the request is more complicated.

- If `password` (local password authentication) *is* included at the end of the authentication order and the remote authentication servers reject the authentication request, the device attempts local password authentication.
- If `password` (local password authentication) is *not* included in the authentication order and the remote authentication servers reject the authentication request, the request ends with the rejection.

NOTE: In Junos OS Evolved Release 20.4R1 and earlier releases, if the remote authentication servers reject the request, the device still attempts local password authentication.

Thus, the device must include `password` as a final authentication order option for the device to attempt local password authentication in the event that the remote authentication servers reject the request.

If the authentication order is set to `authentication-order password`, then the device uses only local password authentication.

Using Remote Authentication

You can configure Junos OS Evolved to be a RADIUS or TACACS+ authentication client (or both).

NOTE: Junos OS Evolved does not support LDAPS.

If an authentication method included in the `authentication-order` statement is not available, or if the authentication method is available but the corresponding authentication server returns a reject response, Junos OS Evolved tries the next authentication method included in the `authentication-order` statement.

The LDAP, RADIUS, or TACACS+ server authentication might fail for one or more of the following reasons:

- The authentication method is configured, but the corresponding authentication servers are not configured. For instance, the RADIUS and TACACS+ authentication methods are included in the `authentication-order` statement, but the corresponding RADIUS or TACACS+ servers are not configured at the respective `[edit system radius-server]` and `[edit system tacplus-server]` hierarchy levels.
- The authentication server does not respond before the configured timeout value for that server, or before the default timeout, if no timeout is configured.
- The authentication server is not reachable because of a network problem.

The authentication server might return a reject response for one or both of the following reasons:

- The user profile of a user accessing a router or switch is not configured on the authentication server.
- The user enters incorrect logon credentials.

How to Use Local Password Authentication

You can explicitly configure the `password` authentication method in the `authentication-order` statement or use this method as a fallback mechanism when remote authentication servers fail. The `password` authentication method consults the local user profiles configured at the `[edit system login]` hierarchy level. Users can log in to a router or switch using their local username and password in the following scenarios:

- The `password` authentication method (`password`) is explicitly configured as one of the authentication methods in the `authentication-order` statement.

In this case, the device tries local password authentication if no previous authentication method accepts the logon credentials. This is true whether the previous authentication methods fail to respond or they return a reject response because of an incorrect username or password.

- The password authentication method is not explicitly configured as one of the authentication methods in the `authentication-order` statement.

In this case, the operating system only tries local password authentication if all configured authentication methods fail to respond. The operating system does not use local password authentication if any configured authentication method returns a reject response because of an incorrect username or password.

NOTE: In Junos OS Evolved Release 20.4R1 and earlier releases, Junos OS Evolved still tries local password authentication whether the other authentication methods return a reject response or fail to respond.

Order of Authentication Attempts

Table 9 on page 127 describes how the `authentication-order` statement at the `[edit system]` hierarchy level determines the procedure that Junos OS uses to authenticate users for access to a device.

Table 9: Order of Authentication Attempts

Syntax	Order of Authentication Attempts
<code>authentication-order radius;</code>	<div><div>1. Try configured RADIUS authentication servers.</div><div>2. If a RADIUS server is available and authentication is accepted, grant access.</div><div>3. If a RADIUS server is available but authentication is rejected, deny access.</div><div>NOTE: In Junos OS Evolved Release 20.4R1 and earlier releases, if a RADIUS server is available but authentication is rejected, try local password authentication.</div><div>4. If no RADIUS servers are available, try local password authentication.</div></div>

Table 9: Order of Authentication Attempts *(Continued)*

Syntax	Order of Authentication Attempts
authentication-order [radius password];	<ol style="list-style-type: none"> 1. Try configured RADIUS authentication servers. 2. If a RADIUS server is available and authentication is accepted, grant access. 3. If the RADIUS servers fail to respond or the servers return a reject response, try local password authentication, because it is explicitly configured in the authentication order.
authentication-order [radius tacplus];	<ol style="list-style-type: none"> 1. Try configured RADIUS authentication servers. 2. If a RADIUS server is available and authentication is accepted, grant access. 3. If the RADIUS servers fail to respond or the servers return a reject response, try configured TACACS+ servers. 4. If a TACACS+ server is available and authentication is accepted, grant access. 5. If a TACACS+ server is available but authentication is rejected, deny access. NOTE: In Junos OS Evolved Release 20.4R1 and earlier releases, if a TACACS+ server is available but authentication is rejected, try local password authentication. 6. If no RADIUS or TACACS+ servers are available, try local password authentication.

Table 9: Order of Authentication Attempts *(Continued)*

Syntax	Order of Authentication Attempts
authentication-order [radius tacplus password];	<ol style="list-style-type: none"> 1. Try configured RADIUS authentication servers. 2. If a RADIUS server is available and authentication is accepted, grant access. 3. If the RADIUS servers fail to respond or the servers return a reject response, try configured TACACS+ servers. 4. If a TACACS+ server is available and authentication is accepted, grant access. 5. If the TACACS+ servers fail to respond or the servers return a reject response, try local password authentication, because it is explicitly configured in the authentication order.
authentication-order tacplus;	<ol style="list-style-type: none"> 1. Try configured TACACS+ authentication servers. 2. If a TACACS+ server is available and authentication is accepted, grant access. 3. If a TACACS+ server is available but authentication is rejected, deny access. NOTE: In Junos OS Evolved Release 20.4R1 and earlier releases, if a TACACS+ server is available but authentication is rejected, try local password authentication. 4. If no TACACS+ servers are available, try local password authentication.
authentication-order [tacplus password];	<ol style="list-style-type: none"> 1. Try configured TACACS+ authentication servers. 2. If a TACACS+ server is available and authentication is accepted, grant access. 3. If the TACACS+ servers fail to respond or the servers return a reject response, try local password authentication, because it is explicitly configured in the authentication order.

Table 9: Order of Authentication Attempts (*Continued*)

Syntax	Order of Authentication Attempts
authentication-order [tacplus radius];	<ol style="list-style-type: none"> 1. Try configured TACACS+ authentication servers. 2. If a TACACS+ server is available and authentication is accepted, grant access. 3. If the TACACS+ servers fail to respond or the servers return a reject response, try configured RADIUS servers. 4. If a RADIUS server is available and authentication is accepted, grant access. 5. If a RADIUS server is available but authentication is rejected, deny access. <p>NOTE: In Junos OS Evolved Release 20.4R1 and earlier releases, if a RADIUS server is available but authentication is rejected, try local password authentication.</p> <ol style="list-style-type: none"> 6. If no TACACS+ or RADIUS servers are available, try local password authentication.
authentication-order [tacplus radius password];	<ol style="list-style-type: none"> 1. Try configured TACACS+ authentication servers. 2. If a TACACS+ server is available and authentication is accepted, grant access. 3. If the TACACS+ servers fail to respond or the servers return a reject response, try configured RADIUS servers. 4. If a RADIUS server is available and authentication is accepted, grant access. 5. If the RADIUS servers fail to respond or the servers return a reject response, try local password authentication, because it is explicitly configured in the authentication order.

Table 9: Order of Authentication Attempts *(Continued)*

Syntax	Order of Authentication Attempts
authentication-order password;	<ol style="list-style-type: none"> 1. Try to authenticate the user using the password configured at the [edit system login] hierarchy level. 2. If the authentication is accepted, grant access. 3. If the authentication is rejected, deny access.

NOTE: If SSH public keys are configured, SSH user authentication first tries to perform public key authentication before using the authentication methods configured in the authentication-order statement. If you want SSH logins to use the authentication methods configured in the authentication-order statement without first trying to perform public key authentication, do not configure SSH public keys.

Configure the Authentication Order for LDAPS, RADIUS, TACACS+ and Local Password Authentication

Using the authentication-order statement, you can prioritize the order in which Junos OS Evolved tries the different authentication methods when verifying user access to a router or switch. If you do not set an authentication order, by default, users are verified based on their locally configured passwords.

When configuring a password using plain text and relying on Junos OS Evolved to encrypt it, you are still sending the password over the Internet in plain text. Using pre-encrypted passwords is more secure because it means that the plain text of the password never has to be sent over the internet. Also, with passwords, only one user can be assigned to a password at a time.

On the other hand, LDAPS, RADIUS, and TACACS+ encrypt passwords. These authentication methods let you assign a set of users at a time instead of assigning users one by one. But here are how these authentication systems differ:

- RADIUS uses UDP; TACACS+ and LDAPS use TCP.
- RADIUS encrypts only the password during transmission, whereas TACACS+ and LDAPS encrypt the entire session.

- RADIUS and LDAPS combine authentication (device) and authorization (user), whereas TACACS+ separates authentication, authorization, and accountability.

In short, TACACS+ is more secure than RADIUS. However, RADIUS has better performance and is more interoperable. RADIUS is widely supported, whereas TACACS+ is a Cisco proprietary product and not widely supported outside of Cisco.

LDAPS is more secure than RADIUS and TACACS+, as it relies on a private key mechanism instead of the shared key used in the case of RADIUS and TACACS+. The TLS protocol secures the transmission of data effectively between the LDAP client and the LDAP server.

You can configure the authentication order based on your system, its restrictions, and your IT policy and operational preferences.

To configure the authentication order, include the `authentication-order` statement at the `[edit system]` hierarchy level.

```
[edit system]
user@host# set authentication-order [authentication-methods ]
```

For a list of hierarchy levels at which you can include this statement, see the statement summary section for this statement.

The following are the possible authentication order entry options:

- `radius`—Verify the user using RADIUS authentication servers.
- `tacplus`—Verify the user using TACACS+ authentication servers.
- `password`—Verify the user using the username and password configured locally in the authentication statement at the `[edit system login user]` hierarchy level.

The Challenge Handshake Authentication Protocol (CHAP) authentication sequence cannot take more than 30 seconds. If it takes longer than 30 seconds to authenticate a client, the authentication is abandoned and a new sequence is initiated.

For example, assume that you configure three RADIUS servers so that the router or switch attempts to contact each server three times. Assume further that, with each retry, the server times out after 3 seconds. In this scenario, the maximum time given to the RADIUS authentication method before CHAP considers it a failure is 27 seconds. If you add more RADIUS servers to this configuration, they might not be contacted because the authentication process might be abandoned before these servers are tried.

Junos OS Evolved enforces a limit on the number of standing authentication server requests that the CHAP authentication can have at one time. Thus, an authentication server method—RADIUS, for example—might fail to authenticate a client when this limit is exceeded. If authentication fails, the

authentication sequence is reinitiated by the router or switch until authentication succeeds and the link is established. However, if the RADIUS servers are unavailable and additional authentication methods such as tacplus or password are also configured, the next authentication method is tried.

The following example shows how to configure radius and password authentication:

```
[edit system]
user@switch# set authentication-order [ radius password ]
```

The following example shows how to insert the tacplus statement after the radius statement:

```
[edit system]
user@switch# insert authentication-order tacplus after radius
```

The following example shows how to delete the radius statement from the authentication order:

```
[edit system]
user@switch# delete authentication-order radius
```

Example: Configure Authentication Order

IN THIS SECTION

- [Requirements | 133](#)
- [Overview | 134](#)
- [Configuration | 134](#)
- [Verification | 136](#)

This example shows how to configure authentication order for user login.

Requirements

Before you begin, perform the initial device configuration. See the Getting Started Guide for your device.

Overview

You can configure the authentication method order that a device uses to verify user access to the device. For each login attempt, the device tries the authentication methods in the order configured, until the password matches or all authentication methods have been tried. If you do not configure remote authentication, users are verified based on their configured local passwords.

This example configures the device to attempt user authentication with RADIUS authentication services first, then with TACACS+ authentication services, and finally with local password authentication.

When you use local password authentication, you must create a local user account for every user who wants to access the system. However, when you use remote authentication servers, you can create template accounts (for authorization purposes) that a set of users shares. When a user is assigned to a template account, the command-line interface (CLI) username is the login name; however, the user inherits the privileges, file ownership, and effective user ID from the template account.

Configuration

IN THIS SECTION

- [Procedure | 134](#)

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` in configuration mode.

```
delete system authentication-order
set system authentication-order radius
insert system authentication-order tacplus after radius
insert system authentication-order password after tacplus
```

Step-by-Step Procedure

To configure authentication order:

1. Delete any existing authentication-order statement.

```
[edit]
user@host# delete system authentication-order
```

2. Add RADIUS authentication to the authentication order.

```
[edit]
user@host# set system authentication-order radius
```

3. Add TACACS+ authentication to the authentication order.

```
[edit]
user@host# insert system authentication-order tacplus after radius
```

4. Add local password authentication to the authentication order.

```
[edit]
user@host# insert system authentication-order password after tacplus
```

Results

In configuration mode, confirm your configuration by entering the `show system authentication-order` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show system authentication-order
authentication-order [ radius tacplus password ];
```

After you configure the device, enter `commit` in configuration mode.

NOTE: To completely set up RADIUS or TACACS+ authentication, you must configure at least one RADIUS or TACACS+ server and create user accounts or user template accounts.

- Configure a RADIUS server. See ["Example: Configure a RADIUS Server for System Authentication" on page 143](#).
- Configure a TACACS+ server. See ["Example: Configure a TACACS+ Server for System Authentication" on page 170](#).
- Configure a user. See ["Example: Configure New User Accounts" on page 20](#).
- Configure template accounts. See ["Example: Create Template Accounts" on page 119](#).

Verification

IN THIS SECTION

- [Verify the Authentication Order Configuration | 136](#)

Confirm that the configuration is working properly.

Verify the Authentication Order Configuration

Purpose

Verify that the device uses the authentication methods in the order configured.

Action

Create a test user that has a different password for each authentication method. Log in to the device using the different passwords. Verify that the device queries subsequent authentication methods when the previous methods reject the password or fail to respond.

Alternatively, in a test environment, you can deactivate the authentication server configuration or the local user account configuration (or both) to test each authentication method. For example, to test the TACACS+ server, you can deactivate the RADIUS server configuration and the user's local account. However, if you deactivate the user's local account, you must ensure that the user still maps to a local user template account such as the `remote` user template.

Release History Table

Release	Description
20.4R1	Starting in Junos OS Evolved Release 20.4R2 and 21.1R1, password authentication behavior is updated to match the password authentication behavior of Junos OS: If the authentication order includes LDAPS, RADIUS, or TACACS+ servers, but the servers do not respond to a request, Junos OS Evolved always defaults to trying local password authentication as a last resort.

RADIUS Authentication

IN THIS SECTION

- [Configure RADIUS Server Authentication | 137](#)
- [Example: Configure a RADIUS Server for System Authentication | 143](#)
- [Juniper Networks Vendor-Specific RADIUS and LDAP Attributes | 147](#)
- [Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands | 152](#)
- [Understanding RADIUS Accounting | 156](#)
- [Configure RADIUS System Accounting | 157](#)

Junos OS Evolved supports RADIUS for central authentication of users on network devices. To use RADIUS authentication on the device, you (the network administrator) must configure information about one or more RADIUS servers on the network. You can also configure RADIUS accounting on the device to collect statistical data about the users logging in to or out of a LAN and send the data to a RADIUS accounting server.

Configure RADIUS Server Authentication

IN THIS SECTION

- [Why Use RADIUS | 138](#)

- [Configure RADIUS Server Details | 138](#)
- [Configure RADIUS to Use the Management Instance | 142](#)

RADIUS authentication is a method of authenticating users who attempt to access a network device. The following sections describe why you would use RADIUS and how to configure it.

Why Use RADIUS

You (the network administrator) can use different protocols for the central authentication of users on network devices including RADIUS and TACACS+. We recommend RADIUS because it is a multivendor IETF standard and its features are more widely accepted than those of TACACS+ or other proprietary systems. In addition, we recommend using a one-time-password system for increased security, and all vendors of these systems support RADIUS.

You should use RADIUS when your priorities are interoperability and performance:

- Interoperability—RADIUS is more interoperable than TACACS+, primarily because of the proprietary nature of TACACS+. While TACACS+ supports more protocols, RADIUS is universally supported.
- Performance—RADIUS is much lighter on your routers and switches. For this reason, network engineers generally prefer RADIUS over TACACS+.

Configure RADIUS Server Details

To use RADIUS authentication on the device, configure information about one or more RADIUS servers on the network by including one `radius-server` statement at the `[edit system]` hierarchy level for each RADIUS server. The device queries the RADIUS servers in the order in which they are configured. If the primary server (the first one configured) is unavailable, the device attempts to contact each server in the list until it receives a response.

The network device can map RADIUS-authenticated users to a locally defined user account or user template account, which determines authorization. By default, Junos OS Evolved assigns RADIUS-authenticated users to the user template account `remote`, if configured, when:

- The authenticated user does not have a user account configured on the local device.
- The RADIUS server either does not assign the user to a local user template, or the template that the server assigns is not configured on the local device.

The RADIUS server can assign an authenticated user to a different user template to grant different administrative permissions to that user. The user retains the same login name in the CLI but inherits the login class, access privileges, and effective user ID from the assigned template. If the RADIUS-

authenticated user does not map to any locally defined user account or user template, and the `remote` template is not configured, then authentication fails.

NOTE: The `remote` username is a special case in Junos OS Evolved and must always be lowercase. It acts as a template for users who are authenticated by a remote server but do not have a locally configured user account on the device. Junos OS Evolved applies the permissions of the `remote` template to those authenticated users without a locally defined account. All users mapped to the `remote` template are in the same login class.

Because you configure remote authentication on multiple devices, it is common to configure it inside of a configuration group. The steps shown here are in a configuration group called `global`. Using a configuration group is optional.

To configure authentication by a RADIUS server:

1. Configure the IPv4 address or the IPv6 address of the RADIUS authentication server.

```
[edit groups global system radius-server]
user@host# set server-address
```

For example:

```
[edit groups global system radius-server]
user@host# set 192.168.17.28
```

```
[edit groups global system radius-server]
user@host# set 2001:db8:0:f101::8
```

2. (Optional) Configure the packet source address for requests sent to the RADIUS server.

```
[edit groups global system radius-server server-address]
user@host# set source-address source-address
```

For example:

```
[edit groups global system radius-server 192.168.17.28]
user@host# set source-address 192.168.17.1
```

```
[edit groups global system radius-server 2001:db8:0:f101::8]
user@host# set source-address 2001:db8:0:f101::1
```

The source address is a valid IPv4 address or IPv6 address configured on one of the router interfaces or switch interfaces. If the network device has several interfaces that can reach the RADIUS server, assign an IP address that the device can use for all its communication with the RADIUS server. Doing this sets a fixed address as the source address for locally generated IP packets.

3. Configure the shared secret password that the network device uses to authenticate with the RADIUS server.

The configured password must match the password that is configured on the RADIUS server. If the password contains spaces, enclose it in quotation marks. The device stores the password as an encrypted value in the configuration database.

```
[edit groups global system radius-server server-address]
user@host# set secret password
```

For example:

```
[edit groups global system radius-server 192.168.17.28]
user@host# set secret Radiussecret1
```

4. (Optional) Specify the port on which to contact the RADIUS server, if different from the default. The default port is 1812 (as specified in RFC 2865).

```
[edit groups global system radius-server server-address]
user@host# set port port-number
```

For example:

```
[edit groups global system radius-server 192.168.17.28]
user@host# set port 51812
```

NOTE: You can also configure the `accounting-port` statement to specify to which RADIUS server port to send accounting packets. The default is 1813 (as specified in RFC 2866).

5. (Optional) Configure the number of times that the device attempts to contact the RADIUS server and the amount of time that the device waits to receive a response from the server.

By default, the device attempts to contact the server three times and waits three seconds. You can configure the `retry` value from 1 through 100 times and the `timeout` value from 1 through 1000 seconds.

```
[edit groups global system radius-server server-address]
user@host# set retry number
user@host# set timeout seconds
```

For example, to contact a RADIUS server 2 times and wait 10 seconds for a response:

```
[edit groups global system radius-server 192.168.17.28]
user@host# set retry 2
user@host# set timeout 10
```

6. Specify the authentication order, and include the `radius` option.

```
[edit groups global system]
user@host# set authentication-order [ authentication-methods ]
```

In the following example, whenever a user attempts to log in, Junos OS Evolved first queries the RADIUS server for authentication. If that fails, it queries the TACACS+ server. If that fails, it attempts authentication with locally configured user accounts.

```
[edit groups global system]
user@host# set authentication-order [ radius tacplus password ]
```

7. Assign a login class to RADIUS-authenticated users who do not have a locally defined user account. You configure a user template account in the same way as a local user account, except that you do not configure a local authentication password because the RADIUS server authenticates the user.

- To use the same permissions for all RADIUS-authenticated users, configure the `remote user` template.

```
[edit groups global system login]
user@host# set user remote class class
```

For example:

```
[edit groups global system login]
user@host# set user remote class super-user
```

- To use different login classes for different RADIUS-authenticated users, granting them different permissions:

- a. Create multiple user templates in the Junos OS Evolved configuration. For example:

```
[edit groups global system login]
user@host# set user RO class read-only
user@host# set user OP class operator
user@host# set user SU class super-user
user@host# set user remote full-name "default remote access user template"
user@host# set user remote class read-only
```

- b. Configure the RADIUS server to map the authenticated user to the appropriate user template.

Set the Juniper-Local-User-Name Juniper VSA (vendor-specific attribute) (Vendor 2636, type 1, string) to the name of a user template configured on the device, which in the previous example is RO, OP, or SU. The RADIUS server includes the attribute in the RADIUS Access-Accept message. Authentication fails if the device cannot assign a user to a local user account or user template, and the `remote user` template is not configured.

Configure RADIUS to Use the Management Instance

By default, Junos OS Evolved routes authentication, authorization, and accounting packets for RADIUS through the default routing instance. You can also route RADIUS packets through a management interface in a non-default VRF instance.

To route RADIUS packets through the `mgmt_junos` management instance:

1. Enable the `mgmt_junos` management instance.

```
[edit system]
user@host# set management-instance
```

2. Configure the routing-instance `mgmt_junos` statement for the RADIUS authentication server and the RADIUS accounting server, if configured.

```
[edit system]
user@host# set radius-server server-address routing-instance mgmt_junos
user@host# set accounting destination radius server server-address routing-instance mgmt_junos
```

Example: Configure a RADIUS Server for System Authentication

IN THIS SECTION

- [Requirements | 143](#)
- [Overview | 143](#)
- [Configuration | 144](#)
- [Verification | 146](#)

This example configures system authentication through a RADIUS server.

Requirements

Before you begin:

- Perform the initial device configuration. See the Getting Started Guide for your device.
- Set up at least one RADIUS server on your network.

Overview

In this example, you add a new RADIUS server with an IP address of 172.16.98.1. You specify the shared secret password of the RADIUS server as `Radiussecret1`. The device stores the secret in the configuration database as an encrypted value. Finally, you specify the source address that the device

uses in RADIUS server requests. In most cases, you can use the loopback address of the device, which in this example is 10.0.0.1.

You can configure support for multiple user authentication methods, such as local password authentication, RADIUS, and TACACS+, on the network device. When you configure multiple authentication methods, you can prioritize the order in which the device tries the different methods. In this example, you configure the device to use RADIUS authentication services first and then, if that fails, to attempt local password authentication.

A RADIUS-authenticated user must map to a local user account or a local user template account on the network device, which determines authorization. By default, if a RADIUS-authenticated user does not map to a local user account or a specific user template, the user is assigned to the remote user template, if configured. This example configures the remote user template.

Configuration

IN THIS SECTION

- [Procedure | 144](#)

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

```
set system radius-server 172.16.98.1
set system radius-server 172.16.98.1 secret Radiussecret1
set system radius-server 172.16.98.1 source-address 10.0.0.1
set system authentication-order [radius password]
set system login user remote class operator
```

Step-by-Step Procedure

To configure a RADIUS server for system authentication:

1. Add a new RADIUS server and set its IP address.

```
[edit system]
user@host# set radius-server 172.16.98.1
```

2. Specify the shared secret (password) of the RADIUS server.

```
[edit system]
user@host# set radius-server 172.16.98.1 secret Radiussecret1
```

3. Specify the device's loopback address as the source address.

```
[edit system]
user@host# set radius-server 172.16.98.1 source-address 10.0.0.1
```

4. Specify the device's order of authentication, and include the radius option.

```
[edit system]
user@host# set authentication-order [radius password]
```

5. Configure the remote user template and its login class.

```
[edit system]
user@host# set login user remote class operator
```

Results

In configuration mode, confirm your configuration by entering the `show system` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

The following output includes only those portions of the configuration hierarchy that are relevant to this example.

```
[edit]
user@host# show system
login {
```

```
user remote {  
    class operator;  
}  
}  
authentication-order [ radius password ];  
radius-server {  
    172.16.98.1 {  
        secret "$9$ABC123"; ## SECRET-DATA  
        source-address 10.0.0.1;  
    }  
}
```

After configuring the device, enter `commit` in configuration mode.

Verification

IN THIS SECTION

- [Verify the RADIUS Server Configuration | 146](#)

Confirm that the configuration is working properly.

Verify the RADIUS Server Configuration

Purpose

Verify that the RADIUS server authenticates users.

Action

Log in to the network device, and verify that the login is successful. To verify that the device uses the RADIUS server for authentication, you can attempt to log in with an account that does not define a local authentication password in the configuration.

Juniper Networks Vendor-Specific RADIUS and LDAP Attributes

Junos OS Evolved supports configuring Juniper Networks RADIUS vendor-specific attributes (VSAs) on the RADIUS server. These VSAs are encapsulated in a RADIUS vendor-specific attribute with the vendor ID set to the Juniper Networks ID number, 2636.

NOTE: Junos OS Evolved does not support LDAP Authentication.

Table 10 on page 147 lists the Juniper Networks VSAs that you can configure.

Some of the attributes accept extended regular expressions, as defined in POSIX 1003.2. If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. For more information, see:

- ["Regular Expressions to Allow and Deny Operational Mode Commands, Configuration Statements, and Hierarchies" on page 51](#)
- ["Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands" on page 176](#)

Table 10: Juniper Networks Vendor-Specific RADIUS and LDAP Attributes

Name	Description	Type	Length	String
Juniper-Local-User-Name	Indicates the name of the user template assigned to this user when the user logs in to a device. This attribute is used only in Access-Accept packets.	1	≥3	One or more octets containing printable ASCII characters.
Juniper-Allow-Commands	Contains an extended regular expression that enables the user to run commands in addition to those commands authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	2	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.

Table 10: Juniper Networks Vendor-Specific RADIUS and LDAP Attributes (Continued)

Name	Description	Type	Length	String
Juniper-Deny-Commands	Contains an extended regular expression that denies the user permission to run commands authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	3	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
Juniper-Allow-Configuration	Contains an extended regular expression that enables the user to view and modify configuration statements in addition to those statements authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	4	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
Juniper-Deny-Configuration	Contains an extended regular expression that denies the user permission to view or modify configuration statements authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	5	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
Juniper-Interactive-Command	Indicates the interactive command entered by the user. This attribute is used only in Accounting-Request packets.	8	≥3	One or more octets containing printable ASCII characters.

Table 10: Juniper Networks Vendor-Specific RADIUS and LDAP Attributes *(Continued)*

Name	Description	Type	Length	String
Juniper-Configuration-Change	Indicates the interactive command that results in a configuration (database) change. This attribute is used only in Accounting-Request packets.	9	≥3	One or more octets containing printable ASCII characters.

Table 10: Juniper Networks Vendor-Specific RADIUS and LDAP Attributes (*Continued*)

Name	Description	Type	Length	String
Juniper-User-Permissions	<p>Contains information the server uses to specify user permissions. This attribute is used only in Access-Accept packets.</p> <p>NOTE: When the RADIUS or LDAP server defines the Juniper-User-Permissions attribute to grant the maintenance permission or all permission to a user, the user's list of group memberships does not automatically include the UNIX wheel group. Some operations such as running the <code>su root</code> command from a local shell require wheel group membership permissions. However, when the network device defines a local user account with the permissions maintenance or all, the user is automatically granted membership to the UNIX wheel group. Therefore, we recommend that you create a user template account with the required permissions and associate individual user accounts with the user template account.</p>	10	≥3	<p>One or more octets containing printable ASCII characters.</p> <p>The string is a list of permission flags separated by a space. The exact name of each flag must be specified in its entirety.</p> <p>See "Access Privilege Levels Overview" on page 41.</p>

Table 10: Juniper Networks Vendor-Specific RADIUS and LDAP Attributes (Continued)

Name	Description	Type	Length	String
Juniper-Authentication-Type	Indicates the authentication method (local database, LDAP or RADIUS server) used to authenticate a user. If the user is authenticated using a local database, the attribute value shows 'local'. If the user is authenticated using a RADIUS or LDAP server, the attribute value shows 'remote'.	11	≥5	One or more octets containing printable ASCII characters.
Juniper-Session-Port	Indicates the source port number of the established session.	12	size of integer	Integer
Juniper-Allow-Configuration-Regexps (RADIUS only)	Contains an extended regular expression that enables the user to view and modify configuration statements in addition to those statements authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	13	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
Juniper-Deny-Configuration-Regexps (RADIUS only)	Contains an extended regular expression that denies the user permission to view or modify configuration statements authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	14	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.

For more information about the VSAs, see RFC 2138, *Remote Authentication Dial In User Service (RADIUS)*.

Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands

Junos OS Evolved can map RADIUS- and TACACS+-authenticated users to a locally defined user account or user template account, which defines the user's access privileges. You can also optionally configure a user's access privileges by defining Juniper Networks RADIUS and TACACS+ vendor-specific attributes (VSAs) on the respective authentication server.

A user's login class defines the set of permissions that determines which operational mode and configuration mode commands a user is authorized to execute and which areas of the configuration a user can view and modify. A login class can also define regular expressions that allow or deny a user the ability to execute certain commands or view and modify certain areas of the configuration, in addition to what the permission flags authorize. A login class can include the following statements to define user authorization:

- `permissions`
- `allow-commands`
- `allow-commands-regexps`
- `allow-configuration`
- `allow-configuration-regexps`
- `deny-commands`
- `deny-commands-regexps`
- `deny-configuration`
- `deny-configuration-regexps`

Similarly, a RADIUS or TACACS+ server configuration can use Juniper Networks VSAs to define specific permissions or regular expressions that determine a user's access privileges. For the list of supported RADIUS and TACACS+ VSAs, see the following:

- ["Juniper Networks Vendor-Specific RADIUS and LDAP Attributes" on page 147](#)
- ["Juniper Networks Vendor-Specific TACACS+ Attributes" on page 174](#)

You can define user permissions on the RADIUS or TACACS+ server as a list of space-separated values.

- A RADIUS server uses the following attribute and syntax:

```
Juniper-User-Permissions += "flag1 flag2 flag3",
```

For example:

```
Juniper-User-Permissions += "interface interface-control configure",
```

- A TACACS+ server uses the following attribute and syntax:

```
user-permissions = "flag1 flag2 flag3"
```

For example:

```
user-permissions = "interface interface-control configure"
```

A RADIUS or TACACS+ server can also define Juniper Networks VSAs that use a single extended regular expression (as defined in POSIX 1003.2) to allow or deny a user the ability to execute certain commands or view and modify areas of the configuration. You enclose multiple commands or configuration hierarchies in parentheses and separate them using a pipe symbol. If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. When you configure authorization parameters both locally and remotely, the device merges the regular expressions received during TACACS+ or RADIUS authorization with any regular expressions defined on the local device.

- A RADIUS server uses the following attributes and syntax:

```
Juniper-Allow-Commands += "(cmd1)|(cmd2)|(cmdn)",
Juniper-Deny-Commands += "(cmd1)|(cmd2)|(cmdn)",
Juniper-Allow-Configuration += "(config1)|(config2)|(confign)",
Juniper-Deny-Configuration += "(config1)|(config2)|(confign)",
```

For example:

```
Juniper-Allow-Commands += "(test)|(ping)|(quit)",
Juniper-Deny-Commands += "(request)|(restart)",
Juniper-Allow-Configuration += "(groups re0)|(system radius-server)",
Juniper-Deny-Configuration += "(system radius-options)|(system accounting)",
```

- A TACACS+ server uses the following attributes and syntax:

```
allow-commands = "(cmd1)|(cmd2)|(cmdn)"
deny-commands = "(cmd1)|(cmd2)|(cmdn)"
allow-configuration = "(config1)|(config2)|(confign)"
deny-configuration = "(config1)|(config2)|(confign)"
```

For example:

```
allow-commands = "(test)|(ping)|(quit)"
deny-commands = "(request)|(restart)"
allow-configuration = "(groups re0)|(system tacplus-server)"
deny-configuration = "(system tacplus-options)|(system accounting)"
```

RADIUS and TACACS+ servers also support configuring attributes that correspond to the same *-regexps statements that you can configure on the local device. The *-regexps TACACS+ attributes and the *-Regexps RADIUS attributes use the same regular expression syntax as the previous attributes, but they enable you to configure regular expressions with variables.

- A RADIUS server uses the following attributes and syntax:

```
Juniper-Allow-Configuration-Regexps += "(config1)|(config2)|(confign)",
Juniper-Deny-Configuration-Regexps += "(config1)|(config2)|(confign)",
```

- A TACACS+ server uses the following attributes and syntax:

```
allow-commands-regexps = "(cmd1)|(cmd2)|(cmdn)"
deny-commands-regexps = "(cmd1)|(cmd2)|(cmdn)"
allow-configuration-regexps = "(config1)|(config2)|(confign)"
deny-configuration-regexps = "(config1)|(config2)|(confign)"
```

For example, the TACACS+ server configuration might define the following attributes:

```
allow-commands-regexps = "(show cli .*)|(ping 10.1.1..*)"
deny-commands-regexps = "(configure .*)|(edit)|(commit)|(rollback .*)"
```

On a RADIUS or TACACS+ server, you can also define the attributes using a simplified syntax where you specify each individual expression on a separate line.

For a RADIUS server, specify the individual regular expressions using the following syntax:

```
Juniper-User-Permissions += "permission-flag1",
Juniper-User-Permissions += "permission-flag2",
Juniper-User-Permissions += "permission-flagn",
Juniper-Allow-Commands += "cmd1",
Juniper-Allow-Commands += "cmd2",
Juniper-Allow-Commands += "cmdn",
Juniper-Deny-Commands += "cmd1",
Juniper-Deny-Commands += "cmd2",
Juniper-Deny-Commands += "cmdn",
Juniper-Allow-Configuration += "config1",
Juniper-Allow-Configuration += "config2",
Juniper-Allow-Configuration += "confign",
Juniper-Deny-Configuration += "config1",
Juniper-Deny-Configuration += "config2",
Juniper-Deny-Configuration += "confign",
```

For a TACACS+ server, specify the individual regular expressions using the following syntax:

```
user-permissions1 = "permission-flag1"
user-permissions2 = "permission-flag2"
user-permissionsn = "permission-flagn"
allow-commands1 = "cmd1"
allow-commands2 = "cmd2"
allow-commandsn = "cmdn"
deny-commands1 = "cmd1"
deny-commands2 = "cmd2"
deny-commandsn = "cmdn"
allow-configuration1 = "config1"
allow-configuration2 = "config2"
allow-configurationn = "confign"
deny-configuration1 = "config1"
deny-configuration2 = "config2"
deny-configurationn = "confign"
```

NOTE:

- In the TACACS+ server syntax, numeric values 1 through n must be unique but need not be sequential. For example, the following syntax is valid:

```
allow-commands1="cmd1"
allow-commands3="cmd3"
allow-commands2="cmd2"
deny-commands3="cmd3"
deny-commands2="cmd2"
deny-commands1="cmd1"
```

- The RADIUS or TACACS+ server imposes a limit on the number of individual regular expression lines.
- When you issue the `show cli authorization` command, the command output displays the regular expression in a single line, even if you specify each individual expression on a separate line.

Users can verify their class, permissions, and command and configuration authorization by issuing the `show cli authorization` operational mode command.

```
user@host> show cli authorization
```

NOTE: When you configure the authorization parameters both locally on the network device and remotely on the RADIUS or TACACS+ server, the device merges the regular expressions received during TACACS+ or RADIUS authorization with any locally configured regular expressions. If the final expression contains a syntax error, the overall result is an invalid regular expression.

Understanding RADIUS Accounting

Network devices support IETF RFC 2866, *RADIUS Accounting*. You can configure RADIUS accounting on a device to collect statistical data about users logging in to or out of a LAN and send the data to a RADIUS accounting server. The statistical data can be used for general network monitoring, analyzing and tracking usage patterns, or billing a user based on the duration of the session or type of services accessed.

To configure RADIUS accounting, specify:

- One or more RADIUS accounting servers to receive the statistical data from the device

- The type of accounting data to collect

You can use the same server for both RADIUS accounting and authentication, or you can use separate servers. You can specify a list of RADIUS accounting servers. The device queries the servers in the order in which they are configured. If the primary server (the first one configured) is unavailable, the device attempts to contact each server in the list until it receives a response.

The RADIUS accounting process between the device and a RADIUS server works like this:

1. A RADIUS accounting server listens for User Datagram Protocol (UDP) packets on a specific port. The default port for RADIUS accounting is 1813.
2. The device forwards an *Accounting-Request* packet containing an event record to the accounting server. The event record associated with this supplicant contains an *Acct-Status-Type* attribute whose value indicates the beginning of user service for this supplicant. When the supplicant's session ends, the accounting request contains an *Acct-Status-Type* attribute value indicating the end of user service. The RADIUS accounting server records this as a stop-accounting record containing session information and the length of the session.
3. The RADIUS accounting server logs these events in a file as start-accounting or stop-accounting records. On FreeRADIUS, the filename is the server's address, such as 192.0.2.0.
4. The accounting server sends an *Accounting-Response* packet to the device confirming that it has received the accounting request.
5. If the device does not receive an *Accounting-Response* packet from the server, it continues to send accounting requests until the server returns a response.

You can view the statistics collected through this process on the RADIUS server. To see those statistics, access the log file configured to receive them.

Configure RADIUS System Accounting

IN THIS SECTION

- [Configure Auditing of User Events on a RADIUS Server | 158](#)

When you enable RADIUS accounting, Juniper Networks devices, acting as RADIUS clients, can notify the RADIUS server about user activities such as software logins, configuration changes, and interactive commands. The framework for RADIUS accounting is described in RFC 2866, *RADIUS Accounting*.

Configure Auditing of User Events on a RADIUS Server

To configure RADIUS accounting:

1. Configure the events to audit.

```
[edit system accounting]
user@host# set events [ events ]
```

For example:

```
[edit system accounting]
user@host# set events [ login change-log interactive-commands ]
```

events can include one or more of the following:

- login—Audit logins
- change-log—Audit configuration changes
- interactive-commands—Audit interactive commands (any command-line input)

2. Enable RADIUS accounting.

```
[edit]
user@host# set system accounting destination radius
```

3. Configure the address for one or more RADIUS accounting servers.

```
[edit system accounting destination radius]
user@host# set server server-address
```

For example:

```
[edit system accounting destination radius]
user@host# set server 192.168.17.28
```

NOTE: If you do not configure any RADIUS servers at the [edit system accounting destination radius] hierarchy level, the device uses the RADIUS servers configured at the [edit system radius-server] hierarchy level.

4. (Optional) Configure the source address for RADIUS accounting requests.

```
[edit system accounting destination radius server server-address]
user@host# set source-address source-address
```

For example:

```
[edit system accounting destination radius server 192.168.17.28]
user@host# set source-address 192.168.17.1
```

The source address is a valid IPv4 address or IPv6 address configured on one of the router interfaces or switch interfaces. If the network device has several interfaces that can reach the RADIUS server, assign an IP address that the device can use for all its communication with the RADIUS server. Doing this sets a fixed address as the source address for locally generated IP packets.

5. Configure the shared secret password that the network device uses to authenticate with the RADIUS accounting server.

The configured password must match the password that is configured on the RADIUS server. If the password contains spaces, enclose it in quotation marks. The device stores the password as an encrypted value in the configuration database.

```
[edit system accounting destination radius server server-address]
user@host# set secret password
```

For example:

```
[edit system accounting destination radius server 192.168.17.28]
user@host# set secret Radiussecret1
```

6. (Optional) If necessary, specify to which RADIUS accounting server port to send accounting packets, if different from the default (1813).

```
[edit system accounting destination radius server server-address]
user@host# set accounting-port port-number
```


NOTE: If you enable RADIUS accounting at the [edit access profile *profile-name* accounting-order] hierarchy level, accounting is triggered on the default port of 1813 even if you do not specify a value for the accounting-port statement.

7. (Optional) Configure the number of times that the device attempts to contact a RADIUS accounting server and the amount of time that the device waits to receive a response from a server.

By default, the device attempts to contact the server three times and waits three seconds. You can configure the retry value from 1 through 100 times and the timeout value from 1 through 1000 seconds.

```
[edit system accounting destination radius server server-address]
user@host# set retry number
user@host# set timeout seconds
```

For example, to contact a server 2 times and wait 10 seconds for a response:

```
[edit system accounting destination radius server 192.168.17.28]
user@host# set retry 2
user@host# set timeout 10
```

8. (Optional) To route RADIUS accounting packets through the non-default management instance instead of the default routing instance, configure the routing-instance `mgmt_junos` statement.

```
[edit system accounting destination radius server server-address]
user@host# set routing-instance mgmt_junos
```

9. (Optional) Configure the enhanced-accounting statement at the [edit system radius-options] hierarchy level to include additional accounting attributes, including access method, remote port, and access privileges, for user login events.

```
[edit system radius-options]
user@host# set enhanced-accounting
```

NOTE: To limit the number of attribute values to audit, configure the enhanced-avs-max *<number>* statement at the [edit system accounting] hierarchy level.

The following example configures three servers (10.5.5.5, 10.6.6.6, and 10.7.7.7) for RADIUS accounting:

```
system {
  accounting {
    events [ login change-log interactive-commands ];
    destination {
      radius {
        server {
          10.5.5.5 {
            accounting-port 3333;
            secret $ABC123;
            source-address 10.1.1.1;
            retry 3;
            timeout 3;
          }
          10.6.6.6 secret $ABC123;
          10.7.7.7 secret $ABC123;
        }
      }
    }
  }
}
```

TACACS+ Authentication

IN THIS SECTION

- [Configure TACACS+ Authentication | 162](#)
- [Configure Periodic Refresh of the TACACS+ Authorization Profile | 169](#)
- [Example: Configure a TACACS+ Server for System Authentication | 170](#)
- [Juniper Networks Vendor-Specific TACACS+ Attributes | 174](#)
- [Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands | 176](#)
- [Configuring TACACS+ System Accounting | 181](#)

Junos OS Evolved supports TACACS+ for central authentication of users on network devices. To use TACACS+ authentication on the device, you (the network administrator) must configure information about one or more TACACS+ servers on the network. You can also configure TACACS+ accounting on the device to collect statistical data about the users logging in to or out of a LAN and send the data to a TACACS+ accounting server.

Configure TACACS+ Authentication

IN THIS SECTION

- [Configure TACACS+ Server Details | 162](#)
- [Configure TACACS+ to Use the Management Instance | 167](#)
- [Configure the Same Authentication Service for Multiple TACACS+ Servers | 167](#)
- [Configure Juniper Networks Vendor-Specific TACACS+ Attributes | 168](#)

TACACS+ authentication is a method of authenticating users who attempt to access a network device.

To configure TACACS+, perform the following tasks:

Configure TACACS+ Server Details

To use TACACS+ authentication on the device, configure information about one or more TACACS+ servers on the network by including one `tacplus-server` statement at the `[edit system]` hierarchy level for each TACACS+ server. The device queries the TACACS+ servers in the order in which they are configured. If the primary server (the first one configured) is unavailable, the device attempts to contact each server in the list until it receives a response.

The network device can map TACACS+-authenticated users to a locally defined user account or user template account, which determines authorization. By default, Junos OS Evolved assigns TACACS+-authenticated users to the user template account `remote`, if configured, when:

- The authenticated user does not have a user account configured on the local device.
- The TACACS+ server either does not assign the user to a local user template, or the template that the server assigns is not configured on the local device.

The TACACS+ server can assign an authenticated user to a different user template to grant different administrative permissions to that user. The user retains the same login name in the CLI but inherits the login class, access privileges, and effective user ID from the assigned template. If the TACACS+-

authenticated user does not map to any locally defined user account or user template, and the `remote` template is not configured, then authentication fails.

NOTE: The `remote` username is a special case in Junos OS Evolved and must always be lowercase. It acts as a template for users who are authenticated by a remote server but do not have a locally configured user account on the device. Junos OS Evolved applies the permissions of the `remote` template to those authenticated users without a locally defined account. All users mapped to the `remote` template are in the same login class.

Because remote authentication is configured on multiple devices, it is commonly configured inside of a configuration group. The steps shown here are in a configuration group called `global`. Using a configuration group is optional.

To configure authentication by a TACACS+ server:

1. Configure the IPv4 address or IPv6 address of the TACACS+ authentication server.

```
[edit groups global system tacplus-server]
user@host# set server-address
```

For example:

```
[edit groups global system tacplus-server]
user@host# set 192.168.17.28
```

```
[edit groups global system tacplus-server]
user@host# set 2001:db8:0:f101::8
```

2. (Optional) Configure the packet source address for requests sent to the TACACS+ server.

```
[edit groups global system tacplus-server server-address]
user@host# set source-address source-address
```

For example:

```
[edit groups global system tacplus-server 192.168.17.28]
user@host# set source-address 192.168.17.1
```

```
[edit groups global system tacplus-server 2001:db8:0:f101::8]
user@host# set source-address 2001:db8:0:f101::1
```

The source address is a valid IPv4 address or IPv6 address configured on one of the router interfaces or switch interfaces. If the network device has several interfaces that can reach the TACACS+ server, assign an IP address that the device can use for all its communication with the TACACS+ server. Doing this sets a fixed address as the source address for locally generated IP packets.

3. Configure the shared secret password that the network device uses to authenticate with the TACACS+ server.

The configured password must match the password that is configured on the TACACS+ server. If the password contains spaces, enclose it in quotation marks. The device stores the password as an encrypted value in the configuration database.

```
[edit groups global system tacplus-server server-address]
user@host# set secret password
```

For example:

```
[edit groups global system tacplus-server 192.168.17.28]
user@host# set secret Tacplussecret1
```

4. (Optional) Specify the port on which to contact the TACACS+ server, if different from the default port (49).

```
[edit groups global system tacplus-server server-address]
user@host# set port port-number
```

For example:

```
[edit groups global system tacplus-server 192.168.17.28]
user@host# set port 50049
```

5. (Optional) Configure the amount of time that the device waits to receive a response from the TACACS+ server.

By default, the device waits 10 seconds. You can configure the `timeout` value from 1 through 90 seconds.

```
[edit groups global system tacplus-server server-address]
user@host# set timeout seconds
```

For example, to wait 15 seconds for a response from the server:

```
[edit groups global system tacplus-server 192.168.17.28]
user@host# set timeout 15
```

6. (Optional) Configure the device to maintain one open TCP connection to the server for multiple requests rather than opening a separate connection for each connection attempt.

```
[edit groups global system tacplus-server 192.168.17.28]
user@host# set single-connection
```

NOTE: Early versions of the TACACS+ server do not support the `single-connection` option. If you specify this option and the server does not support it, the device will be unable to communicate with that TACACS+ server.

7. (Optional) To route TACACS+ packets through a specific routing instance, configure the `routing-instance` statement and specify a valid routing instance.

By default, Junos OS Evolved routes authentication, authorization, and accounting packets for TACACS+ through the default routing instance.

```
[edit groups global system tacplus-server server-address]
user@host# set routing-instance routing-instance
```

8. Specify the authentication order, and include the `tacplus` option.

```
[edit groups global system]
user@host# set authentication-order [ authentication-methods ]
```

In the following example, whenever a user attempts to log in, Junos OS Evolved first queries the TACACS+ server for authentication. If that fails, it queries the RADIUS server. If that fails, it attempts authentication with locally configured user accounts.

```
[edit groups global system]
user@host# set authentication-order [ tacplus radius password ]
```

9. Assign a login class to TACACS+-authenticated users who do not have a locally defined user account. You configure a user template account in the same way as a local user account, except that you do not configure a local authentication password because the TACACS+ server authenticates the user.

- To use the same permissions for all TACACS+-authenticated users, configure the `remote user` template.

```
[edit groups global system login]
user@host# set user remote class class
```

For example:

```
[edit groups global system login]
user@host# set user remote class super-user
```

- To use different login classes for different TACACS+-authenticated users, granting them different permissions:

- a. Create multiple user templates in the Junos OS Evolved configuration. For example:

```
[edit groups global system login]
user@host# set user RO class read-only
user@host# set user OP class operator
user@host# set user SU class super-user
user@host# set user remote full-name "default remote access user template"
user@host# set user remote class read-only
```

- b. Configure the TACACS+ server to map the authenticated user to the appropriate user template.

For example, set the `local-user-name` Juniper vendor-specific attribute (VSA) to the name of a user template configured on the device, which in the previous example is RO, OP, or SU.

Authentication fails if the device cannot assign a user to a local user account or user template and the remote user template is not configured.

Configure TACACS+ to Use the Management Instance

By default, Junos OS Evolved routes authentication, authorization, and accounting packets for TACACS+ through the default routing instance. You can also route TACACS+ packets through a management interface in a non-default VRF instance.

To route TACACS+ packets through the `mgmt_junos` management instance:

1. Enable the `mgmt_junos` management instance.

```
[edit system]
user@host# set management-instance
```

2. Configure the routing-instance `mgmt_junos` statement for the TACACS+ authentication server and the TACACS+ accounting server, if configured.

```
[edit system]
user@host# set tacplus-server server-address routing-instance mgmt_junos
user@host# set accounting destination tacplus server server-address routing-instance
mgmt_junos
```

Configure the Same Authentication Service for Multiple TACACS+ Servers

You can configure the same authentication service for multiple TACACS+ servers by including statements at the `[edit system tacplus-server]` and `[edit system tacplus-options]` hierarchy levels.

To assign the same authentication service to multiple TACACS+ servers:

1. Configure the TACACS+ servers as described in ["Configure TACACS+ Authentication" on page 162](#).
2. Configure the `service-name` statement at the `[edit system tacplus-options]` hierarchy level.

service-name is the name of the authentication service, which by default is `junos-exec`.

```
[edit system tacplus-options]
user@host set service-name service-name
```


For example:

```
[edit system tacplus-options]
service-name bob;
```

The following example shows how to configure the same authentication service for multiple TACACS+ servers:

```
[edit system]
tacplus-server {
  10.2.2.2 secret "$ABC123"; ## SECRET-DATA
  10.3.3.3 secret "$ABC123"; ## SECRET-DATA
}
tacplus-options {
  service-name bob;
}
```

Configure Juniper Networks Vendor-Specific TACACS+ Attributes

Junos OS Evolved can map TACACS+-authenticated users to a locally defined user account or user template account, which determines authorization. You can also optionally configure a user's access privileges by defining Juniper Networks vendor-specific TACACS+ attributes on the TACACS+ server. You define the attributes in the TACACS+ server configuration file on a per-user basis. The network device retrieves these attributes through an authorization request of the TACACS+ server after authenticating a user.

To specify these attributes, include a service statement of the following form in the TACACS+ server configuration file:

```
service = junos-exec {
  local-user-name = <username-local-to-router>
  allow-commands = "<allow-commands-regex>"
  allow-configuration-regexps = "<allow-configuration-regex>"
  deny-commands = "<deny-commands-regex>"
  deny-configuration-regexps = "<deny-configuration-regex>"
}
```

You can define the service statement in a user statement or a group statement.

Configure Periodic Refresh of the TACACS+ Authorization Profile

When you configure a device running Junos OS Evolved to use a TACACS+ server for authentication, the device prompts users for login information, which is verified by the TACACS+ server. After a user is successfully authenticated, the network device sends an authorization request to the TACACS+ server to obtain the authorization profile for the user. Authorization profiles specify the access permissions for authenticated users or devices.

The TACACS+ server sends the authorization profile as part of an authorization REPLY message. The remote user configured on the TACACS+ server is mapped to a local user or user template configured on the device running Junos OS Evolved. Junos OS Evolved combines the user's remote authorization profile and locally configured authorization profile, the latter of which is configured at the `[edit system login class]` hierarchy level.

By default, the exchange of authorization request and reply messages occurs only once, after successful authentication. You can configure the devices so that Junos OS Evolved periodically fetches the remote authorization profile from the TACACS+ server and refreshes the locally stored authorization profile. This periodic refresh ensures that the local device reflects any change in the authorization parameters without requiring that the user restart the authentication process.

To enable periodic refresh of the authorization profile, you must set the time interval at which the local device checks the authorization profile configured remotely on the TACACS+ server. If the remote authorization profile changes, the device fetches the authorization profile from the TACACS+ server and the authorization profile configured under the login class hierarchy. The device refreshes the authorization profile stored locally by combining the remote and locally configured authorization profiles.

You can configure the refresh time interval locally on the device running Junos OS Evolved or directly on the TACACS+ server. The time interval can range from 15 through 1440 minutes.

- To configure periodic refresh of the authorization profile on the local device, include the `authorization-time-interval` statement at the `[edit system tacplus-options]` hierarchy level, as follows:

```
[edit system tacplus-options]
user@host# set authorization-time-interval minutes
```

- To configure periodic refresh on the TACACS+ server, add the `refresh-time-interval` parameter in the authorization profile using the following syntax:

```
refresh-time-interval=minutes
```

Use the following guidelines to determine which time interval configuration takes precedence:

- If the refresh time interval is configured only on the TACACS+ server or only on the device running Junos OS Evolved, then the configured value takes effect.
- If the refresh time interval is configured on both the TACACS+ server and the device running Junos OS Evolved, the value configured on the TACACS+ server takes precedence.
- If no refresh time interval is configured on either the TACACS+ server or the device running Junos OS Evolved, then no periodic refresh occurs.
- If the refresh time interval configured on the TACACS+ server is out of range or invalid, the locally configured refresh time interval takes effect. If no refresh time interval is configured locally, then no periodic refresh occurs.

After the periodic refresh time interval is set, if the user changes the refresh interval before the authorization request is sent from the local device, the updated refresh interval takes effect after the next immediate periodic refresh.

Example: Configure a TACACS+ Server for System Authentication

IN THIS SECTION

- [Requirements | 170](#)
- [Overview | 171](#)
- [Configuration | 171](#)
- [Verification | 173](#)

This example configures system authentication through a TACACS+ server.

Requirements

Before you begin:

- Perform the initial device configuration. See the Getting Started Guide for your device.
- Set up at least one TACACS+ server on your network.

Overview

In this example, you add a new TACACS+ server with an IP address of 172.16.98.1. You specify the shared secret password of the TACACS+ server as Tacacssecret1. The device stores the secret in the configuration database as an encrypted value. Finally, you specify the source address that the device uses in TACACS+ server requests. In most cases, you can use the loopback address of the device, which in this example is 10.0.0.1.

You can configure support for multiple user authentication methods, such as local password authentication, TACACS+, and RADIUS, on the network device. When you configure multiple authentication methods, you can prioritize the order in which the device tries the different methods. In this example, you configure the device to use TACACS+ authentication services first and, if that fails, to then attempt local password authentication.

A TACACS+-authenticated user must map to a local user account or a local user template account on the network device, which determines authorization. By default, if a TACACS+-authenticated user does not map to a local user account or a specific user template, the user is assigned to the remote user template, if configured. This example configures the remote user template.

Configuration

IN THIS SECTION

- [Procedure | 171](#)

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` in configuration mode.

```
set system tacplus-server 172.16.98.1
set system tacplus-server 172.16.98.1 secret Tacacssecret1
set system tacplus-server 172.16.98.1 source-address 10.0.0.1
set system authentication-order [tacplus password]
set system login user remote class operator
```

Step-by-Step Procedure

To configure a TACACS+ server for system authentication:

1. Add a new TACACS+ server and set its IP address.

```
[edit system]
user@host# set tacplus-server 172.16.98.1
```

2. Specify the shared secret (password) of the TACACS+ server.

```
[edit system]
user@host# set tacplus-server 172.16.98.1 secret Tacacssecret1
```

3. Specify the device's loopback address as the source address.

```
[edit system]
user@host# set tacplus-server 172.16.98.1 source-address 10.0.0.1
```

4. Specify the device's order of authentication, and include the tacplus option.

```
[edit system]
user@host# set authentication-order [tacplus password]
```

5. Configure the remote user template and its login class.

```
[edit system]
user@host# set login user remote class operator
```

Results

In configuration mode, confirm your configuration by entering the `show system` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

The following output includes only those portions of the configuration hierarchy that are relevant to this example:

```
[edit]
user@host# show system
login {
  user remote {
    class operator;
  }
}
authentication-order [ tacplus password ];
tacplus-server {
  172.16.98.1 {
    secret "$9$ABC123"; ## SECRET-DATA
    source-address 10.0.0.1;
  }
}
```

After you configure the device, enter `commit` in configuration mode.

Verification

IN THIS SECTION

- [Verify the TACACS+ Server Configuration | 173](#)

Confirm that the configuration is working properly.

Verify the TACACS+ Server Configuration

Purpose

Verify that the TACACS+ server authenticates users.

Action

Log in to the network device, and verify that the login is successful. To verify that the device uses the TACACS+ server for authentication, you can attempt to log in with an account that does not define a local authentication password in the configuration.

Juniper Networks Vendor-Specific TACACS+ Attributes

Junos OS Evolved supports configuring Juniper Networks TACACS+ vendor-specific attributes (VSAs) on the TACACS+ server. [Table 11 on page 174](#) lists the supported Juniper Networks VSAs.

Some of the attributes accept extended regular expressions, as defined in POSIX 1003.2. If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. For more information, see:

- ["Regular Expressions to Allow and Deny Operational Mode Commands, Configuration Statements, and Hierarchies" on page 51](#)
- ["Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands" on page 176](#)

Table 11: Juniper Networks Vendor-Specific TACACS+ Attributes

Name	Description	Length	String
local-user-name	Indicates the name of the user template assigned to this user when the user logs in to a device.	≥3	One or more octets containing printable ASCII characters.
allow-commands	Contains an extended regular expression that enables the user to run commands in addition to those commands authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
allow-commands-regexps	Contains an extended regular expression that enables the user to run commands in addition to those commands authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
allow-configuration	Contains an extended regular expression that enables the user to view and modify configuration statements in addition to those statements authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.

Table 11: Juniper Networks Vendor-Specific TACACS+ Attributes (Continued)

Name	Description	Length	String
allow-configuration-regexps	Contains an extended regular expression that enables the user to view and modify configuration statements in addition to those statements authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
deny-commands	Contains an extended regular expression that denies the user permission to run commands authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
deny-commands-regexps	Contains an extended regular expression that denies the user permission to run commands authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
deny-configuration	Contains an extended regular expression that denies the user permission to view or modify configuration statements authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.
deny-configuration-regexps	Contains an extended regular expression that denies the user permission to view or modify configuration statements authorized by the user's login class permission bits.	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression.

Table 11: Juniper Networks Vendor-Specific TACACS+ Attributes (Continued)

Name	Description	Length	String
user-permissions	<p>Contains information the server uses to specify user permissions.</p> <p>NOTE: When the TACACS+ server defines the user-permissions attribute to grant the maintenance permission or all permission to a user, the user's list of group memberships does not automatically include the UNIX wheel group. Some operations such as running the su root command from a local shell require wheel group membership permissions. However, when the network device defines a local user account with the permissions maintenance or all, the user is automatically granted membership to the UNIX wheel group. Therefore, we recommend that you create a user template account with the required permissions and associate individual user accounts with the user template account.</p>	≥3	<p>One or more octets containing printable ASCII characters.</p> <p>See "Access Privilege Levels Overview" on page 41.</p>
authentication-type	Indicates the authentication method (local database or TACACS+ server) used to authenticate a user. If the user is authenticated using a local database, the attribute value shows 'local'. If the user is authenticated using a TACACS+ server, the attribute value shows 'remote'.	≥5	One or more octets containing printable ASCII characters.
session-port	Indicates the source port number of the established session.	size of integer	Integer

Use Regular Expressions on a RADIUS or TACACS+ Server to Allow or Deny Commands

Junos OS Evolved can map RADIUS- and TACACS+-authenticated users to a locally defined user account or user template account, which defines the user's access privileges. You can also optionally

configure a user's access privileges by defining Juniper Networks RADIUS and TACACS+ vendor-specific attributes (VSAs) on the respective authentication server.

A user's login class defines the set of permissions that determines which operational mode and configuration mode commands a user is authorized to execute and which areas of the configuration a user can view and modify. A login class can also define regular expressions that allow or deny a user the ability to execute certain commands or view and modify certain areas of the configuration, in addition to what the permission flags authorize. A login class can include the following statements to define user authorization:

- `permissions`
- `allow-commands`
- `allow-commands-regexps`
- `allow-configuration`
- `allow-configuration-regexps`
- `deny-commands`
- `deny-commands-regexps`
- `deny-configuration`
- `deny-configuration-regexps`

Similarly, a RADIUS or TACACS+ server configuration can use Juniper Networks VSAs to define specific permissions or regular expressions that determine a user's access privileges. For the list of supported RADIUS and TACACS+ VSAs, see the following:

- ["Juniper Networks Vendor-Specific RADIUS and LDAP Attributes" on page 147](#)
- ["Juniper Networks Vendor-Specific TACACS+ Attributes" on page 174](#)

You can define user permissions on the RADIUS or TACACS+ server as a list of space-separated values.

- A RADIUS server uses the following attribute and syntax:

```
Juniper-User-Permissions += "flag1 flag2 flag3",
```

For example:

```
Juniper-User-Permissions += "interface interface-control configure",
```

- A TACACS+ server uses the following attribute and syntax:

```
user-permissions = "flag1 flag2 flag3"
```

For example:

```
user-permissions = "interface interface-control configure"
```

A RADIUS or TACACS+ server can also define Juniper Networks VSAs that use a single extended regular expression (as defined in POSIX 1003.2) to allow or deny a user the ability to execute certain commands or view and modify areas of the configuration. You enclose multiple commands or configuration hierarchies in parentheses and separate them using a pipe symbol. If the regular expression contains any spaces, operators, or wildcard characters, enclose it in quotation marks. When you configure authorization parameters both locally and remotely, the device merges the regular expressions received during TACACS+ or RADIUS authorization with any regular expressions defined on the local device.

- A RADIUS server uses the following attributes and syntax:

```
Juniper-Allow-Commands += "(cmd1)|(cmd2)|(cmdn)",
Juniper-Deny-Commands += "(cmd1)|(cmd2)|(cmdn)",
Juniper-Allow-Configuration += "(config1)|(config2)|(confign)",
Juniper-Deny-Configuration += "(config1)|(config2)|(confign)",
```

For example:

```
Juniper-Allow-Commands += "(test)|(ping)|(quit)",
Juniper-Deny-Commands += "(request)|(restart)",
Juniper-Allow-Configuration += "(groups re0)|(system radius-server)",
Juniper-Deny-Configuration += "(system radius-options)|(system accounting)",
```

- A TACACS+ server uses the following attributes and syntax:

```
allow-commands = "(cmd1)|(cmd2)|(cmdn)"
deny-commands = "(cmd1)|(cmd2)|(cmdn)"
allow-configuration = "(config1)|(config2)|(confign)"
deny-configuration = "(config1)|(config2)|(confign)"
```

For example:

```
allow-commands = "(test)|(ping)|(quit)"
deny-commands = "(request)|(restart)"
allow-configuration = "(groups re0)|(system tacplus-server)"
deny-configuration = "(system tacplus-options)|(system accounting)"
```

RADIUS and TACACS+ servers also support configuring attributes that correspond to the same *-regexps statements that you can configure on the local device. The *-regexps TACACS+ attributes and the *-Regexps RADIUS attributes use the same regular expression syntax as the previous attributes, but they enable you to configure regular expressions with variables.

- A RADIUS server uses the following attributes and syntax:

```
Juniper-Allow-Configuration-Regexps += "(config1)|(config2)|(config)",
Juniper-Deny-Configuration-Regexps += "(config1)|(config2)|(config)",
```

- A TACACS+ server uses the following attributes and syntax:

```
allow-commands-regexps = "(cmd1)|(cmd2)|(cmdn)"
deny-commands-regexps = "(cmd1)|(cmd2)|(cmdn)"
allow-configuration-regexps = "(config1)|(config2)|(config)"
deny-configuration-regexps = "(config1)|(config2)|(config)"
```

For example, the TACACS+ server configuration might define the following attributes:

```
allow-commands-regexps = "(show cli .*)|(ping 10.1.1..*)"
deny-commands-regexps = "(configure .*)|(edit)|(commit)|(rollback .*)"
```

On a RADIUS or TACACS+ server, you can also define the attributes using a simplified syntax where you specify each individual expression on a separate line.

For a RADIUS server, specify the individual regular expressions using the following syntax:

```
Juniper-User-Permissions += "permission-flag1",
Juniper-User-Permissions += "permission-flag2",
Juniper-User-Permissions += "permission-flagn",
Juniper-Allow-Commands += "cmd1",
```

```

Juniper-Allow-Commands += "cmd2",
Juniper-Allow-Commands += "cmdn",
Juniper-Deny-Commands += "cmd1",
Juniper-Deny-Commands += "cmd2",
Juniper-Deny-Commands += "cmdn",
Juniper-Allow-Configuration += "config1",
Juniper-Allow-Configuration += "config2",
Juniper-Allow-Configuration += "confign",
Juniper-Deny-Configuration += "config1",
Juniper-Deny-Configuration += "config2",
Juniper-Deny-Configuration += "confign",

```

For a TACACS+ server, specify the individual regular expressions using the following syntax:

```

user-permissions1 = "permission-flag1"
user-permissions2 = "permission-flag2"
user-permissionsn = "permission-flagn"
allow-commands1 = "cmd1"
allow-commands2 = "cmd2"
allow-commandsn = "cmdn"
deny-commands1 = "cmd1"
deny-commands2 = "cmd2"
deny-commandsn = "cmdn"
allow-configuration1 = "config1"
allow-configuration2 = "config2"
allow-configurationn = "confign"
deny-configuration1 = "config1"
deny-configuration2 = "config2"
deny-configurationn = "confign"

```

NOTE:

- In the TACACS+ server syntax, numeric values 1 through *n* must be unique but need not be sequential. For example, the following syntax is valid:

```

allow-commands1="cmd1"
allow-commands3="cmd3"
allow-commands2="cmd2"

```

```
deny-commands3="cmd3"
deny-commands2="cmd2"
deny-commands1="cmd1"
```

- The RADIUS or TACACS+ server imposes a limit on the number of individual regular expression lines.
- When you issue the `show cli authorization` command, the command output displays the regular expression in a single line, even if you specify each individual expression on a separate line.

Users can verify their class, permissions, and command and configuration authorization by issuing the `show cli authorization` operational mode command.

```
user@host> show cli authorization
```

NOTE: When you configure the authorization parameters both locally on the network device and remotely on the RADIUS or TACACS+ server, the device merges the regular expressions received during TACACS+ or RADIUS authorization with any locally configured regular expressions. If the final expression contains a syntax error, the overall result is an invalid regular expression.

Configuring TACACS+ System Accounting

IN THIS SECTION

- [Configure TACACS+ Server Accounting | 182](#)

You can configure TACACS+ accounting on a device to collect statistical data about users logging in to or out of a LAN and send the data to a TACACS+ accounting server. The statistical data can be used for general network monitoring, analyzing and tracking usage patterns, or billing a user based on the duration of the session or type of services accessed.

To configure TACACS+ accounting, specify:

- One or more TACACS+ accounting servers to receive the statistical data from the device
- The type of accounting data to collect

You can use the same server for both TACACS+ accounting and authentication, or you can use separate servers. You can specify a list of TACACS+ accounting servers. The device queries the servers in the order in which they are configured. If the primary server (the first one configured) is unavailable, the device attempts to contact each server in the list until it receives a response.

When you enable TACACS+ accounting, Juniper Networks devices, acting as TACACS+ clients, can notify the TACACS+ server about user activities such as software logins, configuration changes, and interactive commands.

Configure TACACS+ Server Accounting

To configure TACACS+ server accounting:

1. Configure the events to audit.

```
[edit system accounting]
user@host# set events [ events ]
```

For example:

```
[edit system accounting]
user@host# set events [ login change-log interactive-commands ]
```

events can include one or more of the following:

- login—Audit logins
 - change-log—Audit configuration changes
 - interactive-commands—Audit interactive commands (any command-line input)
2. Enable TACACS+ accounting.

```
[edit]
user@host# set system accounting destination tacplus
```

3. Configure the address for one or more TACACS+ accounting servers.

```
[edit system accounting destination tacplus]
user@host# set server server-address
```

For example:

```
[edit system accounting destination tacplus]
user@host# set server 192.168.17.28
```

NOTE: If you do not configure any TACACS+ servers at the [edit system accounting destination tacplus] hierarchy level, the device uses the TACACS+ servers configured at the [edit system tacplus-server] hierarchy level.

4. (Optional) Configure the source address for TACACS+ accounting requests.

```
[edit system accounting destination tacplus server server-address]
user@host# set source-address source-address
```

For example:

```
[edit system accounting destination tacplus server 192.168.17.28]
user@host# set source-address 192.168.17.1
```

The source address is a valid IPv4 address or IPv6 address configured on one of the router interfaces or switch interfaces. If the network device has several interfaces that can reach the TACACS+ server, assign an IP address that the device can use for all its communication with the TACACS+ server. Doing this sets a fixed address as the source address for locally generated IP packets.

5. Configure the shared secret password that the network device uses to authenticate with the TACACS+ accounting server.

The configured password must match the password that is configured on the TACACS+ server. If the password contains spaces, enclose it in quotation marks. The device stores the password as an encrypted value in the configuration database.

```
[edit system accounting destination tacplus server server-address]
user@host# set secret password
```

For example:

```
[edit system accounting destination tacplus server 192.168.17.28]
user@host# set secret Tacplussecret1
```

6. (Optional) If necessary, specify to which TACACS+ accounting server port to send accounting packets, if different from the default (49).

```
[edit system accounting destination tacplus server server-address]
user@host# set accounting-port port-number
```

7. (Optional) Configure the amount of time that the device waits to receive a response from the TACACS+ accounting server.

By default, the device waits three seconds. You can configure the `timeout` value from 1 through 90 seconds.

```
[edit system accounting destination tacplus server server-address]
user@host# set timeout seconds
```

For example, to wait 15 seconds for a response from the server:

```
[edit system accounting destination tacplus server 192.168.17.28]
user@host# set timeout 15
```

8. (Optional) Configure the device to maintain one open TCP connection to the server for multiple requests rather than opening a separate connection for each connection attempt.

```
[edit system accounting destination tacplus server server-address]
user@host# set single-connection
```

NOTE: Early versions of the TACACS+ server do not support the `single-connection` option. If you specify this option and the server does not support it, the device will be unable to communicate with that TACACS+ server.

9. (Optional) To route TACACS+ accounting packets through the non-default management instance or another routing instance instead of the default routing instance, configure the `routing-instance` statement and specify the routing instance.

```
[edit system accounting destination tacplus server server-address]
user@host# set routing-instance routing-instance
```

For example:

```
[edit system accounting destination tacplus server 192.168.17.28]
user@host# set routing-instance mgmt_junos
```

10. To ensure that start and stop requests for login events are correctly logged in the TACACS+ server Accounting log file instead of the Administration log file, include either the `no-cmd-attribute-value` statement or the `exclude-cmd-attribute` statement at the `[edit system tacplus-options]` hierarchy level.

```
[edit system tacplus-options]
user@host# set (no-cmd-attribute-value | exclude-cmd-attribute)
```

NOTE: Both statements support the correct logging of accounting requests in the Accounting file instead of the Administration file. If you configure the `no-cmd-attribute-value` statement, the value of the `cmd` attribute is set to a null string in the start and stop requests. If you configure the `exclude-cmd-attribute` statement, the `cmd` attribute is totally excluded from the start and stop requests.

Authentication for Routing Protocols

IN THIS SECTION

- [Authentication Methods for Routing Protocols | 186](#)
- [Example: Configure the Authentication Key for BGP and IS-IS Routing Protocols | 187](#)
- [Configure the Authentication Key Update Mechanism for Routing Protocols | 190](#)

You can configure an authentication method and password for routing protocol messages for many routing protocols including BGP, IS-IS, OSPF, RIP, and RSVP. To prevent the exchange of unauthenticated or forged packets, routers must ensure that they form routing protocol relationships (peering or neighboring relationships) to trusted peers. One way of doing this is by authenticating routing protocol messages. Neighboring routers use the password to verify the authenticity of packets sent by the protocol from the router or from a router interface.

This topic provides a high-level overview and some basic examples for authenticating routing protocols. For detailed information about configuring authentication for a specific routing protocol, see the user guide for that protocol.

Authentication Methods for Routing Protocols

Some routing protocols—BGP, IS-IS, OSPF, RIP, and RSVP—enable you to configure an authentication method and password. Neighboring routers use the password to verify the authenticity of packets that the protocol sends from the router or from a router interface. The following authentication methods are supported:

- Simple authentication (IS-IS, OSPF, and RIP)—Uses a simple text password. The receiving router uses an authentication key (password) to verify the packet. Because the password is included in the transmitted packet, this method of authentication is relatively insecure. We recommend that you *avoid* using this authentication method.
- MD5 and HMAC-MD5 (BGP, IS-IS, OSPF, RIP, and RSVP)—MD5 creates an encoded checksum that is included in the transmitted packet. HMAC-MD5, which combines HMAC authentication with MD5, adds the use of an iterated cryptographic hash function. With both types of authentication, the receiving router uses an authentication key (password) to verify the packet. HMAC-MD5 authentication is defined in RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*.

In general, authentication passwords are text strings consisting of some maximum number of letters and digits. Passwords can include any ASCII characters. If you include spaces in a password, enclose all characters in quotation marks (" ").

Example: Configure the Authentication Key for BGP and IS-IS Routing Protocols

IN THIS SECTION

- [Configure BGP | 187](#)
- [Configure IS-IS | 188](#)

The main task of a router is to use its routing and forwarding tables to forward user traffic to its intended destination. Attackers can send forged routing protocol packets to a router with the intent of changing or corrupting the contents of its routing table or other databases, which in turn can degrade the functionality of the router and the network. To prevent such attacks, routers must ensure that they form routing protocol relationships (peering or neighboring relationships) to trusted peers. One way of doing this is by authenticating routing protocol messages. We strongly recommend using authentication when configuring routing protocols.

Junos OS Evolved supports HMAC-MD5 authentication for BGP, IS-IS, OSPF, RIP, and RSVP. HMAC-MD5 uses a secret key combined with the data being transmitted to compute a hash. The computed hash is transmitted along with the data. The receiver uses the matching key to recompute and validate the message hash. If an attacker has forged or modified the message, the hash will not match, and the data is discarded.

In the following examples, we configure BGP as the exterior gateway protocol (EGP) and IS-IS as the interior gateway protocol (IGP). If you use OSPF, configure it similarly to the IS-IS configuration shown.

Configure BGP

The following example shows the configuration of a single authentication key for the different BGP peer groups. You can also configure BGP authentication at the neighbor or routing instance levels, or for all BGP sessions. As with any security configuration, there is a trade-off between the degree of granularity (and to some extent, the degree of security) and the amount of management necessary to maintain the system.

This example also configures a number of tracing options for routing protocol events and errors, which can be good indicators of attacks against routing protocols. These events include protocol authentication failures, which might point to an attacker. The attacker may be sending spoofed or otherwise malformed routing packets to the router in an attempt to elicit a particular behavior.

```
[edit]
protocols {
  bgp {
    group ibgp {
      type internal;
      traceoptions {
        file bgp-trace size 1m files 10;
        flag state;
        flag general;
      }
      local-address 10.10.5.1;
      log-updown;
      neighbor 10.2.1.1;
      authentication-key "$9$aH1j8gqQ1gjyjjghgjgiiii";
    }
    group ebgp {
      type external;
      traceoptions {
        file ebgp-trace size 10m files 10;
        flag state;
        flag general;
      }
      local-address 10.10.5.1;
      log-updown;
      peer-as 2;
      neighbor 10.2.1.2;
      authentication-key "$9$aH1j8gqQ1gjyjjghgjgiiii";
    }
  }
}
```

Configure IS-IS

Although Junos OS Evolved supports authentication for all IGPs, some IGPs are inherently more secure than others. Most service providers use OSPF or IS-IS to allow fast internal convergence and scalability and to use traffic engineering capabilities with MPLS. Because IS-IS does not operate at the network

layer, it is more difficult to spoof than OSPF. OSPF is encapsulated in IP and is therefore subject to remote spoofing and denial of service (DoS) attacks.

The following example configures authentication for IS-IS. It also configures a number of tracing options for routing protocol events and errors, which can be good indicators of attacks against routing protocols. These events include protocol authentication failures, which might point to an attacker. The attacker may be sending spoofed or otherwise malformed routing packets to the router in an attempt to elicit a particular behavior.

```
[edit]
protocols {
  isis {
    level 1 {
      authentication-key "$9$aH1j8gqQ1gjyjgjhgjgiiii"; # SECRET-DATA
      authentication-type md5;
    }
    interface at-0/0/0.131 {
      lsp-interval 50;
      level 2 disable;
      level 1 {
        metric 3;
        hello-interval 5;
        hold-time 60;
      }
    }
    interface lo0.0 {
      passive;
    }
    traceoptions {
      file isis-trace size 10m files 10;
      flag normal;
      flag error;
    }
  }
}
```

Configure the Authentication Key Update Mechanism for Routing Protocols

IN THIS SECTION

- [Configure Authentication Key Updates | 190](#)
- [Configure BGP and LDP for Authentication Key Updates | 191](#)

You can configure an authentication key update mechanism for the BGP, LDP, and IS-IS routing protocols. This mechanism enables you to update authentication keys without interrupting associated routing and signaling protocols such as OSPF and RSVP.

To configure this feature, include the `authentication-key-chains` statement at the `[edit security]` hierarchy level. To apply the key chain, you must configure the key chain identifier and the key chain algorithm at the appropriate hierarchy level for the protocol.

The following sections provide more information about configuring authentication key updates for routing protocols. For detailed information about configuring authentication key updates for a specific routing protocol, see the user guide for that protocol.

Configure Authentication Key Updates

To configure the authentication key update mechanism, include the `key-chain` statement at the `[edit security authentication-key-chains]` hierarchy level, and specify the `key` option to create a keychain consisting of several authentication keys.

```
[edit security authentication-key-chains]
key-chain key-chain-name {
  key key {
    algorithm (hmac-sha-1 | md5)
    options (basic | isis-enhanced)
    secret secret-data;
    start-time yyyy-mm-dd.hh:mm:ss;
  }
}
```

key-chain—Assign a name to the keychain mechanism. You reference this name at the appropriate hierarchy levels for the protocol to associate unique authentication **key-chain** attributes, as specified using the following options:

- **algorithm**—Authentication algorithm for IS-IS.
- **key**—Integer value that uniquely identifies each key within a keychain. The range is from 0 through 63.
- **options**—(IS-IS only) Protocol transmission encoding format for encoding the message authentication code in routing protocol packets.
- **secret**—Password in encrypted text or plain text format. Even if you enter the secret data in plain-text format, the secret always appears in encrypted format.
- **start-time**—Start time for authentication key transmission, specified in *UTC*. The start time must be unique within the keychain.

Configure BGP and LDP for Authentication Key Updates

To configure the authentication key update mechanism for the BGP and LDP routing protocols, include the **authentication-key-chain** statement within the `[edit protocols (bgp | ldp)]` hierarchy level. Including the **authentication-key-chain** statement associates each routing protocol with the `[edit security authentication-key-chains]` authentication keys. You must also configure the **authentication-algorithm** statement and specify the algorithm. For example:

```
[edit protocols]
bgp {
  group group-name {
    neighbor address {
      authentication-algorithm algorithm;
      authentication-key-chain key-chain-name;
    }
  }
}
ldp {
  session session-addr {
    authentication-algorithm algorithm;
    authentication-key-chain key-chain-name;
  }
}
```


NOTE: When configuring the authentication key update mechanism for BGP, you cannot commit the `0.0.0.0/allow` statement with authentication keys or keychains. If you try this action, the CLI issues a warning, and the commit fails.

5

CHAPTER

Remote Access Management

[Remote Access Overview](#) | 194

[Configuration Guidelines for Securing Console Port Access](#) | 212

Remote Access Overview

IN THIS SECTION

- [System Services Overview | 194](#)
- [Configure Telnet Service for Remote Access to a Router or Switch | 195](#)
- [Configure FTP Service for Remote Access to the Router or Switch | 195](#)
- [Configure Finger Service for Remote Access to the Router | 196](#)
- [Configure SSH Service for Remote Access to the Router or Switch | 196](#)
- [The telnet Command | 200](#)
- [The ssh Command | 201](#)
- [Configure SSH Host Keys for Secure Copying of Data | 202](#)
- [Configure the SSH Service to Support Legacy Cryptography | 205](#)
- [Configure Outbound SSH Service | 207](#)
- [Configure NETCONF-Over-SSH Connections on a Specified TCP Port | 211](#)

You (the network administrator) can access a router, switch, or security device remotely using services such as DHCP, Finger, FTP, rlogin, SSH, and Telnet services. This topic shows you how to configure remote access using Telnet, SSH, FTP, and Finger services.

System Services Overview

For security reasons, remote access to the router is disabled by default. You must configure the router explicitly so that users on remote systems can access it. Users can access the router from a remote system by means of the DHCP, finger, FTP, rlogin, SSH, and Telnet services. In addition, Junos XML protocol client applications can use Secure Sockets Layer (SSL) or the Junos XML protocol-specific clear-text service, among other services.

NOTE: To protect system resources, you can limit the number of simultaneous connections that a service accepts and the number of processes owned by a single user. If either limit is exceeded, connection attempts fail.

Configure Telnet Service for Remote Access to a Router or Switch

To configure the router or switch to accept Telnet as an access service, include the `telnet` statement at the `[edit system services]` hierarchy level:

```
[edit system services]
telnet {
    connection-limit limit;
    rate-limit limit;
}
```

By default, the router or switch supports a limited number of simultaneous Telnet sessions and connection attempts per minute.

Optionally, you can include either or both of the following statements to change the defaults:

- `connection-limit limit`—Maximum number of simultaneous connections per protocol (IPv4 and IPv6). The range is from 1 through 250. The default is 75. When you configure a connection limit, the limit is applicable to the number of telnet sessions per protocol (IPv4 and IPv6). For example, a connection limit of 10 allows 10 IPv6 telnet sessions and 10 IPv4 telnet sessions.
- `rate-limit limit`—Maximum number of connection attempts accepted per minute (from 1 through 250). The default is 150. When you configure a rate limit, the limit is applicable to the number of connection attempts per protocol (IPv4 and IPv6). For example, a rate limit of 10 allows 10 IPv6 telnet session connection attempts per minute and 10 IPv4 telnet session connection attempts per minute.

Configure FTP Service for Remote Access to the Router or Switch

To configure the device to accept FTP as an access service, include the `ftp` statement at the `[edit system services]` hierarchy level:

```
[edit system services]
ftp;
```

You can use passive FTP to access devices that accept only passive FTP services. All commands and statements that use FTP also accept passive FTP. Include the `ftp` statement at the `[edit system services]` hierarchy level to use either active FTP or passive FTP.

To start a passive FTP session, use `pasvftp` (instead of `ftp`) in the standard FTP format (**ftp:// *destination***). For example:

```
request system software add pasvftp://name.com/jinstall.tgz
```

Configure Finger Service for Remote Access to the Router

To configure the router to accept finger as an access service, include the `finger` statement at the `[edit system services]` hierarchy level:

```
[edit system services]
finger;
```

Configure SSH Service for Remote Access to the Router or Switch

IN THIS SECTION

- [Configure the Root Login Through SSH | 198](#)
- [Configure Incoming SFTP Connections | 198](#)
- [Configure the SSH Protocol Version | 199](#)
- [Configure the Client Alive Mechanism | 199](#)
- [Configure the SSH Fingerprint Hash Algorithm | 200](#)

To configure the router or switch to accept SSH as an access service, include the `ssh` statement at the `[edit system services]` hierarchy level:

```
[edit system services]
ssh {
  authentication-order [method 1 method2...];
  authorized-keys-command authorized-keys-command;
  authorized-keys-command-user authorized-keys-command-user;
```

```

ciphers [ cipher-1 cipher-2 cipher-3 ...];
client-alive-count-max number;
client-alive-interval seconds;
connection-limit limit;
fingerprint-hash (md5 | sha2-256);
hostkey-algorithm (algorithm | no-algorithm);
key-exchange [algorithm1 algorithm2...];
log-key-changes log-key-changes;
macs [algorithm1 algorithm2...];
max-pre-authentication-packets number;
max-sessions-per-connection number;
no-challenge-response;
no-password-authentication;
no-passwords;
no-public-keys;
no-tcp-forwarding;
port port-number;
protocol-version [v2];
rate-limit number;
rekey {
    data-limit bytes;
    time-limit minutes;
}
root-login (allow | deny | deny-password);
sftp-server;
tcp-forwarding;
}

```

By default, the router or switch supports a limited number of simultaneous SSH sessions and connection attempts per minute. Use the following statements to change the defaults:

- `connection-limit limit`—Maximum number of simultaneous connections per protocol (IPv4 and IPv6). The range is a value from 1 through 250. The default is 75. When you configure a connection limit, the limit is applicable to the number of SSH sessions per protocol (IPv4 and IPv6). For example, a connection limit of 10 allows 10 IPv6 SSH sessions and 10 IPv4 SSH sessions.
- `max-sessions-per-connection number`—Include this statement to specify the maximum number of SSH sessions allowed per single SSH connection. This allows you to limit the number of cloned sessions tunneled within a single SSH connection. The default value is 10.
- `rate-limit limit`—Maximum number of connection attempts accepted per minute (a value from 1 through 250). The default is 150. When you configure a rate limit, the limit is applicable to the number of connection attempts per protocol (IPv4 and IPv6). For example, a rate limit of 10 allows

10 IPv6 SSH session connection attempts per minute and 10 IPv4 SSH session connection attempts per minute.

- `data-limit`—Data limit before renegotiating session keys (bytes)
- `time-limit`—Time limit before renegotiating session keys (minutes)

By default, a user can create an SSH tunnel over a CLI session to a router running Junos OS via SSH. This type of tunnel can be used to forward TCP traffic, bypassing any firewall filters or access control lists. By bypassing firewall filters or access control lists, this type of tunnel allows access to resources beyond the router. Use the `no-tcp-forwarding` option to prevent a user from creating an SSH tunnel to a router via SSH.

For information about other configuration settings, see the following topics:

Configure the Root Login Through SSH

By default, users are allowed to log in to the router or switch as root through SSH when the authentication method does not require a password. To control user access through SSH, include the `root-login` statement at the `[edit system services ssh]` hierarchy level:

```
[edit system services ssh]
root-login (allow | deny | deny-password);
```

`allow`—Allows users to log in to the router or switch as root through SSH.

`deny`—Disables users from logging in to the router or switch as root through SSH.

`deny-password`—Allows users to log in to the router or switch as root through SSH when the authentication method (for example, RSA) does not require a password.

The default is `deny-password`.

Configure Incoming SFTP Connections

SSH File Transfer Protocol (SFTP) is a network protocol that provides file access, file transfer, and file management over any reliable data stream. Incoming SFTP connections are disabled by default. If desired, you can globally enable incoming SFTP connections by configuring the statement `sftp-server` at the `[edit system services ssh]` hierarchy level.

NOTE: Only the incoming SFTP connections are disabled by default. For example, given devices A and B, you cannot connect through SFTP from B to A by default. However, you can connect through SFTP from device B to device A if you configure `sftp-server` on device A.

The incoming SFTP connections are disabled by default. To enable incoming SFTP connections:

1. Configure the `sftp-server` statement at the `[edit system services ssh]` hierarchy level:

```
[edit system services ssh]
user@host# set sftp-server
```

2. Commit the configuration.

```
[edit system services ssh]
user@host# commit
```

The `sftp-server` statement is now active. Therefore, the incoming SFTP connections are enabled.

Configure the SSH Protocol Version

By default, only version 2 of the SSH protocol is enabled.

To configure the router or switch to use version 2 of the SSH protocol, include the `protocol-version` statement and specify `v2` at the `[edit system services ssh]` hierarchy level:

```
[edit system services ssh]
protocol-version [ v2 ];
```

Configure the Client Alive Mechanism

The client alive mechanism is valuable when the client or server depends on knowing when a connection has become inactive. It differs from the standard keepalive mechanism because the client alive messages are sent through the encrypted channel. The client alive mechanism is not enabled by default. To enable it, configure the `client-alive-count-max` and `client-alive-interval` statements. This option applies to SSH protocol version 2 only.

In the following example, unresponsive SSH clients will be disconnected after approximately 100 seconds (20 x 5):

```
[edit system services ssh]
client-alive-count-max 5;
client-alive-interval 20;
```


Configure the SSH Fingerprint Hash Algorithm

To configure the hash algorithm used by the SSH server when it displays key fingerprints, include the `fingerprint-hash` statement and specify `md5` or `sha2-256` at the `[edit system services ssh]` hierarchy level:

```
[edit system services ssh]
fingerprint-hash (md5 | sha2-256);
```

The telnet Command

You can use the CLI `telnet` command to open a Telnet session to a remote device:

```
user@host> telnet host <8bit> <inet> <inet6> <port port> <routing-instance routing-instance-name>
```

To exit the Telnet session and return to the Telnet command prompt, press `Ctrl-]`.

To exit the Telnet session and return to the CLI command prompt, enter `quit`.

Table 12: CLI telnet Command Options

Option	Description
<code>8bit</code>	Use an 8-bit data path.
<code>host</code>	Open a Telnet session to the specified hostname or IP address.
<code>inet</code>	Force the Telnet session to an IPv4 destination.
<code>inet6</code>	Force the Telnet session to an IPv6 destination.
<code>port port</code>	Specify the port number or service name on the host.
<code>routing-instance routing-instance-name</code>	Use the specified routing instance for the Telnet session.

The ssh Command

You can use the CLI `ssh` command to use the secure shell (SSH) program to open a connection to a remote device:

```
user@host> ssh host <bypass-routing> <inet> <inet6> <interface interface-name> <logical-system>
<routing-instance routing-instance-name> <source address> <v2>
```

Table 13 on page 201 describes the `ssh` command options.

Table 13: CLI ssh Command Options

Option	Description
<code>bypass-routing</code>	Bypass the routing tables and open an SSH connection only to hosts on directly attached interfaces. If the host is not on a directly attached interface, an error message is returned.
<code>host</code>	Open an SSH connection to the specified hostname or IP address.
<code>inet</code>	Force the SSH connection to an IPv4 destination.
<code>inet6</code>	Force the SSH connection to an IPv6 destination.
<code>interface source-interface</code>	Open an SSH connection to a host on the specified interface. If you do not include this option, all interfaces are used.
<code>routing-instance routing-instance-name</code>	Use the specified routing instance for the SSH connection.
<code>logical-system</code>	Use the specified logical system for the SSH connection.
<code>source address</code>	Use the specified source address for the SSH connection.
<code>v2</code>	Force SSH to use version 2 for the connection.

Configure SSH Host Keys for Secure Copying of Data

IN THIS SECTION

- [Configure SSH Known Hosts | 202](#)
- [Configure Support for SCP File Transfer | 203](#)
- [Update SSH Host Key Information | 204](#)

Secure Shell (*SSH*) uses *encryption* algorithms to generate a host, server, and session key system that ensures secure data transfer. You can configure SSH host keys to support secure copy (*SCP*) as an alternative to *FTP* for the background transfer of data such as configuration archives and event logs. To configure SSH support for SCP, you must complete the following tasks:

- Specify SSH known hosts by including hostnames and host key information in the Routing Engine configuration hierarchy.
- Set an SCP URL to specify the host from which to receive data. Setting this attribute automatically retrieves SSH host key information from the SCP server.
- Verify that the host key is authentic.
- Accept the secure connection. Accepting this connection automatically stores host key information in the local host key database. Storing host key information in the configuration hierarchy automates the secure handshake and allows background data transfer using SCP.

Tasks to configure SSH host keys for secure copying of data are:

Configure SSH Known Hosts

To configure SSH known hosts, include the `host` statement, and specify hostname and host key options for trusted servers at the `[edit security ssh-known-hosts]` hierarchy level:

```
[edit security ssh-known-hosts]
host corporate-archive-server {
    dsa-key key;
}
host archive-server-url {
    rsa-key key;
}
host server-with-ssh-version-1 {
```

```

    rsa1-key key;
}

```

Host keys are one of the following:

- dsa-key *key*—Base64 encoded Digital Signature Algorithm (DSA) key for SSH version 2.
- ecdsa-sha2-nistp256-key *key*—Base64 encoded ECDSA-SHA2-NIST256 key.
- ecdsa-sha2-nistp384-key *key*—Base64 encoded ECDSA-SHA2-NIST384 key.
- ecdsa-sha2-nistp521-key *key*—Base64 encoded ECDSA-SHA2-NIST521 key.
- ed25519-key *key*—Base64 encoded ED25519 key.
- rsa-key *key*—Base64 encoded public key algorithm that supports encryption and digital signatures for SSH version 1 and SSH version 2.
- rsa1-key *key*—Base64 encoded RSA public key algorithm, which supports encryption and digital signatures for SSH version 1.

Configure Support for SCP File Transfer

To configure a known host to support background SCP file transfers, include the `archive-sites` statement at the `[edit system archival configuration]` hierarchy level.

```

[edit system archival configuration]
archive-sites {
    scp://username<:password>@host<:port>/url-path;
}

```

NOTE: When specifying a URL in a Junos OS Evolved statement using an IPv6 host address, you must enclose the entire URL in quotation marks (" ") and enclose the IPv6 host address in brackets ([]). For example, "`scp://username<:password>@[host]<:port>/url-path";`

Setting the `archive-sites` statement to point to an SCP URL triggers automatic host key retrieval. At this point, Junos OS Evolved connects to the SCP host to fetch the SSH public key, displays the host key message digest or fingerprint as output to the console, and terminates the connection to the server.

```
user@host# set system archival configuration archive-sites "<scp-url-path>"
The authenticity of host <my-archive-server (<server-ip-address>)> can't be established. RSA key
fingerprint is <ascii-text key>. Are you sure you want to continue connecting (yes/no)?
```

To verify that the host key is authentic, compare this fingerprint with a fingerprint that you obtain from the same host using a trusted source. If the fingerprints are identical, accept the host key by entering **yes** at the prompt. The host key information is then stored in the Routing Engine configuration and supports background data transfers using SCP.

Update SSH Host Key Information

IN THIS SECTION

- [Retrieve Host Key Information Manually | 204](#)
- [Import Host Key Information from a File | 205](#)

Typically, SSH host key information is automatically retrieved when you set a URL attribute for SCP using the `archival configuration archive-sites` statement at the `[edit system]` hierarchy level. However, if you need to manually update the host key database, use one of the following methods.

Retrieve Host Key Information Manually

To manually retrieve SSH public host key information, configure the `fetch-from-server` option at the `[edit security ssh-known-hosts]` hierarchy level. You must specify the host from which to retrieve the SSH public key.

```
user@host# set security ssh-known-hosts fetch-from-server <hostname>
```

Import Host Key Information from a File

To manually import SSH host key information from a **known_hosts** file, include the `load-key-file` option at the `[edit security ssh-known-hosts]` hierarchy level. You must specify the path to the file from which to import host key information.

```
user@host# set security ssh-known-hosts load-key-file /var/tmp/known-hosts
```

Configure the SSH Service to Support Legacy Cryptography

The SSH server in Junos OS Evolved is based on OpenSSH 7 and defaults to a more secure set of ciphers and key-exchange algorithms. OpenSSH 7 omits some legacy cryptography.

NOTE: See the OpenSSH 7 documentation at <https://www.openssh.com/> for more information about these extensions.

Junos OS Evolved supports the following set of ciphers by default:

- chacha20-poly1305@openssh.com
- aes128-ctr
- aes192-ctr
- aes256-ctr
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com

In Junos OS Evolved, the following ciphers are not supported by default, but you can configure your device to support them. They are listed from the most secure to the least secure:

- aes256-cbc
- aes192-cbc
- aes128-cbc
- blowfish-cbc
- cast128-cbc

- arcfour256
- arcfour128
- arcfour

Junos OS Evolved supports the following set of key-exchange methods by default:

- curve25519-sha256
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521
- group-exchange-sha2
- dh-group14-sha1

In Junos OS Evolved, the following key-exchange methods are not supported by default, but you can configure your device to support them:

- group-exchange-sha1
- dh-group1-sha1

To configure the SSH service to support legacy cryptography:

NOTE: By configuring an ordered set of ciphers, key-exchange methods, or message authentication codes (MACs), the newly defined set is applied to both server and client commands. Changes to the defaults affect the `file copy` command when you use Secure Copy Protocol (SCP).

1. Add support for ciphers by using the `set system services ssh ciphers [cipher 1 cipher 2 ...]` command. We recommend that you add the ciphers to the end of the configuration list so that they are among the last options used. In the following example, the `arcfour` and `blowfish-cbc` ciphers are added to the default set:

```
[edit system services ssh]
user@device# set ciphers [ chacha20-poly1305@openssh.com aes128-ctr aes192-ctr aes256-ctr
aes128-gcm@openssh.com aes256-gcm@openssh.com arcfour blowfish-cbc ]
```

2. Add support for key-exchange methods by using the `set system services ssh key-exchange [method 1 method 2 ...]` command. We recommend that you add the key-exchange methods to the end of the

configuration list so that they are among the last options used. In the following example, the `dh-group1-sha1` key-exchange method is added to the default set:

```
[edit system services ssh]
user@device# set key-exchange [ curve25519-sha256 ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-
sha2-nistp521 group-exchange-sha2 dh-group14-sha1 dh-group1-sha1 ]
```

3. Commit the configuration:

```
[edit]
user@device# commit
```

SEE ALSO

| *key-exchange*

Configure Outbound SSH Service

IN THIS SECTION

- [Send the Public SSH Host Key to the Outbound SSH Client | 208](#)
- [Configure Keepalive Messages for Outbound SSH Connections | 209](#)
- [Configure a New Outbound SSH Connection | 210](#)
- [Configure the Outbound SSH Client to Accept NETCONF as an Available Service | 210](#)
- [Configure Outbound SSH Clients | 210](#)
- [Configure Routing Instances for Outbound SSH Clients | 211](#)

You can configure a device running Junos OS Evolved to initiate a TCP/IP connection with a client management application. If the management application does not reach a Juniper Networks device, for example, the device being a firewall. In such cases, `outbound-ssh` can be configured on the Juniper Networks device. An `outbound-ssh` configuration initiates a reverse SSH connection from server to client to the management application. This outbound SSH connection is closed only after the configuration are removed from the device.

NOTE: There is no initiation command with outbound SSH. After you configure and commit outbound SSH, the device begins to initiate an outbound SSH connection based on the committed configuration. The device repeatedly attempts to create this connection until successful. If the connection between the device and the client management application is dropped, the device again attempts to create a new outbound SSH connection until successful. This connection is maintained until the outbound SSH stanza is removed from the configuration.

To configure the device for outbound SSH connections, include the `outbound-ssh` statement at the `[edit system services]` hierarchy level:

```
[edit system services outbound-ssh]
```

The following topics describe the tasks for configuring the outbound SSH service.

Send the Public SSH Host Key to the Outbound SSH Client

Each time the router or switch establishes an outbound SSH connection, it first sends an initiation sequence to the management client. This sequence identifies the router or switch to the management client. Within this transmission is the value of *device-id*.

To configure the device identifier of the router or switch, include the `device-id` statement at the `[edit system services outbound-ssh client client-id]` hierarchy level:

```
[edit system services outbound-ssh client client-id]  
device-id device-id;
```

The initiation sequence when `secret` is not configured:

```
MSG-ID: DEVICE-CONN-INFO\r\n  
MSG-VER: V1\r\n  
DEVICE-ID: <device-id>\r\n
```

During the initialization of an SSH connection, the client authenticates the identity of the device using the public SSH host key of the device. Therefore, before the client can initiate the SSH sequence, the client needs the public SSH key of the device. When you configure the `secret` statement, the device passes its public SSH key as part of the outbound SSH connection initiation sequence.

When the `secret` statement is set and the device establishes an outbound SSH connection, the device communicates its device ID, its public SSH key, and an SHA1 hash derived in part from the `secret` statement. The value of the `secret` statement is shared between the device and the management client.

The client uses the shared secret to authenticate the public SSH host key it is receiving to determine whether the public key is from the device identified by the `device-id` statement.

Using the `secret` statement to transport the public SSH host key is optional. You can manually transport and install the public key onto the client system.

NOTE: Including the `secret` statement means that the device sends its public SSH host key every time it establishes a connection to the client. It is then up to the client to decide what to do with the SSH host key if the client already has an SSH key for that device. We recommend that you replace the client's copy of the SSH host key with the new key. Host keys can change for various reasons. By replacing the key each time a connection is established, you ensure that the client has the latest key.

To send the router's or switch's public SSH host key when the device connects to the client, include the `secret` statement at the `[edit system services outbound-ssh client client-id]` hierarchy level:

```
[edit system services outbound-ssh client client-id]
secret password;
```

The following message is sent by the device when the `secret` attribute is configured:

```
MSG-ID: DEVICE-CONN-INFO\r\n
MSG-VER: V1\r\n
DEVICE-ID: <device-id>\r\n
HOST-KEY: <public-host-key>\r\n
HMAC:<HMAC(pub-SSH-host-key, <secret>>)>\r\n
```

Configure Keepalive Messages for Outbound SSH Connections

After the client application has the router's or switch's public SSH host key, it can initiate the SSH sequence as if it had created the TCP/IP connection. The client can then authenticate the device using its copy of the router's or switch's public host SSH key as part of that sequence. The device authenticates the client user through the mechanisms supported in Junos OS Evolved (RSA/DSA public string or password authentication).

To enable the device to send SSH protocol keepalive messages to the client application, configure the `keep-alive` statement at the `[edit system services outbound-ssh client client-id]` hierarchy level:

```
[edit system services outbound-ssh client client-id]
keep-alive {
```

```

    retry number;
    timeout seconds;
}

```

Configure a New Outbound SSH Connection

When disconnected, the device begins to initiate a new outbound SSH connection. To specify how the device reconnects to the server after a connection is dropped, include the `reconnect-strategy` statement at the `[edit system services outbound-ssh client client-id]` hierarchy level:

```

[edit system services outbound-ssh client-id]
reconnect-strategy (sticky | in-order);

```

You can also specify the number of retry attempts and set the amount of time before the reconnection attempts stop. See ["Configure Keepalive Messages for Outbound SSH Connections" on page 209](#).

Configure the Outbound SSH Client to Accept NETCONF as an Available Service

To configure the application to accept NETCONF as an available service, include the `services netconf` statement at the `[edit system services outbound-ssh client client-id]` hierarchy level:

```

[edit system services outbound-ssh client client-id]
services {
    netconf;
}

```

Configure Outbound SSH Clients

To configure the clients available for this outbound SSH connection, list each client with a separate `address` statement at the `[edit system services outbound-ssh client client-id]` hierarchy level:

```

[edit system services outbound-ssh client client-id]
  address address {
    retry number;
    timeout seconds;
    port port-number;
  }

```

NOTE: Outbound SSH connections support IPv4 and IPv6 address formats.

Configure Routing Instances for Outbound SSH Clients

To use the management routing instance, first enable the `mgmt_junos` routing instance using the `set system management-instance` command.

To use any other routing instance, first configure the routing instance at the `[edit routing-instances]` hierarchy.

If you do not specify a routing instance, your device will establish the outbound SSH connection using the default routing table.

Configure NETCONF-Over-SSH Connections on a Specified TCP Port

Junos OS Evolved enables you to restrict incoming NETCONF connections to a specified TCP port without configuring a firewall. To configure the TCP port used for NETCONF-over-SSH connections, include the port statement at the `[edit system services netconf ssh]` hierarchy level. The configured port accepts only NETCONF-over-SSH sessions. Regular SSH session requests for this port are rejected.

You can either configure the default port 830 for NETCONF connections over SSH, as specified in RFC 4742, *Using the NETCONF Configuration Protocol over Secure Shell (SSH)*, or configure any port from 1 through 65535.

NOTE:

- The default SSH port (22) continues to accept NETCONF sessions even with a configured NETCONF server port. To disable the SSH port from accepting NETCONF sessions, specify this in the login event script.
- We do not recommend configuring the default ports for FTP (21) and Telnet (23) services for configuring NETCONF-over-SSH connections.

RELATED DOCUMENTATION

USB Modems for Remote Management of Security Devices

Secure Web Access for Remote Management

Configuration Guidelines for Securing Console Port Access

IN THIS SECTION

- [Secure the Console Port | 212](#)

We recommend that you (the network administrator) disable the console port to prevent unauthorized access to the device.

Secure the Console Port

You can use the console port on a device to connect to the device through an RJ-45 serial cable. From the console port, you can use the CLI to configure the device. By default, the console port is enabled. To secure the console port, you can configure the device to take the following actions:

- Log out of the console session when you unplug the serial cable connected to the console port.
- Disable root login connections to the console. This action prevents a non-root user from performing password recovery operation using the console.
- Disable the console port. We recommend that you disable the console port to prevent unauthorized access to the device. Preventing unauthorized access is especially important when the device is used as customer premises equipment (CPE) and is forwarding sensitive traffic.

NOTE: It is not always possible to disable the console port, because console access is important during operations such as software upgrades.

To secure the console port:

1. Do one of the following:

- Disable the console port.

```
[edit system ports console]  
user@host# set disable
```

- Disable root login connections to the console.

```
[edit system ports console]  
user@host# set insecure
```

NOTE: After you configure the console port as insecure, if a user tries to perform the password recovery operation by booting in single-user mode, the device will prompt for the root password. This way, only a user who knows the root password will be able to log in to single-user mode for password recovery.

- Log out of the console session when the serial cable connected to the console port is unplugged. Enter

```
[edit system ports console]  
user@host# set log-out-on-disconnect
```

2. After you configure the device, enter `commit` in configuration mode.

6

CHAPTER

Device Discovery

Device Discovery Using LLDP | 215

Device Discovery Using LLDP

IN THIS SECTION

- [Understanding LLDP | 215](#)
- [Configuring LLDP \(CLI Procedure\) | 216](#)

The Link Layer Discovery Protocol (LLDP) is an industry-standard, vendor-neutral method to allow networked devices to advertise capabilities, identity, and other information onto a LAN. It also provides additional types, lengths, and values (TLVs) for capabilities discovery, network policy, Power over Ethernet (PoE), and inventory management. For more information, read this topic.

Understanding LLDP

The device uses LLDP to learn and to distribute device information on network links. The device uses this information to identify a variety of devices quickly. This quick identification results in a LAN that interoperates smoothly and efficiently.

LLDP-capable devices transmit information in type, length, and value (TLV) messages to neighbor devices. Device information can include specifics, such as the chassis identification, the port identification, the system name, and the system capabilities. The TLVs leverage this information from parameters that have already been configured in Junos OS Evolved.

The device supports the following basic TLVs:

- **Chassis Identifier**—The MAC address associated with the local system.
- **Port Identifier**—The port identification for the specified port in the local system.
- **Port Description**—The user-configured port description. The port description can be a maximum of 256 characters.
- **System Name**—The user-configured name of the local system. The system name can be a maximum of 256 characters.
- **System Description**—The system description containing information about the software and current image running on the system. This information is taken from the software. You cannot configure this information.

- **System Capabilities**—The primary function performed by the system, for example, bridge or router. This information cannot be configured, but is based on the model of the product.
- **Management Address**—The IP management address of the local system.

The device supports the following 802.3 TLVs:

- **Power via MDI**—A TLV that advertises media dependent interface (MDI) power support, power source equipment (PSE) power pair, and power class information.
- **MAC/PHY Configuration Status**—A TLV that advertises information about the physical interface, such as autonegotiation status and support and MAU type. The information is based on the physical interface structure. You cannot configure this information.
- **Link Aggregation**—A TLV that advertises whether the port is aggregated and its aggregated port ID.
- **Maximum Frame Size**—A TLV that advertises the Maximum Transmission Unit (MTU) of the interface sending LLDP frames.
- **Port VLAN**—A TLV that advertises the VLAN name configured on the interface.

Configuring LLDP (CLI Procedure)

IN THIS SECTION

- [Enable LLDP on Interfaces | 217](#)
- [Adjust LLDP Advertisement Settings | 217](#)
- [Adjust SNMP Notification Settings of LLDP Changes | 218](#)
- [Specify a Management Address for the LLDP Management TLV | 219](#)
- [Specify a Management Interface for the LLDP Management TLV | 220](#)
- [Configure LLDP Power Negotiation | 220](#)
- [Disable LLDP TLVs | 221](#)

Follow these steps to configure LLDP on your device.

Enable LLDP on Interfaces

LLDP is enabled on all interfaces by default. If you disable it, you can re-enable LLDP by configuring it on all interfaces or on specific interfaces.

- To configure LLDP on all interfaces:

```
[edit protocols lldp]
user@device# set interface all
```

- To configure LLDP on a specific interface:

```
[edit protocols lldp]
user@device# set interface interface-name
```

Adjust LLDP Advertisement Settings

You can adjust the following settings for LLDP advertisements for troubleshooting or verification purposes. LLDP uses the default values when it is enabled. For normal operations, we recommend that you do not change the default values.

- To specify the frequency at which LLDP advertisements are sent (in seconds):

```
[edit protocols lldp]
user@device# set advertisement-interval seconds
```

For example, using the default value of 30 seconds:

```
[edit protocols lldp]
user@device# set advertisement-interval 30
```

- To specify the number of seconds that LLDP information is held before it is discarded:

```
[edit protocols lldp]
user@device# set hold-multiplier number
```

For example, using the default value of 4:

```
[edit protocols lldp]
user@device# set hold-multiplier 4
```

The hold-multiplier value is used in combination with the advertisement-interval value. Using the default values means that the advertisement-interval value of 30 will be multiplied by the hold-multiplier value of 4, resulting in a LLDP hold time of 120 seconds.

- Set the transmit delay to specify the number of seconds the device waits before sending advertisements to neighbors after a change is made in a TLV (element in LLDP or in the state of the local system). A change in state of the local system includes a change in hostname or management address. The transmit delay is enabled by default to reduce the delay in notifying neighbors of a change in the local system. The default transmit delay is 1 second if the advertisement-interval value is set to less than 8 seconds. The default value is 2 seconds if the advertisement-interval value is set to 8 seconds or more.

```
[edit protocols lldp]
user@device# set transmit-delay seconds
```

For example:

```
[edit protocols lldp]
user@device# set transmit-delay 2
```

NOTE: The advertisement-interval value must be greater than or equal to four times the transmit-delay value; otherwise, an error is returned when you attempt to commit the configuration.

Adjust SNMP Notification Settings of LLDP Changes

You can adjust the following settings for SNMP notifications of LLDP changes. If the values are not specified or if the interval values are set to 0, the notifications are disabled.

- To specify the frequency at which LLDP database changes are sent (in seconds):

```
[edit protocols lldp]
user@device# set lldp-configuration-notification-interval seconds
```

For example:

```
[edit protocols lldp]
user@device# set lldp-configuration-notification-interval 600
```

- To configure how long SNMP trap notifications wait for topology changes (in seconds):

```
[edit protocols lldp]
user@device# set ptopo-configuration-trap-interval seconds
```

For example:

```
[edit protocols lldp]
user@device# set ptopo-configuration-trap-interval 600
```

- To specify the holding time (used in combination with the ptopo-configuration-trap-interval value) to maintain dynamic topology entries (in seconds):

```
[edit protocols lldp]
user@device# set ptopo-configuration-maximum-hold-time seconds
```

For example:

```
[edit protocols lldp]
user@device# set ptopo-configuration-maximum-hold-time 2147483647
```

Specify a Management Address for the LLDP Management TLV

You can configure an IPv4 or IPv6 management address to be used in the LLDP Management Address TLV messages. An out-of-band management address must be used as the value for the `management-address` statement.

To configure the management address:

```
[edit protocols lldp]
user@device# set management-address ip-address
```

NOTE: Ensure that the interface with the configured management address has LLDP enabled using the `set protocols lldp interface` command. If you configure a customized management address for LLDP on an interface that has LLDP disabled, the `show lldp local-information` command output does not display the correct interface information.

Specify a Management Interface for the LLDP Management TLV

you can configure an interface to be used in the LLDP Management Address TLV messages.

NOTE: You cannot configure management address and management interface at the same time.

To configure the management interface:

```
[edit protocols lldp]
user@device# set management-interface interface-name
```

If the interface does not have an IP address, the IP address of the default management interfaces is used.

Configure LLDP Power Negotiation

LLDP power negotiation enables the device's Power over Ethernet (PoE) controller to dynamically allocate PoE power to PoE interfaces, based on the needs of the powered device, by negotiating with LLDP-enabled powered devices.

LLDP power negotiation is supported on devices running PoE controller software version 4.04 or later.

LLDP power negotiation is automatically enabled when the PoE management mode is set to class:

- ```
[edit poe]
user@device# set management class
```

To disable LLDP power negotiation:

- On all device interfaces:

```
[edit protocols lldp interface all power-negotiation]
user@device# set disable
```

- On a specific interface:

```
[edit protocols lldp interface interface-name power-negotiation]
user@device# set disable
```

## Disable LLDP TLVs

LLDP sends TLV messages by default. You can configure LLDP to disable non-mandatory TLVs. The mandatory TLVs are: chassis-id, port-id, and time-to-live. In this procedure, any reference to disabling all TLVs means disabling all non-mandatory TLVs.

There are two options for disabling TLVs:

- `tlv-select`—Select which TLVs are allowed to be advertised by LLDP. This approach is useful if you want to allow only a few TLVs and nothing else.
- `tlv-filter`—Filter the TLVs that should not be advertised by LLDP. Use this option if you want to filter only a few TLVs and allow everything else.

**NOTE:** The `tlv-select` and `tlv-filter` options are mutually exclusive and cannot be used on the same configuration stanza at the same time.

You can disable TLVs on specific interfaces or on all interfaces. The configuration under the interface configuration stanza takes precedence over the global configuration.

To select which TLVs are allowed to be advertised by LLDP:

- On all interfaces:

```
[edit protocols lldp]
user@device# set tlv-select tlv-name
```

- On a specific interface:

```
[edit protocols lldp]
user@device# set interface interface-name tlv-select tlv-name
```

To filter TLVs that should not be advertised by LLDP:

- On all interfaces:

```
[edit protocols lldp]
user@device# set tlv-filter tlv-name
```

- On a specific interface:

```
[edit protocols lldp]
user@device# set interface interface-name tlv-filter tlv-name
```

The following example disables all TLVs except port-description:

```
[edit protocols lldp]
user@device# set tlv-select port-description
```

The following example disables the system-description TLV on the et-1/0/1 interface:

```
[edit protocols lldp]
user@device# set interface et-1/0/1 tlv-filter system-description
```

The following example disables all TLVs except port-description and system-description on all interfaces except on the et-1/0/1 interface, where it disables only the system-name TLV:

```
[edit protocols lldp]
user@device# set tlv-select [port-description system-description]
user@device# set interface et-1/0/1 tlv-filter system-name
```

# 7

CHAPTER

## Operational Commands

---

[request security uefi revoke](#) | 224

---



# request security uefi revoke

## IN THIS SECTION

- Syntax | 224
- Description | 224
- Options | 224
- Required Privilege Level | 225
- Release Information | 225

## Syntax

```
request security uefi revoke
<development-keys>
<load-cert-file>
```

## Description

Revoke the platform Unified Extensible Firmware Interface (UEFI) keys.

## Options

**development-keys** (Optional) Revoke the development keys for the operating system.



**WARNING:** If you revoke the development keys, the device will not boot up. You will have to manually reprogram the BIOS image provided by Juniper Networks.

*load-cert-file* (Optional) Revoke the certificate. Whenever any key is compromised, the keys and certificates have to be revoked in a controlled environment.

## Required Privilege Level

view

## Release Information

Statement introduced in Junos OS Evolved Release 18.4R1.