CONFIGURATION GUIDE

Ruckus FastIron FIPS and Common Criteria Configuration Guide, 08.0.70

Supporting FastIron Software Release 08.0.70

Part Number: 53-1005356-02
Publication Date: 3 September 2019

Copyright, Trademark and Proprietary Rights Information

Export Restrictions

These products and associated technical data (in print or electronic form) may be subject to export control laws of the United States of America. It is your responsibility to determine the applicable regulations and to comply with them. The following notice is applicable for all products or technology subject to export control:

These items are controlled by the U.S. Government and authorized for export only to the country of ultimate destination for use by the ultimate consignee or end-user(s) herein identified. They may not be resold, transferred, or otherwise disposed of, to any other country or to any person other than the authorized ultimate consignee or end-user(s), either in their original form or after being incorporated into other items, without first obtaining approval from the U.S. government or as otherwise authorized by U.S. law and regulations.

Disclaimer

THIS CONTENT AND ASSOCIATED PRODUCTS OR SERVICES ("MATERIALS"), ARE PROVIDED "AS IS" AND WITHOUT WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED. TO THE FULLEST EXTENT PERMISSIBLE PURSUANT TO APPLICABLE LAW, COMMSCOPE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT, FREEDOM FROM COMPUTER VIRUS, AND WARRANTIES ARISING FROM COURSE OF DEALING OR COURSE OF PERFORMANCE. CommScope does not represent or warrant that the functions described or contained in the Materials will be uninterrupted or error-free, that defects will be corrected, or are free of viruses or other harmful components. CommScope does not make any warranties or representations regarding the use of the Materials in terms of their completeness, correctness, accuracy, adequacy, usefulness, timeliness, reliability or otherwise. As a condition of your use of the Materials, you warrant to CommScope that you will not make use thereof for any purpose that is unlawful or prohibited by their associated terms of use.

Limitation of Liability

IN NO EVENT SHALL COMMSCOPE, COMMSCOPE AFFILIATES, OR THEIR OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, SUPPLIERS, LICENSORS AND THIRD PARTY PARTNERS, BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER, EVEN IF COMMSCOPE HAS BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, WHETHER IN AN ACTION UNDER CONTRACT, TORT, OR ANY OTHER THEORY ARISING FROM YOUR ACCESS TO, OR USE OF, THE MATERIALS. Because some jurisdictions do not allow limitations on how long an implied warranty lasts, or the exclusion or limitation of liability for consequential or incidental damages, some of the above limitations may not apply to you.

Trademarks

ARRIS, the ARRIS logo, CommScope, Ruckus, Ruckus Wireless, Ruckus Networks, Ruckus logo, the Big Dog design, BeamFlex, ChannelFly, EdgeIron, FastIron, HyperEdge, ICX, IronPoint, OPENG, SmartCell, Unleashed, Xclaim, and ZoneFlex are trademarks of CommScope, Inc. and/or its affiliates. Wi-Fi Alliance, Wi-Fi, the Wi-Fi logo, Wi-Fi Certified, the Wi-Fi CERTIFIED logo, Wi-Fi Protected Access, the Wi-Fi Protected Setup logo, Wi-Fi Protected Setup, Wi-Fi Multimedia and WPA2 and WMM are trademarks or registered trademarks of Wi-Fi Alliance. All other trademarks are the property of their respective owners.

Contents

Preface

Document Conventions

The following table lists the text conventions that are used throughout this guide.

TABLE 1 Text Conventions

Convention	Description	Example
`monospace`	Identifies command syntax examples	`device(config)# interface ethernet 1/1/6`
bold	User interface (UI) components such as screen or page names, keyboard keys, software buttons, and field names	On the **Start** menu, click **All Programs**.
italics	Publication titles	Refer to the *Ruckus Small Cell Release Notes* for more information.

Notes, Cautions, and Warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A NOTE provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An ATTENTION statement indicates some information that you must read before continuing with the current action or task.

CAUTION
A CAUTION statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER
A DANGER statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Command Syntax Conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.
[ ]	Syntax components displayed within square brackets are optional.
	Default responses to system prompts are enclosed in square brackets.
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Document Feedback

Ruckus is interested in improving its documentation and welcomes your comments and suggestions.

You can email your comments to Ruckus at #Ruckus-Docs@commscope.com.

When contacting us, include the following information:

- Document title and release number
- Document part number (on the cover page)
- Page number (if appropriate)

For example:

- Ruckus SmartZone Upgrade Guide, Release 5.0
- Part number: 800-71850-001 Rev A
- Page 7

Ruckus Product Documentation Resources

Visit the Ruckus website to locate related documentation for your product and additional Ruckus resources.

Release Notes and other user documentation are available at https://support.ruckuswireless.com/documents. You can locate the documentation by product or perform a text search. Access to Release Notes requires an active support contract and a Ruckus Support Portal user account. Other technical documentation content is available without logging in to the Ruckus Support Portal.

White papers, data sheets, and other product documentation are available at https://www.ruckuswireless.com.

Online Training Resources

To access a variety of online Ruckus training modules, including free introductory courses to wireless networking essentials, site surveys, and Ruckus products, visit the Ruckus Training Portal at https://training.ruckuswireless.com.

Contacting Ruckus Customer Services and Support

The Customer Services and Support (CSS) organization is available to provide assistance to customers with active warranties on their Ruckus products, and customers and partners with active support contracts.

For product support information and details on contacting the Support Team, go directly to the Ruckus Support Portal using https://support.ruckuswireless.com, or go to https://www.ruckuswireless.com and select **Support**.

What Support Do I Need?

Technical issues are usually described in terms of priority (or severity). To determine if you need to call and open a case or access the self-service resources, use the following criteria:

- Priority 1 (P1)—Critical. Network or service is down and business is impacted. No known workaround. Go to the **Open a Case** section.
- Priority 2 (P2)—High. Network or service is impacted, but not down. Business impact may be high. Workaround may be available. Go to the **Open a Case** section.
- Priority 3 (P3)—Medium. Network or service is moderately impacted, but most business remains functional. Go to the **Self-Service Resources** section.
- Priority 4 (P4)—Low. Requests for information, product documentation, or product enhancements. Go to the **Self-Service Resources** section.

Open a Case

When your entire network is down (P1), or severely impacted (P2), call the appropriate telephone number listed below to get help:

- Continental United States: 1-855-782-5871
- Canada: 1-855-782-5871
- Europe, Middle East, Africa, Central and South America, and Asia Pacific, toll-free numbers are available at https://support.ruckuswireless.com/contact-us and Live Chat is also available.
- Worldwide toll number for our support organization. Phone charges will apply: +1-650-265-0903

We suggest that you keep a physical note of the appropriate support number in case you have an entire network outage.

Self-Service Resources

The Ruckus Support Portal at https://support.ruckuswireless.com offers a number of tools to help you to research and resolve problems with your Ruckus products, including:

- Technical Documentation—https://support.ruckuswireless.com/documents

- Community Forums—https://forums.ruckuswireless.com/ruckuswireless/categories
- Knowledge Base Articles—https://support.ruckuswireless.com/answers
- Software Downloads and Release Notes—https://support.ruckuswireless.com/#products_grid
- Security Bulletins—https://support.ruckuswireless.com/security

Using these resources will help you to resolve some issues, and will provide TAC with additional data from your troubleshooting analysis if you still require assistance through a support case or RMA. If you still require help, open and manage your case at https://support.ruckuswireless.com/case_management.

About This Document

Supported hardware and software

- To determine if the Ruckus device and current software version are FIPS-certified, refer to http://csrc.nist.gov/groups/STM/cmvp/validation.html.
- To determine if the Ruckus device and current software version is Common Criteria certified, refer to https://www.niap-ccevs.org/Product/.

Federal Information Processing Standards

FIPS Overview

A Ruckus device in Federal Information Processing Standards (FIPS) mode is compliant with the standards established by the United States government and the National Institute of Standards and Technology (NIST).

> **NOTE**
> Not all software releases support FIPS. Refer to the Release notes for the software you are running to see if it supports FIPS.

The FIPS Publication 140-2 is a technical standard and worldwide de-facto standard for the implementation of cryptographic modules. The FIPS Publication 140-2 contains security standards developed by the United States government and the National Institute of Standards and Technology (NIST) for use by all non-military government agencies and by government contractors. Due to their importance within the security industry, these standards form a baseline for many security requirements.

In FIPS mode, the network processing occurs in the kernel and in privileged daemons.

> **NOTE**
> To determine if the device and current software version are FIPS-certified, refer to http://csrc.nist.gov/groups/STM/cmvp/validation.html.

You can configure the device to run in FIPS mode to ensure that the device is operating according to the standards stated in FIPS Publication 140-2.

A device is FIPS 140-2-compliant when the following requirements have been met:

- The device software is placed in FIPS mode with the FIPS security policy applied.

For a list of physical ports and interfaces for each device, refer to the Security Policy.

How FIPS Works

You place a device in FIPS mode by entering the **fips enable** CLI command on the management station while the station is connected to the device console port with a serial cable. After you enter the **fips enable** command, the device is administratively in FIPS mode and by default runs in strict FIPS-compliant mode upon reload.

The default FIPS policy is for the system to run in a strict mode that fully supports FIPS 140-2 specifications. However, the device allows you the flexibility to configure a modified FIPS policy according to your network requirements. .

> **NOTE**
> A FIPS policy that varies from the default policy weakens the intent of the FIPS 140-2 specifications; when implemented, the device is not operating in full compliance with these specifications. Refer to Modifying the FIPS policy on page 39

The default FIPS policy enforces the following actions for strict FIPS compliance:

- Disables TFTP access
- Disables monitor access to memory access commands
- Returns 0 or null for SNMP MIBs for passwords or keys (referred to as critical security parameter objects)
- Zeroizes shared secrets and passwords

The device performs the following functions automatically during reboot after the **fips enable** command is entered:

- Disables Telnet
- Enables SCP access
- Disables the HTTP server
- Disables SNMP access to critical security parameter (CSP) MIB objects

After defining the FIPS policy, save the configuration, and reboot the device. While the device is booting, several tests are run to ensure the device is FIPS-compliant.

After these tests are completed successfully, the device reloads and is operationally in FIPS mode.

All the optional FIPS policy commands are provided to perform various non-approved FIPS operations when FIPS is enabled.

NOTE
If any of these policy commands are configured, the module is not operating in the approved FIPS mode.

NOTE
Web server is not supported in FIPS/CC mode.

Upgrading and Downgrading Software on FIPS-enabled Devices

Upgrading FIPS-enabled devices

FIPS 140-2 compliance is a combination of implemented hardware procedures and the activation of a software-based security policy.

FIPS 140-2 certification is achieved when the device meets certain physical security and software conditions:

- FIPS software compliance: The devices are configured to run in FIPS operational mode with the default FIPS security policy.

  NOTE
  Although commands to alter the FIPS security policy exist, altering the default FIPS security policy is not recommended.

  NOTE
  After enabling FIPS mode on your device, you cannot disable it without losing the device configuration. To disable FIPS mode, it is recommended that you contact Ruckus Technical Support and perform the procedure under qualified guidance.

Preparing for a FIPS software upgrade

Before upgrading the software on the device, consider the following important information and notes.

FastIron devices can store two Full Layer 3 images or two Layer 2 or Base Layer 3 images.

NOTE
Signature file auto-copy is not supported. Only image files will be updated and the FIPS check will fail if the corresponding signature file is not present in the same partition.

NOTE
Manual update of firmware is not supported. If you attempt to update firmware manually, a message similar to the one shown in the following example is displayed.

```
device# copy tftp service-module 10.1.1.1 1/3 abc.bin      <-- Invalid attempt
IPSEC firmware update is not supported in FIPS mode        <-- System response
```

In addition, the error message is recorded in the system log as shown in the following example.

```
SYSLOG: <12> Dec 19 09:59:17 device IPsec: IPSEC firmware update is not supported in FIPS mode
```

Image verification in FIPS or CC mode

For a FIPS- or CC-enabled device running FastIron 08.0.10 or later, uploading a flash or boot code triggers a FIPS Integrity Qualification test that performs a digital signature verification of the flash or boot code using a signature file. In earlier FastIron versions, the test is triggered only when the FIPS- or CC-enabled device boots.

- FIPS devices running FastIron 08.0.10 or later support the digital signature files generated using the SHA-256/RSA-2048 algorithm.
- FastIron versions earlier than 08.0.10 in FIPS or CC mode support the digital signature files generated using the SHA-1/DSA-1024 algorithm.

Verifying the currently active software version

Use the **show version** command to check the active software version on a FastIron device. The following example displays the current software version of an ICX 7450 as version 08.0.70T213 and provides additional details on the image file and the modules installed in the device.

```
Device# show version
  Copyright (c) 2017 Ruckus Wireless, Inc. All rights reserved.
    UNIT 1: compiled on Dec 11 2017 at 06:20:06 labeled as SPR08070
      (29327100 bytes) from Primary SPR08070.bin
        SW: Version 08.0.70T213
      Compressed Boot-Monitor Image size = 786944, Version:10.1.09T215 (spz10109b009)
       Compiled on Fri Nov  4 18:01:15 2016

  HW: Stackable ICX7450-48
  Internal USB: Serial #: 9900614021200057
      Vendor: ATP Electronics, Total size = 1919 MB
==========================================================================
UNIT 1: SL 1: ICX7450-48 48-port Management Module
      Serial  #:CYQ3316K00A
      License: ICX7450_L3_SOFT_PACKAGE   (LID: easIIGLmFFc)
      License Compliance: ICX7450-PREM-LIC-SW is Compliant
      P-ASIC  0: type B548, rev 01  Chip BCM56548_A0
==========================================================================
UNIT 1: SL 3: ICX7400-SERVICE-MOD Module
      FW Version  #:2.09
==========================================================================
UNIT 1: SL 4: ICX7400-4X10GF 4-port 40G Module
      Serial  #:CYV3315K003
==========================================================================
 1000 MHz ARM processor ARMv7 88 MHz bus
 8192 KB boot flash memory
 2048 MB code flash memory
 2048 MB DRAM
STACKID 1  system uptime is 3 day(s) 22 hour(s) 59 minute(s) 26 second(s)
The system started at 09:09:31 GMT+00 Thu Dec 14 2017

The system : started=warm start   reloaded=by "reload"
```

Checking the inactive software version in secondary storage

Use the **show flash** command to verify the version of the inactive image loaded in secondary flash. The show flash command displays the image version for both primary and secondary flash partitions as shown in the following example.

```
Device# show flash
Stack unit 1:
  Compressed Pri Code size = 29327100, Version:08.0.70T213 (SPR08070.bin)
  Compressed Sec Code size = 29326864, Version:08.0.70BT213 (SPS08070.bin)
  Compressed Boot-Monitor Image size = 786944, Version:10.1.09T215
  Code Flash Free Space = 44478464
```

Performing a FIPS or CC software upgrade

To upgrade the FastIron software image to support a FIPS or CC environment, perform the following steps.

1. Place the new flash signature file and the new flash image in an SCP client directory to which the FastIron device has access.
2. **NOTE**
   In FIPS mode, SSH and SCP use diffie-hellman-group-exchange-sha256 by default for Key Exchange. The SCP client used should be able to support this option. Any client with OpenSSH 7.2 or higher supports this option as does Putty 0.67 or higher.

   If the device is FIPS- or CC-enabled, enter the following command to copy the SHA-256/RSA-2048 signature file from the SCP client into flash memory:

   Syntax: scp *signaturefilename username***@***ipaddress***:file:primary.sig |secondary.sig**

   ```
   $ scp SPR08070.sig test@1.20.2.2:file:secondary.sig
   ```

   NOTE
   If the device is not FIPS- or CC-enabled, refer to FIPS Configuration on page 19 to enable FIPS or CC mode.
3. To copy the flash code from the scp client into flash memory, enter the following command.

   Syntax: scp *image.bin username***@***ipaddress***:flash:pri | sec**:*filename*

   ```
   $ scp SPR08070.bin test@1.20.66.70:flash:sec:SPR08070.bin
   ```
4. To copy the boot image and appropriate signature file for the boot image, enter the following command.

   ```
   $ scp swz10110.bin test@10.20.66.70:flash:bootrom
   ```
5. Verify that the flash code has been successfully copied by examining the console log or entering the **show flash** command at any level of the CLI.

   ```
   --FIPS: secondary image verification success
   ```
6. Save the running configuration by entering the **write memory** command.
7. Reload the configuration to run the FIPS-enabled or CC-enabled image by entering the **reload** command.
8. Repeat Step 2 with the SHA-256/RSA-2048 signature file and then repeat Step 3. This ensures digital signature verification of the flash code with the SHA-256/RSA-2048 signature file.

SSH connection after upgrading a FIPS or CC operational device to FastIron 08.0.01 or later

If you are using key authentication for SSH connection, you should re-import the new format public keys into the device after upgrading a device in FIPS or CC operational mode.

For details on how to import public keys, refer to the section Usernames and SSH public key authentication on page 23.

Downgrading from FIPS to non-FIPS

While a FIPS-supported image is running on the device, at any time the image can be running in FIPS or non-FIPS operational mode. To change from FIPS mode to non-FIPS mode, you must copy the signature file.

Downgrading from FIPS mode to non-FIPS mode clears all shared secrets, host passwords, SSH and HTTPS host keys and HTTPS certificates.

NOTE
Before upgrading or downgrading a major software version, zeroize the keys by executing the **crypto key zeroize** command.

NOTE
Once FIPS mode is enabled on the system, even if the mode is disabled later, a firmware integrity test will always be carried out on the device when the image is copied.

To place a device in non-FIPS mode and then use TFTP or SCP to download and initialize an older image, complete the following steps.

1. Log in to the device by entering your user name and password.
2. Disable FIPS by entering the **no fips enable** or **no fips enable common-criteria** command at the prompt.
3. To copy the desired non-FIPS binary image into flash memory, enter the following command:

   copy tftp flash *ip-addr image-name*

   NOTE
   The device will perform a checksum validation on the newly downloaded image because the currently running image does not support FIPS and does not require a signature file.

4. Reload the configuration by entering the **reload** command.

   If the unit goes into rolling reboots when there is a mismatch between the image and signature files, the following set of commands can be used at the boot-loaded prompt to reset the device to factory default so the non-FIPS image can be loaded.

   remote address A.B.C.D/M

   remote default-gateway W.X.Y.Z

   boot system flash *primary* | *secondary*

The commands configure the switch to the factory default state. The FIPS configuration is removed, the switch image can be copied, and the switch will load with this image without signature verification.

Once the switch is rebooted, refer to Placing the device in FIPS mode on page 31 to enable FIPS.

FIPS Configuration

User roles in FIPS mode

Configuring FIPS mode on the Ruckus devices complies with the standards established by the United States government and the National Institute of Standards and Technology (NIST).

A Ruckus device in FIPS mode supports three user roles:

- Crypto-officer role: The Crypto-officer role on the device in FIPS mode is equivalent to the administrator role, or the super-user role in non-FIPS mode.
- Port Configuration Administrator role: The Port Configuration Administrator on the device in FIPS mode is equivalent to the port configuration user in non-FIPS mode and has write access to the interface configuration mode only.
- User role: The User role on the device in FIPS mode has read-only privileges and no configuration mode access.

Concurrent operators are supported, but no limit is enforced. The number of concurrent users is only limited by the system resources.

Commands disabled in FIPS mode

The device in FIPS mode does not support the following commands:

- **enable password-display**
- **enable strict-password-enforcement**

  NOTE
  Strict password enforcement is enabled by default when the device is in FIPS mode and it cannot be disabled. The password must be at least eight characters long.

- **web-management allow-no-password**
- **telnet server**
- **ip ssh scp disable**
- **ip ssh key-authentication no**

- **ip ssh permit-empty-password yes**
- **web-management http**
- **ip ssh encryption disable-aes-cbc**
- **authentication key-id**

A device in FIPS mode does not support TFTP commands, including:

- **copy tftp flash** *ip*
- **boot system tftp** *ip file*
- **ip ssh pub-key-file tftp** *ip* {*file* | *pubkey*}
- **ip ssl certificate-data-file tftp** *ip file*
- **ip ssl private-key file tftp** *tftp file*

The following JITC command is not supported because JITC is disabled by default in FIPS mode:

- **jitc enable**

Hidden files in FIPS mode

Hidden files are not displayed when the device is in FIPS mode. Hidden files are displayed only when the device is in non-FIPS mode.

Cryptographic algorithms in FIPS mode

The device in FIPS mode supports the following FIPS 140-2-approved cryptographic algorithms:

- Advanced Encryption Algorithm (AES)
- Secure Hash Algorithm (SHA) (including variants the module supports: SHA-256, SHA-384, and SHA-512)
- Keyed-Hash Message Authentication Code (HMAC)
- Deterministic Random Bit Generator (DRBG)
- Rivest, Shamir, and Adleman public key encryption algorithm (RSA)
- Elliptic curve Digital Signature Algorithm (ECDSA)
- Key-Based Key Derivation Function (KBKDF)
- SNMPv3
- IKEv2 KDF SP800-135

Allowed exceptions include:

- RSA Key Wrapping
- Diffie-Hellman (DH)
- Message Digest 5 (MD5)
- Hash Message Authentication Codes - Message Digest 5 (HMAC-MD5) as used in RADIUS
- Non-Deterministic Random Number Generator (NDRNG)
- SSHv2 Key Derivation Function (KDF)

The device in FIPS mode does not support the following cryptographic algorithms:

- RC4

- 3-DES

SSH

To enable SSH, you must generate RSA encryption keys using the following command.

```
device(config)# crypto key generate rsa  modulus 2048
device(config)#
Creating RSA key pair, please wait...
RSA Key pair is successfully created
```

To confirm that SSH is enabled, use the **show ip ssh** command.

```
device# show ip ssh
No SSH sessions are currently established
SSH-v2.0 enabled; hostkey: RSA(2048)
```

NOTE
If you zeroize the RSA keys, the SSH server is disabled.

Zeroize the keys as shown in the following example.

```
device(config)# crypto key zeroize  rsa
RSA Key pair is successfully deleted
```

Use the **show ip ssh** command to confirm that SSH is disabled.

```
device(config)# show ip ssh
No SSH sessions are currently established
SSH-v2.0 disabled
```

To establish a connection from the client side, enable a local user and set a user password. The following rules apply to passwords:

1. Passwords must be at least eight characters long.
2. Passwords must consist of characters from three or more of these character classes: uppercase, lowercase, numerical, ASCII non-alphanumeric.
   a. The password must not begin with the only uppercase character in the password.
   b. The password must not end with the only numeric character in the password.

   NOTE
   In CC mode, after three incorrect attempts to enter the password, user access is disabled until login recovery time is reached. The default recovery time is three seconds.

To enable a user, enter commands similar to the following.

```
device# configure terminal
device(config)# enable password-min-length 9
device(config)# user test password tesT123$$
```

The example enables the user "test" and defines the minimum password length as 9 characters. It then sets the user password to "tesT123$$".

NOTE
The user configured in the previous example, "test," is a crypto officer who will be able to modify or delete the configuration. To create a read-only user, use a command similar to the following example thta includes the keywords **privilege 5**.

```
device(config)# user test privilege 5 password tesT123$$
```

Login

When a user tries to log in with correct username and password, the login is successful. The following syslog message is generated for a successful login attempt:

```
SYSLOG: <14> Dec 18 09:03:26 Device Security: SSH login by admin from src IP 15.15.15.1 from src MAC
0200.8801.8132 to USER EXEC mode using RSA as Server Host Key.
```

Logout

When the user "admin" closes the session from the TOE, the session is disconnected. The following syslog message is generated for a successful logout attempt:

```
SYSLOG: <14> Dec 18 09:09:11 Device Security: SSH logout by admin from src IP 15.15.15.1 from src MAC
0200.8801.8132 from USER EXEC mode using RSA as Server Host Key.
```

Failed login attempts

By default, the user is disabled after three failed attempts to login. Each time a user connects with a wrong password, a syslog message is displayed. The following example indicates three unsuccessful login attempts.

```
SYSLOG: <14> Dec 18 09:18:14 Device Security: SSH access by user admin from src IP 15.15.15.1 rejected, 1
attempt(s)

SYSLOG: <14> Dec 18 09:18:15 Device Security: SSH access by user admin from src IP 15.15.15.1 rejected, 2
attempt(s)

SYSLOG: <14> Dec 18 09:18:16 Device Security: SSH access by user admin from src IP 15.15.15.1 rejected, 3
attempt(s)
```

The number of attempts and the time for which the user is disabled is configurable. Use the **enable user disable-on-login-failure** command with appropriate parameters to configure the number of login attempts before a user is disabled and the amount of time the system is blocked before the user is allowed to attempt login again.

The following example allows four failed login attempts before the user is disabled and the recovery time of five seconds begins.

```
device# configure terminal
device(config)# enable user disable-on-login-failure 4 login-recovery-time in-secs 5
```

Syntax: [no] enable user { **disable-on-login-failure** [*invalid-attempts* **login-recovery-time** { **in-hours** | **in-mins** | **in-secs** } *recovery-time*] }

NOTE
By default, the user is allowed three login attempts. In CC mode, the default recovery time for re-enabling user accounts is three seconds.

Login attempts can be any decimal value from 1 through 10.

Login recovery time can be specified in hours, minutes, or seconds.

NOTE
You must configure users before enabling the aaa console; otherwise, you may be logged off and locked out of the system.

SSH clients

SSH clients must be FIPS 186-3-compliant. You can use an SSH client that is equivalent to OpenSSH v7.5 or later.

Usernames and SSH public key authentication

The device stores or uses the username that is provided by the SSH client when public-key authentication is used. Therefore, the username is mentioned in the login and logout syslogs.

The FastIron devices save the username from the public-key authentication request. The username is used in the login and logout syslogs. When FIPS mode is operational, the FastIron device uses the username to match against the username attached to the SSH client public key stored on the device. If the two usernames do not match, the authentication request is denied.

Implementation

The client public key file format allows for a username to be provided in the "Subject" field the SSH2 public key. Additional private headers can be used. There are three privilege levels: 0 READ-WRITE/ADMINISTRATOR, 4 PORT-CONFIG, and 5 READ-ONLY. The following public key example shows the two headers that are used by the device. No continuation lines are allowed in the file for these headers.

```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "rsa-key-20121206"
Subject: brcd
x-device-privilege-level: 0
AAAAB3NzaC1yc2EAAAABJQAAAQEAkwiApY1x4T/DHII5JzR2OgqcF5vjlubNcvSE
UjkGmiRBDSOicjxS0ZLm1b2xFpVzw8XxSSy8cxvntfs5ortOt80QzynqgL+H2zJa
Lb4Qbu6/1vakJbPb/VUJE66Zezh0c8mze6zTbiP4iQ/Wn2lxpSmlS5cdowmFlZ7B
97xcagJIBl+7JKuvj8P+85ESUf2/pcrogqx7gdr1IpP2nev5s4xwCWFGtr2R/yMF
Q9h0xLcc4A7vLTDuY/h1GzLdICgtNYdqpUhpw+w0DkTKbQuDPd0gkwHkoFwg85lE
4VCDevdC/DeOCNJjNp9NbVD+SW6uL4NymmV7/i0YbPyl3gTESQ==
---- END SSH2 PUBLIC KEY ----
```

After decoding the base64 encoded public keys to binary format, a SHA256 hash of the binary format key is created. This hash is saved to memory. The hash is verified as unique compared to all the hashes of client public keys that have already been parsed. Non-empty usernames are also verified as unique compared to the usernames already parsed in the public key. Access is denied if the usernames are mismatched.

The username has the following restrictions:

- The username cannot contain control characters, spaces, ", ?, |, or characters above ASCII code 0x7F.
- The username must be less than or equal to 48 characters.
- The username must be specified with the public key for that key to allow access. The user must specify a non-empty username in the login request.

Restrictions

- No EXEC authorization from AAA: No EXEC authorization through the AAA server is available because the privilege level is obtained from the public key file private header field (x-device-privilege-level) as shown in the public key example in Implementation on page 23.
- No EXEC accounting from AAA
- No system accounting from AAA

Protocol changes in FIPS mode

The following table lists the protocols that undergo changes while the device is in FIPS mode with the default policy applied.

TABLE 2 Protocol changes

Protocols/ Algorithms	Supported in FIPS mode	Supported in Non-FIPS mode	For more information on individual protocol changes, refer to the following sections:
BGP	Yes	Yes	BGP on page 24
HTTP	No	Yes	HTTP on page 24
HTTPS client	Yes, with limitations (HTTPS client only)	Yes	HTTPS on page 24
IPsec	Yes, with limitations	Yes	IKEv2/IPsec on page 25
MD5 password encryption	Yes, with limitations (MD5 password encryption is supported for AAA and RADIUS).	Yes	RADIUS protocol in CC mode on page 43
NTP	Yes, with limitations (supports SHA1 only)	Yes (supports SHA1 and MD5 hash only)	NTP on page 31
OSPFv2	Yes	Yes	OSPFv2 on page 26
PKI	Yes	Yes	PKI on page 27
Proprietary 2-way encryption algorithms	No	Yes	Proprietary 2-way encryption algorithms on page 28
RADIUS	Yes	Yes	RADIUS protocol in FIPS mode on page 28
SCP	Yes	Yes	SCP on page 28
SNMP	Yes, with limitations	Yes	SNMP on page 29
SSHv2	Yes, with limitations	Yes	SSHv2 on page 30
Telnet	No	Yes	Telnet on page 30
TFTP	No	Yes	TFTP on page 30

BGP

Border Gateway Protocol (BGP) allows peer-to-peer authentication or client-to-server authentication.

To authorize an authentication, use a command such as the following to configure shared secret keys for BGP:

```
device(config-bgp-router)# neighbor 192.168.1.2 password P@$$w0rd
```

Syntax: [no] neighbor {*ip-addr* | *peer-group-name*} **password** *string*

HTTP

HTTP is not supported on FastIron devices in FIPS mode.

HTTPS

The following HTTPS configurations are affected in FIPS mode:

The FIPS 140-2 cipher suites consist of the following algorithms:

- Triple-DES (FIPS 46-3) or AES (FIPS 197) for symmetric key encryption and decryption.
- Secure Hash Standard (SHA-256, SHA-384, and SHA-512) (FIPS 180-2) for hashing).
- HMAC (FIPS 198) for keyed hash
- Random number generator Hash DRBG (NIST SP800-90).
- Diffie-Hellman, EC Diffie-Hellman, or Key Wrapping using RSA keys for key establishment
- DSA (FIPS 186-2 with Change Notice 1), RSA (PKCS #1 v2.1), or ECDSA (ANSI X9.62) for signature generation and verification.

The following cipher suites are allowed in FIPS mode:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

TLS implementation in FastIron devices

By default, all TLS versions are supported on devices that act as an HTTPS server.

For devices that act as an SSL server or HTTPS server, the default connection is with TLS1.2. For devices that acts as an SSL client or syslog, OpenFlow, or secure AAA client, during session negotiation, the TLS version is decided based on the server support.

You can configure the minimum TLS version on FastIrondevices using the **ip ssl min-version**{ **tls_1_1** | **tls_1_2** } command.

The following example configures TLS 1.2.

```
device(config)# ip ssl min-version
  tls_1_1   TLS Version 1.1
  tls_1_2   TLS Version 1.2
device(config)# ip ssl min-version tls_1_2
```

Use the **show ip ssl** command to identify the TLS version that is configured on the device.

```
device(config)# show ip ssl
Session Protocol  Source IP      Source Port  Remote IP     Remote Port
1       TLS_1_2   10.20.157.102  634          10.25.105.201 60892
```

IKEv2/IPsec

The ICX7400-SERVICE-MOD interface module supports creation of virtual private network (VPN) using the IPsec protocol. The IKEv2 protocol is used to negotiate the IPsec service parameters for the VPN.

IPsec critical security parameters

The following parameters make up the IPsec critical security parameters.

- IKEv2 DH Group-14 Private Key 2048 bit MODP
- IKEv2 DH Group-14 Shared Secret 2048 bit MODP
- IKEv2 DH Group-14 Public Key 2048 bit MODP
- IKEv2 ECDH Group-19 Private Key (P-256)
- IKEv2 ECDH Group-19 Shared Secret (P-256)

- IKEv2 ECDH Group-19 Public Key (P-256)
- IKEv2 ECDH Group-20 Private Key (P-384)
- IKEv2 ECDH Group-20 Shared Secret (P-384)
- IKEv2 ECDH Group-20 Public Key (P-384)
- IKEv2 ECDSA Private Key (P-256)
- IKEv2 ECDSA Private Key (P-384)
- IKEv2 ECDSA Public Key (P-256)
- IKEv2 ECDSA Public Key (P-384)
- IKEv2 Encrypt/Decrypt Key
- IKEv2/IPSec Authentication Key
- IKEv2 KDF State
- IKEv2 Pre-Shared Key (PSK)

OSPFv2

OSPFv2 protocol uses IPsec with IP ESP and HMAC-SHA-196, and is allowed in FIPS mode. OSPF allows peer-to-peer authentication or client-to-server authentication. You can apply OSPFv2 IPsec authentication at the interface level using commands similar to the following example.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ipv6 ospf authentication ipsec spi 256 esp sha1
12345678901234567890123456789012345678901234567890
```

Use the following syntax to configure the IPsec-SHA1 key:

Syntax: ipv6 ospf authentication ipsec spi *spinum* **esp sha1** *key*

Use the following syntax to remove the IPsec-SHA1 key configuration:

Syntax: no ipv6 ospf authentication ipsec spi *spinum* **esp sha1 encrypt** *key*

The key is retained even when unconfigured and appears in the running configuration as shown in the following example.

```
ICX7750-48C Router# show running-config interface ethernet 1/1/1
interface Ethernet 1/1/1
ipv6 address 2002::20/64
ipv6 ospf area 1
ipv6 ospf authentication ipsec spi 500 esp sha1 encrypt
$Wnw4M09tWVd7UVp8ODNPbVlXe1FafDgzT21ZV3tRWnw4M09tWVd7UQ==
```

Related OSPFv2 authentication command options are available as shown in the following syntax statements.

Syntax: ip ospf authentication { **hmac-sha-1** | **hmac-sha-256** } **key-id** *key-id-value* **key** *key-string*

Syntax: no ip ospf authentication { **hmac-sha-1** | **hmac-sha-256** } **key-id** *key-id-value* **key** *key-string*

Syntax: ip ospf authentication keychain *keychain-name*

Syntax: no ip ospf authentication keychain *keychain-name*

OSPFv3

The OSPFv3 protocol uses IPsec with IP ESP and HMAC-SHA-196 and HMAC-SHA-256 and is allowed in FIPS mode. OSPF allows peer-to-peer authentication or client-to-server authentication. You can apply OSPFv3 IPsec authentication at the interface level using commands similar to the following example.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ipv6 ospf authentication ipsec spi 256 esp sha1
1234567890123456789012345678901234567890
```

Use the following syntax to configure the IPsec-SHA1 key:

Syntax: ipv6 ospf authentication ipsec spi *spinum* **esp sha1** *key*

Use the following syntax to remove the IPsec-SHA1 key configuration:

Syntax: no ipv6 ospf authentication ipsec spi *spinum* **esp sha1 encrypt** *key*

The key is retained when unconfigured and appears in the running configuration as shown in the following example.

```
ICX7750-48C Router# show running-config interface ethernet 1/1/1
interface Ethernet 1/1/1
ipv6 address 2002::20/64
ipv6 ospf area 1
ipv6 ospf authentication ipsec spi 500 esp sha1 encrypt
$Wnw4M09tWVd7UVp8ODNPbVlXe1FafDgzT21ZV3tRWnw4M09tWVd7UQ==key
```

Related OSPFv3 authentication command options are available as shown in the following syntax statements.

Syntax: ipv6 ospf authentication { **hmac-sha-1** | **hmac-sha-256** } **key-id** *key-id-value* **key** *key-string*

Syntax: no ipv6 ospf authentication { **hmac-sha-1** | **hmac-sha-256** } **key-id** *key-id-value* **key** *key-string*

Syntax: ipv6 ospf authentication keychain *keychain-name*

Syntax: no ipv6 ospf authentication keychain *keychain-name*

The following commands are entered in IPv6 OSPF router configuration mode.

Syntax: area *area-id* **authentication** { **hmac-sha-1** | **hmac-sha-256** } **key-id** *key-id-value* **key** *key-string*

Syntax: no area *area-id* **authentication** { **hmac-sha-1** | **hmac-sha-256** } **key-id** *key-id-value* **key** *key-string*

Syntax: area *area-id* **authentication keychain** *keychain-name*

Syntax: no area *area-id* **authentication keychain** *keychain-name*

PKI

Public Key Infrastructure (PKI) operates on the module that allows automated certificate authentication during the IKEv2 session setup. IKEv2 sessions are established on the ICX7400-SERVICE-MOD interface module.

The following parameters make up the PKI critical security parameters (CSPs):

- PKI SCEP Enrollment RSA 2048-bit Private Key
- PKI SCEP Enrollment RSA 2048-bit Public Key

NOTE
OCSP does not use an RSA private key.

Proprietary 2-way encryption algorithms

The routing protocols OSPFv2 and BGP, and the management protocol SNMP save authentications parameters using one of the following two proprietary algorithms:

- Global encoding scheme
- Base 64 encoding scheme

These proprietary algorithms are not supported in FIPS mode. When the default FIPS policy is applied, these authentication parameters are zeroized.

RADIUS protocol in FIPS mode

HMAC-MD5 authentication used in RADIUS is allowed in FIPS mode.

RADIUS allows client-to-server authentication. To authorize an authentication, use commands such as the following to configure shared secret keys for RADIUS:

```
device(config)# radius-server host 1.2.3.4 auth-port < auth-port > 1812 acct-port 1813 default
key 1 Example01
```

[no] radius-server host { *ip-addr* | *server-name* } [**auth-port** *number* [**acct-port** *number*] [**authentication-only** | **accounting-only** | **default**] [**key [0 1 2]** *string* [**dot1x**]]]

SCP

Using SCP to copy a digital certificate to a FastIron device

To copy a digital certificate and private keys to a FastIron device using secure copy, enter commands in the following format:

```
C:> scp < Cert_file > user@< ip address >:file: < cert >.pem
C:> scp < Private_key_file > user@< ip address > :file: < private >.pem
```

Syntax: scp *source_filename* **user@***ip_address* **:file:** *cert_filename***.pem**

Syntax: scp *private_key_file* **user@***ip_address* **:file:** *private_filename***.pem**

The *source_filename* variable is the filename of the digital certificate that the user is copying to the device.

The *private_key_file* is the private key that the user is copying to the device.

The *ip-address* variable is the IP address of the ICX device to which the digital certificate file is downloaded.

The keyword **:file:** is required.

The *cert_filename* variable is the filename to which the digital certificate is copied.

The *private_filename* variable is the filename to which the private key is copied.

The destination filename must have the suffix **.pem**; otherwise, the file will not be copied to the device.

SNMP

In the FIPS mode of operation, the device uses the existing SNMP configuration. However, MIB objects related to keys and passwords output NULL or a 0 value.

SNMP allows peer-to-peer authentication or client-to-server authentication. To authorize an authentication, use commands such as the following to configure shared secret keys for SNMP:

```
device(config)# snmp-server group admingrp v3 priv read all write all notify all
```

Syntax: [no] snmp-server group *groupname* { **v3 auth** | **noauth** | **priv** [**access** *standard-ACL-id*] [**read** *viewstring* | **write** *viewstring* | **notify** *viewstring*] }

```
device(config)# snmp-server user adminuser admingrp v3 encrypted auth md5
c1c510d4f3c6bec15ff14f9c0f3ec120 priv encrypted aes
b0c4c6c05cded8cfe3a335299347c71b
```

Syntax: [no] snmp-server user *name groupname* **v3** [[**access** *standard-ACL-id*[[**encrypted**] [**auth** | **sha** *sha-password*] [**priv** [**encrypted**] **aes** *aes-password-key*]]]]

SNMP CSP objects

The following SNMP MIB objects represent the critical security parameter (CSP) entities that are restricted in FIPS mode.

Enterprise MIB objects:

- snRadiusKey
- snRadiusServerRowKey
- snVrrpIfAuthPassword
- snAgGblPassword
- snAgGblReadOnlyCommunity
- snAgGblReadWriteCommunity
- snAgGblTelnetPassword
- snAgentUserAccntPassword
- fdryRadiusServerRowKey
- snOspfIfAuthKey
- snOspfIfMd5AuthKey
- snOspfIf2AuthKey
- snOspfIf2Md5authKey
- snOspfVirtIfAuthKey
- snOspfVirtIfMd5AuthKey
- snOspfIfStatusAuthKey
- snOspfIfStatusMd5AuthKey
- snOspfVirtIfStatusAuthKey
- snOspfVirtIfStatusMd5AuthKey
- snBgp4NeighGenCfgPass
- snVrrpIf2AuthPassword
- snVsrpIfAuthPassword

Standard MIB objects:

- rip2IfConfAuthKey
- vrrpOperAuthKey
- dvmrpInterfaceKey

SSHv2

Secure Shell version 2 (SSHv2) is allowed in FIPS mode.

The following SSH commands are affected when the FastIron device is in FIPS mode:

- The **ip ssh encryption aes-only** command is disabled.
- The **ip ssh key-authentication** command is disabled.
- The **ip ssh permit-empty-password** command is disabled.
- The **ip ssh pub-key-file tftp** command is disabled.
- The **ip ssh scp** command ensures that SCP is enabled to run in FIPS mode. SCP is needed for file communication and the **ip ssh scp disable** command is disabled in FIPS mode and displays the following message:

  ```
  FIPS Compliance: SCP needs to be enabled
  ```

- The **crypto key zeroize** command removes configured SSH keys.

  NOTE
  The following encryption methods are supported in FIPS mode:
  - aes256-ctr
  - aes128-ctr

Use the **show ip ssh config** command to display SSH configuration information.

SSH key generation time is affected by the increased security of authentication and encryption algorithms both in and out of FIPS mode.

Telnet

Telnet is disabled in FIPS mode as part of the default FIPS policy on the device. Attempts to start the Telnet server fail in FIPS mode.

TFTP

The following TFTP commands are disabled and return an error when TFTP operation is not allowed on the device in FIPS mode:

- All **copy tftp** commands
- The **boot system tftp** *ip-address filename* command

The following TFTP commands are disabled. Use SCP commands with equivalent functionality instead. Refer to SCP on page 28.

- **ip ssl certificate-data-file tftp** *ip-address certificate-filename*
- **ip ssl private-key-file tftp** *ip-address key-filename*
- **ip ssh pub-key-file tftp** *ip-address key-filename*

NTP

FastIron FIPS devices support Network Time Protocol (NTP) using SHA1.

```
device (config-ntp)# authentication-key key-id 1 sha1
```

Syntax: [no] authentication-key key-id *decimal* **sha1**

NOTE
FIPS mode and CC mode do not support MD5 hash algorithm.

System reset and boot up in FIPS mode

POST testing takes place as the device progresses through the boot sequence.

The following actions and limitations take effect when the device is operationally in FIPS mode according to the FIPS default policy:

- Boot up from TFTP is disabled.
- The monitor mode memory access command set is disabled. Configure an alternative FIPS policy to the default policy to access the command set. Refer to Modifying the FIPS policy on page 39.
- Boot monitor access during a cold boot is disabled with the exception of the option to access monitor mode during the boot sequence.
- Access to memory test mode is disabled.
- Debug commands are disabled from the application prompt in FIPS mode.

Debugging in FIPS mode

The device reloads automatically when it encounters a system reset and enters FIPS failure state. The cause of failure logs on the console and the device performs a self-reboot.

You can conduct debugging when a flexible FIPS policy is applied on the device. In the event of continuous reload, contact technical support.

Placing the device in FIPS mode

Placing the device in FIPS mode is a multiple-step process that begins with enabling FIPS mode on the device.

This places the device administratively in FIPS mode. To operate the device in FIPS mode, save the configuration, and reboot the device. Always back up the desired configuration to ensure it is saved in the event of a system reset.

General steps to place the device in FIPS mode

Perform the following steps to place the device in FIPS mode.

1. Disable the AAA authentication method **none** option if used in the device configuration.
2. Copy the needed signature files. Refer to step 3 in Performing a FIPS or CC software upgrade on page 17.
3. Perform a FIPS self test to verify the correct signature files were copied. Refer to Performing a FIPS self-test on page 38.

4. Assume Crypto Officer role.
5. Enable FIPS mode. Refer to Enabling FIPS mode on page 32.
6. Zeroize shared secrets and host keys. Refer to Zeroizing shared secrets and host keys on page 34.
7. Save the configuration. Refer to Saving the configuration on page 37.
8. Reload the device. Refer to Reloading the device on page 37.

Enabling FIPS mode

Perform the following steps to enable FIPS mode.

1. Attach a management station (PC or terminal) to the management module serial (console) port using a serial cable.

   When the device is not in a console session, FIPS-related commands return errors.
2. Verify that the device is in non-FIPS mode by using the **fips show** command.

   ```
   device(config)# fips show
   ```

   Syntax: fips show

   The **fips show** command lists the current configuration of the device and can be run in both FIPS mode and non-FIPS modes to establish whether the device is truly in FIPS mode.

   The output of the **fips show** command confirms that the device is in FIPS mode and identifies the device as either administratively or operationally in FIPS mode.

   NOTE
   If the FastIron device is in JITC mode, then you cannot enable FIPS on the device.

   The following example shows the output of the **fips show** command before the **fips enable** command is entered. Administrative status and operational status are off.

   ```
   device(config)# fips show
   FIPS mode: Administrative Status: OFF, Operational Status: OFF
   ```

   If the device is already in administrative FIPS mode, you can modify the FIPS policy. Refer to Modifying the FIPS policy on page 39.

3. Use the **fips enable** command to place the device administratively in FIPS mode.

```
device(config)# fips enable
```

Syntax: [no] fips enable

The following example shows sample output of the **fips enable** command.

> **NOTE**
> Beginning with FastIron 08.0.20a, the RSA key pair is deleted when the FIPS mode is enabled. Use the **crypto key generate** command to generate the RSA key once the device is in FIPS mode.

```
device(config)# fips enable
All keys incompatible with FIPS 140-2 standard will be deleted.

RSA Key pair not found

RSA client Key pair not found
This device is now running in FIPS administrative mode.
At this time you can alter this system's FIPS default security policy
and then enter FIPS operational mode.

Note: Making changes to the default FIPS security policy weakens
the security of the device and makes the device non-compliant with
FIPS 140-2 Level 1
The default security policy defined in the FIPS
Security Policy Document ensures that the device complies with all
FIPS 140-2 specifications. Commands to alter the default security policy
are available to the crypto-officer; however, Ruckus does not recommend
making changes to the default security policy at any time.
====================================

To enter FIPS mode, complete the following steps:
1. Install the signature file now if not already done. Failure to install
   signature or wrong signature file can cause continuous resets.
   Also, optionally, configure FIPS policy commands that meets your network
   requirements. You must explicitly configure the following services if you want
   to use them when the device is operational in FIPS mode:
FIPS: SCP is already enabled

      - Allow TFTP access.
          Current status: Enabled
      - Allow SNMP Access to the Critical Security Parameter (CSP) MIB objects.
          Current status : Disabled
      - Allow access to all commands within the monitor mode.
          Current status: Enabled
      - Retention of shared secret keys for all protocols and the host passwords.
          Current status: Clear
      - Retention of SSH DSA host keys.
          Current status: Clear
      - Retention of SSH RSA host keys and HTTPS certificate.
          Current status: Clear

2. Enter the "fips zeroize all" command, which zeroes out the shared secrets
    used by various networking protocols, including the host access passwords,
    SSH and HTTPS host-keys with the digital signature based on the configured
    FIPS Security Policy.
3. Save the running configuration.
4. Reload the device.
5. Do not press "b" during reload, else FIPS or CC will not be enabled properly.
6. Enter the "fips show" command to verify that the device entered
    FIPS or CC operational mode.
====================================

The system will disable the following services or commands after reload:
1. Telnet server will be disabled. The "telnet server" command will be removed.
2. SCP will be enabled. The "ip ssh scp disable" command will be removed.
3. HTTP server will be disabled. The "web-management http" command will be removed.
4. HTTPS server will change as follows:
```

```
      -SSL 3.0 will be disabled.
      -TLS version 1.0 and greater will be used.
      -RC4 cipher will be disabled.
      -Passwords will be required; the "web-management allow-no-password"
        command will be removed.
5. SNMP server will change as follows:
        -SNMP support for v1 and v2 versions will be disabled.
        -For SNMPv3 version md5 key and DES privacy password will be disabled.
6. NTP md5 authentication will be diabled.
Passwords/Keys which dont comply FIPS standards will be removed on reload.
Please see FIPS config guide for complete details.
```

4. You can verify the status of the device as administratively in FIPS mode by using the **fips show** command.

   The following example shows the output of the **fips show** command on a FastIron device after the **fips enable** command is entered and administrative status is on and operational status is off:

   The following example shows the output of the **fips show** command on a CER devices after the fips enable command is entered and administrative status is on and operational status is off:

```
device# fips show
Cryptographic Module Version: BRCD-IP-CRYPTO-VER-4.0
FIPS mode: Administrative status ON: Operational status OFF
Common-Criteria: Administrative status OFF: Operational status OFF
System Specific
OS monitor access status is: Disabled

Management Protocol Specific:
Telnet server: Disabled
Telnet client: Disabled
TFTP client: Disabled
HTTPS SSL 3.0: Disabled
SNMP Access to security objects: Disabled

Critical security Parameter updates across FIPS boundary:
Protocol Shared secret and host passwords: Clear
Password Display: Disabled

HTTPS RSA Host Keys and Signature: Clear
SSH DSA Host keys: Clear
SSH RSA Host keys: Clear
```

Zeroizing shared secrets and host keys

After you have reviewed the FIPS policy, use the **fips zeroize** command to zeroize the shared secrets and host keys used by various networking protocols.

```
device# fips zeroize all
```

Syntax: fips zeroize { **all** | **shared-secret** | **host-keys** | **pki-certs** }

The **all** option zeroizes all shared secrets and host keys. The **shared-secret** option zeroizes shared secret keys only. The **host-keys** option zeroizes host keys only.

For example, entering **fips zeroize shared-secret** zeroizes only the shared secret keys of various networking protocols and host access passwords.

> **NOTE**
> The **fips zeroize** command may cause operational failure within networking protocols using shared secrets and should be used with careful consideration.

The default FIPS policy calls for the zeroization of all keys using the **fips zeroize all** command option. When you apply a less strict FIPS policy than the default, zeroize at your discretion.

NOTE
The **fips zeroize all** command zeroizes all keys irrespective of the configured FIPS policy.

The following table lists the various keys used in the system that are zeroized in compliance with FIPS.

TABLE 3 Key zeroization

Keys used	Command option handling
DH private keys	Host-keys
FCSP Challenge Handshake Authentication Protocol (CHAP) secret	Host-keys
SSH session key	Host-keys
SSH RSA private key	Host-keys
RNG seed key	N/A
Passwords	Shared-secret
TLS private key	Host-keys
TLS pre-master secret	Host-keys
TLS session key	Host-keys
TLS authentication key	Host-keys
RADIUS secret	Shared-secret
Authentication passwords for various networking protocols	Shared-secret

TABLE 4 Key zeroization for IKEv2 or IPsec on interface module

Keys used	Command option handling
Password for MM	NA
ECDH shared secret	NA
ECDSA private key	SP800-90
ECDH private	SP800-90
IKE Encrypt/Decrypt	IKE v2 KDF
IKE authentication key	IKE v2 KDF
ESP Encrypt/Decrypt	IKE v2 KDF
ESP authentication key	IKE v2 KDF
IKE KDF state	IKE v2 PRF
Pre-Shared Key (PSK)	RFC 5996
DRBG state	SP800-90
Entropy data	NDRNG

Configuring user authentication

Ruckus FastIron devices support role-based authentication. A device can perform authentication and authorization (role selection) using RADIUS and local configuration database. FastIron devices also support multiple authentication methods for each service.

To implement one or more authentication methods for securing access to the device, you configure authentication-method lists that set the order in which the authentication methods are consulted.

In an authentication-method list, you specify the access method (SSHv2, SNMP, and so on) and the order in which the device tries one or more of the following authentication methods:

- Line password authentication
- Enable password authentication
- Local user authentication
- RADIUS authentication

When a list is configured, the device attempts the first method listed to provide authentication. If that method is not available (for example, the device cannot reach a RADIUS server), the device tries the next method until a method in the list is available or all methods have been tried.

FastIron devices allow multiple concurrent operators through SSHv2 and the console. One operator's configuration changes can overwrite the changes of another operator.

Line password authentication

The password authentication method uses the Telnet password to authenticate an operator. To use line authentication, a Crypto-officer must set the Telnet password.

NOTE
When operating in the FIPS approved mode, Telnet is disabled and line authentication is not available.

Enable password authentication

The enable method uses a password corresponding to each role to authenticate an operator. An operator must enter the read-only password to select the User role. An operator enters the port-config password to the Port Configuration Administrator role. An operator enters the super-user password to select the Crypto-officer Role.

To use enable authentication method, a Crypto-officer must set the password for each privilege level.

Local user authentication

The local method of authentication uses a password associated with a user name to authenticate an operator. An operator enters a user name and corresponding password. The FastIron device assigns the role associated with the user name to the operator when authentication is successful.

To use local authentication, a Crypto-officer must define user accounts. The definition includes a user name, password, and privilege level (which determines the role).

RADIUS authentication

The RADIUS method uses one or more RADIUS servers to verify user names and passwords. The FastIron device prompts an operator for user name and password. The device sends the user name and password to the RADIUS server. Upon successful authentication, the RADIUS server returns the operator's privilege level, which determines the operator's role. If a RADIUS server does not respond, the FastIron device sends the user name and password information to the next configured RADIUS server.

FastIron series devices support additional command authorization with RADIUS authentication. The following events occur when RADIUS command authorization takes place.

1. A user previously authenticated by a RADIUS server enters a command on the FastIron device.

2. The FastIron device looks at its configuration to see if the command is at a privilege level that requires RADIUS command authorization.
3. If the command belongs to a privilege level that requires authorization, the FastIron device looks at the list of commands returned to it when RADIUS server authenticated the user.

After RADIUS authentication takes place, the command list resides on the FastIron device. The device does not consult the RADIUS server again once the operator has been authenticated. This means that any changes made to the operator's command list on the RADIUS server are not reflected until the next time the RADIUS server authenticates the operator, and the server sends a new command list to the FastIron device.

NOTE
Radius over TLS is supported in FIPS mode.

To use RADIUS authentication, a Crypto-officer must configure RADIUS server settings along with authentication and authorization settings.

Saving the configuration

NOTE
Keep a backup copy of the startup configuration in the event of system reset.

After zeroizing, use the **write memory** command to save the configuration.

```
device(config)# write memory
```

Reloading the device

NOTE
Before upgrading to a new image in FIPS mode, ensure that the corresponding signature file is available in the flash memory.

After you have saved the configuration, reload the device using the **reload** command:

```
device# reload
```

Various tests, including Power-On Self Tests (POSTs) and Known Answer Tests (KATs), are run by the FastIron device during reload, during the transition between non-FIPS mode and FIPS mode.

POSTs check for the consistency of the FIPS-approved algorithms implemented on the device.

KATs are used to exercise various features of FIPS-approved algorithms.

All interfaces on the device are down until the tests are completed successfully.

Possible POST failure messages indicating that the device did not pass the tests successfully include the following messages:

```
Crypto module initialization and KNown Answer Test (KAT) failed with reason:(Error
Code 0x80000000)'CKR_VENDOR_DEFINED'

FIPS: Primary image verification failed

FIPS: Secondary image verification failed
```

If there is a failure while the POSTs are being run, the device reboots. Monitor mode can be accessed to troubleshoot the issue.

After all tests are completed successfully, the device reloads in FIPS mode and FIPS mode is successfully enabled and operational on the FastIron device.

You can verify the status of the device as operationally in FIPS mode by using the **fips show** command.

```
device(config)# fips show
```

The following example shows **fips show** command after the device reloads successfully in the default strict FIPS mode. Administrative status and operational status are on.

```
device# fips show
Cryptographic Module Version: BRCD-IP-CRYPTO-VER-4.0
FIPS mode: Administrative status ON: Operational status ON
Common-Criteria: Administrative status OFF: Operational status OFF
System Specific
OS monitor access status is: Disabled

Management Protocol Specific:
Telnet server: Disabled
Telnet client: Disabled
TFTP client: Disabled
HTTPS SSL 3.0: Disabled
SNMP Access to security objects: Disabled

Critical security Parameter updates across FIPS boundary:
Protocol Shared secret and host passwords: Clear
Password Display: Disabled

HTTPS RSA Host Keys and Signature: Clear
SSH DSA Host keys: Clear
SSH RSA Host keys: Clear
```

Performing a FIPS self-test

Use the FIPS self-test to verify the sanity of FIPS software.

For more information on the FIPS self-test, refer to Running FIPS self-test on page 40.

NOTE
During FIPS self-test, the CPU usage is high. The **fips self-tests** command should only be executed prior to the device being placed into FIPS operational or administrative modes. Execution of the **FIPS self-tests** command in FIPS operational or administrative modes may result in the device rebooting as per the FIPS criteria.

- From the Privileged EXEC level of the CLI on the console, use the **fips self-tests** command to verify that the FIPS Software and Firmware Integrity Test passes.

  Syntax: fips self-tests

- The following example shows the FIPS Software and Firmware Integrity Test as passed:

```
device# fips self-test
Running FIPS Power On Self Tests and Software/Firmware Integrity Test.
FIPS Power On Self Tests and Software/Firmware Integrity tests successful.
.....<output truncated>.......
```

If the test fails, make sure that the correct signature file was copied for the correct image file and version, and recopy as needed.

NOTE
The FIPS self-test must pass before saving the configuration and reloading the device.

Modifying the FIPS policy

After the device is administratively in FIPS mode, you can modify the default FIPS policy.

> **NOTE**
> Making changes to the default FIPS policy on the device is not recommended and weakens the security of the device. Any modification of the default FIPS policy places the device in a state that is not in compliance with FIPS 140-2.

The output of the **fips enable** command displays which protocols that constitute the FIPS policy are set in compliance with FIPS standards by default and can be adjusted to set a more flexible policy. The remaining protocols that constitute the FIPS policy are set to the appropriate status automatically during reload due to the **fips enable** command. The default FIPS policy is detailed in How FIPS Works on page 13.

When you make no changes to the FIPS policy, the default FIPS policy is applied on the device and the device operates in strict FIPS mode upon reload, in full compliance with FIPS 140-2 specifications.

To set a more flexible FIPS policy on the FastIron device, use the following commands as desired to modify the default FIPS policy.

- Allow TFTP access:

```
device(config)# fips policy allow tftp-access
```

Syntax: [no] fips policy allow tftp-access

- Allow SNMP access to the critical security parameter (CSP) MIB objects:

```
device(config)# fips policy allow snmp-csp-access
```

Syntax: [no] fips policy allow snmp-csp-access

- Allow access to monitor mode for debugging both from application and boot prompts:

```
device (config)# fips policy allow monitor-full-access
```

Syntax: [no] fips policy allow monitor-full-access

> **NOTE**
> During an application reset, monitor access is restored to allow debugging.

- Retain the shared secret keys for all protocols and the host passwords:

```
device(config)# fips policy retain shared-secrets
```

Syntax: [no] fips policy retain shared-secrets

- Retain the SSH DSA host keys:

```
device(config)# fips policy retain dsa-host-keys
```

Syntax: [no] fips policy retain dsa-host-keys

- Retain the HTTPS RSA host keys and the HTTPS server digital certificate:

```
device(config)# fips policy retain rsa-host-keys
```

Syntax: [no] fips policy retain rsa-host-keys

Disabling FIPS mode

NOTE
Even after disabling FIPS mode, you should always load the signature file before loading the software image file.

Use the **no fips enable** command to disable FIPS mode on the FastIron device.

```
device(config)# no fips enable
```

After you enter the command, a warning displays that FIPS mode will be disabled.

This command performs the following policy-related operations:

- Enables TFTP access.
- Re-enables SNMP access to critical security parameter (CSP) MIB objects.
- Re-enables SNMPv3 encryption protocol DES for future SNMPv3 user configuration.
- Re-enables access to monitor mode.
- Zeroizes shared secrets, SSH and HTTPS host keys, and the HTTPS certificate based on the configured FIPS policy.

The **no fips enable** command also performs the non-policy-related operation of re-enabling the RC4 cipher for the HTTPS server.

Changes to the running configuration are not saved to the startup configuration; therefore, when the device reloads, it returns to FIPS mode.

Use the **write memory** command to save the running configuration.

Running FIPS self-test

Use the **FIPS self-tests** command either in FIPS mode or non-FIPS mode to run the Known Answer Tests (KATs) and conditional tests on demand in both FIPS mode and non-FIPS mode.

```
device(config)# fips self-tests
```

Syntax: fips self-tests

The following log message is generated when the KAT is completed, but no trap messages are generated because the system is not fully operational.

```
"Crypto module initialization and Known Answer Test (KAT) passed".
```

Common Criteria Certification

Common Criteria Overview

Common Criteria certification for a device enforces a set of security standards and feature limitations on a device to be compliant with the Common Criteria standards, similar to placing the device in FIPS mode. These restrictions are in addition to the requirements of FIPS mode. When the device is placed in Common Criteria mode, several security features that are available in FIPS mode are unavailable on the device. Because Common Criteria mode enforces security restrictions additional to FIPS mode, procedures and information are provided in relation to those for the FIPS mode.

For information about enabling FIPS mode on the device, refer to FIPS Configuration on page 19.

For additional information on features available in both FIPS and CC mode and their configuration, refer to the related FIPS sections.

For information on SSH, refer to the following sections:

- SSH on page 21
- SSH clients on page 23
- Usernames and SSH public key authentication on page 23.

For information on self-tests, refer to Running FIPS self-test on page 40.

NOTE
Common Criteria mode becomes available once a device is FIPS-enabled.

NOTE
To determine if the FastIron device and current software version is Common Criteria-certified, refer to https://www.niap-ccevs.org/CCEVS_Products/pcl.cfm. The Security Targets identified in the Ruckus PCL entries define the scope of features that were evaluated. Refer to the release notes for the software version running on the device to verify that the software is FIPS- and Common Criteria-certified.

NOTE
Administrators must be careful to use features only applicable to the evaluation of interest. Instructions throughout this document to configure IPsec and packet filtering features should be followed only if the evaluation of interest is a VPN Gateway and the administrator is configuring an ICX 7450 w/ VPN module.

The following table summarizes support for Common Criteria protection profiles by FastIron device.

TABLE 5 Common Criteria protection profiles supported by device

Platform	NDcPP20	VPNGWEP21
ICX7150	Yes	No
ICX7250	Yes	No
ICX7450	Yes	Yes[1]
ICX7650	Yes	No
ICX7750	Yes	No

You can enable Common Criteria mode on a device directly from non-FIPS mode, or on a device already in FIPS mode. The following table summarizes the transitions.

TABLE 6 Transition to Common Criteria mode

From	To non-FIPS mode	To FIPS mode	To Common Criteria mode
Non-FIPS mode	Not applicable	Use the **fips enable** command	Use the **fips enable common-criteria** command
FIPS mode	Use the **no fips enable** command	Not applicable	Use the **fips enable common-criteria** command
Common Criteria mode	Use the **no fips enable** or **no fips enable common-criteria** command	Use the following commands in a sequence: 1. **no fips enable** 2. **reload device** 3. **fips enable**	Not applicable

Be advised of the following considerations:

- Disabling FIPS mode from the Common Criteria mode using the **no fips enable** command downgrades the device directly into non-FIPS mode.
- You cannot directly transition from Common Criteria mode to FIPS mode. To transition to FIPS mode, you must disable FIPS mode, reload the device, and then enable FIPS mode.

Features unavailable in Common Criteria mode

Some of the security features that are allowed in FIPS mode are disabled in Common Criteria mode:

- SSHv2: Host and client key generation methods using DSA and the RSA-1024 key size are not supported (only RSA 2048 and higher key sizes are supported). Therefore, the following commands are not supported:
  - **crypto key generation dsa**
  - **crypto key client generation dsa**
  - **crypto key zero dsa**
  - **crypto key client zero dsa**
  - **crypto key gen rsa modulus 1024**
  - **crypto key zero rsa modulus 1024**

[1] VPNGWEP21 is supported only on ICX 7450 devices with an IPsec service module installed.

- TLS and HTTPS: The RSA 1024 key size for SSL or TLS private key generation is not supported (FastIron devices support only 2048 and above key sizes).
- SSH key exchange: The SSH key exchange method DiffieHellmanGroup1Sha1 is not supported. Only DiffieHellmanGroup14Sha1 is supported.
- Web Management: Web Management is not supported.

Features available in Common Criteria mode

The following features are available in Common Criteria mode.

1. Secure Syslog: The secure syslog feature uses TLS or IPsec to securely send the log messages to the log server.
2. Radius Authentication over Secure tunnel: All user authentication uses a TLS or IPsec tunnel to communicate with radius server to authenticate the user.
3. All services and features that are available in FIPS mode and are not disallowed as presented in Features unavailable in Common Criteria mode on page 42 are also available in Common Criteria mode.

Supported algorithms for SSH client

An SSH client can connect to a device in Common Criteria mode only with the AES-128 and AES-256 encryption algorithms.

Supported cipher suites

The following mandatory cipher suites are supported for TLS 1.1 and TLS 1.2 in Common Criteria mode.

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

RADIUS protocol in CC mode

HMAC-MD5 authentication used in RADIUS is allowed in CC mode.

RADIUS allows client-to-server authentication. To authorize an authentication, follow the procedure described in "Configuring the SSL profile for use with logging and RADIUS server hosts."

SCP for Common Criteria

Use SCP as described in the FIPS section SCP on page 28.

Enabling Common Criteria mode

When you enable Common Criteria mode on the device, it enters the Common Criteria Administrative mode. Similar to FIPS, Common Criteria also has administrative and operational modes:

- Common Criteria Administrative mode: Log in to the device console and enable the Common Criteria mode. You can optionally modify the default Common Criteria security policy in this mode.

NOTE
When you execute the command to reload the device, the validation of the software image with the signature file is triggered. If signature verification fails, the device continuously reboots after device reload.

- Common Criteria Operational mode: Transition to Common Criteria operational mode from Common Criteria Administrative mode. After you transition the device to the Operational mode, you must save the configuration and reboot the device.

Entering Common Criteria Administrative mode

Use the following command to enter Common Criteria mode.

```
device(config)# fips enable common-criteria
```

Syntax: [no] fips enable common-criteria

The following example includes the detailed banner that is displayed after you enter the command.

```
device(config)# fips enable common-criteria

All keys incompatible with FIPS 140-2 standard will be deleted.

RSA Key pair not found

RSA Client Key pair is successfully deleted
This device is now running in CC administrative mode.
At this time you can alter this system's CC default security policy
and then enter CC operational mode.

Note: Making changes to the default policy makes the device non-compliant
with CC and FIPS 140-2 Level 1
The default security policy defined in the FIPS
Security Policy Document ensures that the device complies with all
FIPS 140-2 specifications. Commands to alter the default security policy
are available to the crypto-officer; however, Ruckus does not recommend
making changes to the default security policy at any time.
====================================

To enter CC mode, complete the following steps:
1. Optionally, configure FIPS policy commands that meets your network
   requirements. You must explicitly configure the following services if you want to use them when the
device is operational in CC mode:
Note: ip ssl client continues to be in enabled mode if enabled
FIPS: SCP is already enabled

      - Allow TFTP access.
          Current status: Enabled
      - Allow SNMP Access to the Critical Security Parameter (CSP) MIB objects.
          Current status : Disabled
      - Allow access to all commands within the monitor mode.
          Current status: Enabled
      - Retention of shared secret keys for all protocols and the host passwords.
          Current status: Clear
      - Retention of SSH DSA host keys.
          Current status: Clear
      - Retention of SSH RSA host keys and HTTPS certificate.
          Current status: Clear

2. Enter the "fips zeroize all" command, which zeroes out the shared secrets
    used by various networking protocols, including the host access passwords,
    SSH and HTTPS host-keys with the digital signature based on the configured
    FIPS Security Policy. If SSH ReKey Exchange value was not configured then
    the default value of 30Mins and 500MB will be configured
3. Save the running configuration.
4. Reload the device.
5. Do not press "b" during reload, else FIPS or CC will not be enabled properly.
```

```
6. Enter the "fips show" command to verify that the device entered
    FIPS or CC operational mode.
====================================

The system will disable the following services or commands after reload:
1. Telnet server will be disabled. The "telnet server" command will be removed.
2. SCP will be enabled. The "ip ssh scp disable" command will be removed.
3. HTTP server will be disabled. The "web-management http" command will be removed.
4. HTTPS server will be disabled. The "web-management https" command will be removed.
Passwords/Keys which dont comply FIPS standards will be removed on reload.
aaa authentication method must be configured.
Disabling user when invalid password is entered is default in FIPS and above modes.
Default value of login recovery time is 3 secs.
No command sets the login recovery time to 3 secs and disables the user after 3 invalid attempts.
Default values of 3 attempts and 3 secs are not displayed in running config.
Please see FIPS config guide for complete details.
====================================
Additionally, in CC mode, the system will disable the following
services or commands after reload:
UDP Syslog servers will be deleted from configuration(only in the CC operational mode).
DSA keys will be deleted from configuration, and will be disabled .
RSA key sizes will be restricted to 2048 and above in the configuration.
Non-TLS TACACS+ servers will be disabled from configuration.
For SSH Key Exchange, only diffie-hellman-group14 algorithm is allowed.


Note: ip ssl client continues to be in enabled mode if enabled
All keys incompatible with FIPS 140-2 standard will be deleted.

RSA Key pair not found

RSA client Key pair not found
This device is now running in CC administrative mode.
At this time you can alter this system's CC default security policy
and then enter CC operational mode.

Note: Making changes to the default policy makes the device non-compliant
with CC and FIPS 140-2 Level 2, design assurance Level 3
The default security policy defined in the FIPS
Security Policy Document ensures that the device complies with all
FIPS 140-2 specifications. Commands to alter the default security policy
are available to the crypto-officer; however, Ruckus does not recommend
making changes to the default security policy at any time.
====================================

To enter CC mode, complete the following steps:
1. Optionally, configure FIPS policy commands that meets your network requirements. You must explicitly
configure the following services if you
want to use them when the device is operational in CC mode:
FIPS: SCP is already enabled

            - Allow TFTP access.
                      Current status: Disabled
            - Allow SNMP Access to the Critical Security Parameter (CSP) MIB objects.
                      Current status : Disabled
            - Allow access to all commands within the monitor mode.
                      Current status: Disabled
            - Retention of shared secret keys for all protocols and the host passwords.
                      Current status: Clear
            - Retention of SSH DSA host keys.
                      Current status: Clear
            - Retention of SSH RSA host keys and HTTPS certificate.
                      Current status: Clear

2. Enter the "fips zeroize all" command, which zeroes out the shared secrets used by various networking
protocols, including the host access passwords,
SSH and HTTPS host-keys with the digital signature based on the configured FIPS Security Policy.
3. Save the running configuration.
4. Reload the device.
5. Do not press "b" during reload, else FIPS or CC will not be enabled properly.
6. Enter the "fips show" command to verify that the device entered
FIPS or CC operational mode.
```

```
====================================

The system will disable the following services or commands after reload:
1. Telnet server will be disabled. The "telnet server" command will be removed.
2. SCP will be enabled. The "ip ssh scp disable" command will be removed.
3. HTTP server will be disabled. The "web-management http" command will be removed.
4. HTTPS server will be disabled.
************************************
************************************
5. aaa authentication method must be configured.
6. Disabling user when invalid password is entered is default in FIPS and above modes.
7. Default value of login recovery time is 3 secs.
No command sets the login recovery time to 3 secs and disables the user after 3 invalid attempts.
Default values of 3 attempts and 3 secs are not displayed in running config.

**********************************
**********************************

====================================
Additionally, in CC mode, the system will disable the following services or commands after reload:
UDP Syslog servers will be deleted from configuration(only in the CC operational mode).
DSA keys will be deleted from configuration, and will be disabled .
RSA key sizes will be restricted to 2048 and above in the configuration.
Non-TLS TACACS+ servers will be disabled from configuration.
```

General considerations when the device is in the Common Criteria Administrative mode

The following general considerations apply when the device is in the Common Criteria administrative mode (applies only to the VPNGW mode, and IPsec must be used for VPNGW).

- Use RADIUS/UDP over the IPsec tunnel configured for managing the device.
- IPsec stack is not available on the management port.
- Configure the VPN gateway separately since it requires logging into the device.
- Configure the VPN gateway NAT translation separately.
- VPN gateway allows Syslog to use IPsec instead of TLS.
- Extended IKEv2 and extended PKI logging needs to be enabled to log the entire contents of packets associated with establishing a session with an IPsec peer.

  NOTE
  You must copy the signature file before copying the corresponding image file; otherwise, image validation fails.

SSH rekey exchange

In SSH2 implementation, if an SSH session is authenticated and established, the session remains connected until the user closes it or until it is closed after the configured idle time limit. Prolonged usage of the session key negotiated at connection startup poses several security issues and exposes SSH connections to man-in-middle attacks. To safeguard existing SSH connections from security vulnerabilities, new keys should be exchanged frequently.

SSH rekeying is the process of exchanging the session keys at a configured interval, based on a time limit or a data limit for the SSH session. SSH rekeying is triggered when the time limit (maximum minutes) or the data limit has been reached for the session. Rekey can be initiated by either the client or the server. While the key exchange renegotiation is taking place, data does not pass through the SSH connection. The algorithm that was used at connection startup is used during rekey.

In FIPS and CC modes, the SSH rekey feature is enabled by default and cannot be disabled. The default value for time is 30 minutes, and the default limit for data is 500 Mbytes in both FIPS and CC modes. If the rekey configuration is removed in either FIPS or CC mode, the default values are applied. The default values are not displayed in the configuration.

When moving from non-FIPS mode to FIPS/CC mode:, if SSH rekey is enabled in non-FIPS mode, the configured values are applied while moving to FIPS mode. If SSH rekey is not configured in non-FIPS mode, the default values in FIPS and CC mode will be applied.

In transitioning from FIPS or CC mode to non-FIPS mode, the SSH rekey configuration is removed, and the feature is disabled in non-FIPS mode.

SSH rekey configuration notes

- The encryption method must not be modified during the rekey process.
- When rekey configuration has changed, the change has no impact on the existing session until the next rekey exchange for the session occurs.
- When a rekey exchange occurs, the value of data and time for the corresponding SSH session is reset to the configured rekey value.
- SSH sessions established without rekey configuration do not have the rekey functionality.
- When rekey is enabled, the existing SSH session does not have the rekey functionality until the rekey exchange occurs from the other side.
- When the rekey configuration is removed, the default values are applied.

SSH rekey configuration examples

The following example configures rekeying of the outbound SSH session every hour.

```
device# configure terminal
device(config)# ip ssh rekey client time 60
```

The following example configures rekeying on the inbound SSH session whenever 10,000 Kilobytes of data are transmitted.

```
device# configure terminal
device(config)# ip ssh rekey server data 10000
```

The following example resets SSH rekey exchange to default settings (and does not disable the function). The defaults can be restored from either the client or the server side.

```
device# configure terminal
device(config)# no ip ssh rekey client time 60
```

Syntax: ip ssh rekey { **client** | **server** } { **data** *Kbytes* | **time** *minutes* }

Syntax: no ip ssh rekey { **client** | **server** } { **data** *Kbytes* | **time** *minutes* }

CLI banner configuration

FastIron devices can be configured to display a greeting message on users' terminals when they enter the Privileged EXEC CLI level.

Setting a message of the day banner

Use the **banner motd** command to configure the FastIron device to display a message on a user terminal when a Telnet CLI session is established. The banner motd command allows you to define a delimiting character to be used at the beginning and end of the text to be used as the banner. The delimiting character cannot appear in the message and can be any character

except a double quotation mark ("). The dollar sign ($) is used as the delimiting character in the example. The banner text can be up to 4,000 characters long and can contain multiple lines.

For example, to display the message “Welcome to ICX!” when a Telnet CLI session is established, enter the following commands.

```
Device# configure terminal
Device(config)# banner motd $
Enter text message. End with character '$'.
Welcome to ICX!$
Device(config)#
```

Use the **no banner motd** command to remove the banner.

Entering Common Criteria Operational mode

When the device is in Common Criteria Administrative mode, perform the following steps to place the device into Common Criteria Operational mode.

1. Configure the local user accounts as secure and delete non-secure user accounts. A local user account is secure when it has a password with characters from three or more character classes. These character classes are uppercase, lowercase, numeric, and ASCII non-alphanumeric characters.
2. Configure secure logging by setting up the encrypted syslog server. For details, refer to Encrypted syslog servers in Common Criteria mode on page 49.
3. Use the **enable aaa console** command to ensure user authentication during the next reload. This also requires that you have enabled AAA authentication with the **aaa authentication login default** command.
4. Use **logging cli-command** to allow you to log all syntactically valid CLI commands from each user session into the system log.
5. Configure a motd banner using the **banner motd** command. Refer to CLI banner configuration on page 47 for more information.
6. Use the **write memory** command to save the configuration.
7. Use the **reload** command to reload the device.

On successful completion of these steps, the device will be in Common Criteria Operational mode.

NOTE
FIPS self-tests are executed as part of device bootup. If any of the self-tests fails, the device reloads automatically. If there are continuous reloads, please contact Ruckus technical support.

NOTE
Use the **exit** command to terminate a local or remote interactive session.

Displaying Common Criteria information

After you have enabled Common Criteria Administrative mode on the device, you can display the information with the **fips show** command.

After you have enabled Common Criteria operational mode by reloading the device, enter the **fips show** command to verify the operational mode status:

```
device# fips show
FIPS mode: Administrative status ON: Operational status ON
Common-Criteria: Administrative status ON: Operational status ON
System Specific
OS monitor access status is: Disabled
```

```
Management Protocol Specific:
Telnet server: Disabled
Telnet client: Disabled
TFTP client: Disabled
HTTPS SSL 3.0: Disabled
SNMP Access to security objects: Disabled

Critical security Parameter updates across FIPS boundary:
Protocol Shared secret and host passwords: Clear
Password Display: Disabled

HTTPS RSA Host Keys and Signature: Clear
SSH DSA Host keys: Clear
SSH RSA Host keys: Clear
```

Encrypted syslog servers in Common Criteria mode

FastIron devices in any mode send the generated syslog messages in real time to the local log storage on the device and to a syslog server (only if a syslog server is configured and available).

A FastIron device running in Common Criteria operational mode queues the syslog messages if a syslog server is not available or configured for the device. This queue is not related to the local syslog messages store and it is cleared when the syslog messages in the queue are forwarded to the syslog server. The queue cannot hold more than 3,000 syslog messages. On reaching the maximum message limit, the device displays an error message and no further syslog messages are queued.

Parameters that are defined for syslog server connections, such as specifying the hold time for queued messages and traps when the device reloads or switches over, are applicable for encrypted syslog connections as well.

The following table summarizes the transitions to and from Common Criteria mode.

TABLE 7 Syslog server connections during transition to and from Common Criteria mode

From	To non-FIPS mode	To FIPS mode	To Common Criteria Operational mode
Non-FIPS mode	Not applicable	No change. FIPS mode does not support encrypted syslog servers.	Both UDP-based and encrypted syslog server connections are allowed in CC Operational mode.
FIPS mode	No change	Not applicable	Both UDP-based and encrypted syslog server connections are allowed in CC Operational mode.
Common Criteria mode	All the SSL servers are removed. Non-FIPS mode does not support encrypted syslog server connections.	Not allowed. You must disable Common Criteria mode to revert to non-FIPS mode, and then re-enable FIPS mode. FIPS mode does not support encrypted syslog server connections.	Not applicable

AAA servers in Common Criteria mode

Common Criteria mode requires that devices support NDcPP version 2.0. This standard requires the communication of the device with AAA servers to take place over a TLS-encrypted session.

Even though you can configure multiple TLS-encrypted RADIUS servers, only one connection can be active at any time. If another TLS-encrypted RADIUS session is attempted at the same time as the first RADIUS session, the connection attempt is rejected.

When the device is in Common Criteria Operational mode, and the device has been configured for a TLS encrypted RADIUS server for authentication, only one administrator is able to administer the device. In addition, accounting and authorizing using the TLS-encrypted RADIUS server are disabled.

> **NOTE**
> You must configure users before enabling the aaa console; otherwise, you may be logged off and locked out of the system.

After configuring users, enter the **aaa** command and enable the AAA console as shown in the following example.

```
device# configure terminal
device(config)# aaa authentication login default local
device(config)# enable aaa console
```

Modifying the Common Criteria policies to use non-encrypted AAA servers

If required, you can modify the Common Criteria policies to allow AAA servers that do not use TLS encryption to be configured, such as RADIUS servers. When non-encrypted AAA servers are allowed, you cannot configure TLS-encrypted TACACS+ servers on the device.

> **NOTE**
> Modifying the default Common Criteria policy makes the device noncompliant with Common Criteria standards.

To allow any AAA server to work with the device in Common Criteria mode, enter the following command:

```
device# fips policy allow common-criteria aaa-server-any
```

Syntax: [no] fips policy allow common-criteria aaa-server-any

Use the **[no]** form of the command to remove non-encrypted AAA servers. If any non-encrypted AAA servers were available on the device, they are removed when Common Criteria mode is enabled on the device.

Downgrading from Common Criteria mode to non-FIPS mode

Downgrading a device from Common Criteria mode either to FIPS mode or to non-FIPS mode uses the same command. You cannot directly downgrade to FIPS mode. You must first downgrade to non-FIPS mode and then enable FIPS mode using the procedures detailed in the previous chapter.

After the device is placed in non-FIPS mode, you can use SCP to download and initialize an older image. Use the following steps to revert to a non-FIPS-compliant image.

1. Log in to the device by entering your username and password.

2. Disable Common Criteria mode by entering the **no fips enable** or **no fips enable common-criteria** command.
3. Regenerate SSH host keys or other shared secrets as needed for access after reload.
4. To replace the startup configuration with the **no fips enable** configuration, enter the **write memory** command.
5. Reload the configuration by entering the **reload** command.

Commercial Solutions for Classified program

The Commercial Solutions for Classified (CSfC) program was established by the United States government to enable commercial networking applications to be used in layered solutions that protect classified National Security Systems (NSS) data. The CSfC program provides the ability to communicate securely based on commercial standards in a solution. Ruckus, as a networking company, supports CSfC and many of the products listed in the CSfC component list. Ruckus devices must be approved by the CSfC program to be deployed in government or federal networks.

To determine if the FastIron device and current software version are CSfC certified, refer to the following URL:

https://www.nsa.gov/resources/everyone/csfc/components-list/#ipsec-vpn-gateway

Configuring NTP

NTP services are disabled on all interfaces by default. To enable NTP in both NTP client and server mode, enter the **ntp** command in global configuration mode as shown in the following example.

```
Device# configure terminal
Device(conf)# ntp
Device(conf-ntp)#
```

Syntax: [no] ntp

Use the **no** form of the command to disable NTP and remove the NTP configuration. The no ntp command removes all manual and statistical configuration as well as learned associations from NTP neighbors.

By default, no NTP servers are configured. To configure the device in client mode and specify the NTP servers to synchronize the system clock, use the **server** command in NTP configuration mode as shown in the following example.

```
Device(config-ntp)# server < server_IP_address >
```

A maximum of eight NTP servers can be configured. To remove NTP server configuration, use the **no** form of the **server** command.

The following example configures an NTP server with the IP address 129.6.15.30. The configuration generates syslog messages similar to those shown.

```
Device(config-ntp)# server 129.6.15.30
SYSLOG: <14> Dec 18 09:33:57 Device NTP: client association is mobilized for 129.6.15.30.
SYSLOG: <14> Dec 18 09:33:57 Device NTP: The system clock is not synchronized to any time source.
```

As soon as the system time is synchronized with the server, the following syslog messages are displayed.

```
SYSLOG: <14> Dec 18 09:35:01 Device Security: Time is updated by NTP server "129.6.15.30" from
"09:36:18.710 GMT+00 Mon Dec 18 2017 "
to "09:35:01.760 GMT+00 Mon Dec 18 2017 "

SYSLOG: <14> Dec 18 09:35:10 Device NTP: System clock is synchronized to 129.6.15.30.
```

NTP limitations

Consider the following limitations when configuring NTP on FastIron devices:

- A FastIron device cannot operate as the primary time server (Stratum 1). It can only serves as the secondary time server (Stratum 2 to 15).
- The NTP server and client cannot communicate using hostnames.
- NTP is not supported on VRF-enabled interfaces.
- Autokey public key authentication is not supported.
- The NTP version 4 extension fields are not supported. Any packets containing the extension fields are discarded.
- NTP packets in control (6) or private (7) packet mode are not supported. NTP packets with control and private modes are discarded.
- On reboot or switchover, all NTP state information is lost, and time synchronization starts fresh.
- NTP multicast server and client and manycast functionalities are not supported.
- NTP versions 1 and 2 are not supported.
- NTP MIB is not supported.

Configuring PKI

You can create PKI entities for use in certificate authentication. To participate in certificate authentication with a Certificate Authority (CA), PKI entities must be enrolled. Entities can be enrolled through an automatic enrollment process, which allows them to send a certificate signing request (CSR) and to receive the required X.509 certificates from the CA in response. Entities can also be enrolled manually as described in the section "PKI manual import." The following procedure explains how to configure automatic enrollment.

Configuring PKI involves the following tasks:

- Generate a cryptographic key using either the **ec** (elliptical key pair) or the **rsa** key pair option for PKI.
- Create a PKI entity.
- Configure the PKI profile.
- Configure the PKI trustpoint.
- Authenticate the PKI.
- Enroll the PKI.

Perform the following steps to complete these tasks.

1. Create a cryptographic key as shown in the following example.

   The first example generates a key pair using the rsa option for the PKI. The second example generates an elliptical key pair for the PKI.

```
device# configure terminal
device(config)# crypto key generate rsa label < name >

device# configure terminal
device(config)# crypto key generate ec label < name >
```

2. Enter PKI entity configure submode to configure end user parameters.

   NOTE
   PKI entity configuration is used for auto-enrollment only.

   The following example enters configuration submode for the PKI entity named entity1.

```
device (config)# pki entity entity1
device(config-pki-entity-entity1)#
```

3. Configure PKI entity details, including common name, country name, state name, and organization name. Country names use a two-letter abbreviation.

   NOTE
   It is recommended that you use quotes around text strings. Quotes are required when a name includes a space.

   The first example below provides command syntax for entering PKI entity details.

   The second example configures realistic parameters for entity1.

```
device(config-pki-entity-entity1)# common-name < name >
device(config-pki-entity-entity1)# country-name < country-name >
device(config-pki-entity-entity1)# state-name < state-name >
device(config-pki-entity-entity1)# org-unit-name < unit-name >
device(config-pki-entity-entity1)# org-name < org-name >
device(config-pki-entity-entity1)# email-id < email-address >
device(config-pki-entity-entity1)# location < location-name >
device(config-pki-entity-entity1)#
device(config-pki-entity-entity1)# exit

device(config-pki-entity-entity1)# common-name "tester1"
device(config-pki-entity-entity1)# country-name "IN"
device(config-pki-entity-entity1)# state-name "KA"
device(config-pki-entity-entity1)# org-unit-name "FI"
device(config-pki-entity-entity1)# org-name "Ruckus"
device(config-pki-entity-entity1)# email-id "user@ruckus.com"
device(config-pki-entity-entity1)# location "BG"
device(config-pki-entity-entity1)#
device(config-pki-entity-entity1)# exit
```

4. Configure the PKI enrollment profile for use later in the enrollment process, including the following items:
   - Profile name
   - An authentication URL for the CA server where authentication requests are sent (for automatic enrollment only)
   - An enrollment URL for the CA server where enrollment requests are sent (for automatic enrollment only)
   - The challenge password obtained from the CA

   The following example provides profile enrollment syntax.

```
device(config)# pki profile-enrollment < profile-name >
device(config-pki-profile-enrollment-profile1)# authentication-url < URL >
device(config-pki-profile-enrollment-profile1)# authentication-command < command >
device(config-pki-profile-enrollment-profile1)# enrollment-url < URL >
device(config-pki-profile-enrollment-profile1)# password < password >
```

   The following example configures the PKI enrollment profile named profile1.

```
device(config)# pki profile-enrollment profile1
device(config-pki-profile-enrollment-profile1)# authentication-url http://WIN-
N6C3R0LUDAJ.englab.ruckus.com/CertSrv/mscep/mscep.dll
device(config-pki-profile-enrollment-profile1)# authentication-command WIN-
N6C3R0LUDAJ.englab.ruckus.com_englab-WIN-N6C3R0LUDAJ-CA-15
device(config-pki-profile-enrollment-profile1)# enrollment-url http://WIN-
N6C3R0LUDAJ.englab.ruckus.com/CertSrv/mscep/mscep.dll
device(config-pki-profile-enrollment-profile1)# password DB6E1F091AEF0244
device(config-pki-profile-enrollment-profile1)# exit
```

5. Configure the trustpoint name and details, including the following items:
   - The enrollment option (automatic)
   - Enrollment retry-period (1 through 60 minutes)
   - Name of enrollment profile to used in the enrollment process
   - Name of the pre-configured PKI entity to be enrolled
   - Key pair type and label (for key generated previously using crypto commands)
   - Digital fingerprint for rootca (obtained from the rootca certificate)
   - OCSP transport protocol (HTTP) and method (post)

   The following example provides PKI trustpoint command syntax.

```
device(config)# pki trustpoint < trustpoint-name >
device(config-pki-trustpoint-trust1)# auto-enroll
device(config-pki-trustpoint-trust1)# enrollment retry-period < number >
device(config-pki-trustpoint-trust1)# enrollment profile < profile-name >
device(config-pki-trustpoint-trust1)# pki-entity < entity-name >
device(config-pki-trustpoint-trust1)# { eckeypair | rsakeypair } key-label < label >
device(config-pki-trustpoint-trust1)# fingerprint < fingerprint-value >
device(config-pki-trustpoint-trust1)# ocsp http post
device(config-pki-trustpoint-trust1)# exit
```

   The following example configures the PKI trustpoint named trust1.

```
device(config)# pki trustpoint trust1
device(config-pki-trustpoint-trust1)# auto-enroll
device(config-pki-trustpoint-trust1)# enrollment retry-period 2
device(config-pki-trustpoint-trust1)# enrollment profile profile1
device(config-pki-trustpoint-trust1)# pki-entity entity1
device(config-pki-trustpoint-trust1)# eckeypair key-label eckeyAuto
device(config-pki-trustpoint-trust1)# fingerprint 36:0c:92:6e:df:b2:72:eb:59:e8:63:73:2a:98:a8:91:cb:
50:94:d9
device(config-pki-trustpoint-trust1)# ocsp http post
device(config-pki-trustpoint-trust1)# exit
```

6. Authenticate the CA (trust1 in this example) to the FastIron device by obtaining the self-signed certificate from the CA.

   The following example authenticates previously configured trustpoint trust1.

   ```
   device(config)# pki authenticate trust1
   ```

7. Once the trustpoint has been authenticated, enroll the FastIron device with the PKI trustpoint to obtain a local certificate signed by the CA server. The **pki enroll** command sends a CSR request to the CA with the configured keypair and entity values. The CA server signs it and sends back the client certificate for the given trustpoint.

   The following example enrolls the PKI trustpoint trust1.

   ```
   device(config)# pki enroll trust1
   ```

   The FastIron device, once enrolled, requests certificates from the CA for each of its key pairs. The CA sends the response in the form of a local certificate.

The following example creates a PKI entity, configures a PKI enrollment profile, and specifies automatic enrollment as the enrollment method. It then configures an enrollment profile. Next, it creates a trustpoint containing the previously configured enrollment profile and PKI entity. Finally, it authenticates and enrolls the trustpoint.

```
device# configure terminal
device (config)# pki entity entity1
device(config-pki-entity-entity1)# common-name "tester1"
device(config-pki-entity-entity1)# country-name "IN"
device(config-pki-entity-entity1)# state-name "KA"
device(config-pki-entity-entity1)# org-unit-name "FI"
device(config-pki-entity-entity1)# org-name "Ruckus"
device(config-pki-entity-entity1)# email-id "user@ruckus.com"
device(config-pki-entity-entity1)# location "BG"
device(config-pki-entity-entity1)# exit
device(config)# pki profile-enrollment profile1
device(config-pki-profile-enrollment-profile1)# authentication-url http://WIN-N6C3R0LUDAJ.englab.ruckus.com/
CertSrv/mscep/mscep.dll
device(config-pki-profile-enrollment-profile1)# authentication-command WIN-
N6C3R0LUDAJ.englab.ruckus.com_englab-WIN-N6C3R0LUDAJ-CA-15
device(config-pki-profile-enrollment-profile1)# enrollment-url http://WIN-N6C3R0LUDAJ.englab.ruckus.com/
CertSrv/mscep/mscep.dll
device(config-pki-profile-enrollment-profile1)# password DB6E1F091AEF0244
device(config-pki-profile-enrollment-profile1)# exit
device(config)# pki trustpoint trust1
device(config-pki-trustpoint-trust1)# auto-enroll
device(config-pki-trustpoint-trust1)# enrollment retry-period 2
device(config-pki-trustpoint-trust1)# enrollment profile profile1
device(config-pki-trustpoint-trust1)# pki-entity entity1
device(config-pki-trustpoint-trust1)# eckeypair key-label eckeyAuto
device(config-pki-trustpoint-trust1)# fingerprint 36:0c:92:6e:df:b2:72:eb:59:e8:63:73:2a:98:a8:91:cb:
50:94:d9
device(config-pki-trustpoint-trust1)# ocsp http post
device(config-pki-trustpoint-trust1)# exit
device(config)# pki authenticate trust1
device(config)# pki enroll trust1
```

PKI manual import

To import X.509 certificates, you need the following items:

1. Root CA Certificate, which will be used with the configured trustpoint on the TOE.
2. The intermediate CA certificate.
3. The local TOE certificate, which is signed by the RootCA or any intermediate CA of the RootCA.
4. The key for the TOE's local certificate.

The listed certificates and keys should be copied into flash on the TOE.

Perform the following steps to import the certificates manually.

1. Create a trustpoint. Here, the trustpoint corresponds to ROOT CA. Set the fingerprint for the trustpoint (which can be copied from the rootCA certificate).

   The following example provides the syntax for the commands.

   ```
   device# configure terminal
   device(config)# pki trustpoint < trustpointname >
   device(config-pki-trustpointname)# fingerprint < value >
   ```

   The following example creates a trustpoint called abcd and sets the fingerprint to the value copied from the rootCA certificate.

   ```
   device# configure terminal
   device(config)# pki trustpoint abcd
   device(config-pki-trustpoint-abcd)# fingerprint 3C:EA:EC:E6:F1:DD:3B:86:65:DE:58:F4:A2:75:D8:63:6D:
   23:68:40
   device(config-pki-trustpoint-abcd)# exit
   ```

2. Import the key to the PKI database from flash memory of the FastIron device using one of the following command.

   NOTE
   The key file can be a plain text or an encrypted file (encryption is recommended). Only aes256 encryption is supported for key file encryption.
   The following examples provide command syntax. The key type may be either **rsa** or **ec**.
   If the key file being used is plain text, the following command format should be used:

   ```
   device(config)# pki import key < keytype > < keylabel > pem url flash: < keyname >.key.pem
   ```

   If the key file is encrypted using a password, the following command format should be used:

   ```
   device(config)# pki import key < keytype > < keylabel > pem url flash: < keyname >.key.pem
   password pwd_value
   ```

   The following example imports the RSA key from the flash location specified.

   ```
   device(config)# pki import key rsa rsakey pem url flash: dut.key.pem
   device(config)# end
   ```

   The following example imports the EC key eckey1 from flash using a password string associated with the encrypted file. (You will be prompted for the password if you later try to open the imported file.)

   ```
   device(config)# pki import key ec eckey1 pem url flash: dut.eckey.pem password f1234yzztk!
   ```

3. Use the following commands to import the CA and local certificates.

   The following example provides command syntax.

   ```
   device(config)# pki import < trustpointname > pem url flash: rootca.pem
   device(config)# pki import < trustpointname > pem url flash: localcert.pem
   ```

   The following command imports the CA certificate (rootca.pem) and the local certificate (localcert.pem) from the trustpoint named abcd.

   ```
   device(config)# pki import abcd pem url flash: rootca.pem
   device(config)# pki import abcd pem url flash: localcert.pem
   ```

4. Attach the imported key label to the trustpoint using the following commands.

   The following example shows command syntax.

```
device(config)# pki trustpoint < trustpointname >
device(config-pki-trustpoint-trustpointname)# { rsakeypair | eckeypair } key-label <label >
```

   The following example binds the rsakey label to the CA trustpoint named abcd.

```
device(config)# pki trustpoint abcd
device(config-pki-trustpoint-abcd)# rsakeypair key-label rsakey
```

5. Authenticate the rootCA trustpoint using the following command.

   The following example provides the syntax for authenticating the trustpoint.

```
device(config-pki-trustpoint-trustpointname)# exit
device(config)# pki authenticate < trustpointname >
```

   The following example authenticates the previously configured trustpoint abcd.

```
device(config-pki-trustpoint-abcd)# exit
device(config)# pki authenticate abcd
```

NOTE

Because the manual process does not generate a CSR on the FastIron device (the TOE), it is not necessary to execute the **pki enroll** command. after the trustpoint is authenticated.

6. Use the following commands to check the local and CA certificates on the FastIron device.

```
device(config)# show pki certificates local
device(config)# show pki certificates trustpoint < trustpointname >
```

The following example displays information for the local certificate and for certificates from the trustpoint abcd.

```
device# show pki certificates local

----------------PKI LOCAL CERTIFICATE ENTRY-----------------
CA: TLS-ABCD
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4125 (0x101d)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=CA, L=SJ, O=ROOTCA-CC, OU=SQA, CN=ROOTCA-CC/emailAddress=user@arris.com
        Validity
            Not Before: Nov  7 02:24:18 2017 GMT
            Not After : Nov 17 02:24:18 2018 GMT
        Subject: CN=DUTFIPSCC, ST=CA, C=US/emailAddress=user@arris.com, O=DUTFIPSCC, OU=SQA
device#


device# show pki certificates trustpoint

----------------PKI TRUSTPOINT CERTIFICATE ENTRY-----------------
CA: TLS-ABCD
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            bd:fa:4f:da:bd:89:4a:5d
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=CA, L=SJ, O=ROOTCA-CC, OU=SQA, CN=ROOTCA-CC/emailAddress=user@arris.com
        Validity
            Not Before: Nov  7 02:10:00 2017 GMT
            Not After : Nov  7 02:10:00 2022 GMT
        Subject: C=US, ST=CA, L=SJ, O=ROOTCA-CC, OU=SQA, CN=ROOTCA-CC/emailAddress=user@arris.com
device#
```

7. Validate the certificates using the following command.

The following example provides the syntax for validating the imported certificates.

```
device(config)# pki cert-validate < trustpointname >
```

The following example successfully validates certificates imported from the trustpoint abcd.

```
device(config)# pki cert-validate abcd
PKI: Successfully validated the local certificate for trustpoint: abcd
```

The following example imports certificates for the configured trustpoint abcd, authenticates the trustpoint, and validates imported certificates.

```
device# configure terminal
device(config)# crypto key generate rsa
device(config)# pki import key rsa rsakey pem url flash: dut.key.pem
device(config)# pki import abcd pem url flash: rootca.pem
device(config)# pki import abcd pem url flash: localcert.pem
device(config)# pki trustpoint abcd
device(config-pki-trustpoint-abcd)# ocsp http post
device(config-pki-trustpoint-abcd)# revocation-check ocsp
device(config-pki-trustpoint-abcd)# ocsp-url http://10.21.40.39:2560
device(config-pki-trustpoint-abcd)# fingerprint 3C:EA:EC:E6:F1:DD:3B:86:65:DE:58:F4:A2:75:D8:63:6D:23:68:40
device(config-pki-trustpoint-abcd)# exit
device(config)# pki authenticate abcd
device(config)# pki cert-validate abcd
PKI: Successfully validated the local certificate for trustpoint: abcd
```

Revocation check for peer certificates

FastIron devices in Common Criteria mode support OCSP for checking the revocation status of a certificate received from a peer. The revocation status is checked for both the intermediate and the peer certificate. The following actions are taken in revocation checks:

- The OCSP signing bit should be set for the OCSP responder.
- If the OCSP responder responds that the certificate status is good, the TLS or IPSec session comes up.
- If the OCSP responder responds that the certificate status is revoked, the TLS or IPSec session does not come up.
- If the OCSP server is not reachable and the response is not received, then the TLS or IPSec session does not come up between the peers.

The OCSP URL is picked up from the certificates received from the peer. The default configuration is "revocation-check none" and the status of the revoked certificate is not checked. When the command **revocation-check ocsp** is configured as part of the trustpoint configuration, OCSP protocol is used to check the status of certificates received from the peer. The command "ocsp-url http://15.1.1.1:2560" is used to identify the URL for the OCSP responder.

By default, an HTTP "get" command is used to reach the OCSP responder. This method can be changed to an HTTP post using the command "ocsp http post" as shown in the following example.

```
device# configure terminal
device(config)# pki trustpoint abcd
device(config-pki-trustpoint-abcd)# ocsp http post    <-- Optional command. Applies only to Linux.
device(config-pki-trustpoint-abcd)# revocation-check ocsp
device(config-pki-trustpoint-abcd)# ocsp-url http://15.1.1.1:2560
device(config-pki-trustpoint-abcd)# rsakeypair key-label rsakey
device(config-pki-trustpoint-abcd)# fingerprint 3C:EA:EC:E6:F1:DD:3B:86:65:DE:58:F4:A2:75:D8:63:6D:23:68:40
device(config-pki-trustpoint-abcd)# exit
device(config)#
```

Network Device Collaborative Protection Profile with VPN gateway

The Network Collaborative Device Protection Profile (NDcPP) standards provide a set of rules that define the security requirements for network devices. The main purpose of these requirements is to minimize and reduce threats to network devices. NDcPP requires SSH or TLS for syslog and authentication server communications. NDcPP with VPN gateway requires IPsec for syslog and authentication server communications.

FastIron devices support creation of a VPN using the IPsec protocol and also the setup of security associations (SAs) for the IPsec protocol suite. Once the SAs are set up, the IPsec protocol is used to set up and operate encrypted tunnels between two endpoints. NDcPP with VPN gateway allows the FastIron device to be used as a VPN gateway within highly secure and high security federal networks. These networks should be approved by the Commercial Solutions for Classified (CSfC) program.

NOTE
The connections syslog and AAA servers need to be over TLS or IPsec. For more information, see the output of **fips enable common-criteria** command with lines prefixed with "CC".

NDcPP with VPN gateway requirements

- Management and control traffic, including SSH and HTTPS, should be transported over the IPsec network provided by the FastIron device.
- Elliptic curve-based key establishment support curves.

- The TSF should ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), and 20 (384-bit Random ECP).
- New audit events for IKE and IPsec.
- Support for IKEv2 as defined in RFC 5996.
- Support for AES-256-CBC and AES-128-CBC for IKEv2.
- Support for AES256-GCM and AES-128-GCM for IPsec.
- Support for binary bits-based PSK for IKEv2 authentication.
- Support for X509v3 certificate for authentication of IKEv2 endpoints using ECDSA P-384 and P-256 curves.
- Support for NAT traversal.
- TOE supports IPv4 per RFC 791, IPv6 per RFC 2460, TCP per RFC 793, and UDP per RFC 768.

  NOTE
  If the **enable strict-password-enforcement** command is enabled, users have up to three login attempts. If a user fails to login after three attempts, that user is locked out (disabled). Enable the user by entering the **username name enable** command.

NAT traversal in IKE and IPsec

The evaluated configuration for the VPN gateway Extended Profile (EP) requires that IKEv2 and the IPsec tunnel support Network Address Translation (NAT) traversal. The peer device is set up behind a NAT device or a NAT firewall to protect its identity.

IPsec protocol protects the integrity of the IP packet in addition to providing encapsulation and encryption. When an IP packet passes through a NAT device, there is a change in the IP/TCP header leading to an integrity violation that causes the packet to be discarded by the IPsec tunnel end nodes. To overcome this issue, determine whether the IKE peers are capable of supporting NAT traversal or whether there is a NAT device between the IKE peers.

The following commands are introduced as part of NAT traversal in IKE and IPsec:

- **ikev2 nat-disable**
- **ikev2 nat-keepalive** *time in seconds*

  NOTE
  NAT is enabled by default.

  NOTE
  IPsec supports only IKEv2 protocol; IKEv1 protocol is not supported.

Selecting the encryption algorithm

You select the encryption algorithm for the tunnel when configuring the IPsec proposal.

The following example uses the AES-GCM-128 algorithm for the IPsec proposal named *ipsec_proposal*. Because the AES-GCM-256 algorithm is used by default, it must be disabled.

```
device(config)# ipsec proposal ipsec_proposal
device(config-ipsec-proposal-ipsec_proposal)# no encryption-algorithm aes-gcm-256
device(config-ipsec-proposal-ipsec_proposal)# encryption-algorithm aes-gcm-128
```

The following example displays the configuration of an IPsec proposal named *ipsec_proposal*. AES-GCM-128 is configured as the accepted encryption algorithm.

```
device# show ipsec proposal ipsec_proposal
================================================================================
Name                : ipsec_proposal
Protocol            : ESP
Encryption          : aes-gcm-128
Authentication      : NULL
ESN                 : Disable
Mode                : Tunnel
Ref Count           : 0
```

Audit logging

FastIron devices support logging of IKE and PKI transaction details. The logs are automatically generated syslog messages that contain the IKEv2 and PKI transaction details.

There are two types or levels of logging. Standard logging is enabled by default. The second type of logging is called extended logging, which you must enable using commands. This type of logging allows you to log additional IKE or PKI transaction details.

Support for Logging IKE and PKI Transaction Details

FastIron devices support logging of IKE and PKI transaction details. The log files are automatically generated syslog messages that contain the transaction details.

There are two types or levels of logging. Standard logging is enabled by default. The second type of logging is called extended logging, which you must enable using commands. This type of logging allows you to log additional IKE or PKI transaction details.

Required hardware

The hardware requirements are identical for default logging and extended logging. The following table lists the required hardware.

TABLE 8 Required hardware for IPsec

Device	Module
FastIron ICX 7450	ICX7400-SERVICE-MOD Module **NOTE** The FastIron device should have at least one interface module through which the external syslog server can be reached.

Limitations

All of the current limitations of the logging feature on FastIron devices and the limitations of the IPsec security feature apply to the logging of IKE and PKI transaction details.

In addition, there are some limitations specific to the feature for logging IKE and PKI transaction details. The following table lists the current limitations for this feature.

TABLE 9 IKE and PKI logging limitations

Default and Extended Logging	Description
IKE transaction details (send and receive packets)	An IKEv2 packet that together with protocol headers totals more than 250 bytes will be logged in multiple syslog messages.

Management commands

The following list of commands and command variants are required for administration of the TOE. These commands are available only after an administrator has successfully logged into the TOE.

TABLE 10 Management commands

Command	Tested Command Variants	Description
aaa	**aaa** authentication **aaa** authentication enable default radius local **aaa** authentication login default radius local **aaa** authentication web-server default local	Configures the AAA authentication functions
access-list	**access-list** deny host *IP address* **access-list** 1 deny 10.157.29.12 **access-list** 1 deny host IPHost1 **access-list** 1 permit any	Creates ACL rules
banner	**banner** motd+	Manages the login banner
clock	**clock** *set time*	Manages the internal clock
config	**config** terminal	Switches to configuration mode
crypto	**crypto key generate**	Invokes cryptographic functions.
crypto-ssl	**crypto-ssl** certificate generate	Manages web server properties.
enable	**enable** aaa **enable** password-min-length 15	Enables console login features.
exit	**exit**	Logs out or exits current session.
fips	**fips** enable common-criteria **fips** show **fips** zeroize all	Manages FIPS and common criteria configuration.
ikev2	**ikev2 auth-proposal** **ikev2 nat keepalive** **ikev2 nat-enable** **ikev2 proposal** **ikev2 profile** **ikev2 policy**	Configures IKEv2 properties.
interface	**interface** ethernet 4/12 **interface** mac access-group 400 in **tunnel**	Configures an interface or associates an ACL with an interface.
ip / ipv6	**access-list** **access-group** **address** **ssl profile** (IPv4 only)	Configures IPv4 and IPv6 parameters.

TABLE 10 Management commands (continued)

Command	Tested Command Variants	Description
ipsec	**ipsec** proposal **ipsec** profile	Configures IPsec properties.
lifetime (IKEv2)	**lifetime** *lifetime in minutes*	Configures the IKEv2 security association (SA) lifetime value in minutes in the IKEv2 profile config mode. For example: `device(config-ike-profile-ipsec-abcd)# lifetime 400`. Default is 43200. Valid range is 10 through 43200 (decimal).
lifetime (IPsec)	**lifetime** *lifetime in minutes*	Configures the IPsec SA lifetime value in minutes in the IPsec profile config mode. For example: `device(config-ipsec-profile-ipsec-abcd)# lifetime 600` Default is 480. Valid range is 60 through 480.
logging	**logging host** *ip-address* **ssl-port** *port-number* **profile** *profile-name*	Configures the audit logging host.
logout	**logout**	Used to terminate both local console and remote SSH sessions.
ntp	**ntp**	Switches to NTP configuration mode.
openssl	**openssl s_server**	Configures secure connections (for example with syslog).
pki	**pki** **authenticate** - Authenticates CA to router by obtaining the self-signed certificate of the CA. **cert-validate** - Determines if a trustpoint has been successfully authenticated. **enroll** - Requests certificates from the CA for each key pair of your router. **entity** - Configures PKI end-user parameters. **export** - Exports a PKI certificate manually. **import** - Imports a PKI certificate manually. **profile-enrollment** - Configures PKI enrollment parameters. **trustpoint** - Configures PKI CA parameters.	Configures Public Key Infrastructure parameters.
radius-server	**radius-server host** *ip-address* **ssl-auth-port** *port* **profile** *profile-name* **authentication key** *value* **radius-server** retransmit *retransmit period* **radius-server** timeout *timeout period* **radius-server** key *key name*	Configures the RADIUS server.
reload	**reload**	Reloads the current flash image.
server	**server** *ntp server ip*minpoll *time*	Configures external services.
show	**show** flash **show** version **show** clock **show** ip client-pub-key **show** ip ssl	Displays information about specified configuration.

TABLE 10 Management commands (continued)

Command	Tested Command Variants	Description
	show logging **show** pki **show** run	
timeout	**console timeout** *time* **ip ssh timeout** *time*	Configures the console timeout in minutes. Default is 0. Valid values are 0-240. [3] Configures the SSH timeout in seconds. Default is 120. Valid values are 1-120.
tunnel	**tunnel** protection ipsec ipv4 *ipsec profile name*	Enables IPsec on an interface.
username	**username** *user* password	Manages user accounts.
write	**write memory**	Writes to persistent storage.

IPsec configuration

You can configure an IPsec IPv4 or IPv6 tunnel to transmit secure IP packets.

The following steps are required to configure IPsec for use in Common Criteria mode:

- Configure an IKEv2 proposal and policy.
- Configure an IKEv2 authentication proposal. The authentication proposal may be based on either a pre-shared key or on X.509 certification.
  - In Common Criteria mode, there is no default value for a pre-shared key. You must specify the preshared key.
  - If you use X.509 certification, you must also configure PKI. PKI can be configured for either automatic or manual enrollment.
- Configure the IKEv2 profile.
- Configure the IPsec profile.
- Configure the IPsec tunnel and apply the IPsec profile.

Configuring an IKEv2 proposal and policy

The initial steps in the following procedure describe how to modify standard IKEv2 settings. The last step in the procedure associates a pre-configured IKEv2 proposal to an IKEv2 policy.

1. (Optional) Enter the following **ikev2** commands at the global configuration level to enable the non-default NAT-T option, and to specify the NAT keep-alive time interval.

   In this example, the option is enabled and the keep-alive time interval is set to 30 seconds (the default is 20 seconds).

```
device(config)# ikev2 nat-enable
device(config)# ikev2 nat keepalive 30
```

[3] Inactivity timeouts are not supported for IPsec tunnels.

2. (Optional) Enter the **ikev2 proposal** command to designate a non-default proposal for the initial phase of the IKEv2 peer negotiation and to enter IKEv2 proposal configuration mode. (You must be in IKEv2 proposal configuration mode to configure a non-default proposal.)

   In this example, an IKEv2 proposal named a1 is defined.

   ```
   device(config)# ikev2 proposal a1
   device(config-ike-proposal-a1)#
   ```

3. (Optional) In IKE proposal configuration sub-mode, select a non-default Diffie Hellman (DH) group, or groups. (The default DH group is 20. The non-default groups you can select are groups 14 or 19.)

   In this example, DH group 14 is selected and the default (DH group 20) is disabled. Only DH group 14 will be included in the IKEv2 proposal. The default is disabled to ensure it is not selected during the negotiation because if multiple DH groups are selected, the first matching DH group supported by both ends is automatically selected.)

   ```
   device(config-ike-proposal-a1)# dhgroup 14
   device(config-ike-proposal-a1)# no dhgroup 20
   device(config-ike-proposal-a1)# exit
   ```

4. (Optional) In IKEv2 proposal configuration sub-mode, configure a pseudorandom function (PRF) for the proposal. This defines the hash size algorithm for the IKEv2.

   In this example, the SHA-256 algorithm is added to the PRF algorithms configured for a1. Because the SHA-384 algorithm is configured by default, both the SHA-384 and SHA-256 algorithms are configured for IKEv2 proposal a1 after executing this step. Configuration of multiple PRF algorithms is allowed.

   ```
   device(config-ike-proposal-a1)# prf sha256
   ```

5. (Optional) In IKEv2 proposal configuration sub-mode, enter the **integrity** command followed by the encryption algorithm to configure an integrity algorithm for the proposal as shown in the following example.

   This step adds the SHA-256 algorithm to the integrity algorithms configured for the proposal a1. Because the SHA-384 algorithm is configured by default, both the SHA-384 and SHA-256 algorithms are configured for a1 after executing this step. Configuration of multiple integrity algorithms is allowed.

   When you want to configure the SHA-256 algorithm only for the proposal, you must first add the SHA-256 algorithm and then remove the default algorithm by using the command **no integrity sha384**.

   ```
   device(config-ike-proposal-a1)# integrity sha256
   ```

6. (Optional) In IKEv2 proposal configuration sub-mode, enter the **encryption** command to configure an encryption algorithm for the proposal as shown in the following example.

   In the example, the AES-CBC-128 algorithm is added to the encryption algorithms configured for proposal a1. Because the AES-CBC-256 algorithm is configured by default, both the AES-CBC-256 and AES-CBC-128 algorithms are configured for IKEv2 Proposal a1 after executing this step. Configuration of multiple encryption algorithms is allowed.

   When you want to configure the AES-CBC-128 algorithm only for the proposal, you must first add the AES-CBC-128 algorithm and then remove the default algorithm using the command **no encryption aes-cbc-256**.

   ```
   device(config-ike-proposal-a1)# encryption aes-cbc-128
   ```

7. Enter the **ikev2 policy** command in general configuration mode to designate an IKEv2 policy, to enter IKEv2 policy configuration mode, and to bind a pre-configured authentication proposal to protect IKE during negotiations.

   ```
   device(config)# ikev2 policy test-policy
   device(config-ike-test-policy)# proposal a1
   device(config-ike-test-policy)# exit
   ```

Configuring an IKEv2 authentication proposal for use in an IPsec profile

Follow these steps to configure an IKEv2 authentication proposal.

1. Based on your certification method, do one of the following:
   - To use a pre-defined shared secret (PSK), configure an IKEv2 authentication proposal.

     NOTE
     You must define a pre-shared key in Common Criteria mode, as no default value is configured. A warning message is displayed if no pre-shared key has been defined in the authentication proposal.

   - For X.509 certification, configure a PKI trustpoint.

   The following example provides command syntax for specifying a pre-shared key.

```
device# configure terminal
device(config)# ikev2 auth-proposal < proposalname >
device(config-ike-auth-proposalname)# pre-shared-key < valid-key >
device(config-ike-auth-proposalname)# exit
```

   NOTE
   Valid keys must be at least eight characters in length. They must contain a mix of at least three character types (uppercase alpha characters, lowercase alpha characters, numeric characters, non-alpha ASCII characters). The first character must not be the only uppercase alpha character in the key, and the last character must not be the only numeric character in the key. The system returns an error if the pre-shared key is not valid.

   The following example defines a valid pre-shared key for use with IKEv2 authentication proposal "withKey-L2."

```
device# configure terminal
device(config)# ikev2 auth-proposal withKeyL2
device(config-ike-auth-withKeyL2)# pre-shared-key m!XYZ#79L
device(config-ike-auth-withKeyL2)# exit
```

   NOTE
   Bit-based pre-shared key are supported on ICX 7450 devices. To specify a bit-based preshared key in the IKE auth-proposal, enter auth-proposal bit-based configuration submode, and add the prefix "0x" to a hexadecimal value.

   The following example creates a bit-based pre-shared key on an ICX 7450 device.

```
ICX_7450_device# configure terminal
ICX_7450_device(config)# ike auth-proposal bit-based
ICX_7450_device(config-ike-auth-proposal-bit-based)# 0xabcd10908
```

   The following example provides command syntax for specifying a trustpoint for use with X.509 certification.

```
device# configure terminal
device(config)# ikev2 auth-proposal < proposalname >
device(config-ike-auth-proposalname)# pki-trustpoint < trustpointname > [ sign | verify ]
```

   The following example configures a PKI trustpoint for the Certificate Authority for use in X.509 certification. In the example, the server abcd is established as the PKI trustpoint for both certificate signature and verification.

```
device# configure terminal
device(config)# ikev2 auth-proposal abcd-CA
device(config-ike-auth-proposal-abcd-CA)# pki-trustpoint abcd-CA sign
device(config-ike-auth-proposal-abcd-CA)# pki-trustpoint abcd-CA verify
```

   NOTE

A full example of setting up for X.509 certification is presented in the section "Configuration example: creating an IPsec profile for tunnels that use X.509 certificates."

2. Configure an IKEv2 profile that uses IP addresses or distinguished names as local and remote identifiers.

   NOTE
   DN is the subject name of the PKI local certificate. Domain name may contain common name, state, country, organization name and organization unit name of the entity for which the PKI certificate is presented.

   The following example provides command syntax.

```
device(config)# ikev2 profile < profilename >
device(config-ike-profile-profilename)# authentication < auth-proposalname >
device(config-ike-profile-profilename)# local-identifier address < ip-address | ipv6-address | dn >
device(config-ike-profile-profilename)# remote-identifier address < ip-address | ipv6-address | dn >
device(config-ike-profile-profilename)# match-identity local address < ip-address | ipv6-address |
dn >
device(config-ike-profile-profilename)# match-identity remote address < ip-address | ipv6-address |
dn >
device(config-ike-profile-profilename)# exit
```

   The example below uses IP addresses as identifiers.

```
device(config)# ikev2 profile withKeyL2
device(config-ike-profile-withKeyL2)# authentication withKeyL2
device(config-ike-profile-withKeyL2)# local-identifier address 15.1.1.2
device(config-ike-profile-withKeyL2)# remote-identifier address 15.1.1.1
device(config-ike-profile-withKeyL2)# match-identity local address 15.1.1.2
device(config-ike-profile-withKeyL2)# match-identity remote address 15.1.1.1
device(config-ike-profile-withKeyL2)# exit
```

   The following example uses distinguished names as identifiers.

```
device(config)# ikev2 profile with_standalone
device(config-ike-profile-with_standalone)# authentication withCert
device(config-ike-profile-with_standalone)# local-identifier dn "CN=SPATHA_STANDALONE, ST=CA, C=US,
O=SPATHA_SPATHA_ORG_NAME, OU=SPATHA_ALONE_ORG_UNIT"
device(config-ike-profile-with_standalone)# remote-identifier dn "CN=SPATHA48F, ST=CA, C=US,
O=SPATHA48F_ORG_NAME"
device(config-ike-profile-with_standalone)# match-identity local dn "CN=SPATHA_STANDALONE, ST=CA,
C=US, O=SPATHA_SPATHA_ORG_NAME, OU=SPATHA_ALONE_ORG_UNIT"
device(config-ike-profile-with_standalone)# match-identity remote dn "CN=SPATHA48F, ST=CA, C=US,
O=SPATHA48F_ORG_NAME"
device(config-ike-profile-with_standalone)#  exit
```

3. Configure an IPsec profile that uses a previously configured IKEv2 profile.

```
device(config)# ipsec profile withKeyL2
device(config-ipsec-profile-withKeyL2)# ike-profile withKeyL2  <--- pre-configured IKEv2 profile
device(config-ipsec-profile-withKeyL2)# exit
```

   NOTE
   To use the IPsec profile, you must associate it with a specific tunnel. This process is described in the section "Configuring an IPv4 or IPv6 IPsec tunnel."

The following example configures IPsec tunnel 1 to use an IPsec profile that references IKEv2 settings from a specific IKEv2 profile.

```
device# configure terminal
device(config)# ikev2 auth-proposal withKeyL2
device(config-ike-auth-withKeyL2)# pre-shared-key 2
device(config-ike-auth-withKeyL2)# exit
device(config)# ikev2 profile withKeyL2
device(config-ike-profile-withKeyL2)# authentication withKeyL2            <--- authentication proposal name
device(config-ike-profile-withKeyL2)# local-identifier address 15.1.1.2
device(config-ike-profile-withKeyL2)# remote-identifier address 15.1.1.1
device(config-ike-profile-withKeyL2)# match-identity local address 15.1.1.2
device(config-ike-profile-withKeyL2)# match-identity remote address 15.1.1.1
device(config-ike-profile-withKeyL2)# exit
device(config)# ipsec profile withKeyL2
device(config-ipsec-profile-withKeyL2)# ike-profile withKeyL2             <--- previously configured IKEv2
profile
device(config-ipsec-profile-withKeyL2)# exit
device(config)# interface tunnel 1
device(config-tnif-1)# interface tunnel 1
device(config-tnif-1)# tunnel mode ipsec ipv4
device(config-tnif-1)# tunnel protection ipsec profile withKeyL2           <---- previously configured
IPsec profile
device(config-tnif-1)# tunnel source 15.1.1.2
device(config-tnif-1)# tunnel destination 15.1.1.1
device(config-tnif-1)# ip address 15.15.15.1 255.255.255.0
```

Configuration example: creating an IPsec profile for tunnels that use X.509 certificates

You can use X.509 certificates to establish an IPsec tunnel by configuring the Certificate Authority (CA server) with an IKEv2 profile that is then referenced in an IPsec profile.

The following steps are an example of how to configure an IPsec profile that can be used to invoke a CA server to sign and verify X.509 certificates.

1. Create an IKEv2 authentication proposal.

   ```
   device# configure terminal
   device(config)# ikev2 auth-proposal withCert
   ```

2. In the proposal, define the method to be used for authentication.

   ```
   device(config-ike-auth-proposal-withCert)# method remote rsa
   device(config-ike-auth-proposal-withCert)# method local rsa
   ```

3. Next, define the PKI trustpoint for the Certificate Authority.

   In the example, the server abcd-CA is established as the PKI trustpoint for both certificate signature and verification.

   ```
   device(config-ike-auth-proposal-withCert)# pki-trustpoint abcd-CA sign
   device(config-ike-auth-proposal-withCert)# pki-trustpoint abcd-CA verify
   ```

4. Configure an IKEv2 profile that specifies a local and remote ID for the CA server, using IP addresses or distinguished name information.

   The example uses distinguished names to identify the server.

```
device(config)# ikev2 profile with_standalone
device(config-ike-profile-with_standalone)# authentication withCert
device(config-ike-profile-with_standalone)# local-identifier dn "CN=SPATHA_STANDALONE, ST=CA, C=US,
O=SPATHA_SPATHA_ORG_NAME, OU=SPATHA_ALONE_ORG_UNIT"
device(config-ike-profile-with_standalone)# remote-identifier dn "CN=SPATHA48F, ST=CA, C=US,
O=SPATHA48F_ORG_NAME"
device(config-ike-profile-with_standalone)# match-identity local dn "CN=SPATHA_STANDALONE, ST=CA,
C=US, O=SPATHA_SPATHA_ORG_NAME, OU=SPATHA_ALONE_ORG_UNIT"
device(config-ike-profile-with_standalone)# match-identity remote dn "CN=SPATHA48F, ST=CA, C=US,
O=SPATHA48F_ORG_NAME"
device(config-ike-profile-with_standalone)#  exit
!
```

5. Configure an IPsec profile that references the IKEv2 profile you have already configured. The profile can be used when you define parameters for a specific IPsec tunnel to bring the tunnel up using X.509 certificates.

```
!
device(config)# ipsec profile withCert
device(config-ipsec-profile-withCert)# ike-profile with_standalone
!
```

The following example creates the IKEv2 authentication proposal called withcert to use RSA as the authentication method and establishes the PKI trustpoint abcd-CA to sign and verify certificates. The example also configures an IKEv2 profile called with_standalone, which specifies distinguished name identifiers for the device used as Certificate Authority for both verification and signature. The IKEv2 profile is then referenced in one of the configured IPsec profiles, withCert. The IPsec profile can be used to establish an IPsec tunnel using X.509 certificates held by the server configured in the example.

In addition to the X.509 certificate example just defined, the example also configures the profile described previously in the section "Configuring an IKEv2 authentication proposal for use in an IPsec profile." The second proposal and profile can be used to establish a tunnel using a pre-shared key.

```
device# configure terminal
device(config)# ikev2 auth-proposal withCert
device(config-ike-auth-proposal-withCert)# method remote rsa
device(config-ike-auth-proposal-withCert)# method local rsa
device(config-ike-auth-proposal-withCert)# pki-trustpoint abcd-CA sign
device(config-ike-auth-proposal-withCert)# pki-trustpoint abcd-CA verify
device(config-ike-auth-proposal-withCert)# exit
!
device(config)# ikev2 profile with_standalone
device(config-ike-profile-with_standalone)# authentication withCert
device(config-ike-profile-with_standalone)# local-identifier dn "CN=SPATHA_STANDALONE, ST=CA, C=US,
O=SPATHA_SPATHA_ORG_NAME, OU=SPATHA_ALONE_ORG_UNIT"
device(config-ike-profile-with_standalone)# remote-identifier dn "CN=SPATHA48F, ST=CA, C=US,
O=SPATHA48F_ORG_NAME"
device(config-ike-profile-with_standalone)# match-identity local dn "CN=SPATHA_STANDALONE, ST=CA, C=US,
O=SPATHA_SPATHA_ORG_NAME, OU=SPATHA_ALONE_ORG_UNIT"
device(config-ike-profile-with_standalone)# match-identity remote dn "CN=SPATHA48F, ST=CA, C=US,
O=SPATHA48F_ORG_NAME"
device(config-ike-profile-with_standalone)#  exit
!
device(config)# ipsec profile withKey-L2
device(config-ipsec-profile-withKey-L2)# ike-profile withKey-L2
device(config-ipsec-profile-withKey-L2)# exit
!
device(config)# ipsec profile withCert
device(config-ipsec-profile-withCert)# ike-profile with_standalone
!
```

Configuring IPv4 and IPv6 IPsec tunnels

To set up an IPsec IPv4 or IPv6 tunnel, you must specify the tunnel interface by tunnel number, designate the tunnel mode (IPv4 or IPv6), and set the IKE and IPsec options for the tunnel. Once the tunnel is created, you can use it to transmit IP packets securely.

Pre-requisites:

- **Use of PKI**: If your IPsec tunnel configuration involves the use of PKI options (for example, a PKI entity or PKI trustpoint), make sure you complete the PKI configuration before you begin setting up the IPsec tunnel. The PKI options must be configured before you can select them as part of the tunnel setup process.
- **Use of global IKEv2 parameters**: If you need to configure any global IKEv2 parameters, make sure you complete the configuration before you begin setting up the IPsec tunnel.

  NOTE
  You can use an IPsec IPv4 tunnel to transmit IPv6 packets, but you cannot use an IPsec IPv6 tunnel to transmit IPv4 packets.

  NOTE
  Only IKEv2 is supported; IKEv1 is not supported.

  NOTE
  ICX 7450 devices do not support transport mode for IPsec tunnel configuration; they support "tunnel" mode only.

  NOTE
  The network administrator must ensure that the IKE cryptographic algorithms and key sizes that are configured for a tunnel are stronger than the IPsec cryptographic algorithms and key sizes used by the same tunnel.

Effect of authentication method on IKEv2 profile settings

The type or method of authentication you select for IKE transactions affects IKEv2 profile options you should select when setting up IPsec tunnels.

The recommended IKEv2 profile options are:

- **PKI-based authentication**: When using PKI-based authentication, it is recommended that you select Distinguished Name (DN).
- **Pre-shared key authentication**: When using pre-shared key authentication (PSK), use IP addresses as local and remote identifiers.

Complete the following steps to set up the IPsec tunnel.

1. Enter one of the following commands in tunnel interface configuration mode to select the tunnel mode for the IPsec virtual terminal interface (VTI).
   - ipv4: **tunnel mode ipsec ipv4**
   - ipv6: **tunnel mode ipsec ipv6**

   The following example specifies IPv4 as the IPsec tunnel mode for Tunnel 1.

   ```
   device(config)# interface tunnel 1
   device(config-tnif-1)# tunnel mode ipsec ipv4
   ```

2. Enter the **tunnel protection ipsec profile** command in tunnel interface configuration mode to assign a previously configured IPsec profile that will be used to encapsulate outgoing packets. (This binds the profile to the VTI.)

   The following example assigns the previously created IPsec profile test-profile to Tunnel 1.

   ```
   device(config)# interface tunnel 1
   device(config-tnif-1)# tunnel protection ipsec profile test-profile
   ```

3. Enter the **tunnel source** command to specify the tunnel source. This is the local endpoint of the tunnel. If you are specifying an IP address, it should be consistent with the tunnel mode you have specified.

   The tunnel source can be one of the following:

   - The IPv4 address of a physical, virtual, or loopback interface as shown in the following example.

     ```
     device(config)# interface tunnel 1
     device(config-tnif-1)# tunnel source 192.168.1.2
     ```

   - The global IPv6 address of a physical, virtual, or loopback interface as shown in the following example.

     ```
     device(config) interface tunnel 3
     device(config-tnif-3)# tunnel mode ipsec ipv6
     device(config-tnif-3)# tunnel source 10:1:1::1/64
     ```

   - The interface on which the required tunnel source IPv4 address or IPv6 address has been configured as shown in the following example.

     ```
     device(config) interface tunnel 1
     device(config-tnif-1)# tunnel source ethernet 1/1/1
     ```

4. Enter the **tunnel destination** command to specify the tunnel destination. This is the remote endpoint of the tunnel. Use an IP address consistent with the tunnel mode you have specified.

   The following example specifies an IPv4 tunnel destination.

   ```
   device(config-tnif-1)# tunnel destination 10.1.1.2
   ```

   The following example specifies an IPv6 destination.

   ```
   device(config-tnif-3)# tunnel destination 10:1:1::2/64
   ```

5. Specify the tunnel address. This is the IPv4 or IPv6 address of the tunnel port, not a tunnel endpoint.
   - For an IPv4 tunnel, enter the **ip address** command followed by the IPv4 tunnel address.
     - As an option, so that the IPv4 tunnel can also transmit IPv6 packets, also enter an IPv6 address for the same IPv4 tunnel interface.
   - For an IPv6 tunnel, enter the **ipv6 address** command followed by the IPv6 tunnel address.

   The following example configures an IPv4 address for the tunnel port.

   ```
   device(config-tnif-1)# ip address 36.0.8.108/32
   ```

   The following example configures an IPv6 address for the tunnel port.

   ```
   device(config-tnif-3)# ipv6 address 10:10:10::1/120
   ```

   The following example, when configured on an IPv4 tunnel interface, adds an IPv6 address as a tunnel port, which enables the tunnel to transmit IPv6 packets.

   ```
   device(config-tnif-1)# ipv6 address 36:1:1::1/120
   ```

6. In global configuration mode, enter the **ipsec profile** command followed by the IPsec profile name to enter IPsec profile configuration mode. In this mode, specify a previously configured IPsec proposal and a previously configured IKEv2 profile for use.

   The following example provides required syntax.

```
device(config)# ipsec profile < ipsec_profile_name >
device(config-ipsec-profile-name) proposal < ipsec_proposal >
device(config-ipsec-profile-name) ike-profile < ike_profile_name >
```

   The following example configures the IPsec profile a1 to use the IPsec proposal test-proposal and the IKEv2 profile test-profile.

```
device(config)# ipsec profile a1
device(config-ipsec-profile-a1) proposal test-proposal
device(config-ipsec-profile-a1) ike-profile test-profile
device(config-ipsec-profile-a1)# exit
```

Summary Examples

The following example shows the steps and syntax needed to configure an IPv4 IPsec tunnel.

```
device(config)# interface tunnel < tunnel_id >
device(config-tnif-id)# tunnel mode ipsec ipv4
device(config-tnif-id)# tunnel protection ipsec profile < ipsec_profile_name >
device(config-tnif-id)# tunnel protection ipsec profile < profile-name >
device(config-tnif-id)# tunnel source < ipv4-address >  <-- ethernet < port > may be used instead.
device(config-tnif-id)# tunnel destination < tunnel destination ipv4-address >
device(config-tnif-id)# ip address < tunnel ip-address >
```

The following example applies the IPsec profile a1 to the tunnel with IP address 10.0.0.1 255.255.255.0. It also specifies the source (local endpoint) and destination addresses to be used for IPsec connections over the tunnel.

```
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel mode ipsec ipv4
device(config-tnif-1)# tunnel protection ipsec profile a1
device(config-tnif-1)# tunnel source 10.1.1.1
device(config-tnif-1)# tunnel destination 10.1.1.2
device(config-tnif-1)# ip address 10.0.0.1 255.255.255.0
```

The following example adds an IPv6 address to the IPv4 tunnel interface shown in the previous example, which enables the tunnel to transmit IPv6 packets.

```
device(config-tnif-1)# ipv6 address 36:1:1::1/120
device(config-tnif-1)# exit
```

The following example shows the steps and syntax needed to configure an IPv6 IPsec tunnel.

```
device# configure terminal
device(config)# interface tunnel < tunnel_id >
device(config-tnif-id)# tunnel mode ipsec ipv6
device(config-tnif-id)# tunnel protection ipsec profile < ipsec_profile_name >
device(config)# interface tunnel < tunnel_id >
device(config-tnif-id)# tunnel source < ipv6_address >         <-- ethernet < port > may be used instead.
device(config-tnif-id)# tunnel destination < ipv6_address >
device(config-tnif-id)# ip address < tunnel_port_ipv6_address >
```

The following example creates an IPv6 IPsec tunnel and applies the previously configured IPsec profile ipv6prof.

```
device# configure terminal
device(config)# interface tunnel 3
device(config-tnif-3)# tunnel mode ipsec ipv6
device(config-tnif-3)# tunnel protection ipsec profile ipv6prof
device(config-tnif-3)# tunnel source 10::1
device(config-tnif-3)# tunnel destination 10::2
device(config-tnif-3)# ipv6 address 10:10:10::1/120
```

The following example outlines the full process of creating an authentication proposal (here, using a previously defined pre-shared key), including the proposal in an IKEv2 profile, associating the IKEv2 profile with an IPsec profile, and configuring an IPsec tunnel protected by the IPsec profile.

```
device# configure terminal
device(config)# ikev2 auth-proposal preshared
device(config-ike-preshared)# pre-shared-key 2
!

!
device(config)# ikev2 profile p2
device(config-ike-profile-p2)# authentication preshared
device(config-ike-profile-p2)# local-identifier address 10::1
device(config-ike-profile-p2)# remote-identifier address 10::2
device(config-ike-profile-p2)# match-identity local address 10::1
device(config-ike-profile-p2)# match-identity remote address 10::2
device(config-ike-profile-p2)# exit
!
device(config)# ipsec profile ipv6prof
device(config-ipsec-profile-ipv6prof)# ike-profile p2
device(config-ipsec-profile-ipv6prof)# exit

device# configure terminal
device(config)# interface tunnel 3
device(config-tnif-3)# tunnel mode ipsec ipv6
device(config-tnif-3)# tunnel protection ipsec profile ipv6prof
device(config-tnif-3)# tunnel source 10::1
device(config-tnif-3)# tunnel destination 10::2
device(config-tnif-3)# ipv6 address 10:10:10::1/120


device(config-tnif-3)#  show running-config interface tunnel 3
!
interface tunnel 3
tunnel mode ipsec ipv6
tunnel protection ipsec profile ipv6prof
tunnel source 10::1
tunnel destination 10::2
ipv6 address 10:10:10::1/120
```

IPsec SPD rules

Each SPD rule requires two ACL rules, one defining the traffic flows on the tunnel, and one defining the traffic flows within the tunnel. The protocol to which the SPD rule applies must be identical for both ACL rules, and the sequence number used with the ACL rules, must be consecutive.

TABLE 11 SPD rules and related actions

SPD rule	Action	Address	Protocol	Sequence number
BYPASS	Permit	Public address	Applicable protocol	N
	Deny	Tunnel internal address	Applicable protocol	N + 1
PROTECT	Deny	Public address	Applicable protocol	N
	Permit	Tunnel internal address	Applicable protocol	N + 1
DISCARD	Deny	Public address	Applicable protocol	N
	Deny	Tunnel internal address	Applicable protocol	N + 1

ACL rules

To process traffic packets, the traffic needs to be segregated using access-list (ACL) rules. The access lists are used to define the traffic pattern rule to drop, permit, or bypass the traffic pattern. An ACL rule can also be set to log matches in the system log.

Extended ACLs allow you to permit or deny packets based on the following information:

- IP protocol
- Source IP address or host name
- Destination IP address or host name
- Source TCP or UDP port (if the IP protocol is TCP or UDP)
- Destination TCP or UDP port (if the IP protocol is TCP or UDP)

The IP protocol can be one of the following well-known names or any IP protocol number from 0 - 255:

- Internet Control Message Protocol (ICMP)
- Internet Group Management Protocol (IGMP)
- Internet Gateway Routing Protocol (IGRP)
- Internet Protocol (IP)
- Open Shortest Path First (OSPF)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

Syntax for ACL rules

The following syntax is used to define ACL rules.

Syntax: [**no**] **ip access-list** { **standard** | **extended** } { *acl-num* | *acl-name* }

ip access-list command parameters and guidelines

The following parameters are used in the **ip access-list** command.

- **standard**: Creates a standard access control list. Contains rules that permit or deny traffic based on source addresses that you specify. The rules are applicable to all ports of the specified address.
- **extended**: Contains rules that permit or deny traffic according to source and destination addresses, as well as other parameters. For example, you can also filter by port, protocol (TCP or UDP), and TCP flags.
- **acl-num**: Specifies the ACL number for a standard or extended access list. The value can be from 1 through 99 for standard IPv4 ACLs and from 100 through 199 for extended IPv4 ACLs.
- **acl-name**: Specifies a unique IPv4 ACL name. The name can be up to 255 characters, and must begin with an alphabetic character. If the name contains spaces, put it within quotation marks. Otherwise, no special characters are allowed, except for underscores and hyphens.

The following guidelines apply to the **ip access-list** command.

You can also create numbered IPv4 ACLs, using the **access-list** command; however, the **ip access-list** command is recommended.

An ACL name must be unique among IPv4 and IPv6 standard and extended ACL types.

After you create an IPv4 ACL, enter one or more **permit** or **deny** commands to create filtering rules for that ACL.

An IPv4 ACL starts functioning only after it is applied to an interface using the **ip access-group** command.

The system supports the following IPv4 ACL resources:

- IPv4 numbered standard ACLs: 99
- IPv4 numbered extended ACLs: 100

- IPv4 named standard ACLs: 99
- IPv4 named extended ACLs: 100
- Maximum filter-rules per IPv4 or IPv6 ACL: 2000. You can change the maximum up to 8192 using the **system-max ip-filter-sys** command.

The **no** form of the command deletes the ACL. You can delete an IPv4 ACL only after you first remove it from all interfaces to which it is applied, using the **no ip access-group** command.

ACL examples

The following example provides syntax for applying an ACL to an interface.

```
device# configure terminal
device(config)# interface ethernet < unit/slot/port >
device(config int-e-xxx-unit/slot/port)# [no] ip access-group < num > [ in | out ]
```

The following rules form a sample ACL.

```
10: permit tcp host 18.1.1.3 host 19.1.1.2 log
11: permit tcp host 18.1.1.2 any log
20: permit tcp any host 19.1.1.4 log
30: permit tcp any any eq 1490 log
40: deny tcp host 18.1.1.5 host 19.1.1.5 log
50: deny tcp host 18.1.1.6 any log
60: deny tcp any host 19.1.1.6 log
70: deny tcp any any eq 1590 log
80: permit ip any any log
```

The following example applies the previously configured ACL called tcp-ex to incoming traffic on port 1/4/4.

```
device# configure terminal
device(config)# interface ethernet 1/4/4
device(config-if-e10000-1/4/4)# ip access-group tcp-ex in
```

When the traffic starts to flow, it is segregated based on the ACL applied to the interface. When the traffic matches a rule configured for the ACL, a syslog message similar to the following messages is generated.

```
SYSLOG: <12> Oct  6 19:09:50 device ACL: ACL: List tcp-ex permitted tcp 18.1.1.8(1024)(Ethernet 1/4/4
0010.9400.0002)
-> 19.1.1.8(1490), 1 event(s)

SYSLOG: <12> Oct  6 19:11:14 device ACL: ACL: List tcp-ex denied tcp 18.1.1.5(1024)(Ethernet 1/4/4
0010.9400.0002)
-> 19.1.1.5(1024), 1 event(s)
```

ACLs can also be defined for IPv6 traffic as shown in the following example.

```
device(config)# ipv6 access-list ipv6_test
device(config-ipv6-access-list ipv6_test)# deny tcp host 2001:DB8:e0bb::2 any eq telnet log
device(config-ipv6-access-list ipv6_test)# permit ipv6 any any log
device(config-ipv6-access-list ipv6_test)# exit
```

Access an interface on which you need to apply the ACL.

```
device(config)# interface ethernet 1/1/1
```

If needed, enable IPv6 on that interface.

```
device(config-if-e1000-1/1/1)# ipv6 enable
```

The following example applies the previously configured ACL access group called ipv6_test to outgoing traffic on port 1/1/1.

```
device(config-if-e1000-1/1/1)# ip access-group ipv6_test out
```

Logging ACL rules

To enable syslog for IPv4 access-lists, add the keyword **log** to an ACL rule, and enable logging on the interface to which the access-list is applied with the **acl-logging** command as shown in the following example.

```
device# configure terminal
device(config)# permit tcp host 18.1.1.3 host 19.1.1.2 log
device(config)# interface ethernet 1/4/4
device(config-if-e10000-1/4/4)# acl-logging
```

To enable syslog for IPv6 access lists, enter the command **logging-enable** as part of the access list in access-list configuration sub-mode, and include the keyword **log** as part of the ACL rule as shown in the following example.

```
device# configure terminal
device(config)# ipv6 access-list acl1
device(config-ipv6-access-list acl1)# logging-enable
device(config-ipv6-access-list acl1)# permit ipv6 any host 19::2 log
```

Configuring Logging and RADIUS Server Hosts

Logging servers

When a logging server is configured, the syslogs are simultaneously stored locally and forwarded to the external syslog server if it is reachable.

If connectivity to the logging server is lost, the device automatically tries to reconnect.

Configuring an SSL profile for use with logging and RADIUS server hosts for NDcPP

Before configuring a server host, you must configure an SSL profile.

1. Name the SSL profile and enter profile configuration mode.

```
device# configure terminal
device(config)# ip ssl profile tls01
```

Syntax: ip ssl profile *profile-name*

Syntax: no ip ssl profile *profile-name*

2. Specify the trustpoint (CA server) that will be associated with the profile.

```
device(config-ssl-tls01)# trustpoint TLS-ABCD
```

Syntax: trustpoint *trustpoint-name*

Syntax: no trustpoint *trustpoint-name*

3. Configure the remote domain name that the end user certificate issues to the server.

```
device(config-ssl-tls01)# remotedomain ruckus.com
```

Syntax: remotedomain *domain-name*

Syntax: no remotedomain *domain-name*

The following example configures the SSL profile tls01 and associates it with the trustpoint TLS-ABCD with ruckus.com as the remote domain name that the end user certificate issues to the server.

```
device# configure terminal
device(config)# ip ssl profile tls01
device(config-ssl-tls01)# trustpoint TLS-ABCD
device(config-ssl-tls01)# remotedomain ruckus.com
```

Logging and RADIUS server host configuration for NDcPP

A FastIron ICX device acting as a syslog client sends out audit logs over a trusted and secure tunnel.

To set up the secure tunnel and authenticate the syslog client and server, you must perform the following set of tasks:

- Configure PKI (Refer to Configuring PKI on page 52 in this guide for more information.)
- Configure one or both of the following:
  - Logging host
  - RADIUS server host for user authentication.

Configuring a logging host for NDcPP

Configure a logging host with the local or remote IP address of the syslog server and its remote port number. On the same line, specify a pre-configured SSL profile.

```
device# configure terminal
device(config)# logging host < ip-address | server-name > ssl-port < port-number > profile < profile-name >
```

Syntax: logging host { *ip-address* | *server-name* } **ssl-port** *port-number* **profile** *profile-name*

When audit logs are generated, the FastIron device establishes a secure TLS tunnel.

During the handshake with the server, the FastIron device receives the server certificate and obtains validation for the certificate from the CA server through the PKI infrastructure.

If validation is successful, the handshake continues to look for the client certificate. If the server has requested a client certificate, the FastIron device sends the client certificate, and the server validates it using Verify protocol logic.

If the client and server certificate validations are successful, the TLS tunnel is established, and audit logs are sent to the server over the secure and trusted tunnel. Subsequent log messages use the established TLS tunnel.

The following example configures a logging host with an IP address of 192.168.10.10. Its SSL port is 5002, and it uses the profile tls03.

```
device# configure terminal
device(config)# logging host 192.168.10.10 ssl-port 5002 profile tls03
```

Configuring a RADIUS server host for NDcPP

Configure a radius-server host for user authentication by specifying its local or remote IP address, the remote port of the RADIUS server, and the name of the previously configured SSL profile to be used.

```
device(config)# radius-server host < ip-address | server-name > ssl-auth-port < port-number > profile <
profile-name > authentication key < radius-key >
```

The following syntax statement includes only information relevant to encryption.

Syntax: radius-server { **host** { *ip-address* | *server-name* } **ssl-auth-port** *port-number* **profile** *profile-name* **authentication key** *radius-key* }

When the user tries to log into the FastIron device, he is first authenticated by the radius server. Before the FastIron device sends out the Radius request to the server, the FastIron device establishes a secure TLS tunnel.

During the handshake with the server, the FastIron device receives the server certificate, and it is validated by the CA server through the PKI infrastructure.

If validation is successful, the handshake continues to look for the client certificate. If the server has requested the client certificate, the FastIron device sends the client certificate, and the server validates it using Verify protocol logic.

If the client and server certificate validations are successful, a TLS tunnel is established, and the Radius authentication request and response are sent over the secure and trusted tunnel.

If TLS tunnel establishment fails, the FastIron device attempts to establish the tunnel and authenticate the user when the user tries to log in again.

The following example configures a RADIUS server host with an IP address of 10.20.158.104. The SSL authentication port is 8001. The profile used is tls03, and the RADIUS authentication key is tesT123$$.

```
device(config)# radius-server host 10.20.158.104 ssl-auth-port 8001 profile tls03 authentication key
tesT123$$
```

Setting up logging hosts for VPN gateway configurations

For VPN gateway configurations, set up logging hosts (Syslog servers) and Radius servers over an IPsec tunnel.

Perform the following steps to configure logging hosts and Radius servers for VPN gateway configurations.

1. Enter global configuration mode.

   ```
   device# configure terminal
   device(config)#
   ```

2. To configure a Syslog server, enter the **logging host** command followed by the IP address of the logging host (Syslog) server.

   The following example configures a logging host with the IP address 192.168.10.10.

   ```
   device(config)# logging host 192.168.10.10
   ```

   Syntax: [no] logging host < logging-host-ip-address >

3. To configure a Radius server, enter the **radius-server host** command followed by identifying parameters. Specify the host IP address, the UDP authentication port, the UDP accounting port, the server as a default available for any AAA server operation, and the key password.

   The following example configures a Radius server with the IP address 15.15.15.3.

   ```
   device(config)# radius-server host 15.15.15.3 auth-port 1812 acct-port 1813 default key tEst1234#
   ```

   Syntax: [no] radius-server host < host-ip-address > auth-port < UDP-port-id > acct-port < UDP-port-id > default key < PASSWORD >

The following example configures both a logging host and a Radius server for use in a VPN gateway configuration.

```
device# configure terminal
device(config)# logging host 192.168.10.10
device(config)# radius-server host 15.15.15.3 auth-port 1812 acct-port 1813 default key tEst1234#
```

Syslog Messages

Syslog messages in FIPS and CC modes

The following table lists some of the syslog messages in FIPS and CC mode.

TABLE 12 FIPS and CC syslog messages

Message level	Message	Explanation
Alert	Time is updated by NTP *server ip-address* from NO_CLOCK to <*new time*> GMT+00 <*new date*>	Indicates time is updated by an NTP server.
Alert	Clock Changed from old time <*old time*> GMT +00 <*old date*> to new time <*new time*> GMT +00 <*new date*>	Indicates time is updated using the **clock set** command.
Informational	FIPS: [primary/secondary] image verification success	The image copy and verification to primary or secondary flash are successful.
Informational	FIPS: [primary/secondary] image verification failed	Image verification in primary or secondary flash has failed.
Informational	SSH login by *user* from *src* IP *ip-address*, *src* MAC *mac-address* to USER EXEC mode using RSA as Server Host Key.	Indicates entry into the "user exec" mode for all sessions for the mentioned user. Similar message is logged for “privileged exec” mode.
Informational	Certificate has expired.	Certificate time (validity period) expired.
Informational	Certificate signature failure.	Signature is not valid.
Informational	Unsupported certificate purpose.	Extended Key Usage support does not have expected key purposes.
Informational	Revoked.	Certificate is revoked by CA (applies to both chain/non-chained case).
Informational	Unable to get local issuer certificate.	Wrong root certificate in a certificate chain.
Informational	Hostname mismatch.	Configured DN value doesn’t match with peer certificate remote DN.
Informational	OCSP purpose missing in responder certificate.	OCSP response doesn’t have OCSP signing bit set.
Informational	SSH logout by *user* from *src* IP *ip-address*, *src* MAC *mac-address* from USER EXEC mode using RSA as Server Host Key.	Indicates exit from "user exec" mode for all sessions for the mentioned user. Similar message is logged for “privileged exec” mode.
Informational	SSH timed out by admin from *src* IP *ip-address* from *src* MAC *mac_address* from USER EXEC mode using RSA as Server Host Key.	The SSH session connected to the specified IP address has timed out.
Informational	SSH session closed by *user* from *src* IP *ip-address*, MAC *mac-address* in PRIVILEGED EXEC mode.	Indicates SSH logout has occurred due to termination. Similar message is logged for “user exec” mode.
Informational	SSH session killed for *user src* IP *ip-address*, MAC *mac-address* in PRIVILEGED EXEC mode.	Indicates SSH logout has occurred because the session was killed.
Informational	SSH session 1 from src IP *ip-address* Algorithm Negotiation Failed	SSH session not established for specified source due to negotiation failure.
Informational	SSH Server session 1 initiated key-exchange due to MAX DATA	SSH session key re-exchange was initiated because the specified data limit was reached.

TABLE 12 FIPS and CC syslog messages (continued)

Message level	Message	Explanation
Informational	SSH Server session 1 initiated key-exchange due to MAX Time	SSH session key re-exchange was initiated because the specified time limit expired.
Informational	Super user login success in console session.	Indicates user has logged in with super user password.
Informational	Console timed out by super from PRIVILEGED EXEC mode	Indicates the timeout of a user session at the local console.
Informational	username : *user* is disabled	Specified user account is locked.
Informational	username : *user* is enabled	Specified user account has been unlocked.
Informational	Console login by user *user* failed	Console login for the specified user failed, possibly due to incorrect username or password.
Informational	SSH access by user *user* from src IP ip-address rejected, # attempt(s)	SSH login for the designated user has failed after the specified number of attempts.
Informational	Logging CLI_CMD operation enabled by *user* from console session. "logging cli-command" by *user* from console.	Indicates audit log **logging cli-command** command is enabled.
Informational	Logging CLI_CMD operation disabled by *user* from console session.	Indicates audit log **logging cli-command** command is disabled.
Informational	"reload" by un-authenticated user from console	Indicates initiation of device reload through console.
Informational	<Device_Hostname> SSL session from src IP: *ip-address* failed due to remote disconnect.	Indicates SSL connection failure.
Informational	SSL server *ip-address:port_number* is now connected	Indicates encrypted syslog or radius server is connected in the server end.
Informational	SSL server *ip-address:port_number* is now disconnected	Indicates encrypted syslog or radius server is disconnected in the server end.
Informational	SSH login by *user* from *src* IP *ip-address* from *src* MAC *mac-address* to USER EXEC mode using RSA as Server Host Key. Device# scp -t file: secondary.sig Device# transfer to device completed SSH logout by *user* from *src* IP *ip-address* from *src* MAC *mac-address* from USER EXEC mode using RSA as Server Host Key.	Indicates the SCP transfer.
Informational	*yyyy month dd hh:mm:ss*	Indicates the timestamp format that is used in syslog messages.
Informational	device(config) # write memory Message: "write memory" by *user* from console.	Audit log will display the commands in expanded form.
Informational	console login by *user* to USER EXEC mode.	Displays all "login" events including the user and session details. Similar message is logged for “logout” events and “privileged exec” mode.
Informational	Interface ipsec_tnnl <tunnel_id>, state up	The IPsec tunnel interface has come up .
Informational	Interface ipsec_tnnl <tunnel_id>, state down <reason>	The IPsec tunnel interface has gone down. Reasons that may be displayed: • clear IKE SA

TABLE 12 FIPS and CC syslog messages (continued)

Message level	Message	Explanation
		• clear IPSEC SA • IKE session down • IPSEC session down • tunnel source interface down • tunnel no destination orute • administratively brought down • IPSEC card down • switchover and failover
Informational	CLI CMD: "ip ssl profile <profile_name>" by user from console	Indicates SSL profile is created.
Informational	CLI CMD: "trustpoint < trustpoint_name>" by user from console	Indicates trustpoint is associated with SSL profile.
Informational	CLI CMD: "remotedomain <remotedomain>" by user from console	Indicates remotedomain is associated with SSL profile.
Informational	Configuration for radius-server host *ip-address* has been enable	Added radius server host configuration.
Informational	CLI CMD: " no radius-server host *ip-address* " by user from console	Removed radius server host configuration.
Informational	System: Syslog server *ip-address* added by user from console session.	Added syslog server host configuration.
Informational	System: Syslog server *ip-address* deleted by user from console session.	Removed syslog server host configuration.
Informational	SSL Handshake to server *ip-address:port_number* is failed due to Bad certificate	The server certificate is not valid for the following reasons. • Invalid/revoked server certificate • CN mismatch • SAN mismatch
Informational	PKI: Certificate validation for trustpoint <trustpoint_name> success	The server certificate has been validated successfully.
Informational	PKI: Validation response trustpoint <trustpointname>. status: failed error: certificate validation failed error: certificate has expired	Certificate time (validity period) has expired.
Informational	PKI: Validation response trustpoint <trustpointname>. status: failed error: certificate validation failed error: certificate signature failure	The signature in the received certificate is not valid.
Informational	PKI: Unsupported certificate purpose - Extended key usage validation for trustpoint< Trustpointname> failed	Extended Key Usage support does not have expected key purposes.
Informational	PKI: ocsp reply received from ocsp responder: <OCSPResponder> for trustpoint <trustpointName>. :revoke_status: Certificate not valid, Reason : Revoked, code : 6	Certificate is revoked by CA (applies to both chain/non-chained case).
Informational	PKI: certificate validation: failed, Error: unable to get local issuer certificate	Wrong root certificate in a certificate chain.
Notification	IKEv2: Phase1 failed . Hostname mismatch Source <TunnelSource> Destination	Configured DN value doesn't match peer certificate remote DN.

TABLE 12 FIPS and CC syslog messages (continued)

Message level	Message	Explanation
	<TunnelDestination> VRF <VRFID> Tunnel <TunnelID>	
Informational	PKI: ocsp reply received from ocsp responder: <OCSPResponder> for trustpoint <trustpointname>.:revoke_status: failed error:OCSP purpose missing in responder certificate	OCSP Response doesn't have OCSPSigning bit set.
Informational	PKI: authenticate response received: trustpoint <trustpointName>. auth-status:failed. error:fingerprint match failed	There is a fingerprint mismatch between the retrieved fingerprint and configured fingerprint.
Informational	ACL: List <Access-list name> permitted <Protocol> <sourceIP> <source port number> (Ethernet <Ethernet interface> <source MAC address> -> <destination IP><Destination Port Number >	The traffic from given <source IP, source port> to <destination IP, destination port> is permitted by the given <Access-list name>, which is applied on the given Ethernet interface.
Informational	ACL: List <Access-list name> denied <Protocol> <sourceIP><source port number>(Ethernet <Ethernet interface> <source MAC address> -> <destination IP><Destination Port Number>	The traffic from given <source IP, source port> to <destination IP, destination port> is denied by the given <Access-list name>, which is applied on the given Ethernet interface.

OpenSSL License

OpenSSL License Overview

NOTE
This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

NOTE
OpenSSL has been compiled without the Heartbeat extension.

License

This is a copy of the current LICENSE file inside the CVS repository.

```
LICENSE ISSUES
==============
    The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
    the OpenSSL License and the original SSLeay license apply to the toolkit.
    See below for the actual license texts. Actually both licenses are BSD-style
    Open Source licenses. In case of any license issues related to OpenSSL
    please contact openssl-core@openssl.org.

OpenSSL License
---------------

/* ====================================================================
 * Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1.Redistributions of source code must retain the above copyright
 *         notice, this list of conditions and the following disclaimer.
 *
 * 2.Redistributions in binary form must reproduce the above copyright
 *         notice, this list of conditions and the following disclaimer in
 *         the documentation and/or other materials provided with the
 *         distribution.
 *
 * 3.All advertising materials mentioning features or use of this
 *         software must display the following acknowledgment:
 *         "This product includes software developed by the OpenSSL Project
 *         for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4.The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *         endorse or promote products derived from this software without
 *         prior written permission. For written permission, please contact
 *         openssl-core@openssl.org.
 *
 * 5.Products derived from this software may not be called "OpenSSL"
 *         nor may "OpenSSL" appear in their names without prior written
 *         permission of the OpenSSL Project.
 *
 * 6.Redistributions of any form whatsoever must retain the following acknowledgment:
 *         "This product includes software developed by the OpenSSL Project
 *         for use in the OpenSSL Toolkit (http://www.openssl.org/)"
```

```
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
* ====================================================================
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

Original SSLeay License
-----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to. The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code. The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1.Redistributions of source code must retain the copyright
 *         notice, this list of conditions and the following disclaimer.
 * 2.Redistributions in binary form must reproduce the above copyright
 *         notice, this list of conditions and the following disclaimer in the
 *         documentation and/or other materials provided with the distribution.
 * 3.All advertising materials mentioning features or use of this software
 *         must display the following acknowledgement:
 *         "This product includes cryptographic software written by
 *         Eric Young (eay@cryptsoft.com)"
 *         The word 'cryptographic' can be left out if the rouines from the library
 *         being used are not cryptographic related :-).
 * 4.If you include any Windows specific code (or a derivative thereof) from
 *         the apps directory (application code) you must include an acknowledgement:
 *         "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
```

```
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
```

Ruckus Wireless, Inc., a wholly owned subsidiary of CommScope, Inc.
350 West Java Dr., Sunnyvale, CA 94089 USA
www.ruckuswireless.com