Configure MACsec

This chapter describes how to configure MACsec on Cisco NX-OS devices.

About MACsec

Media Access Control Security (MACsec) an IEEE 802.1AE along with MACsec Key Agreement (MKA) protocol provide secure communications on Ethernet links. It offers the following :

- Provides line rate encryption capabilities.
- Helps to ensure data confidentiality by providing strong encryption at Layer 2.
- Provides integrity checking to help ensure that data cannot be modified in transit.
- Can be selectively enabled using a centralized policy to help ensure that it is enforced where required while allowing non-MACsec-capable components to access the network.

- Encrypts packets on a hop-by-hop basis at Layer 2, allowing the network to inspect, monitor, mark, and forward traffic according to your existing policies (unlike end-to-end Layer 3 encryption techniques that hide the contents of packets from the network devices they cross).

Key Lifetime and Hitless Key Rollover

A MACsec keychain can have multiple pre-shared keys (PSKs), each configured with a key ID and an optional lifetime. A key lifetime specifies at which time the key activates and expires. In the absence of a lifetime configuration, the default lifetime is unlimited. When a lifetime is configured, MKA rolls over to the next configured pre-shared key in the keychain after the lifetime is expired. The time zone of the key can be local or UTC. The default time zone is UTC.

To configure a MACsec keychain, see Configuring a MACsec Keychain and Keys, on page 9.

A key can roll over to a second key within the same keychain by configuring the second key (in the keychain) and configuring a lifetime for the first key. When the lifetime of the first key expires, it automatically rolls over to the next key in the list. If the same key is configured on both sides of the link at the same time, then the key rollover is hitless (that is, the key rolls over without traffic interruption).

Fallback Key

A MACsec session can fail due to a key/key name (CKN) mismatch or a finite key duration between the switch and a peer. If a MACsec session does fail, a fallback session can take over if a fallback key is configured. A fallback session prevents downtime due to primary session failure and allows a user time to fix the key issue causing the failure. A fallback key also provides a backup session if the primary session fails to start. This feature is optional.

To configure a MACsec fallback key, see Configuring MACsec Fallback Key, on page 11.

Licensing Requirements for MACsec

Product	License Requirement
Cisco NX-OS	MACsec requires a Security license. For a complete explanation of the Cisco NX-OS licensing scheme to obtain and apply licenses, see the Cisco NX-OS Licensing Guide.

Guidelines and Limitations for MACsec

MACsec has the following guidelines and limitations:

- MACsec is supported on the following interface types:
  - Layer 2 switch ports (access and trunk)
  - Layer 3 routed interfaces (no subinterfaces)

Note Enabling MACsec on the Layer 3 routed interface also enables encryption on all the subinterfaces that are defined under that interface. However, selectively enabling MACsec on a subset of subinterfaces of the same Layer 3 routed interface is not supported.

  - Layer 2 and Layer 3-port channels (no subinterfaces)

- Beginning with Cisco Nexus Release 10.2(1)F, Secure Channel Identifier (SCI) can be disabled from MACSec security tag (SecTAG) on Cisco Nexus 9000 ToR switches.
  - It is supported in FX2 and FX3 platforms.
  - It is supported in FX platforms with XPN cipher suites only
- When the Cisco Nexus ToR switches are downgraded from Cisco NX-OS Release 9.3.7 to Cisco NX-OS Release 9.3.6 and below releases, MACsec is not supported.
- MKA is the only supported key exchange protocol for MACsec. The Security Association Protocol (SAP) is not supported.
- Link-level flow control (LLFC) and priority flow control (PFC) are not supported with MACsec.
- Multiple MACsec peers (different SCI values) for the same interface are not supported.
- You can retain the MACsec configuration when you disable MACsec using the **macsec shutdown** command.
- MACsec sessions are liberal in accepting packets from a key server whose latest Rx and latest Tx flags have been retired after Tx SA installation for the first time. The MACsec session then converges into a secure state.
- Beginning with Cisco NX-OS Release 9.2(1), the following configurations are allowed:
  - Allowing MACSec policy to be modified while the policy is referenced by an interface.
  - Allowing different MACsec policies across different lanes of a breakout port.
- Beginning with Cisco Nexus Release 9.2(1), MACsec is supported on Cisco Nexus 93180YC-FX and Cisco Nexus 3264C-E switches.
- Beginning with Cisco Nexus Release 9.3(1), MACsec is supported on the Cisco Nexus 9364C, 9332C, and 9348GC-FXP switches. The following limitations are applicable when you use MACsec with these switches:
  - Cisco Nexus C9364C—MACsec is supported on 16 ports (Ports 49–64).
  - Cisco Nexus C9332C—MACsec is supported on 8 ports (Ports 25–32).
  - Cisco Nexus 9348GC-FXP—MACsec is supported on 6 ports (Ports 49–54).

Note On the Cisco Nexus 9364C, and 9332C platform switches, when MACsec is either configured or unconfigured on a port, there will be a port-flap occurrence irrespective of MACsec security-policy type.

- Beginning with Cisco Nexus Release 9.3(1), you cannot apply MACsec configuration directly on port-channel interface. However, you can apply MACsec configurations directly on port-channel member ports. This applies to both NX-OS and vPC port-channels.
- Beginning with Cisco Nexus Release 9.3(3), MACsec is supported on Cisco Nexus 93216TC-FX2, Cisco Nexus 93360YC-FX2.
- Beginning with Cisco NX-OS Release 9.3(5), MACsec is supported on the following switches and line cards:
  - Cisco Nexus 93180YC-FX3S switches - MACsec is supported on all ports.
  - Cisco Nexus X9732C-FX, and X9788TC-FX line cards
- The N9K-X9736C-FX, N9K-X9732C-FX, N9K-C9348GC-FXP, N9K-C93180YC-FX, N9K-C93108TC-FX, N9K-X9788TC-FX, N9K-C9336C-FX2, N9K-C93240YC-FX2, N9K-C93216TC-FX2, N9K-C93360YC-FX2, N9K-C9364C, and N9K-C9332C cards and switches do not support MACsec on 1G ports. MACsec is not supported on any port on a mac block that has 1G ports on it.
- Beginning with Cisco NX-OS Release 10.1(1), the Cisco Nexus 93180YC-FX3, and 93108TC-FX3P switches support MACsec on all port speeds including 1G and 10G port speeds.
- MACsec is supported on Cisco Nexus 93240YC-FX2, 9336C-FX2, 93108TC-FX, 93180YC-FX switches and the X9736C-FX, and X9732C-EXM line cards.
- Cisco Nexus 9000 Series switches do not support MACsec on any of the MACsec capable ports when QSA is being used.
  - Beginning with Cisco NX-OS Release 9.3(7), MACsec is supported by Cisco Nexus 9364C and 9336C-FX2 switches when QSA is being used.
  - Beginning with Cisco NX-OS Release 10.1(1), MACsec is supported by Cisco Nexus 9336C-FX2, 9336C-FX2-E, and 9364C switches when QSA is being used.
  - Beginning with Cisco NX-OS Release 10.1(2), MACsec is supported by Cisco Nexus 9300-FX3 platform switches when QSA is being used.
- Beginning with Cisco Nexus Release 10.1(1), MACsec is supported on Cisco Nexus 9336C-FX2-E.
- Beginning with Cisco Nexus Release 10.2(1)F, MACsec is supported on Cisco Nexus X9716D-GX.
- Beginning with Cisco NX-OS Release 10.2(1q)F, MACsec is supported on ports 25-32 of Cisco Nexus 9332D-GX2B switches.
- Beginning with Cisco NX-OS Release 10.2(2)F, MACsec is supported on the Cisco Nexus N9K-C9348D-GX2A switches on 1-48 ports.
- Beginning with Cisco NX-OS Release 10.2(2)F, MACsec supports Cisco Nexus X9736C-FX, and X9736Q-FX line cards with 10G QSA links.

- Beginning with Cisco NX-OS Release 10.2(2)F, MACsec is supported on ports 1-16 of Cisco Nexus 9364D-GX2A switches.
- On the Cisco Nexus 9332D-GX2B, 9364D-GX2A and 9348D-GX2A switches and Cisco Nexus X9836DM-A line card, when MACsec is either configured or unconfigured on a port, a port-flap occurs irrespective of MACsec security-policy type.
- Beginning with Cisco NX-OS Release 10.3(1)F, MACsec is supported on Cisco Nexus X9836DM-A line card of Cisco Nexus 9800 platform switches.
- Beginning with Cisco NX-OS Release 10.3(2)F, MACsec is supported on Cisco Nexus 9408 switches with LEM modules X9400-16W and X9400-8D on all supported links.
- Beginning with Cisco Nexus Release 10.3(3)F, the cipher key enforcement feature provides the option to define the supported cipher suites from the most preferred to the least preferred on the Cisco Nexus 9332D-GX2B, 9336C-FX2, 93180YC-FX, and 93180YC-FX3 switches with following limitations:
  - Cipher Key Enforcement feature will work effectively only if it is prioritized as key server, else it will be in **init** or **pending** state of the session.
  - Cipher Key Enforcement feature is only supported for direct connections between 2 peers. If MKA session is with multiple peers, this feature is not expected to work properly.
  - During a peer cipher suite allowance change, session may not secure on the most preferred supported cipher suite.
  - When changing cipher from any to an enforced-peer cipher on a policy that is used on any secured MACsec session, it is recommended to flap the port after changing the cipher to reach expected behavior. If flapping is not done, the session shows secured on the switch while peer session shows pending on unsupported ciphers. It could also cause session to not get secured immediately even if supported cipher is present in enforce-peer cipher suites.
  - The Allowed Peer Cipher Suites (APSC) can not be empty or cannot have duplicates.
  - Cipher-suite and cipher-suite enforce-peer commands cannot coexist under the same policy.
  - While waiting for SAK Cipher-Enforcing Timer to timeout to try the next cipher suite in line, the data and control traffic might face one way traffic interruptions even with should secure mode. The interruption will only recover when session is secured.
- Beginning with Cisco Nexus Release 10.4(1)F, MACsec is supported on ports 49 to 54 of Cisco Nexus 9348GC-FX3 and 9348GC-FX3PH switches.
- Beginning with Cisco NX-OS Release 10.4(1)F, MACsec is supported on all front panel ports (port 1 to 32) of Cisco Nexus 9332D-H2R platform switches. However, MACsec is not supported on Ethernet1/33 and Ethernet1/34.
- Beginning with Cisco Nexus Release 10.4(2)F, MACsec is supported on the below switches:
  - Cisco Nexus 93400LD-H1 on all ports.
  - Cisco Nexus 93108TC-FX3 on ports 49 to 54.
- Beginning with Cisco Nexus Release 10.4(3)F, MACsec is supported on the Cisco Nexus 9364C-H1 switches on ports 49 to 64.
- If the MACsec feature is configured, non-disruptive ISSU is not supported.

Keychain restrictions:

- You cannot overwrite the octet string for a MACsec key. Instead, you must create a new key or a new keychain.
- A new key in the keychain is configured when you enter **end** or **exit**. The default timeout for editor mode is 6 seconds. If the key is not configured with the key octet string or/and the send lifetime within the 6-second window, incomplete information may be used to bring up the MACsec session and could result in the session being stuck in an Authorization Pending state. If the MACsec sessions are not converged after the configuration is complete, you might be advised to shut/no shut the ports.
- For a given keychain, key activation times should overlap to avoid any period of time when no key is activated. If a time period occurs during which no key is activated, session negotiation fails and traffic drops can occur. The key with the latest start time among the currently active keys takes precedence for a MACsec key rollover.
- In addition to enabling the MACsec feature, a MACsec keychain must be configured on at least one interface to consume the security add-on license.

Fallback restrictions:

- If a MACsec session is secured on an old primary key, it does not go to a fallback session in case of mismatched latest active primary key. So the session remains secured on the old primary key and will show as rekeying on the old CA under status. And the MACsec session on the new key on primary PSK will be in init state.
- Use only one key with infinite lifetime in the fallback key chain. Multiple keys are not supported.
- The key ID (CKN) used in the fallback key chain must not match any of the key IDs (CKNs) used in the primary key chain.
- Once configured, fallback configuration on an interface cannot be removed, unless the complete MACsec configuration on the interface is removed.

MACsec policy restrictions:

- BPDU packets can be transmitted before a MACsec session becomes secure.

Layer 2 Tunneling Protocol (L2TP) restrictions:

- MACsec is not supported on ports configured for dot1q tunneling or L2TP.
- L2TP does not work if STP is enabled on trunk ports for non-native VLANs.

Statistics restrictions:

- Few CRC errors should occur during the transition between MACsec and non-MACsec mode (regular port shut/no shut).
- Secy statistics are cumulative and polled every 30 seconds.
- The IEEE8021-SECY-MIB OIDs secyRxSAStatsOKPkts, secyTxSAStatsProtectedPkts, and secyTxSAStatsEncryptedPkts can carry only up to 32 bits of counter values, but the traffic may exceed 32 bits.
- On Cisco Nexus 9300-FX3 platform switches, the **show macsec secy statistics** command supports rate statistics and the following rate related "CISCO-SECY-EXT-MIB" OIDs beginning with Cisco NX-OS Release 10.4(2)F.

- cseSecyIfRxUncontrolledPktRate,
- cseSecyIfRxControlledPktRate,
- cseSecyIfTxUncontrolledPktRate,
- cseSecyIfTxControlledPktRate
- cseSecyIfRxControlledOctetRate
- cseSecyIfTxControlledOctetRate
- cseSecyIfRxUnControlledOctetRate
- cseSecyIfTxUnControlledOctetRate

Interoperability restrictions:

- Interoperability of N9K-X9732C-EXM and other peer switches (other Cisco and non-Cisco switches) is supported only with the XPN cipher suite.
- MACsec peers must run the same Cisco NX-OS release in order to use the AES_128_CMAC cryptographic algorithm. For interoperability between previous releases and Cisco NX-OS Release 9.2(1), you must use keys with the AES_256_CMAC cryptographic algorithm.
- For interoperability between previous releases and Cisco NX-OS Release 9.2(1), pad the MACsec key with zeros if it is less than 32 octets.
- On any Cisco NX-OS switch, you can configure only one unique combination of an alternate MAC address and Ethernet type on all interfaces.
- When using 1G optics on MACSEC capable module, it is recommended to change diagnostics mode to 'minimal'.
- When you attempt to downgrade from Cisco NX-OS Release 9.3(1) to a Cisco NX-OS release without per port channel member MACsec configuration support, when the switch has MACsec configurations on members of the same port channel interface that are different from each other, you may see the following error message:

```
Asymmetric macsec config is present on port-channel members. Please use
symmetric macsec config across members to perform Non-disruptive ISSU.
```

- Software support for MACsec and 50G is not available on the Cisco Nexus X9400-22L LEM card.

EAPOL has the following guidelines and limitations:

- In Cisco NX-OS Release 9.3(1), EAPOL configuration is not supported on Cisco Nexus 9332C and 9364C Series switches.
- Within the same slice of the forwarding engine, EAPOL ethertype and dot1q ethertype cannot have the same value.
- For enabling EAPOL configuration, the range of ethernet type between 0 to 0x599 is invalid.
- For enabling EAPOL configuration, on N9K-X9836DM-A line card, the only supported EAPOL mac addresses are range 0x0180c2000000 to 0x0180c20000ff.

- While configuring EAPOL packets, the following combinations must not be used:
  - Mac address 0100.0ccd.cdd0 with any ethertype
  - Any mac address with Ether types: 0xfff0, 0x800, 0x86dd
  - The default destination MAC address, 0180.c200.0003 with the default Ethernet type, 0x888e
  - Different EAPOL DMAC addresses on both MACsec peers. The MACsec session works only if the MACsec peer is sending MKAPDUs with the DMAC configured locally.
- Beginning with Cisco NX-OS Release 10.2(1)F, EAPOL is supported on Cisco Nexus 9300-FX3 Series switches.

Enabling MACsec

Before you can access the MACsec and MKA commands, you must enable the MACsec feature.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature macsec** **Example:** `switch(config)# feature macsec`	Enables MACsec and MKA on the device.
Step 3	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Disabling MACsec

Beginning with Cisco NX-OS Release 9.2(1), disabling the MACsec feature only deactivates this feature and does not remove the associated MACsec configurations.

Disabling MACsec has the following conditions:

- MACsec shutdown is global command and is not available at the interface level.
- The macsec shutdown, show macsec mka session/summary, show macsec mka session detail, and show macsec mka/secy statistics commands will display the 'Macsec is shutdown' message. However, the show macsec policy and show key chain commands will display the output.

- Consecutive MACsec status changes from macsec shutdown to no macsec shutdown and vice versa needs a 30 seconds time interval in between the status change.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**macsec shutdown** **Example:** `switch(config)# macsec shutdown`	Disables the MACsec configuration on the device. The **no** option restores the MACsec feature.
Step 3	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Copies the running configuration to the startup configuration. This step is required only if you want to retain the MACsec in the shutdown state after the switch reload.

Configuring a MACsec Keychain and Keys

You can create a MACsec keychain and keys on the device.

Note Only MACsec keychains will result in converged MKA sessions.

Before you begin

Make sure that MACsec is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	(Optional) [**no**] **key-chain macsec-psk no-show** **Example:** `switch(config)# key-chain macsec-psk no-show`	Hides the encrypted key octet string in the output of the **show running-config** and **show startup-config** commands by replacing the string with a wildcard character. By default, PSK keys are displayed in encrypted format and

	Command or Action	Purpose
		can be easily decrypted. This command applies only to MACsec keychains. **Note** The octet string is also hidden when you save the configuration to a file.

Step 3	**key chain** *name* **macsec**
	Example:
	`switch(config)# key chain 1 macsec`
	`switch(config-macseckeychain)#`
		Creates a MACsec keychain to hold a set of MACsec keys and enters MACsec keychain configuration mode.

Step 4	**key** *key-id*
	Example:
	`switch(config-macseckeychain)# key 1000`
	`switch(config-macseckeychain-macseckey)#`
		Creates a MACsec key and enters MACsec key configuration mode. The range is from 1 to 32 octets, and the maximum size is 64.
		Note The key must consist of an even number of characters.

Step 5	**key-octet-string** *octet-string* **cryptographic-algorithm** {**AES_128_CMAC** | **AES_256_CMAC**} **Example:** `switch(config-macseckeychain-macseckey)# key-octet-string abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789 cryptographic-algorithm AES_256_CMAC`	Configures the octet string for the key. The *octet-string* argument can contain up to 64 hexadecimal characters. The octet key is encoded internally, so the key in clear text does not appear in the output of the **show running-config macsec** command. The key octet string includes the following: • 0 Encryption Type - No encryption (default) • 6 Encryption Type - Proprietary (Type-6 encrypted). For more information, see Enabling Type-6 Encryption on MACsec Keys. • 7 Encryption Type - Proprietary WORD key octet string with maximum 64 characters **Note** MACsec peers must run the same Cisco NX-OS release in order to use the AES_128_CMAC cryptographic algorithm. To interoperate between previous releases and Cisco NX-OS Release 7.0(3)I7(2) or a later release, you must use keys with the AES_256_CMAC cryptographic algorithm.

	Command or Action	Purpose
Step 6	**send-lifetime** *start-time* **duration** *duration* **Example:** `switch(config-macseckeychain-macseckey)# send-lifetime 00:00:00 Oct 04 2016 duration 100000`	Configures a send lifetime for the key. By default, the device treats the start time as UTC. The *start-time* argument is the time of day and date that the key becomes active. The *duration* argument is the length of the lifetime in seconds. The maximum length is 2147483646 seconds (approximately 68 years).
Step 7	(Optional) **show key chain** *name* **Example:** `switch(config-macseckeychain-macseckey)# show key chain 1`	Displays the keychain configuration.
Step 8	(Optional) **copy running-config startup-config** **Example:** `switch(config-macseckeychain-macseckey)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

MACsec Packet-Number Exhaustion

Every MACsec frame contains a 32-bit packet number (PN), and it is unique for a given Security Association Key (SAK). Upon PN exhaustion (after reaching 75% of 2^{32}- 1), SAK rekey takes place automatically to refresh the data plane keys and the PN will wrap around.

For example, on 10G full line rate @ 64 bytes, the SAK rekey will occur every 216 seconds due to PN exhaustion.

This is applicable when using GCM-AES-PN-128 or GCM-AES-PN-256 cipher-suites.

When GCM-AES-XPN-128 or GCM-AES-XPN-256 cipher-suite is used , the SAK rekey happens automatically when reaching 75% of 2^{64}- 1 , which will take several years to exhaust the packet numbering. The cipher-suite is configurable under the macsec policy and the operational cipher-suite is determined by the key-server device.

It is recommended to use XPN ciphersuite on N9K-X9732C-EXM line card

Configuring MACsec Fallback Key

Beginning with Cisco NX-OS Release 9.2(1), you can configure a fallback key on the device to initiate a backup session if the primary session fails as a result of a key/key name (CKN) mismatch or a finite key duration between the switch and peer.

Before you begin

Make sure that MACsec is enabled and a primary and fallback keychain and key ID are configured. See Configuring a MACsec Keychain and Keys.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters the global configuration mode.

Step 2	**interface** *name*
	Example:
	`switch(config)# interface ethernet 1/1`
	`switch(config-if)#`
		Specifies the interface that you are configuring. You can specify the interface type and identity. For an Ethernet port, use ethernet slot/port.

Step 3	**macsec keychain** *keychain-name* **policy** *policy-name* **fallback-keychain** *keychain-name* **Example:** `switch(config-if)# macsec keychain kc2 policy abc fallback-keychain fb_kc2`	Specifies the fallback keychain to use after a MACsec session failure due to a key/key ID mismatch or a key expiration. The fallback key ID should not match any key ID from a primary keychain. Fallback keychain configuration for each interface can be changed on the corresponding interface, without removing the MACsec configuration, by reissuing the same command with the fallback keychain name changed. **Note** The command must be entered exactly the same as the existing configuration command for the interface, except for the fallback keychain name. See Configuring a MACsec Keychain and Keys.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring a MACsec Policy

You can create multiple MACsec policies with different parameters. However, only one policy can be active on an interface.

Before you begin

Make sure that MACsec is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**macsec policy** *name*
	Example:
	`switch(config)# macsec policy abc`
	`switch(config-macsec-policy)#`
		Creates a MACsec policy.

Step 3	(Optional) **[no] cipher-suite** { { **enforce-peer** <*allowed-peer-cipher-suite1*> [*allowed-peer-cipher-suite2*> [*allowed-peer-cipher-suite3*> [*allowed-peer-cipher-suite4*>]]] } | <**suite**>}
	Example:
	`switch(config-macsec-policy)#`
	`cipher-suite enforce-peer`
	`GCM-AES-XPN-256 GCM-AES-XPN-128`
		Configures the sequence for the following cipher suites from the most preferred to the least preferred. The session gets secured on the most preferred cipher suite that is supported by the peer.: GCM-AES-128, GCM-AES-256, GCM-AES-XPN-128, or GCM-AES-XPN-256.
		To unconfigure, you can either use the **no** form or overwrite the existing sequence with the required sequence preference.
		Note
		• For this feature to work, ensure that the Cisco NX-OS switch is set as key server.
		• If the peer supports cipher suites that are not included in the set of cipher suites defined in the **cipher-suite enforce-peer** command, the MKA session state will not be secured instead it will be pending.

Step 4	(Optional) **[no] include-sci**
	Example:
	`switch(config-macsec-policy)# no`
	`include-sci`
		Disables SCI in SecTAG. By default, SCI is always enabled.
		Note
		To prevent packet drops, ensure that SCI tagging settings are consistent at both ingress and egress points.

Step 5	(Optional) **key-server-priority** *number*
	Example:
	`switch(config-macsec-policy)#`
	`key-server-priority 0`
		Configures the key server priority to break the tie between peers during a key exchange. The range is from 0 (highest) and 255 (lowest), and the default value is 16.

	Command or Action	Purpose
Step 6	(Optional) **security-policy** *name* **Example:** `switch(config-macsec-policy)# security-policy should-secure`	Configures one of the following security policies to define the handling of data and control packets: • **must-secure**—Packets not carrying MACsec headers will be dropped. • **should-secure**—Packets not carrying MACsec headers will be permitted. This is the default value.
Step 7	(Optional) **window-size** *number* **Example:** `switch(config-macsec-policy)# window-size 512`	Configures the replay protection window such that the secured interface will not accept any packet that is less than the configured window size. The range is from 0 to 596000000.
Step 8	(Optional) **sak-expiry-time** *time* **Example:** `switch(config-macsec-policy)# sak-expiry-time 100`	Configures the time in seconds to force an SAK rekey. This command can be used to change the session key to a predictable time interval. The default is 0.
Step 9	(Optional) **conf-offset** *name* **Example:** `switch(config-macsec-policy)# conf-offset CONF-OFFSET-0`	Configures one of the following confidentiality offsets in the Layer 2 frame, where encryption begins: CONF-OFFSET-0, CONF-OFFSET-30, or CONF-OFFSET-50. This command might be necessary for intermediate switches to use packet headers {dmac, smac, etype} like MPLS tags.
Step 10	(Optional) **show macsec policy** **Example:** `switch(config-macsec-policy)# show macsec policy`	Displays the MACsec policy configuration.
Step 11	(Optional) **copy running-config startup-config** **Example:** `switch(config-macsec-policy)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring MACsec EAP

Beginning with Cisco NX-OS Release 10.4(1)F, you can use MACsec EAP profile for 802.1X authentication.

Before you begin

- Enable the 802.1X feature on the Cisco NX-OS device.

• Configure MACsec command which specifies should-secure (default) or must-secure macsec policy

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface ethernet** *slot / port*
	Example:
	`switch(config)# interface ethernet 1/30`
	`switch(config-if)#`
		Selects the interface to configure and enters interface configuration mode.

Step 3	**[no] macsec eap policy** *policy name*
	Example:
	`switch(config-if)# macsec eap policy P1`
	`switch(config-eap-profile)#`
		Creates the MACsec eap profile.
		The **no** form of the command is used to disable the MACsec eap profile.

Step 4	**[no] dot1x supplicant eap profile** *eap profile name* } **Example:** `switch(config-if)# dot1x supplicant eap profile`	Enters global configuration mode. Configures the eap profile to be used by the supplicant.

QKD integration with SKIP on MACsec

About Quantum-Safe Encryption

Recent advancements in quantum computing have exposed vulnerabilities in various cryptographic algorithms, making them unsecured for future applications. The RSA (integer factorization) and DHE (discrete logarithms) public key algorithms, which rely on computational complexity, are now at risk of being solved by quantum computers using Shor's or Grover's algorithm.

As a result, establishing a shared secret key between communicating parties has become a significant challenge. To avoid this issue, you can configure quantum-safe algorithms or implement a Quantum Key Distribution (QKD).

About QKD Integration with Secure Key Integration Protocol

Integrating Secure Key Integration Protocol (SKIP) protocol to the switches empowers to establish communication with external quantum devices. This advancement allows for the utilization of Quantum Key Distribution (QKD) devices in the exchange of MACsec encryption keys between switches.

QKD operates on the principles of quantum physics, utilizing the quantum state of photons to encode and share information through an optical link. Additionally, an authenticated classical channel is used for sharing

measurements. The change in quantum states helps the two end parties of the communication channel to identify any interception of their key.

QKD is a secured key exchange mechanism against quantum attacks even in the future advancements in cryptanalysis or quantum computing. QKD doesn't require continual updates based on discovered vulnerabilities.

Guidelines and Limitations

The integration of QKD with SKIP for MACsec communication has the following guidelines and limitations:

- Beginning with Cisco NX-OS Release 10.4(3)F, Secure Key Integration Protocol is supported on the following Cisco Nexus switches:
  - N9K-C9348GC-FXP
  - N9K-C93216TC-FX2
  - N9K-C93360YC-FX2
  - N9K-C9336C-FX2
  - N9K-C9348GC-FX3
  - N9K-C9348D-GX2A
  - N9K-C9332D-H2R
- You can use the SKIP protocol only in a point-to-point MACsec link encryption scenario.
- The SKIP protocol is available only on the interfaces that support MACsec encryption.
- Ensure that the QKD server is accessible through the management interface if switches have an HTTPS connection that is established with it.
- If MACsec peers are connected to two different QKD servers, the QKD servers synchronize the keys to establish an MKA session. This synchronization ensures that the MACsec key (CKN) and key-string (CAK) are the same at both ends.
- To establish a secure Transport Layer Security (TLS) connection and enable mutual authentication, you must install trustpoint certificates on the switch. These certificates allow the switch to obtain keys from the server. For more information, see chapter Configuring PKI in *Cisco Nexus 9000 Series NX-OS Security Configuration Guide.*
- MACsec PPK session and EAP-TLS sessions are not supported on the same interface.
- QKD MACsec session fails due to configuration or invalid key errors. There is no FALLBACK mechanism for the preshared key (PSK).
- A switch can connect to only one QKD server and one QKD profile per switch.
- For SKIP protocol, only a single remoteSystemID is supported.
- For QKD connectivity, IPv6 is not supported.
- MACsec peers exchanging QKD Keys must be Cisco NX-OS switches.

- Once the MACsec session is established, any modifications to the QKD profile will cause traffic loss in **MUST SECURE MACSEC** mode.
- Once the MACsec session is up with QKD keys derived from KME Server, trustpoint modifications that are part of the QKD profile will not have any effect on the current sessions.
- The remoteSystemID attribute is mandatory for the capability response.

Configuring point-to-point MACsec Link Encryption Using SKIP

In point-to-point MACsec Link Encryption, secure encryption is established by using SKIP in switches. This encryption is set up between two interfaces in peer switches and requires the assistance of a QKD device network. Instead of the switches network, the QKD network shares the MACsec encryption key. Therefore, when a switch is required to create a MACsec link between peer switch interfaces, it contacts the external QKD device and requests the key. The external QKD device then generates a key pair consisting of the key ID and the key.

The Key ID acts as the unique ID string for the key (Shared Secret). The QKD device shares both the key ID and Key with the switch, while the switch only shares the key ID with its peer. The Peer switch uses this Key ID to retrieve encryption keys from its QKD device. Hence, Quantum networks always securely communicate encryption keys.

Enabling Postquantum Cryptography

Before you begin

- Configure MACsec Pre-Shared Key (PSK).
- Configure MACsec in the PPK mode.
- An external QKD device network.
- Add the QKD server CA to the trustpoint in the switch and import the QKD server root CA certificate to the switch.

Procedure

	Command or Action	Purpose

Step 1	switch# **configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	switch(config)# **feature cryptopqc** **Example:** `switch(config)# feature cryptopqc`	Enables post quantum cryptography (cryptopqc) on the switch.

	Command or Action	Purpose
Step 3	(Optional) switch(config)# **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Enabling MACsec and MKA features

Procedure

	Command or Action	Purpose

Step 1	switch# **configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	switch(config)# **feature macsec** **Example:** `switch(config)# feature macsec`	Enables MACsec and MKA on the switch.
Step 3	(Optional) switch(config)# copy running-config startup-config **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Configuring Quantum Key Distribution Profile

Procedure

	Command or Action	Purpose
Step 1	switch (config)# **crypto qkd profile***name* **Example:** `switch(config)# crypto qkd profile ppk1`	Creates a QKD profile with name ppk1.
Step 2	switch (config)# **kme server**<*hostname*/*IP*> **port** *portnumber* **Example:** `switch(config-crypto-qkd-profile)# kme server 172.0.0.2 port 6000`	Configures Key Management Engine (KME)/QKD server IP and TCP port number. **Note** Port number is optional. By default, port number will be 443.

	Command or Action	Purpose
Step 3	switch(config)#**transport tls authentication-type trustpoint**<*trustpoint name*> **Example:** `switch(config-crypto-qkd-profile)# transport tls authentication-type trustpoint tp1`	Configures the CA (Certificate Authority) trustpoint. To create a trustpoint, refer to Configuring PKI section.
Step 4	(Optional) switch(config)#**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Enabling MACsec and MKA features

Procedure

	Command or Action	Purpose
Step 1	switch(config)# **macsec policy**<*name*> **Example:** `switch(config)# [no] macsec policy test-policy`	Creates a MACsec policy.
Step 2	switch(config)# **ppk crypto-qkd-profile**<*name*> **Example:** `switch(config-macsec-policy)# [no] ppk crypto-qkd-profile ppk1`	Configures the PPK profile name.
Step 3	(Optional) switch(config)# copy running-config startup-config **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Configuration Examples

The following examples shows configuration of QKD profile and display of the configured details:

- Configuring QKD profile

```
switch(config)# feature cryptopqc
switch(config)#
switch(config)# crypto qkd profile ppk1
switch(config-crypto-qkd-profile)# kme server 168.20.1.2 port 5000
```

```
switch(config-crypto-qkd-profile)# transport tls authentication-type trustpoint tp1
switch(config-crypto-qkd-profile)# end
switch#
```

- Displaying QKD configuration

```
switch# show running-config cryptopqc
!Command: show running-config cryptopqc
!Running configuration last done at: Mon Jan 29 22:19:16 2024
!Time: Mon Jan 29 22:19:35 2024
version 10.4(3) Bios:version 05.51
feature cryptopqc
crypto qkd profile ppk1
kme server 168.20.1.2 port 5000
transport tls authentication-type trustpoint tp1
switch#
```

The following examples shows configuration of PPK profile on MACsec policy and display of the configured details:

- Configuring PPK profile on MACsec policy

```
switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# macsec policy test
switch(config-macsec-policy) # ppk crypto-qkd-profile ppk1
switch(config-macsec-policy)# sak-expiry-time 1800
switch(config-macsec-policy)# exit
switch(config)# end
```

- Display of configured MACsec policy

```
switch# show macsec policy test
MACSec Cipher     Pri Window Offset Security     SAKRekey   timeICV     Policy Indicator
     Include-SCI
------------------------------ ---------------- ---- ------------ --------
-------------- -------------- ------------- ------------
test   GCM-AES-XPN-256 16     148809600 0         should-secure 1800         FALSE
            TRUE
MACSec Policy     PPK Crypto-QKD-Profile Name
--------------------------------- ---------------------------------
test                ppk1
switch#
```

The following example shows configuration of key chain, MACsec policy on an interface, and display of configured details:

- Configuring key chain

```
switch(config)# key chain KC1 macsec
switch(config-macseckeychain)#key 10100000
switch(config-macseckeychain-macseckey)#key-octet-string
F123456789ABCDEF0123456789ABCDEFF123456789ABCDEF0123456789ABCDEF cryptographic-algorithm
 AES_256_CMAC
switch(config-macseckeychain-macseckey)#exit
```

- Configuring MACsec policy to an interface

```
switch(config)# interface Ethernet 1/21
switch(config-if)# macsec keychain KC1 policy test
```

- Displaying MACsec session

```
switch(config)# show macsec mka session
Interface     Local-TxSCI #     Peers     Status     Key-Server Auth Mode
```

```
----------------- ------------------------------- ----------------- -----------------
 ------------------ ------------------
Ethernet1/21  6cb2.ae9f.e766/0001 1     Secured         No     PRIMARY-PPK
```

The following examples show configuring point-to-point MACsec QKD profile, binding QKD profile to MACsec policy and binding the MACsec policy to the interface:

Note Make sure that the KME1 and KME2 servers must be active for connection through management port.

Configuring Swtich 1

```
switch1# configure terminal
switch1(config)# crypto ca trustpoint tp1
switch1config-trustpoint)# end
switch1#

switch1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch1(config)# feature cryptopqc
switch1(config)#
switch1(config)# crypto qkd profile PPK1
switch1(config-crypto-qkd-profile)# kme server KME1 port 7010
switch1(config-crypto-qkd-profile)# transport tls authentication-type trustpoint tp1
switch1(config-crypto-qkd-profile)# end
switch1#

switch1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch1(config)# feature macsec
switch1(config)#

switch1(config)# macsec policy MP1
switch1(config-macsec-policy)# ppk crypto-qkd-profile PPK1
switch1(config-macsec-policy)#exit
switch1(config-if)# interface Ethernet1/21
switch1(config-if)# macsec keychain KC1 policy MP1
switch1(config-if)#

switch1(config-if)# interface Ethernet1/22
switch1(config-if)# macsec keychain KC1 policy MP1
switch1(config-if)#
switch1(config-if)# end
switch1#
```

Configuring Swtich 2

```
switch2# configure terminal
switch2(config)# crypto ca trustpoint tp1
switch2config-trustpoint)# end
switch2#

switch2# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch2(config)# feature cryptopqc
switch2(config)#
switch2(config)# crypto qkd profile PPK1
switch2(config-crypto-qkd-profile)# kme server KME2 port 7010
switch2(config-crypto-qkd-profile)# transport tls authentication-type trustpoint tp1
switch2(config-crypto-qkd-profile)#
switch2(config-crypto-qkd-profile)# end
```

```
switch2#

switch2# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch2(config)# feature macsec
switch2(config)#

switch2(config)# macsec policy MP1
switch2(config-macsec-policy)# ppk crypto-qkd-profile PPK1
switch2(config-macsec-policy)# exit

switch2(config-if)# interface Ethernet1/53
switch2(config-if)# macsec keychain KC1 policy MP1
switch2(config-if)#
switch2(config-if)# interface Ethernet1/54
switch2(config-if)# macsec keychain KC1 policy MP1
switch2(config-if)# end
switch2#
```

The following shows the output of configuration on Switch 1 and Switch 2:

Switch 1:

```
switch1#
switch1# show macsec mka session
Interface     Local-TxSCI #     Peers     Status     Key-Server     Auth Mode
------------------ ---------------------------------- ------------------ ------------------
------------------ ------------------
Ethernet1/22   3c8b.7ffe.0244/0001 1     Secured         Yes        PRIMARY-PPK
Ethernet1/21   3c8b.7ffe.0240/0001 1     Secured         Yes        PRIMARY-PPK
N9K3K STANDARD TEMPLATE FOR FEATURE REVIEWS
------------------ ---------------------------------- ------------------ ------------------
------------------ ------------------
Total Number of Sessions : 2
Secured Sessions : 2
Pending Sessions : 0
switch1#
```

Switch 2:

```
switch2#
switch2# show macsec mka session
Interface     Local-TxSCI #     Peers     Status     Key-Server     Auth Mode
------------------ ---------------------------------- ------------------ ------------------
------------------ ------------------
Ethernet1/53  5451.deb8.62b4/0001 1         Secured     No         PRIMARY-PPK
Ethernet1/54  5451.deb8.62b8/0001 1         Secured     No         PRIMARY-PPK
------------------ ---------------------------------- ------------------ ------------------
------------------ ------------------
Total Number of Sessions : 2
Secured Sessions : 2
Pending Sessions : 0
switch2#
```

About Configurable EAPOL Destination and Ethernet Type

Beginning Cisco NX-OS Release 9.2(2), Cisco enables networks with WAN MACsec to change the Extensible Authentication Protocol (EAP) over LAN (EAPOL) protocol destination address, and the Ethernet type values to nonstandard values.

Configurable EAPOL MAC and Ethernet type provides you the ability to change the MAC address and the Ethernet type of the MKA packet, inorder to allow CE device to form MKA sessions over the ethernet networks that consume the standard MKA packets.

The EAPOL destination Ethernet type can be changed from the default Ethernet type of 0x888E to an alternate value or, the EAPOL destination MAC address can be changed from the default DMAC of 01:80:C2:00:00:03 to an alternate value, to avoid being consumed by a provider bridge.

This feature is available at the interface level and the alternate EAPOL configuration can be changed on any interface at any given time as follows:

- If the MACsec is already configured on an interface, the sessions will come up with a new alternate EAPOL configuration.
- When MACsec is not configured on an interface, the EAPOL configuration is applied to the interface and is effective when MACsec is configured on that inferface.

Enabling EAPOL Configuration

You can enable the EAPOL configuration on any available interface.

Before you begin

Make sure that MACsec is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface** *name*
	Example:
	`switch(config)# interface ethernet 1/1`
	`switch(config-if)#`
		Specifies the interface that you are configuring. You can specify the interface type and identity. For an Ethernet port, use ethernet slot/port.

Step 3	**eapol mac-address** *mac_address* [**ethertype** *eth_type*]	Enables the EAPOL configuration on the specified interface type and identity. **Note** If the ethernet type is not specified, the default ethernet type of MKA packets, which is 0x888e, is considered.
Step 4	**eapol mac-address broadcast-address** [**ethertype** *eth_type*]	Enables the broadcast address as the alternate mac address.
Step 5	(Optional) **copy running-config startup-config** **Example:**	Copies the running configuration to the startup configuration.

	Command or Action	Purpose
	`switch(config-macseckeychain-macseckey)# copy running-config startup-config`	
Step 6	**show macsec mka session detail**	Displays the EAPOL settings.

Disabling EAPOL Configuration

You can disable the EAPOL configuration on any available interface.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface** *name*
	Example:
	`switch(config)# interface ethernet 1/1`
	`switch(config-if)#`
		Specifies the interface that you are configuring. You can specify the interface type and identity. For an Ethernet port, use ethernet slot/port.

Step 3	[**no**] **eapol mac-address** *mac_address* [**ethertype** *eth_type*]	Disables the EAPOL configuration on the specified interface type and identity.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config-macseckeychain-macseckey)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Verifying the MACsec Configuration

To display MACsec configuration information, perform one of the following tasks:

Command	Purpose
show key chain *name*	Displays the keychain configuration.
show macsec mka session [**interface** *type slot*/*port*] [**detail**]	Displays information about the MACsec MKA session for a specific interface or for all interfaces.
show macsec mka session details	Displays information about the MAC address and the ethernet type that is currently used by the interfaces for all EAPOL packets.
show macsec mka summary	Displays the MACsec MKA configuration.

Command	Purpose
show macsec policy [*policy-name*]	Displays the configuration for a specific MACsec policy or for all MACsec policies.
show running-config macsec	Displays the running configuration information for MACsec.

The following example displays information about the MACsec MKA session for all interfaces. .

```
switch# show macsec mka session
Interface          Local-TxSCI                     #Peers            Status
Key-Server         Auth Mode
------------------ ------------------------------- ------------------ -----------------
------------------ ------------------
Ethernet2/2        2c33.11b8.7d14/0001             1                  Secured
Yes                PRIMARY-PSK
Ethernet2/3        2c33.11b8.7d18/0001             1                  Secured
Yes                PRIMARY-PSK
------------------ ------------------------------- ------------------ -----------------
------------------ ------------------
Total Number of Sessions : 2
        Secured Sessions : 2
        Pending Sessions : 0
```

The following example displays information about the MACsec MKA session for a specific interface. In addition to the common elements of the table as described in the previous example, the following also identifies the authentication mode which defines the current MACsec session type.

```
switch# show macsec mka session interface ethernet 1/1

Interface      Local-TxSCI          # Peers   Status        Key-Server     Auth Mode
-------------- -------------------- --------- ------------- -------------- -------------
Ethernet1/1    70df.2fdc.baf4/0001      0     Pending       Yes            PRIMARY-PSK
Ethernet1/1    70df.2fdc.baf4/0001      1     Secured       No             FALLBACK-PSK
```

The following example displays detailed information about the MACsec MKA session for a specific Ethernet interface:

```
Interface Name          : Ethernet2/2
    Session Status                    : SECURED - Secured MKA Session with MACsec
    Local Tx-SCI                      : 2c33.11b8.7d14/0001
    Local Tx-SSCI                     : 2
    MKA Port Identifier               : 2
    CAK Name (CKN)                    : 12
    CA Authentication Mode            : PRIMARY-PSK
    Member Identifier (MI)            : B54263EF7949A561E25CE617
    Message Number (MN)               : 523
    MKA Policy Name                   : tests2
    Key Server Priority               : 16
    Key Server                        : Yes
    Include ICV                       : No
    SAK Cipher Suite                  : GCM-AES-XPN-256
    SAK Cipher Suite (Operational)    : GCM-AES-XPN-256
    Replay Window Size                : 148809600
    Confidentiality Offset            : CONF-OFFSET-0
    Confidentiality Offset (Operational): CONF-OFFSET-0
    Latest SAK Status                 : Rx & TX
    Latest SAK AN                     : 0
    Latest SAK KI                     : B54263EF7949A561E25CE61700000001
    Latest SAK KN                     : 1
    Last SAK key time                 : 12:59:38 PST Tue Mar 19 2019
```

```
    CA Peer Count                       : 1
    Eapol dest mac                      : 0180.c200.0003
    Ether-type                          : 0x888e
Peer Status:
    Peer MI                             : 2C2C090E62A96F4D6E018210
    RxSCI                               : 2c33.11b8.8b88/0001
    Peer CAK                            : Match
    Latest Rx MKPDU                     : 13:16:54 PST Tue Mar 19 2019
```

The following example displays the MACsec MKA configuration:

```
switch# show macsec mka summary
Interface          MACSEC-policy                    Keychain
------------------ -------------------------------- ------------------------
Ethernet2/13       1                                1/10000000000000000
Ethernet2/14       1                                1/10000000000000000
```

The following example displays the configuration for all MACsec policies:

```
switch# show macsec policy
MACSec Policy        Cipher          Pri  Window     Offset   Security   SAK Rekey time ICV
 Indicator  Include-SCI
------------------ -------------     --- --------   --------  --------  --------------
------------- -------------
KC256-Po117b        GCM-AES-256      16   148809600    0    should-secure  pn-rollover
FALSE        True
pol1                GCM-AES-XPN-256  100  148809600    30   must-secure    60
FALSE        True
pol256-FanO         GCM-AES-XPN-256  16   148809600    0    must-secure    60
FALSE        True
pol256-MCT          GCM-AES-XPN-256  16   148809600    0    should-secure  60
FALSE        FALSE
system-default-
macsec-policy       GCM-AES-XPN-256  16   148809600    0    should-secure  pn-rollover
FALSE        FALSE
test1               GCM-AES-XPN-256  16   148809600    0    should-secure  pn-rollover
FALSE        True
```

The following example displays the key octet string in the output of the **show running-config** and **show startup-config** commands when the **key-chain macsec-psk no-show** command is not configured:

```
key chain KC256-1 macsec
  key 2000
   key-octet-string 7 075e701e1c5a4a5143475e5a527d7c7c706a6c724306170103555a5c57510b051e47080
a05000101005e0e50510f005c4b5f5d0b5b070e234e4d0a1d0112175b5e cryptographic-algorithm
AES_256_CMAC
```

The following example displays the key octet string in the output of the **show running-config** and **show startup-config** commands when the **key-chain macsec-psk no-show** command is configured:

```
key chain KC256-1 macsec
  key 2000
    key-octet-string 7 ****** cryptographic-algorithm AES_256_CMAC
```

Displaying MACsec Statistics

You can display MACsec statistics using the following commands.

Command	Description
show macsec mka statistics [**interface** *type slot*/*port*]	Displays MACsec MKA statistics.
show macsec secy statistics [**interface** *type slot*/*port*]	Displays MACsec security statistics.

The following example shows the MACsec MKA statistics for a specific Ethernet interface:

```
switch# show macsec mka statistics interface ethernet 2/2

Per-CA MKA Statistics for Session on interface (Ethernet2/2) with CKN 0x10
================================================================================
CA Statistics
   Pairwise CAK Rekeys..... 0

SA Statistics
   SAKs Generated.......... 0
   SAKs Rekeyed............ 0
   SAKs Received........... 0
   SAK Responses Received.. 0

MKPDU Statistics
   MKPDUs Transmitted...... 1096
      "Distributed SAK".. 0

   MKPDUs Validated & Rx... 0
      "Distributed SAK".. 0

MKA Statistics for Session on interface (Ethernet2/2)
============================================================
CA Statistics
   Pairwise CAK Rekeys..... 0

SA Statistics
   SAKs Generated.......... 0
   SAKs Rekeyed............ 0
   SAKs Received........... 0
   SAK Responses Received.. 0

MKPDU Statistics
   MKPDUs Transmitted...... 1096
      "Distributed SAK".. 0
   MKPDUs Validated & Rx... 0
      "Distributed SAK".. 0
   MKPDUs Tx Success.......... 1096
   MKPDUs Tx Fail............. 0
   MKPDUS Tx Pkt build fail... 0
   MKPDUS No Tx on intf down.. 0
   MKPDUS No Rx on intf down.. 0
   MKPDUs Rx CA Not found..... 0
   MKPDUs Rx Error............ 0
   MKPDUs Rx Success.......... 0

MKPDU Failures
   MKPDU Rx Validation ..................... 0
   MKPDU Rx Bad Peer MN..................... 0
   MKPDU Rx Non-recent Peerlist MN.......... 0
   MKPDU Rx Drop SAKUSE, KN mismatch........ 0
   MKPDU Rx Drop SAKUSE, Rx Not Set......... 0
   MKPDU Rx Drop SAKUSE, Key MI mismatch.... 0
   MKPDU Rx Drop SAKUSE, AN Not in Use...... 0
   MKPDU Rx Drop SAKUSE, KS Rx/Tx Not Set... 0
   MKPDU Rx Drop Packet, Ethertype Mismatch. 0
```

```
SAK Failures
   SAK Generation................... 0
   Hash Key Generation.............. 0
   SAK Encryption/Wrap.............. 0
   SAK Decryption/Unwrap............ 0

CA Failures
   ICK Derivation................... 0
   KEK Derivation................... 0
   Invalid Peer MACsec Capability... 0

MACsec Failures
   Rx SA Installation............... 0
   Tx SA Installation............... 0
```

The following example shows the MACsec security statistics for a specific Ethernet interface.

Note The following differences exist for uncontrolled and controlled packets in Rx and Tx statistics:

- Rx statistics:
  - Uncontrolled = Encrypted and unencrypted
  - Controlled = Decrypted
- Tx statistics:
  - Uncontrolled = Unencrypted
  - Controlled = Encrypted
  - Common = Encrypted and unencrypted

```
switch(config)# show macsec secy statistics interface e2/28/1

Interface Ethernet2/28/1 MACSEC SecY Statistics:
-----------------------------------------
Interface Rx Statistics:
   Unicast Uncontrolled Pkts: 14987
   Multicast Uncontrolled Pkts: 1190444
   Broadcast Uncontrolled Pkts: 4
   Uncontrolled Pkts - Rx Drop: 0
   Uncontrolled Pkts - Rx Error: 0
   Unicast Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Multicast Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Broadcast Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Controlled Pkts: 247583
   Controlled Pkts - Rx Drop: N/A (N9K-X9736C-FX not supported)
   Controlled Pkts - Rx Error: N/A (N9K-X9736C-FX not supported)
   In-Octets Uncontrolled: 169853963 bytes
   In-Octets Controlled: 55027017 bytes
   Input rate for Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
   Input rate for Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
   Input rate for Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Input rate for Controlled Pkts: N/A (N9K-X9736C-FX not supported)

Interface Tx Statistics:
   Unicast Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
```

```
   Multicast Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
   Broadcast Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
   Uncontrolled Pkts - Rx Drop: N/A (N9K-X9736C-FX not supported)
   Uncontrolled Pkts - Rx Error: N/A (N9K-X9736C-FX not supported)
   Unicast Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Multicast Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Broadcast Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Controlled Pkts: 205429
   Controlled Pkts - Rx Drop: N/A (N9K-X9736C-FX not supported)
   Controlled Pkts - Rx Error: N/A (N9K-X9736C-FX not supported)
   Out-Octets Uncontrolled: N/A (N9K-X9736C-FX not supported)
   Out-Octets Controlled: 20612648 bytes
   Out-Octets Common: 151787484 bytes
   Output rate for Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
   Output rate for Uncontrolled Pkts: N/A (N9K-X9736C-FX not supported)
   Output rate for Controlled Pkts: N/A (N9K-X9736C-FX not supported)
   Output rate for Controlled Pkts: N/A (N9K-X9736C-FX not supported)

SECY Rx Statistics:
  Transform Error Pkts: N/A (N9K-X9736C-FX not supported)
   Control Pkts: 952284
   Untagged Pkts: N/A (N9K-X9736C-FX not supported)
   No Tag Pkts: 0
   Bad Tag Pkts: 0
   No SCI Pkts: 0
   Unknown SCI Pkts: 0
   Tagged Control Pkts: N/A (N9K-X9736C-FX not supported)

SECY Tx Statistics:
   Transform Error Pkts: N/A (N9K-X9736C-FX not supported)
   Control Pkts: 967904
   Untagged Pkts: N/A (N9K-X9736C-FX not supported)

SAK Rx Statistics for AN [3]:
   Unchecked Pkts: 0
   Delayed Pkts: 0
   Late Pkts: 0
   OK Pkts: 1
   Invalid Pkts: 0
   Not Valid Pkts: 0
   Not-Using-SA Pkts: 0
   Unused-SA Pkts: 0
   Decrypted In-Octets: 235 bytes
   Validated In-Octets: 0 bytes

SAK Tx Statistics for AN [3]:
   Encrypted Protected Pkts: 2
   Too Long Pkts: N/A (N9K-X9736C-FX not supported)
   SA-not-in-use Pkts: N/A (N9K-X9736C-FX not supported)
   Encrypted Protected Out-Octets: 334 bytes
switch(config)#
```

Configuration Example for MACsec

The following example shows how to configure a user-defined MACsec policy and then apply the policy to interfaces:

```
switch(config)# macsec policy 1
switch(config-macsec-policy)# cipher-suite GCM-AES-256
switch(config-macsec-policy)# window-size 512
switch(config-macsec-policy)# key-server-priority 0
switch(config-macsec-policy)# conf-offset CONF-OFFSET-0
```

```
switch(config-macsec-policy)# security-policy should-secure
switch(config-macsec-policy)# exit

switch(config)# int e2/13-14
switch(config-if-range)# macsec keychain 1 policy 1
switch(config-if-range)# exit
switch(config)# show macsec mka summary
Interface          MACSEC-policy                    Keychain
------------------ -------------------------------- ------------------------
Ethernet2/13       1                                1/10000000000000000
Ethernet2/14       1                                1/10000000000000000

switch(config)# show macsec mka session
Interface      Local-TxSCI          # Peers    Status     Key-Server
-------------- -------------------- ---------- ---------- -------------
Ethernet2/13   006b.f1be.d31c/0001  1          Secured    Yes
Ethernet2/14   006b.f1be.d320/0001  1          Secured    No

switch(config)# show running-config macsec
!Command: show running-config macsec
!Time: Mon Dec  5 04:53:40 2016

version 9.2(1)feature macsec
macsec policy 1
  cipher-suite GCM-AES-256
  key-server-priority 0
  window-size 512
  conf-offset CONF-OFFSET-0
  security-policy should-secure

interface Ethernet2/13
  macsec keychain 1 policy 1

interface Ethernet2/14
  macsec keychain 1 policy 1
```

The following example shows how to configure a MACsec keychain and then add the system default MACsec policy to the interfaces:

```
switch(config)# key chain 1 macsec
switch(config-macseckeychain)# key 1000
switch(config-macseckeychain-macseckey)# key-octet-string
abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789 cryptographic-algorithm
aes_256_CMAC
switch(config-macseckeychain-macseckey)# exit

switch(config)# int e2/13-14
switch(config-if-range)# macsec keychain 1
switch(config-if-range)# exit
switch(config)#

switch(config)# show running-config macsec
!Command: show running-config macsec
!Time: Mon Dec  5 04:50:16 2016
version 7.0(3)I4(5)
feature macsec
interface Ethernet2/13
  macsec keychain 1 policy system-default-macsec-policy
interface Ethernet2/14
  macsec keychain 1 policy system-default-macsec-policy

switch(config)# show macsec mka session
Interface          Local-TxSCI                      # Peers            Status
Key-Server         Auth Mode
```

```
------------------ -------------------------------- ------------------ ------------------
------------------ ------------------
Ethernet2/2        2c33.11b8.7d14/0001              1                  Secured
Yes                PRIMARY-PSK
Ethernet2/3        2c33.11b8.7d18/0001              1                  Secured
Yes                PRIMARY-PSK
------------------ -------------------------------- ------------------ ------------------
------------------ ------------------
Total Number of Sessions : 2
        Secured Sessions : 2
        Pending Sessions : 0

switch(config)# show macsec mka summary
Interface          Status   Cipher (Operational)   Key-Server   MACSEC-policy   Keychain
Fallback-keychain
------------------ -------- ---------------------- ------------ -------------- ----------
---------------
Ethernet2/1        down       -                      -          tests1           keych1
 no keychain
Ethernet2/2        Secured  GCM-AES-XPN-256        Yes          tests2           keych2
 no keychain
Ethernet2/3        Secured  GCM-AES-256            Yes          tests3           keyc3
 no keychain
```

The following example shows the configuration and output of Peer Enforce Cipher configuration feature MACsec:

```
switch# show key chain
Key-Chain KC1 Macsec
Key 10000000 -- text 7
"0729701e1d5d4c53404a522d26090f010e63647040534355560e007971772a263e30080a040707030353022725 7b7321355655095852 5a771b165038273
4362e2a"
cryptographic-algorithm AES_256_CMAC
send lifetime (always valid) [active]

Key-Chain KC2 Macsec
Key 10100000 -- text 7
"0729701e1d5d4c53404a522d26090f010e63647040534355560e007971772a263e30080a040707030353022725 7b7321355655095852 5a771b165038273
4362e2a"
cryptographic-algorithm AES_256_CMAC
send lifetime (always valid) [active]

switch#
switch# show run macsec

!Command: show running-config macsec
!Running configuration last done at: Mon Apr 17 16:49:57 2023
!Time: Mon Apr 17 16:50:09 2023

version 10.3(3) Bios:version 05.47
feature macsec

macsec policy MP1
cipher-suite enforce-peer GCM-AES-XPN-256 GCM-AES-XPN-128
macsec policy MP2
cipher-suite enforce-peer GCM-AES-256
interface Ethernet1/97/1
macsec keychain KC1 policy MP1

interface Ethernet1/97/2
macsec keychain KC2 policy MP2

switch#
```

```
switch# show macsec policy
MACSec Policy Cipher Pri Window Offset Security SAK Rekey time ICV Indicator Include-SCI
------------------------------ ---------------- ---- ------------ -------- --------------
 -------------- -------------- --------------
MP1 Enforce-Peer 16 148809600 0 should-secure pn-rollover FALSE TRUE
MP2 Enforce-Peer 16 148809600 0 should-secure pn-rollover FALSE TRUE
system-default-macsec-policy GCM-AES-XPN-256 16 148809600 0 should-secure pn-rollover FALSE
 TRUE


MACSec Policy Cipher-Suite Enforce-Peer
------------------------------ ------------------------------------------------------
MP1 GCM-AES-XPN-256 GCM-AES-XPN-128
MP2 GCM-AES-256
switch#
```

The following example shows the sample output of the **show macsec mka session detail** command:

```
switch# show macsec mka session details
Detailed Status for MKA Session
-----------------------------------
Interface Name : Ethernet1/97/1
Session Status : SECURED - Secured MKA Session with MACsec
Local Tx-SCI : c4f7.d530.1484/0001
Local Tx-SSCI : 1
MKA Port Identifier : 1
CAK Name (CKN) : 10000000
CA Authentication Mode : PRIMARY-PSK
Member Identifier (MI) : D94B90E3FDB111CE583E7158
Message Number (MN) : 111
MKA Policy Name : MP1
Key Server Priority : 16
Key Server : Yes
Include ICV : No
SAK Cipher Suite : GCM-AES-XPN-128
SAK Cipher Suite (Operational) : GCM-AES-XPN-128
Replay Window Size : 148809600
Confidentiality Offset : CONF-OFFSET-0
Confidentiality Offset (Operational): CONF-OFFSET-0
Latest SAK Status : Rx & TX
Latest SAK AN : 1
Latest SAK KI : D94B90E3FDB111CE583E715800000001
Latest SAK KN : 1
Last SAK key time : 16:48:41 PST Mon Apr 17 2023
CA Peer Count : 1
Eapol dest mac : 0180.c200.0003
Ether-type : 0x888e
Peer Status:
Peer MI : 00110000000100010000001
RxSCI : 0011.0000.0001/0001
Peer CAK : Match
Latest Rx MKPDU : 16:52:07 PST Mon Apr 17 2023

Interface Name : Ethernet1/97/2
Session Status : SECURED - Secured MKA Session with MACsec
Local Tx-SCI : c4f7.d530.1485/0001
Local Tx-SSCI : 1
MKA Port Identifier : 1
CAK Name (CKN) : 10100000
CA Authentication Mode : PRIMARY-PSK
Member Identifier (MI) : 43AE54C19982238C298E0241
Message Number (MN) : 107
MKA Policy Name : MP2
Key Server Priority : 16
Key Server : Yes
```

```
Include ICV : No
SAK Cipher Suite : GCM-AES-256
SAK Cipher Suite (Operational) : GCM-AES-256
Replay Window Size : 148809600
Confidentiality Offset : CONF-OFFSET-0
Confidentiality Offset (Operational): CONF-OFFSET-0
Latest SAK Status : Rx & TX
Latest SAK AN : 0
Latest SAK KI : 43AE54C19982238C298E024100000001
Latest SAK KN : 1
Last SAK key time : 16:48:42 PST Mon Apr 17 2023
CA Peer Count : 1
Eapol dest mac : 0180.c200.0003
Ether-type : 0x888e
Peer Status:
Peer MI : 00270000000010001000000001
RxSCI : 0027.0000.0001/0001
Peer CAK : Match
Latest Rx MKPDU : 16:52:06 PST Mon Apr 17 2023
switch#
```

XML Examples

MACsec supports XML output for the following **show** commands for scripting purposes using **| xml**:

- **show key chain** *name* **| xml**
- **show macsec mka session interface** *interface slot/port* **details | xml**
- **show macsec mka statistics interface** *interface slot/port* **| xml**
- **show macsec mka summary | xml**
- **show macsec policy** *name* **| xml**
- **show macsec secy statistics interface** *interface slot/port* **| xml**
- **show running-config macsec | xml**

The following are example outputs for each of the preceding **show** commands:

Example 1: Displays the keychain configuration.

```
switch# show key chain "Kc2" | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0:rpm">
 <nf:data>
  <show>
   <key>
    <chain>
     <__XML__OPT_Cmd_rpm_show_keychain_cmd_keychain>
      <keychain>Kc2</keychain>
     </__XML__OPT_Cmd_rpm_show_keychain_cmd_keychain>
    </chain>
   </key>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

Example 2: Displays information about the MACsec MKA session for a specific interface.

```
switch# show macsec mka session interface ethernet 4/31 details | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0">
 <nf:data>
  <show>
   <macsec>
    <mka>
     <session>
      <__XML__OPT_Cmd_show_macsec_mka_session_interface>
       <interface>
        <__XML__INTF_ifname>
         <__XML__PARAM_value>
          <__XML__INTF_output>Ethernet4/31</__XML__INTF_output>
         </__XML__PARAM_value>
        </__XML__INTF_ifname>
       </interface>
       <__XML__OPT_Cmd_show_macsec_mka_session_details>
        <details/>
        <__XML__OPT_Cmd_show_macsec_mka_session___readonly__>
         <__readonly__>
          <TABLE_mka_session_details>
           <ROW_mka_session_details>
            <ifname>Ethernet4/31</ifname>
            <status>Secured</status>
            <sci>0c75.bd03.5360/0001</sci>
            <ssci>1</ssci>
            <port_id>1</port_id>
            <mi>02000000000000000000000000000000000000000000000000000000000000000000
</mi>
            <mi>F511280A765CE41C79458753</mi>
            <mn>2770</mn>
            <policy>am2</policy>
            <ks_prio>0</ks_prio>
            <keyserver>No</keyserver>
            <cipher>GCM-AES-XPN-256</cipher>
            <window>512</window>
            <conf_offset>CONF-OFFSET-0</conf_offset>
            <sak_status>Rx &amp;amp; TX</sak_status>
            <sak_an>1</sak_an>
            <sak_ki>516486241</sak_ki>
            <sak_kn>90</sak_kn>
            <last_sak_rekey_time>07:12:02 UTC Fri Jan 20 2017</last_sak_rekey_ti
me>
           </ROW_mka_session_details>
          </TABLE_mka_session_details>
         </__readonly__>
        </__XML__OPT_Cmd_show_macsec_mka_session___readonly__>
       </__XML__OPT_Cmd_show_macsec_mka_session_details>
      </__XML__OPT_Cmd_show_macsec_mka_session_interface>
     </session>
    </mka>
   </macsec>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

Example 3: Displays MACsec MKA statistics.

```
switch# show macsec mka statistics interface ethernet 4/31 | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0">
 <nf:data>
  <show>
<macsec>
    <mka>
     <statistics>
      <__XML__OPT_Cmd_some_macsec_mka_statistics_interface>
       <interface>
        <__XML__INTF_ifname>
         <__XML__PARAM_value>
          <__XML__INTF_output>Ethernet4/31</__XML__INTF_output>
          <__XML__INTF_output>Ethernet4/31</__XML__INTF_output>
         </__XML__PARAM_value>
        </__XML__INTF_ifname>
       </interface>
       <__XML__OPT_Cmd_some_macsec_mka_statistics___readonly__>
        <__readonly__>
         <TABLE_mka_intf_stats>
          <ROW_mka_intf_stats>
           <TABLE_ca_stats>
            <ROW_ca_stats>
             <ca_stat_ckn>0x2</ca_stat_ckn>
             <ca_stat_pairwise_cak_rekey>0</ca_stat_pairwise_cak_rekey>
             <sa_stat_sak_generated>0</sa_stat_sak_generated>
             <sa_stat_sak_rekey>0</sa_stat_sak_rekey>
             <sa_stat_sak_received>91</sa_stat_sak_received>
             <sa_stat_sak_response_rx>0</sa_stat_sak_response_rx>
             <mkpdu_stat_mkpdu_tx>2808</mkpdu_stat_mkpdu_tx>
             <mkpdu_stat_mkpdu_tx_distsak>0</mkpdu_stat_mkpdu_tx_distsak>
             <mkpdu_stat_mkpdu_rx>2714</mkpdu_stat_mkpdu_rx>
             <mkpdu_stat_mkpdu_rx_distsak>91</mkpdu_stat_mkpdu_rx_distsak>
            </ROW_ca_stats>
           </TABLE_ca_stats>
          </ROW_mka_intf_stats>
         </TABLE_mka_intf_stats>
        </__readonly__>
       </__XML__OPT_Cmd_some_macsec_mka_statistics___readonly__>
       <interface>
        <__XML__INTF_ifname>
         <__XML__PARAM_value>
          <__XML__INTF_output>Ethernet4/31</__XML__INTF_output>
         </__XML__PARAM_value>
        </__XML__INTF_ifname>
       </interface>
       <__XML__OPT_Cmd_some_macsec_mka_statistics___readonly__>
        <__readonly__>
         <TABLE_mka_intf_stats>
          <ROW_mka_intf_stats>
           <TABLE_idb_stats>
            <ROW_idb_stats>
             <ca_stat_pairwise_cak_rekey>0</ca_stat_pairwise_cak_rekey>
             <sa_stat_sak_generated>0</sa_stat_sak_generated>
             <sa_stat_sak_rekey>0</sa_stat_sak_rekey>
             <sa_stat_sak_received>91</sa_stat_sak_received>
             <sa_stat_sak_response_rx>0</sa_stat_sak_response_rx>
             <mkpdu_stat_mkpdu_tx>2808</mkpdu_stat_mkpdu_tx>
             <mkpdu_stat_mkpdu_tx_distsak>0</mkpdu_stat_mkpdu_tx_distsak>
             <mkpdu_stat_mkpdu_rx>2714</mkpdu_stat_mkpdu_rx>
             <mkpdu_stat_mkpdu_rx_distsak>91</mkpdu_stat_mkpdu_rx_distsak>
             <idb_stat_mkpdu_tx_success>2808</idb_stat_mkpdu_tx_success>
             <idb_stat_mkpdu_tx_fail>0</idb_stat_mkpdu_tx_fail>
             <idb_stat_mkpdu_tx_pkt_build_fail>0</idb_stat_mkpdu_tx_pkt_build_fail>
             <idb_stat_mkpdu_no_tx_on_intf_down>0</idb_stat_mkpdu_no_tx_on_intf_down>
```

```
            <idb_stat_mkpdu_no_rx_on_intf_down>0</idb_stat_mkpdu_no_rx_on_intf_down>
            <idb_stat_mkpdu_rx_ca_notfound>0</idb_stat_mkpdu_rx_ca_notfound>
            <idb_stat_mkpdu_rx_error>0</idb_stat_mkpdu_rx_error>
            <idb_stat_mkpdu_rx_success>2714</idb_stat_mkpdu_rx_success>
            <idb_stat_mkpdu_failure_rx_integrity_check_error>0</idb_stat_mkpdu_
failure_rx_integrity_check_error>
            <idb_stat_mkpdu_failure_invalid_peer_mn_error>0</idb_stat_mkpdu_fai
lure_invalid_peer_mn_error>
            <idb_stat_mkpdu_failure_nonrecent_peerlist_mn_error>1</idb_stat_mkp
du_failure_nonrecent_peerlist_mn_error>
            <idb_stat_mkpdu_failure_sakuse_kn_mismatch_error>0</idb_stat_mkpdu_
failure_sakuse_kn_mismatch_error>
<idb_stat_mkpdu_failure_sakuse_rx_not_set_error>0</idb_stat_mkpdu_f
ailure_sakuse_rx_not_set_error>
            <idb_stat_mkpdu_failure_sakuse_key_mi_mismatch_error>0</idb_stat_mk
pdu_failure_sakuse_key_mi_mismatch_error>
            <idb_stat_mkpdu_failure_sakuse_an_not_in_use_error>0</idb_stat_mkpd
u_failure_sakuse_an_not_in_use_error>
            <idb_stat_mkpdu_failure_sakuse_ks_rx_tx_not_set_error>0</idb_stat_m
kpdu_failure_sakuse_ks_rx_tx_not_set_error>
            <idb_stat_mkpdu_failure_sakuse_eapol_ethertype_mismatch_error>0</id
b_stat_mkpdu_failure_sakuse_eapol_ethertype_mismatch_error>
            <idb_stat_sak_failure_sak_generate_error>0</idb_stat_sak_failure_sa
k_generate_error>
            <idb_stat_sak_failure_hash_generate_error>0</idb_stat_sak_failure_h
ash_generate_error>
            <idb_stat_sak_failure_sak_encryption_error>0</idb_stat_sak_failure_
sak_encryption_error>
            <idb_stat_sak_failure_sak_decryption_error>0</idb_stat_sak_failure_
sak_decryption_error>
            <idb_stat_sak_failure_ick_derivation_error>0</idb_stat_sak_failure_
ick_derivation_error>
            <idb_stat_sak_failure_kek_derivation_error>0</idb_stat_sak_failure_
kek_derivation_error>
            <idb_stat_sak_failure_invalid_macsec_capability_error>0</idb_stat_s
ak_failure_invalid_macsec_capability_error>
            <idb_stat_macsec_failure_rx_sa_create_error>0</idb_stat_macsec_fail
ure_rx_sa_create_error>
            <idb_stat_macsec_failure_tx_sa_create_error>0</idb_stat_macsec_fail
ure_tx_sa_create_error>
           </ROW_idb_stats>
          </TABLE_idb_stats>
         </ROW_mka_intf_stats>
        </TABLE_mka_intf_stats>
       </__readonly__>
      </__XML__OPT_Cmd_some_macsec_mka_statistics___readonly__>
     </__XML__OPT_Cmd_some_macsec_mka_statistics_interface>
    </statistics>
   </mka>
  </macsec>
 </show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

Example 4: Displays the MACsec MKA configuration.

```
switch# show macsec mka summary | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0">
 <nf:data>
  <show>
   <macsec>
```

```
    <mka>
     <__XML__OPT_Cmd_some_macsec_summary>
      <__XML__OPT_Cmd_some_macsec___readonly__>
       <__readonly__>
        <TABLE_mka_summary>
         <ROW_mka_summary>
          <ifname>Ethernet2/1</ifname>
          <policy>am2</policy>
<keychain>kc2/020000000000000000000000000000000000000000000000000000000000
00000000</keychain>
         </ROW_mka_summary>
         <ROW_mka_summary>
          <ifname>Ethernet3/1</ifname>
          <policy>am2</policy>
          <keychain>kc2/020000000000000000000000000000000000000000000000000000000000
00000000</keychain>
         </ROW_mka_summary>

[TRUNCATED FOR READABILITY]

<ROW_mka_summary>
          <ifname>Ethernet3/32</ifname>
          <policy>am2</policy>
          <keychain>kc2/020000000000000000000000000000000000000000000000000000000000
00000000</keychain>
         </ROW_mka_summary>
        </TABLE_mka_summary>
       </__readonly__>
      </__XML__OPT_Cmd_some_macsec___readonly__>
     </__XML__OPT_Cmd_some_macsec_summary>
    </mka>
   </macsec>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

Example 5: Displays the configuration for a specific MACsec policy.

```
switch# show macsec policy am2 | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0">
 <nf:data>
  <show>
   <macsec>
    <policy>
     <__XML__OPT_Cmd_some_macsec_policy_name>
      <policy_name>am2</policy_name>
      <__XML__OPT_Cmd_some_macsec___readonly__>
       <__readonly__>
        <TABLE_macsec_policy>
         <ROW_macsec_policy>
          <name>am2</name>
          <cipher_suite>GCM-AES-XPN-256</cipher_suite>
          <keyserver_priority>0</keyserver_priority>
          <window_size>512</window_size>
          <conf_offset>0</conf_offset>
          <security_policy>must-secure</security_policy>
          <sak-expiry-time>60</sak-expiry-time>
         </ROW_macsec_policy>
        </TABLE_macsec_policy>
       </__readonly__>
      </__XML__OPT_Cmd_some_macsec___readonly__>
```

```
     </__XML__OPT_Cmd_some_macsec_policy_name>
    </policy>
   </macsec>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

Example 6: Displays MACsec security statistics.

```
switch# show macsec secy statistics interface ethernet 4/31 | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0">
 <nf:data>
  <show>
   <macsec>
    <secy>
     <statistics>
      <interface>
       <__XML__INTF_ifname>
        <__XML__PARAM_value>
         <__XML__INTF_output>Ethernet4/31</__XML__INTF_output>
        </__XML__PARAM_value>
        <__XML__OPT_Cmd_some_macsec_secy_statistics___readonly__>
         <__readonly__>
          <TABLE_statistics>
           <ROW_statistics>
            <in_pkts_unicast_uncontrolled>0</in_pkts_unicast_uncontrolled>
            <in_pkts_multicast_uncontrolled>42</in_pkts_multicast_uncontrolled>
            <in_pkts_broadcast_uncontrolled>0</in_pkts_broadcast_uncontrolled>
            <in_rx_drop_pkts_uncontrolled>0</in_rx_drop_pkts_uncontrolled>
            <in_rx_err_pkts_uncontrolled>0</in_rx_err_pkts_uncontrolled>
            <in_pkts_unicast_controlled>0</in_pkts_unicast_controlled>
            <in_pkts_multicast_controlled>2</in_pkts_multicast_controlled>
            <in_pkts_broadcast_controlled>0</in_pkts_broadcast_controlled>
            <in_rx_drop_pkts_controlled>0</in_rx_drop_pkts_controlled>
            <in_rx_err_pkts_controlled>0</in_rx_err_pkts_controlled>
            <in_octets_uncontrolled>7230</in_octets_uncontrolled>
            <in_octets_controlled>470</in_octets_controlled>
            <input_rate_uncontrolled_pps>0</input_rate_uncontrolled_pps>
            <input_rate_uncontrolled_bps>9</input_rate_uncontrolled_bps>
            <input_rate_controlled_pps>0</input_rate_controlled_pps>
            <input_rate_controlled_bps>23</input_rate_controlled_bps>
            <out_pkts_unicast_uncontrolled>0</out_pkts_unicast_uncontrolled>
            <out_pkts_multicast_uncontrolled>41</out_pkts_multicast_uncontrolled>
            <out_pkts_broadcast_uncontrolled>0</out_pkts_broadcast_uncontrolled>
            <out_rx_drop_pkts_uncontrolled>0</out_rx_drop_pkts_uncontrolled>
            <out_rx_err_pkts_uncontrolled>0</out_rx_err_pkts_uncontrolled>
            <out_pkts_unicast_controlled>0</out_pkts_unicast_controlled>
            <out_pkts_multicast_controlled>2</out_pkts_multicast_controlled>
            <out_pkts_broadcast_controlled>0</out_pkts_broadcast_controlled>
            <out_rx_drop_pkts_controlled>0</out_rx_drop_pkts_controlled>
            <out_rx_err_pkts_controlled>0</out_rx_err_pkts_controlled>
            <out_octets_uncontrolled>6806</out_octets_uncontrolled>
            <out_octets_controlled>470</out_octets_controlled>
            <out_octets_common>7340</out_octets_common>
            <output_rate_uncontrolled_pps>2598190092</output_rate_uncontrolled_pps>
            <output_rate_uncontrolled_bps>2598190076</output_rate_uncontrolled_bps>
            <output_rate_controlled_pps>0</output_rate_controlled_pps>
            <output_rate_controlled_bps>23</output_rate_controlled_bps>
            <in_pkts_transform_error>0</in_pkts_transform_error>
            <in_pkts_control>40</in_pkts_control>
            <in_pkts_untagged>0</in_pkts_untagged>
```

```
            <in_pkts_no_tag>0</in_pkts_no_tag>
            <in_pkts_badtag>0</in_pkts_badtag>
            <in_pkts_no_sci>0</in_pkts_no_sci>
            <in_pkts_unknown_sci>0</in_pkts_unknown_sci>
            <in_pkts_tagged_ctrl>0</in_pkts_tagged_ctrl>
            <out_pkts_transform_error>0</out_pkts_transform_error>
            <out_pkts_control>41</out_pkts_control>
            <out_pkts_untagged>0</out_pkts_untagged>
            <rx_sa_an>1</rx_sa_an>
            <in_pkts_unchecked>0</in_pkts_unchecked>
            <in_pkts_delayed>0</in_pkts_delayed>
            <in_pkts_late>0</in_pkts_late>
            <in_pkts_ok>1</in_pkts_ok>
            <in_pkts_invalid>0</in_pkts_invalid>
            <in_pkts_not_valid>0</in_pkts_not_valid>
            <in_pkts_not_using_sa>0</in_pkts_not_using_sa>
            <in_pkts_unused_sa>0</in_pkts_unused_sa>
            <in_octets_decrypted>223</in_octets_decrypted>
            <in_octets_validated>0</in_octets_validated>
            <tx_sa_an>1</tx_sa_an>
            <out_pkts_encrypted_protected>1</out_pkts_encrypted_protected>
            <out_pkts_too_long>0</out_pkts_too_long>
            <out_pkts_sa_not_inuse>0</out_pkts_sa_not_inuse>
            <out_octets_encrypted_protected>223</out_octets_encrypted_protected>
          </ROW_statistics>
        </TABLE_statistics>
      </__readonly__>
    </__XML__OPT_Cmd_some_macsec_secy_statistics___readonly__>
   </__XML__INTF_ifname>
  </interface>
 </statistics>
</secy>
</macsec>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

Example 7: Displays the running configuration information for MACsec.

```
switch# show running-config macsec | xml

!Command: show running-config macsec
!Time: Fri Jan 20 07:12:34 2017

version 7.0(3)I4(6)
****************************************
This may take time. Please be patient.
****************************************
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cis
co.com/nxos:7.0.3.I4.6.:configure_" xmlns:m="http://www.cisco.com/nxos:7.0.3.I4.
6.:_exec" xmlns:m1="http://www.cisco.com/nxos:7.0.3.I4.6.:configure__macsec-poli
cy" xmlns:m2="http://www.cisco.com/nxos:7.0.3.I4.6.:configure__if-eth-non-member
" message-id="1">
  <nf:get-config>
    <nf:source>
      <nf:running/>
    </nf:source>
    <nf:filter>
      <m:configure>
        <m:terminal>
          <feature>
            <macsec/>
```

```
          </feature>
          <macsec>
            <policy>
              <__XML__PARAM__policy_name>
                <__XML__value>am2</__XML__value>
                <m1:cipher-suite>
                  <m1:__XML__PARAM__suite>
                    <m1:__XML__value>GCM-AES-XPN-256</m1:__XML__value>
                  </m1:__XML__PARAM__suite>
                </m1:cipher-suite>
                <m1:key-server-priority>
                  <m1:__XML__PARAM__pri>
                    <m1:__XML__value>0</m1:__XML__value>
                  </m1:__XML__PARAM__pri>
                </m1:key-server-priority>
<m1:window-size>
<m1:__XML__PARAM__size>
                    <m1:__XML__value>512</m1:__XML__value>
                  </m1:__XML__PARAM__size>
                </m1:window-size>
                <m1:conf-offset>
                  <m1:__XML__PARAM__offset>
                    <m1:__XML__value>CONF-OFFSET-0</m1:__XML__value>
                  </m1:__XML__PARAM__offset>
                </m1:conf-offset>
                <m1:security-policy>
                  <m1:__XML__PARAM__policy>
                    <m1:__XML__value>must-secure</m1:__XML__value>
                  </m1:__XML__PARAM__policy>
                </m1:security-policy>
                <m1:sak-expiry-time>
                  <m1:__XML__PARAM__ts>
                    <m1:__XML__value>60</m1:__XML__value>
                  </m1:__XML__PARAM__ts>
                </m1:sak-expiry-time>
              </__XML__PARAM__policy_name>
            </policy>
          </macsec>
          <interface>
            <__XML__PARAM__interface>
              <__XML__value>Ethernet2/1</__XML__value>
              <m2:macsec>
                <m2:keychain>
                  <m2:__XML__PARAM__keychain_name>
                    <m2:__XML__value>kc2</m2:__XML__value>
                    <m2:policy>
                      <m2:__XML__PARAM__policy_name>
                        <m2:__XML__value>am2</m2:__XML__value>
                      </m2:__XML__PARAM__policy_name>
                    </m2:policy>
                  </m2:__XML__PARAM__keychain_name>
                </m2:keychain>
              </m2:macsec>
            </__XML__PARAM__interface>
          </interface>

[TRUNCATED FOR READABILITY]

<interface>
            <__XML__PARAM__interface>
              <__XML__value>Ethernet4/31</__XML__value>
              <m2:macsec>
                <m2:keychain>
                  <m2:__XML__PARAM__keychain_name>
```

```
                        <m2:__XML__value>kc2</m2:__XML__value>
                        <m2:policy>
                          <m2:__XML__PARAM__policy_name>
                            <m2:__XML__value>am2</m2:__XML__value>
                          </m2:__XML__PARAM__policy_name>
                        </m2:policy>
                      </m2:__XML__PARAM__keychain_name>
                    </m2:keychain>
                  </m2:macsec>
                </__XML__PARAM__interface>
              </interface>
            </m:terminal>
          </m:configure>
        </nf:filter>
      </nf:get-config>
    </nf:rpc>
    ]]>]]>
```

MIBs

MACsec supports the following MIBs:

- IEEE8021-SECY-MIB
- CISCO-SECY-EXT-MIB

To locate and download supported MIBs, go to the following URL: ftp://ftp.cisco.com/pub/mibs/supportlists/nexus9000/Nexus9000MIBSupportList.html.

Related Documentation

Related Topic	Document Title
Keychain management	Cisco Nexus 9000 Series NX-OS Security Configuration Guide
System messages	Cisco Nexus 9000 Series NX-OS System Messages References