802.1X Port-Based Authentication

The IEEE 802.1X port-based authentication protects the network from unauthorized clients. It blocks all traffic to and from devices at the interface, until the Authentication server authenticates the client. After successful authentication, the port is open for traffic.

This chapter describes how to configure IEEE 802.1X port-based authentication in Cisco NCS 540 Series Routers to prevent unauthorized devices (clients) from gaining access to the network.

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
IEEE 802.1X Port-Based Authentication Support for Multiple Authentication and Multiple Host Modes	Release 7.9.1	The IEEE 802.1X port-based authentication allows only authorized supplicants to access the network. The IEEE 802.1X port-based authentication now supports multiple authentication and multiple host modes to allow multiple hosts or MAC addresses on a single port. Applicable to Cisco N540-24Q8L2DD-SYS router variant.

Table 2: Feature History

Release	Modification
Release 7.2.1	Support for multi-auth and multi-host modes by 802.1X to allow multiple hosts or MAC addresses on a single port was introduced.
Release 6.6.3	This feature was introduced.

- Usage guidelines and restrictions for 802.1X port-based authentication, on page 2
- IEEE 802.1X Device Roles, on page 2
- Understanding 802.1X Port-Based Authentication, on page 3
- 802.1X host-modes, on page 4
- Prerequisites for 802.1X Port-Based Authentication, on page 4
- 802.1X with Remote RADIUS Authentication, on page 5
- 802.1X with Local EAP Authentication, on page 6
- Router as 802.1X Supplicant, on page 10

• Verify 802.1X Port-Based Authentication, on page 11

Usage guidelines and restrictions for 802.1X port-based authentication

Consider these restrictions and usage guidelines when implementing 802.1X port-based authentication on the Cisco 8000 platform:

Port authentication

- 802.1X port authentication must be configured on physical ports.
- Supported modes for 802.1X port-based authentication:
  - Single-host
  - Multi-auth

VLAN sub-interfaces

- VLAN sub-interfaces must have pre-configured VLAN IDs.
- All VLAN-tagged traffic is dropped until successful 802.1X authentication of the port.
- No default VLAN assignment is provided for unauthenticated MAC addresses.
- Authenticated MAC addresses are validated at the main port, independent of VLAN assignment.
- VLAN-tagged traffic is allowed only for authenticated MAC addresses.

Untagged traffic

- Untagged EAPOL traffic is always allowed.
- All other untagged traffic is dropped until successful 802.1X authentication of the port.
- Untagged traffic is allowed only for authenticated MAC addresses.
- No default VLAN assignment is provided for untagged traffic by the port.

IEEE 802.1X Device Roles

The devices in the network have the following specific roles with IEEE 802.1X authentication:

- **Authenticator** - An entity that facilitates authentication of other entities attached to the same LAN.
- **Supplicant** - An entity at one end of a point-to-point LAN segment that seeks to be authenticated by an Authenticator attached to the other end of that link.

- **Authentication Server** - An entity that provides an authentication service to an Authenticator. Based on the credentials provided by the Supplicant, the server determines whether the Supplicant is authorized to access the services provided by the system in which the Authenticator resides.

Understanding 802.1X Port-Based Authentication

IEEE 802.1X port-based authentication is configured on Cisco 8000 series router to prevent unauthorized routers (supplicants) from gaining access to the network. An authentication server validates the supplicant that is connected to an authenticator port, before the services offered by the client or the network is made available to the supplicant.

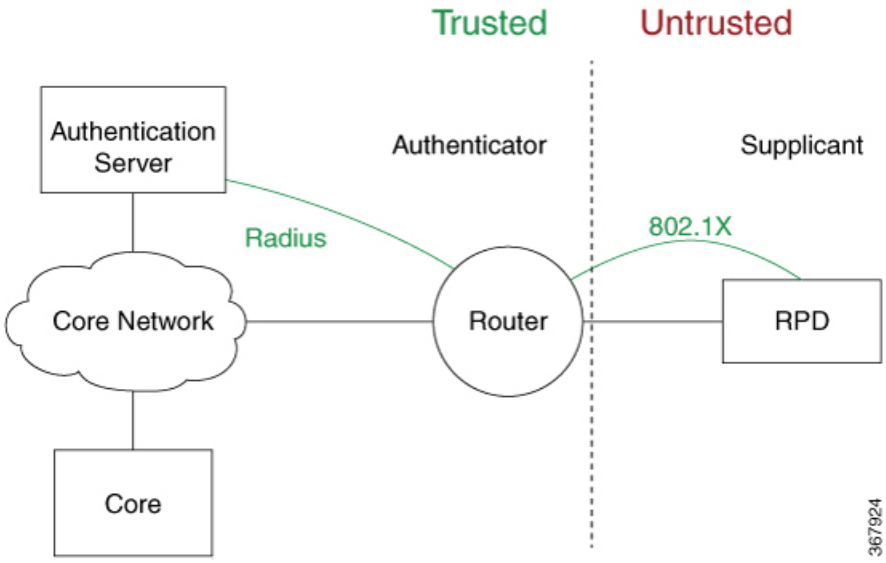
Until the supplicant is authenticated, the port is in *Unauthorized* state, and 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPOL) packets through the port. EAPoL frames can have either default EtherType of 0x888E or Cisco-defined EtherType of 0x876F. After successful authentication of the supplicant, the port transitions to *Authorized* state, and normal traffic passes through the port for the authenticated client.

Periodic reauthentication can be enabled to use either the port-configured value or from authentication server. The authentication server communicates the reauthentication-timer value in Session-Timeout attribute, with the final RADIUS Access-Accept message. On 802.1X reauthentication failure, the port is blocked and moved back to the *Unauthorized* state.

If the link state of a port changes from up to down, or if an EAPOL-logoff frame is received, the port returns to the *Unauthorized* state.

The following figure shows the topology for IEEE 802.1X port-based authentication:

Figure 1: Topology for IEEE 802.1X Port-Based Authentication



By default, the dot1x configured port is in multi-auth mode. However, this behaviour can be altered by changing the host mode under dot1x profile.

Note Port-control is enforced only on the ingress traffic.

802.1X host-modes

The following table describes the three host modes supported by 802.1X:

Table 3: 802.1X host modes

Host modes	Description
Single-host	While in this mode, the port allows a single host to be authenticated and allows only ingress traffic from the authenticated peer. A security violation is detected if more than one client is present.
Multi-auth	This is the default host mode. While in this mode, multiple hosts can independently authenticate through the same port and ingress traffic is allowed from all authenticated peers.
Multi-host	While in this mode, the first device to authenticate will open the port access so that all other hosts can use the port. These hosts need not be authenticated independently. If the authenticated host becomes unauthorized, the port will be closed.

Configure 802.1X host-modes

Use the following steps to configure 802.1X host-modes. Here, `host-mode` is introduced under the authenticator mode in dot1x profile. The default is `multi-auth` mode.

```
Router# configure terminal
Router(config)# dot1x profile {name}
Router(config-dot1x-auth)# pae {authenticator}
Router(config-dot1x-auth-auth)# host-mode
  multi-auth   multiple authentication mode
  multi-host   multiple host mode
  single-host  single host mode
```

Prerequisites for 802.1X Port-Based Authentication

Prerequisites for 802.1X port-based authentication are:

- K9sec RPM is required to enable this feature.
- Ensure that both RADIUS/EAP-server and supplicant are configured with supported EAP methods when remote authentication is used.
- If the device is used as a local EAP server, only EAP-TLS method is supported.
    - Ensure that a Certificate Authority (CA) server is configured for the network with a valid certificate.

- Ensure that the supplicant, authenticator, and CA server are synchronized using Network Time Protocol (NTP). If time is not synchronized on all these devices, certificates may not be validated.

802.1X with Remote RADIUS Authentication

Configure RADIUS Server

To configure RADIUS server pre-shared keys, obtain the pre-shared key values for the remote RADIUS server and perform this task.

Configuration Example

```
Router# configure terminal
Router(config)# radius-server host 209.165.200.225 auth-port 1646 key secret007
Router(config)# radius-server vsa attribute ignore unknown
Router(config)# commit
```

Running Configuration

```
Router# show run radius
radius-server host 209.165.200.225 auth-port 1646
  key 7 00171605165E1F565F76
radius-server vsa attribute ignore unknown
!
```

For more information, see Configure Router to RADIUS Server Communication and Configure RADIUS Server Groups in chapter *Configuring AAA Services*.

Configure 802.1X Authentication Method

You can configure 802.1X authentication method using RADIUS as the protocol. Only default AAA method is supported for 802.1X authentication.

Configuration Example

```
Router# configure terminal
Router(config)# aaa authentication dot1x default group radius
Router(config)# commit
```

Running Configuration

```
Router# show run aaa
  aaa authentication dot1x default group radius
```

Configure 802.1X Authenticator Profile

Configure 802.1X profile on an authenticator.

```
RP/0/RP0/CPU0:router(config)# dot1x profile <auth>
RP/0/RP0/CPU0:router(config-dot1x-auth)# pae authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth)# authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# timer reauth-time 3600
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# host-mode { multi-auth | multi-host |
single-host }
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# commit
```

Running Configuration

The following is a sample output of `show run dot1x` command.

```
RP/0/RSP0/CPU0:router# show run dot1x profile auth
dot1x profile auth
pae authenticator
authenticator
  timer reauth-time 3600
  host-mode multi-auth
!
```

Configure 8021X Profile on Interface

You can attach one of the 802.1X profiles on an interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface <interface-name>
RP/0/RSP0/CPU0:router(config-if)# dot1x profile <profile-name>
RP/0/RSP0/CPU0:router(config-if)# commit
```

Example Configuration

This example provides the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE 0/3/0/0
RP/0/RSP0/CPU0:router(config-if)# dot1x profile auth
RP/0/RSP0/CPU0:router(config-if)# commit
```

This example verifies the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
  dot1x profile auth
```

This example provides the configuration to allow tagged traffic with VLAN IDs 1 & 2:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.1
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.1.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 1
!
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.2
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.2.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)#encapsulation dot1q 2
!
```

802.1X with Local EAP Authentication

In local EAP authentication, the EAP-server is co-located with the authenticator locally on the router. This feature enables the router to authenticate 802.1X clients with EAP-TLS method using TLS Version 1.2. It

provides EAP-TLS based mutual authentication, where a Master Session Key (MSK) is generated on successful authentication.

Generate RSA Key Pair

RSA key pairs are used to sign and encrypt key management messages. This is required before you can obtain a certificate for the node.

```
RP/0/RSP0/CPU0:router#crypto key generate rsa < keypair-label >
```

Running Configuration

The following is a sample output of **show crypto key mypubkey rsa** command.

```
RP/0/RSP0/CPU0:router# show crypto key mypubkey rsa
Key label:  rsa_tp
Type : RSA General purpose
Size : 2048
Data :
30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
00BAA4F5 19C1C41A 4A195B31 6722B853 5271EECA B884CC19 CE75FB23 19DC0346
2F90F9B2 CBCB9BA3 4E4DDD46 2C21F555 4C642E3A 98FE0A2F 587D79F5 1D5B898F
893CEC38 B7C8CB03 48D0AEA1 D554DF2B BA751489 3099A890 1A910D25 7DA78F99
E29526FE 6F84C147 4F872715 D3BDE515 FACB28E8 6375BB38 1F3AFDA8 853C6E57
8BDA1800 7DDADFE3 32ABAB4C 3D078342 36E79F05 CAFCE764 26274F41 25F7BC70
04ABEDFE 96A183EE 23A3D099 2D5741C5 F81747FB 1ED5F672 5449B7AE 8D2E9224
CF12E1CA 9E2373C4 41BF29FA A9DDD930 5A3A2FDE FD1DADE1 2548DEDB 05FC2176
7D5DB337 B1563CA3 A94DF081 5B294D1A A9B70A56 CA5CF7B2 A779F27A 3EE4F568
F1020301 0001
```

For more information, see Generate RSA Key Pair in chapter *Implementing Certification Authority Interoperability*.

Configure Trustpoint

Trustpoints let you manage and track CAs and certificates. A trustpoint includes the identity of the CA, CA-specific configuration parameters, and an association with one, enrolled identity certificate. After you have defined a trustpoint, you can reference it by name in commands requiring that you specify a CA.

```
RP/0/RSP0/CPU0:router# configure terminal
RP/0/RSP0/CPU0:router(config)# crypto ca trustpoint <tp_name>
RP/0/RSP0/CPU0:router(config-trustp)# enrollment url <ca-url>
RP/0/RSP0/CPU0:router(config-trustp)# subject-name <x.500-name>
RP/0/RSP0/CPU0:router(config-trustp)# rsakeypair <keypair-label>
RP/0/RSP0/CPU0:router(config-trustp)# crl optional
RP/0/RSP0/CPU0:router(config-trustp)# commit
```

Running Configuration

The following is a sample output of **show run crypto ca trustpoint** *tp_name* command.

```
crypto ca trustpoint tp
  crl optional
  subject-name CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
  enrollment url http://20.30.40.50
  rsakeypair rsa_tp
   !
```

For more information, see Declare Certification Authority and Configure Trusted Point in chapter *Implementing Certification Authority Interoperability*.

Configure Domain Name

You can configure a domain name, which is required for certificate enrolment.

```
RP/0/RSP0/CPU0:router# domain name ca.cisco.com
```

Running Configuration

The following is a sample output of **show run domain name** command.

```
RP/0/1/CPU0:router# show run domain name
Thu Mar 29 16:10:42.533 IST
domain name cisco.com
```

Certificate Configurations

Certificate enrolment involves the following two steps:

1. Obtain CA certificate for the given trust point, using the **crypto ca authenticate** *tp_name* command.
2. Enroll the device certificate with CA, using the **crypto ca enroll** *tp_name* command.

```
RP/0/RSP0/CPU0:router# crypto ca authenticate <tp_name>
RP/0/RSP0/CPU0:router# crypto ca enroll <tp_name>
```

Running Configuration

The following is a sample output of the **show crypto ca certificates** command.

```
RP/0/RSP0/CPU0:router# show crypto ca certificates
Trustpoint : tp
================================================
CA certificate
Serial Number        : E0:18:F3:E4:53:17:3E:28
Subject              : subject-name CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Issued By            : subject-name CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Validity Start       : 08:17:32 UTC Fri Jun 24 2016
Validity End         : 08:17:32 UTC Mon Jun 22 2026
SHA1 Fingerprint     : 894ABBFAA3B08E5B7D9E470ECFBBC04576B569F2
Router certificate
Key usage            : General Purpose
Status               : Available
Serial Number        : 03:18
Subject              :
serialNumber=cf302761,unstructuredAddress=20.30.40.50,unstructuredName=asr9k,
C=US,ST=NY,L=Newyork,O=Govt,OU=BU,CN=asr9k
Issued By            : CN=asr9k,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
Validity Start       : 13:04:52 UTC Fri Feb 23 2018
Validity End         : 13:04:52 UTC Sat Feb 23 2019
SHA1 Fingerprint     :33B50A59C76CCD87D3D0F0271CD5C81F4A1EE9E1
Associated Trustpoint: tp
```

For more information, see Declare Certification Authority and Configure Trusted Point in chapter *Implementing Certification Authority Interoperability*.

Configure EAP Profile

You can configure multiple EAP profiles.

```
RP/0/RSP0/CPU0:router# configure terminal
RP/0/RSP0/CPU0:router(config)# eap profile <name>
RP/0/RSP0/CPU0:router(config-eap)# identity <user-name>
RP/0/RSP0/CPU0:router(config-eap)# method tls pki-trustpoint <trustpoint-name>
RP/0/RSP0/CPU0:router(config-eap)# commit
```

Note To allow EAP-TLS authentication with peer devices or EAP-server running on TLS 1.0, configure `allow-eap-tls-v1.0` under EAP profile.

Running Configuration

The following is sample output of **show run eap** command.

```
RP/0/RSP0/CPU0:router# show run eap profile <local eap>
eap profile local_eap
method tls
  pki-trustpoint tp
!
identity CE1
```

Configure 802.1X Authenticator Profile

You can configure 802.1X profile on an authenticator.

```
RP/0/RP0/CPU0:router(config)# dot1x profile local_auth
RP/0/RP0/CPU0:router(config-dot1x-auth)# pae authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth)# authenticator
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# eap profile <local_eap>
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# host-mode {multi-auth | multi-host |
single-host}
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# timer reauth-time 3600
RP/0/RP0/CPU0:router(config-dot1x-auth-auth)# commit
```

Running Configuration

The following is a sample output of `show run dot1x` command.

```
RP/0/RSP0/CPU0:router# show run dot1x profile local_auth

dot1x profile local_auth
pae authenticator
  authenticator
     eap profile local_eap
     host-mode multi-host
     timer reauth-time 3600
```

Configure 802.1X Profile on Interface

You can attach one of the 802.1X profiles on an interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface <interface-name>
RP/0/RSP0/CPU0:router(config-if)# dot1x profile <profile-name>
RP/0/RSP0/CPU0:router(config-if)# commit
```

Example Configuration

This example provides the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE 0/3/0/0
RP/0/RSP0/CPU0:router(config-if)# dot1x profile auth
RP/0/RSP0/CPU0:router(config-if)# commit
```

This example verifies the dot1x profile configuration.

```
RP/0/RSP0/CPU0:router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
  dot1x profile auth
```

This example provides the configuration to allow tagged traffic with VLAN IDs 1 & 2:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.1
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.1.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 1
!
RP/0/RSP0/CPU0:router(config)# interface HundredGigE0/3/0/0.2
RP/0/RSP0/CPU0:router(config-subif)# ipv4 address 20.10.2.2 255.255.255.0
RP/0/RSP0/CPU0:router(config-subif)#encapsulation dot1q 2
!
```

Router as 802.1X Supplicant

To configure the router as 802.1X supplicant, make sure that the following configurations are enabled:

- RSA Key Pair: Generate RSA Key Pair, on page 7
- Trust point: Configure Trustpoint, on page 7
- Domain name: Configure Domain Name, on page 8
- Certificates: Certificate Configurations, on page 8
- EAP profile: Configure EAP Profile, on page 8

Configure 802.1X Supplicant Profile

You can configure 802.1X profile on a supplicant.

```
RP/0/RP0/CPU0:router(config)# dot1x profile supp
RP/0/RP0/CPU0:router(config-dot1x-supp)# pae supplicant
RP/0/RP0/CPU0:router(config-dot1x-supp)# supplicant
RP/0/RP0/CPU0:router(config-dot1x-supp-supp)# eap profile eap_supp
RP/0/RP0/CPU0:router(config-dot1x-supp-supp)# commit
```

Running Configuration

The following is a sample output of `show run dot1x` command.

```
RP/0/RSP0/CPU0:router# show run dot1x profile supp
dot1x profile supp
pae supplicant
supplicant
eap profile eap_supp
!
```

Configure 802.1X Profile on Interface

You can attach one of the 802.1X profiles on an interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface <interface-name>
RP/0/RSP0/CPU0:router(config-if)# dot1x profile <profile-name>
RP/0/RSP0/CPU0:router(config-if)# commit
```

Example Configuration

```
Router# show run interface HundredGigE 0/3/0/0
interface HundredGigE 0/3/0/0
  dot1x profile supp
```

Verify 802.1X Port-Based Authentication

The 802.1X authentication can be verified using the following:

- Show command outputs
- Syslog messages

Show Command Outputs

The **show dot1x interface** command verifies whether the 802.1X port-based authentication is successful or not. If the authentication is successful, the traffic is allowed on the configured interface.

```
Router# show dot1x interface HundredGigE 0/0/1/0 detail

Dot1x info for HundredGigE 0/0/1/0
-----------------------------------------------------------------
Interface short name      : Hu 0/0/1/0
Interface handle          : 0x4080
Interface MAC             : 021a.9eeb.6a59
Ethertype                 : 888E
PAE                       : Authenticator
Dot1x Port Status         : AUTHORIZED
Dot1x Profile             : test_prof
L2 Transport              : FALSE
Authenticator:
   Port Control           : Enabled
   Config Dependency      : Resolved
   Eap profile            : None
   ReAuth                 : Disabled
Client List:
      Supplicant          : 027e.15f2.cae7
 Programming Status       : Add Success
      Auth SM State       : Authenticated
      Auth Bend SM State  : Idle
      Last authen time    : 2018 Dec 11 17:00:30.912
      Last authen server  : 10.77.132.66
      Time to next reauth : 0 day(s), 00:51:39
MKA Interface:
   Dot1x Tie Break Role   : NA (Only applicable for PAE role both)
   EAP Based Macsec       : Disabled
   MKA Start time         : NA
   MKA Stop time          : NA
   MKA Response time      : NA
```

```
Router#show dot1x
Mon Jun 15 18:30:49.327 IST

NODE: node0_RP0_CPU0

Dot1x info for TenGigE0/11/0/1
-------------------------------------------------------------------
PAE                       : Authenticator
Dot1x Port Status         : AUTHORIZED (2/2)
Dot1x Profile             : auth
Authenticator:
   Host Mode              : Multi-Auth
   Port Control           : Enabled
   Config Dependency      : Resolved
   Eap profile            : Not Configured
   ReAuth                 : Enabled, 1 day(s), 00:00:00
   Client List:
      Supplicant          : 008a.96a4.b028
      Port Status         : Authorized
      Programming Status  : Add Success
      Auth SM State       : Authenticated
      Auth Bend SM State  : Idle
      Last authen time    : 2020 Jun 15 18:30:42.659
      Last authen server  : 10.105.236.94
      Time to next reauth : 0 day(s), 23:59:52

      Supplicant          : 008a.96a4.c830
      Port Status         : Authorized
      Programming Status  : Add Success
      Auth SM State       : Authenticated
      Auth Bend SM State  : Idle
      Last authen time    : 2020 Jun 15 18:30:42.654
      Last authen server  : 10.105.236.94
      Time to next reauth : 0 day(s), 23:59:52
```

Syslog Messages

Syslogs on Authenticator

When 802.1x configuration is applied on an interface, the port becomes 802.1X controlled, and the following syslog message is displayed:

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled
```

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Single-Host
 mode
```

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Multi-Host
 mode
```

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Multi-Auth
 mode
```

When there is a host-mode violation, the following syslog messages are displayed:

```
%L2-DOT1X-3-HOST_MODE_VIOLATION: Hu0/0/1/0 : multiple clients detected in Single-Host mode,
 dropping supplicant (008a.9686.0058) request
```

```
%L2-DOT1X-3-HOST_MODE_VIOLATION: Hu0/0/1/0 : multiple clients detected in Multi-Host mode,
 dropping supplicant (008a.9686.0058) request
```

After successful authentication of supplicant, the following syslog messages are displayed:

```
%L2-DOT1X-5-AUTH_SUCCESS : Hu0/0/1/0 : Authentication successful for client 027E.15F2.CAE7


%L2-DOT1X-5-PORT_CONTROL_ADD_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Enabled For Client
027E.15F2.CAE7
```

When 802.1X port-based configuration is removed from an interface, the following syslog message is displayed:

```
%L2-DOT1X-5-PORT_CONTROL_DISABLE_SUCCESS : Hu0/0/1/0  : Port Control Disabled
```

When authentication fails, the following syslog messages are displayed:

```
%L2-DOT1X-5-AUTH_FAIL : Hu0/0/1/0 : Authentication fail for client 027E.15F2.CAE7

%L2-DOT1X-5-PORT_CONTROL_REMOVE_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Disabled For Client
 027E.15F2.CAE7
```

When authentication server is unreachable, the following syslog message is displayed:

```
%L2-DOT1X-5-AAA_UNREACHABLE : Hu0/0/1/0 : AAA server unreachable for client 027E.15F2.CAE7
  , Retrying Authentication
```

When authentication method is not configured, the following syslog message is displayed:

```
%L2-DOT1X-4-NO_AUTHENTICATION_METHOD : Hu0/0/1/0 : No authentication method configured
```

Syslogs on Supplicant

```
%L2-DOT1X-5-SUPP_SUCCESS : Hu0/0/1/0 : Authentication successful with authenticator
008a.96a4.b050

%L2-DOT1X-5-SUPP_FAIL : Hu0/0/1/0 : Authentication successful with authenticator
0000.0000.0000.0000

%L2-DOT1X-5-SUPP_FAIL : Hu0/0/1/0 : Authentication successful with authenticator
008a.96a4.b028
```