Configure Local EAP Authentication on Catalyst 9800 WLC

Contents

Introduction

This document describes the configuration of Local EAP on Catalyst 9800 WLCs (Wireless LAN Controllers).

Prerequisites

Requirements

This document describes the configuration of Local EAP (Extensible Authentication Protocol) on Catalyst 9800 WLCs; that is, the WLC perform as RADIUS authentication server for the wireless clients.

This document assumes you are familiar with the basic configuration of a WLAN on the 9800 WLC and only focuses on the WLC operating as Local EAP server for wireless clients.

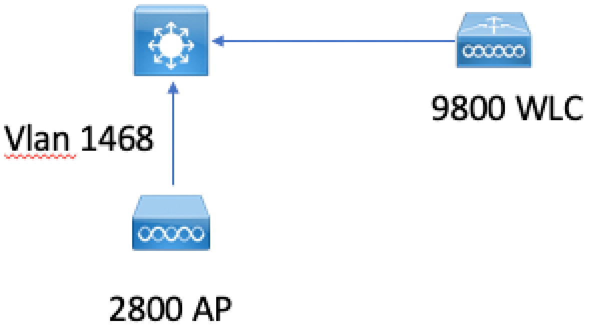
Components Used

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Catalyst 9800 on version 16.12.1s

Configure

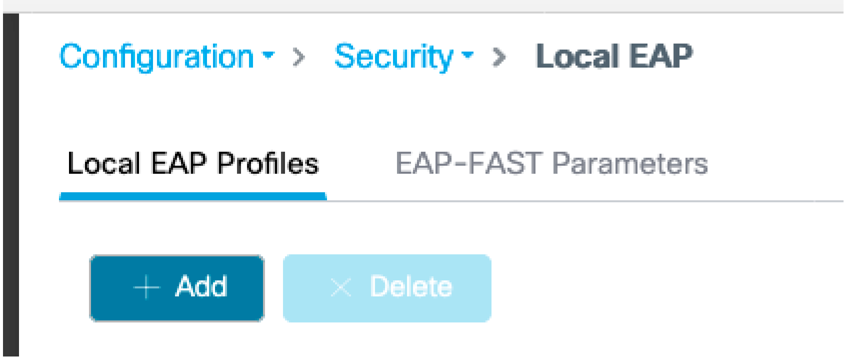
Network Diagram



Main Local EAP configuration

Step 1. Local EAP profile

Go to **Configuration** > **Security** > **Local EAP** in the 9800 web UI.

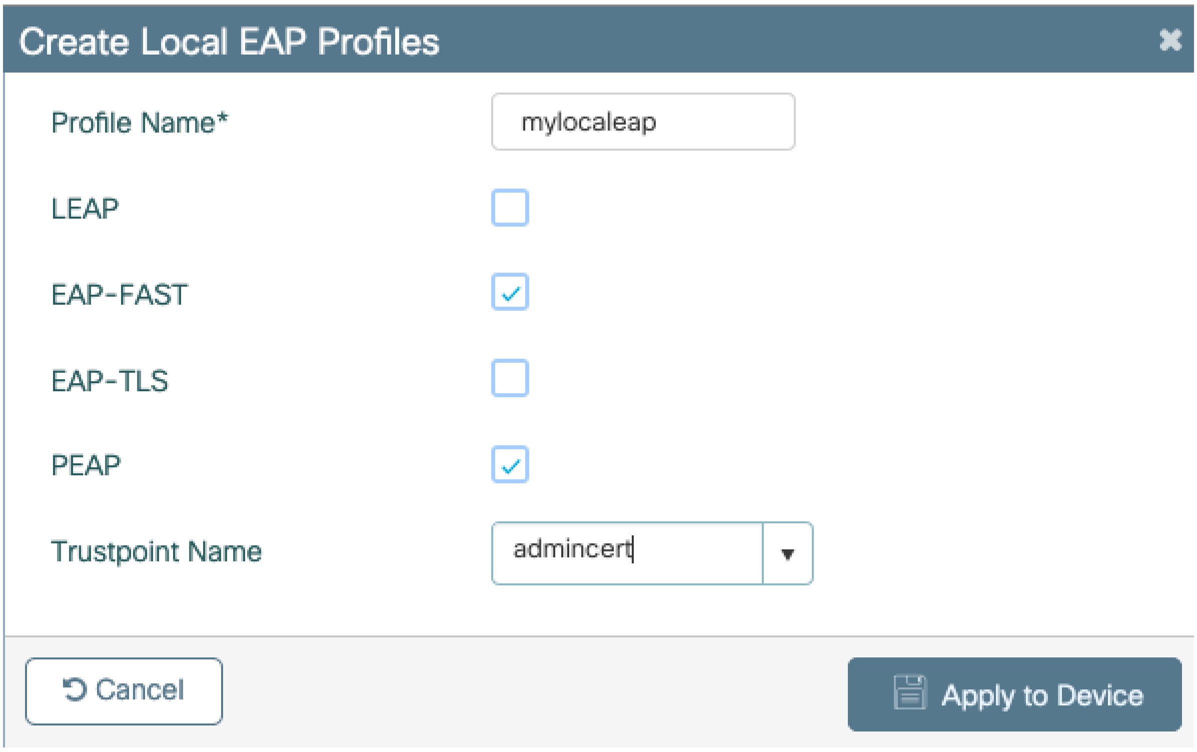


Select **Add**

Enter a profile name.

It is not advised to use LEAP at all due to its weak security. Any of the other 3 EAP methods requires you to configure a trustpoint. This is because the 9800, which acts as authenticator has to send a certificate for the client to trust it.

Clients do not trust the WLC default certificate, so you would need to deactivate server certificate validation on the client side (not advised) or install a certificate trustpoint on the 9800 WLC that the client trusts (or import it manually in the client trust store).



Create Local EAP Profiles

Profile Name*: mylocaleap

LEAP □

EAP-FAST ☑

EAP-TLS □

PEAP ☑

Trustpoint Name: admincert

CLI:

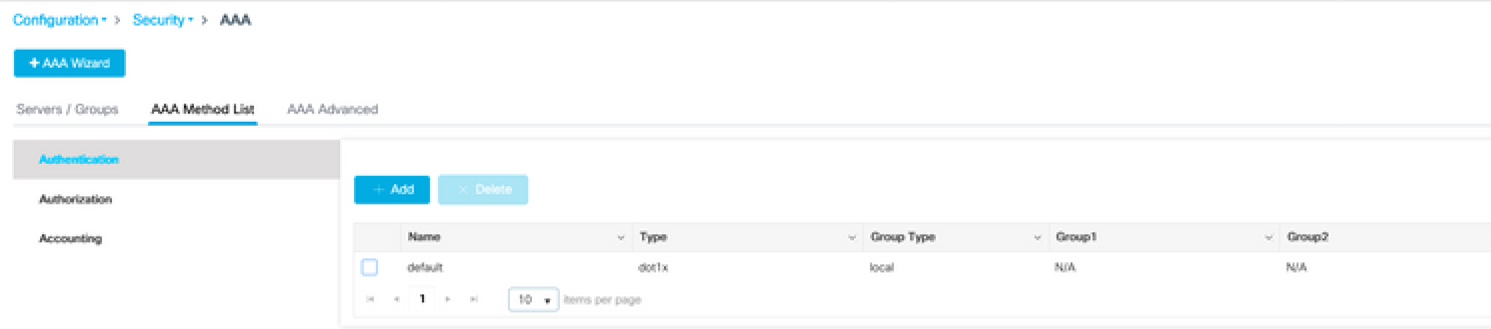
```
(config)#eap profile mylocapeap
(config-eap-profile)#method peap
(config-eap-profile)#pki-trustpoint admincert
```

Step 2. AAA authentication method

You need to configure a AAA dot1x method that points locally as well in order to use the local database of users (but you could use external LDAP lookup for example).

Go to **Configuration> Security > AAA** and go to the **AAA method list** tab for **Authentication**. Select **Add.**

Choose "dot1x" type and local group type.



	Name	Type	Group Type	Group1	Group2
□	default	dot1x	local	N/A	N/A

Step 3. Configure a AAA authorization method

Go to **Authorization** sub-tab and create a new method for type **credential-download** and point it to local.

Do the same for **network** authorization type

CLI:

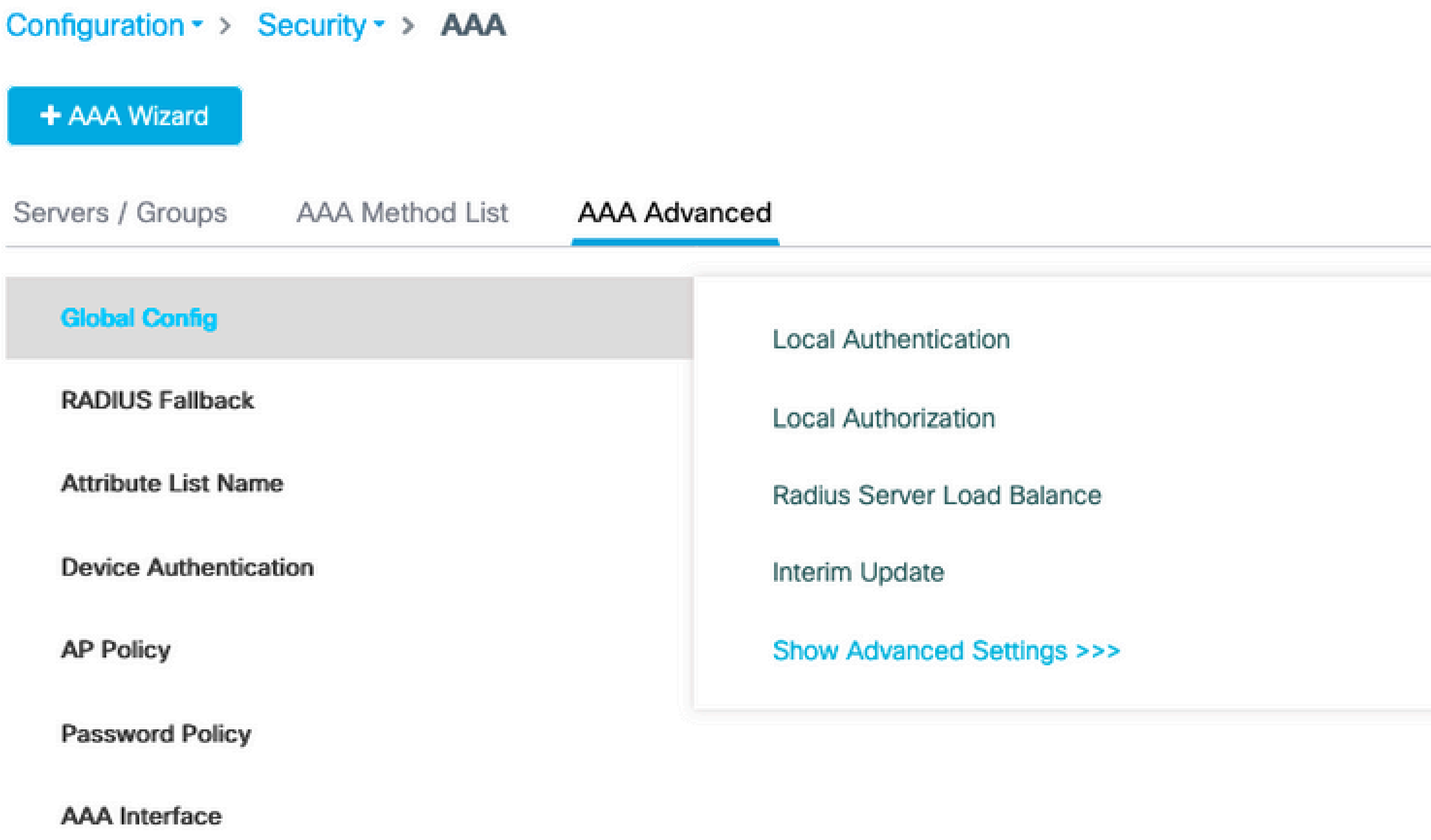
```
(config)#aaa new-model
(config)#aaa authentication dot1x default local
(config)#aaa authorization credential-download default local
(config)#aaa local authentication default authorization default
(config)#aaa authorization network default local
```

Step 4. Configure local advanced methods

Go to the **AAA advanced** tab.

Define the local authentication and authorization method. Since this exampled used the "default" credential-download and "Default" dot1x method, you need to set default for both local authentication and authorization drop down boxes here.

In case you defined named methods, pick "method list" in the dropdown and another field allows you to enter your method name.



CLI:

```
aaa local authentication default authorization default
```

Step 5. Configure a WLAN

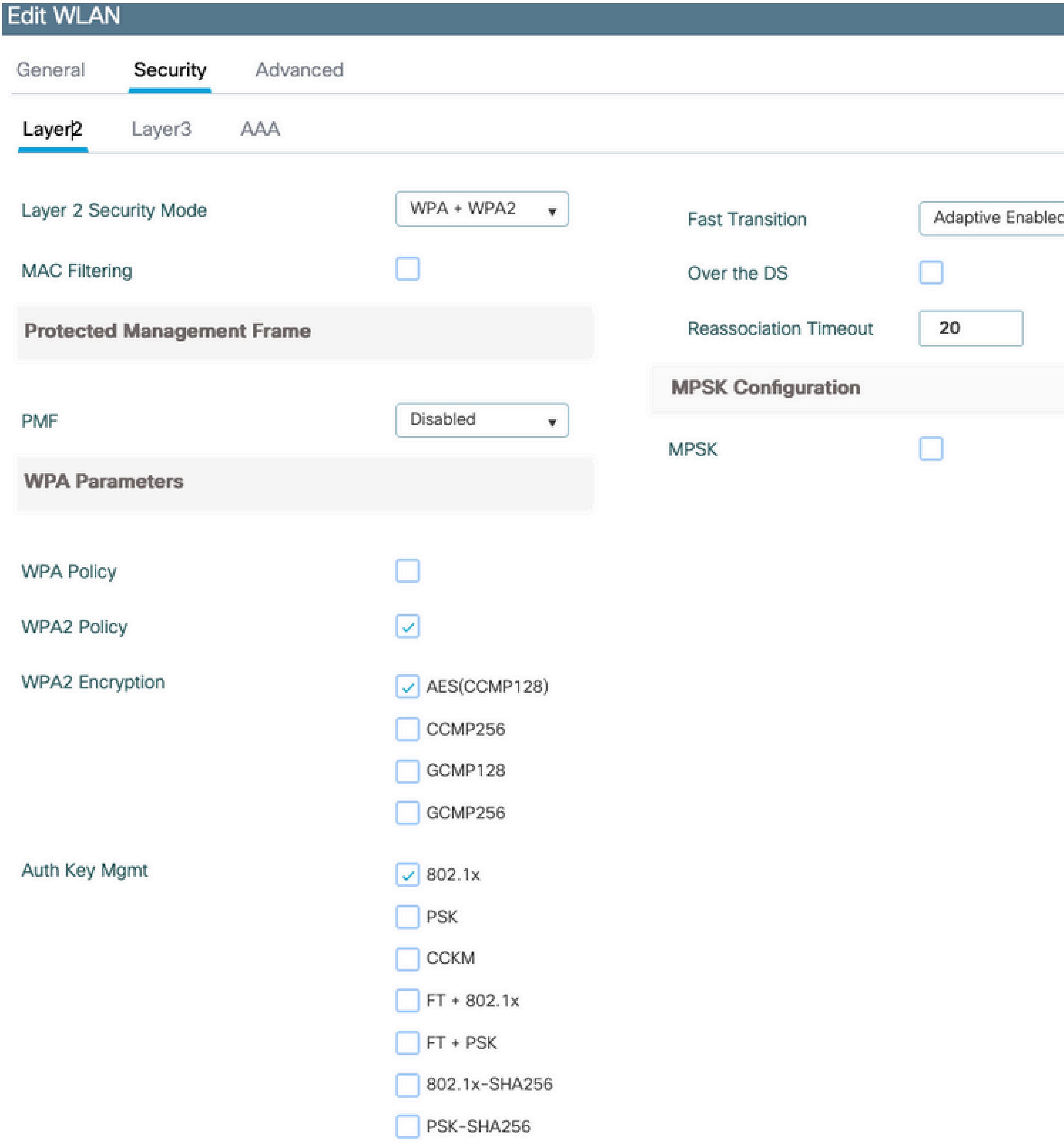
You can then configure your WLAN for 802.1x security against the local EAP profile and AAA authentication method defined in the previous step.

Go to Configuration > Tags and Profiles > WLANs > + Add >

Provide SSID and Profile Name.

Dot1x security is selected by default under Layer 2.

Under AAA, select Local EAP Authentication and choose Local EAP profile and AAA Authentication list from drop-down.


Edit WLAN

Layer 2 Security Mode: WPA + WPA2

MAC Filtering: □

Protected Management Frame

PMF: Disabled

WPA Parameters

WPA Policy: □

WPA2 Policy: ☑

WPA2 Encryption:
- ☑ AES(CCMP128)
- □ CCMP256
- □ GCMP128
- □ GCMP256

Auth Key Mgmt:
- ☑ 802.1x
- □ PSK
- □ CCKM
- □ FT + 802.1x
- □ FT + PSK
- □ 802.1x-SHA256
- □ PSK-SHA256

Fast Transition: Adaptive Enablec

Over the DS: □

Reassociation Timeout: 20

MPSK Configuration

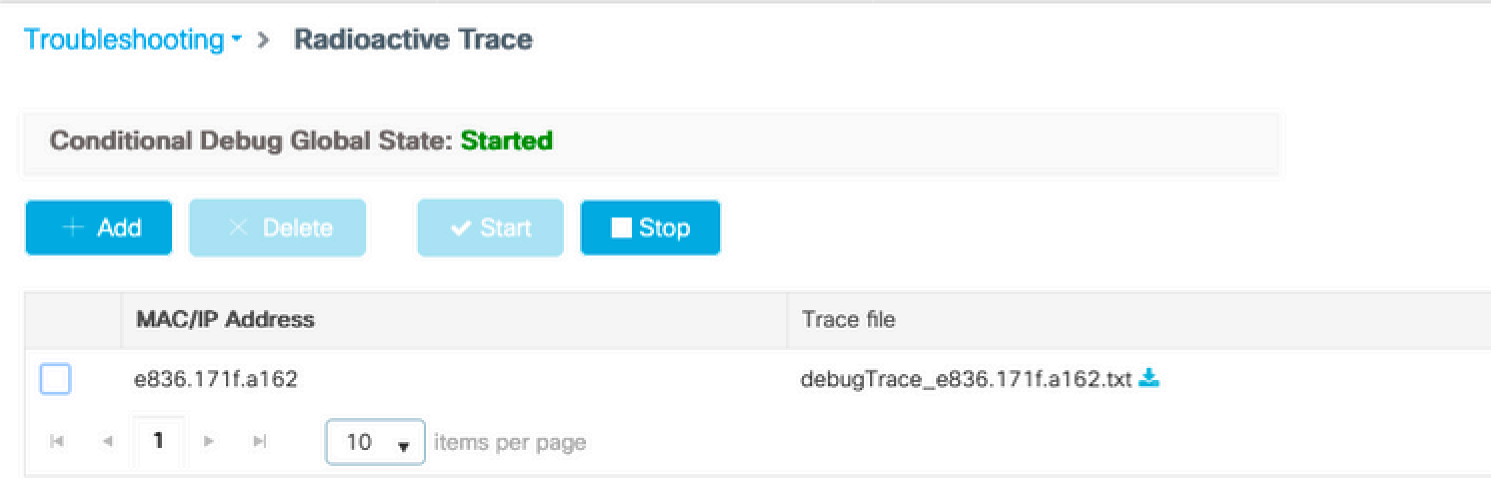
MPSK: □

Edit WLAN

16.12 and earlier releases only support TLS 1.0 for local eap authentication which could cause issues if your client supports only TLS 1.2 as is more and more the norm. Cisco IOS® XE 17.1 and later support TLS 1.2 and TLS 1.0.

In order to troubleshoot a specific client which has trouble connecting, use RadioActive Tracing. Go to **Troubleshooting > RadioActive Trace** and add the client mac address.

Select **Start** to enable the tracing for that client.



Conditional Debug Global State: Started

	MAC/IP Address	Trace file
□	e836.171f.a162	debugTrace_e836.171f.a162.txt

Once the problem is reproduced, you can select the **Generate** button in order to produce a file that contains the debugging output.

Example of a client that fails to connect due to wrong password

```
2019/10/30 14:54:00.781 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Sent
2019/10/30 14:54:00.781 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.784 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rece
2019/10/30 14:54:00.784 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.785 {wncd_x_R0-0}{2}: [caaa-authen] [23294]: (info): [CAAA:AUTHEN:66000006] DEBUG: m
2019/10/30 14:54:00.788 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Sent
2019/10/30 14:54:00.788 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.791 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rece
2019/10/30 14:54:00.791 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.791 {wncd_x_R0-0}{2}: [caaa-authen] [23294]: (info): [CAAA:AUTHEN:66000006] DEBUG: m
2019/10/30 14:54:00.792 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Sent
2019/10/30 14:54:00.792 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.795 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rece
2019/10/30 14:54:00.795 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.795 {wncd_x_R0-0}{2}: [caaa-authen] [23294]: (info): [CAAA:AUTHEN:66000006] DEBUG: m
2019/10/30 14:54:00.796 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Sent
2019/10/30 14:54:00.796 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.804 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rece
2019/10/30 14:54:00.804 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.804 {wncd_x_R0-0}{2}: [caaa-authen] [23294]: (info): [CAAA:AUTHEN:66000006] DEBUG: m
2019/10/30 14:54:00.805 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Sent
2019/10/30 14:54:00.805 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.808 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rece
2019/10/30 14:54:00.808 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.808 {wncd_x_R0-0}{2}: [caaa-authen] [23294]: (info): [CAAA:AUTHEN:66000006] DEBUG: m
2019/10/30 14:54:00.808 {wncd_x_R0-0}{2}: [eap] [23294]: (info): FAST:EAP_FAIL from inner method MSCHAPV
```

```
2019/10/30 14:54:00.808 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Sent
2019/10/30 14:54:00.808 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.811 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rece
2019/10/30 14:54:00.811 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] EAP
2019/10/30 14:54:00.811 {wncd_x_R0-0}{2}: [caaa-authen] [23294]: (info): [CAAA:AUTHEN:66000006] DEBUG: m
2019/10/30 14:54:00.812 {wncd_x_R0-0}{2}: [eap-auth] [23294]: (info): FAIL for EAP method name: EAP-FAST
2019/10/30 14:54:00.812 {wncd_x_R0-0}{2}: [dot1x] [23294]: (info): [e836.171f.a162:capwap_90000004] Rais
2019/10/30 14:54:00.813 {wncd_x_R0-0}{2}: [errmsg] [23294]: (note): %DOT1X-5-FAIL: Authentication failed
2019/10/30 14:54:00.813 {wncd_x_R0-0}{2}: [auth-mgr] [23294]: (info): [e836.171f.a162:capwap_90000004] A
```

Trace on failure

It is possible to check the list of failure events for a given mac address with the trace-on-failure command, even when no debugs are enabled.

In the next example, the AAA method was absent at first (AAA server down event) and then the client used wrong credentials a few minutes later.

The command is **show logging trace-on-failure summary** in release 16.12 and before and is **show logging profile wireless (filter mac <mac>) trace-on-failure** in Cisco IOS® XE 17.1 and later. There is no technical difference apart that 17.1 and later allows you to filter for the client mac address.

```
Nico9800#show logging profile wireless filter mac e836.171f.a162 trace-on-failure
Displaying logs from the last 0 days, 0 hours, 10 minutes, 0 seconds
executing cmd on chassis 2 ...
sending cmd to chassis 1 ...
Collecting files on current[1] chassis.
# of files collected = 30
Collecting files on current[2] chassis.
# of files collected = 30
Collecting files from chassis 1.
Time                        UUID                 Log

----------------------------------------------------------------------------------------------------


2019/10/30 14:51:04.438     0x0      SANET_AUTHC_FAILURE - AAA Server Down username , audit session id 0
2019/10/30 14:58:04.424     0x0      e836.171f.a162 CLIENT_STAGE_TIMEOUT State = AUTHENTICATING, WLAN pr
```