Catalyst 9800 WLC(Wireless LAN Controller)에서 모빌리티 토폴로지 구성

목차

소개

이 문서에서는 Catalyst 9800 WLC(Wireless LAN Controller)와 AireOS WLC 간의 토폴로지를 다루

는 모빌리티 컨피그레이션 시나리오에 대해 설명합니다.

사전 요구 사항

요구 사항

Cisco에서는 다음 항목에 대한 지식을 권장합니다.

- 무선 컨트롤러에 대한 CLI 또는 GUI 액세스.

사용되는 구성 요소

- **AireOS WLC** 버전 8.10 MR1 이상 또한 **Inter Release Controller Mobility (IRCM)** 특수 8.5 이미지
- 9800 WLC, Cisco IOS® XE v17.3.4

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

구성

네트워크 다이어그램

지침 및 제한 사항

1. **Mobility Group** 상자에서 9800의 이름은 "default"입니다.

참고:
1) WLC가 서로 다른 서브넷에 있는 경우 포트 UDP 16666 및 16667이 서로 열려 있는지 확인합니다.
2) 9800 WLC는 모두 동일한 버전을 실행하여 로밍 클라이언트가 레이어 3 로밍 및 게스트 앵커 시나리오 모두에서 일관된 경험을 갖도록 하는 것이 좋습니다.

두 Catalyst 9800 WLC 간의 모빌리티 터널

이 기본 예에서는 2개의 9800 컨트롤러 간에 모빌리티를 설정하는 방법을 설명합니다. 이는 일반적으로 게스트 액세스(앵커) 또는 클라이언트가 컨트롤러 간에 로밍하고 클라이언트 ID를 보존하도록 허용하는 데 사용됩니다.

C9800에서 모빌리티를 구성할 때 먼저 모빌리티 그룹 이름을 선택합니다. 미리 채워진 모빌리티 그룹 이름은 기본값이지만 원하는 값으로 사용자 지정할 수 있습니다.

빠른 레이어 2가 다음과 같은 방식으로 로밍할 경우 컨트롤러 전체에서 동일한 모빌리티 그룹 이름을 구성해야 합니다 **Fast Transition (FT)** 또는 **Cisco Centralized Key Management (CCKM)** 사용 중입니다.

기본적으로 섀시의 기본 이더넷 mac 주소는에 나와 있습니다 **show version** 모빌리티 MAC 주소용 GUI에 반영됩니다.

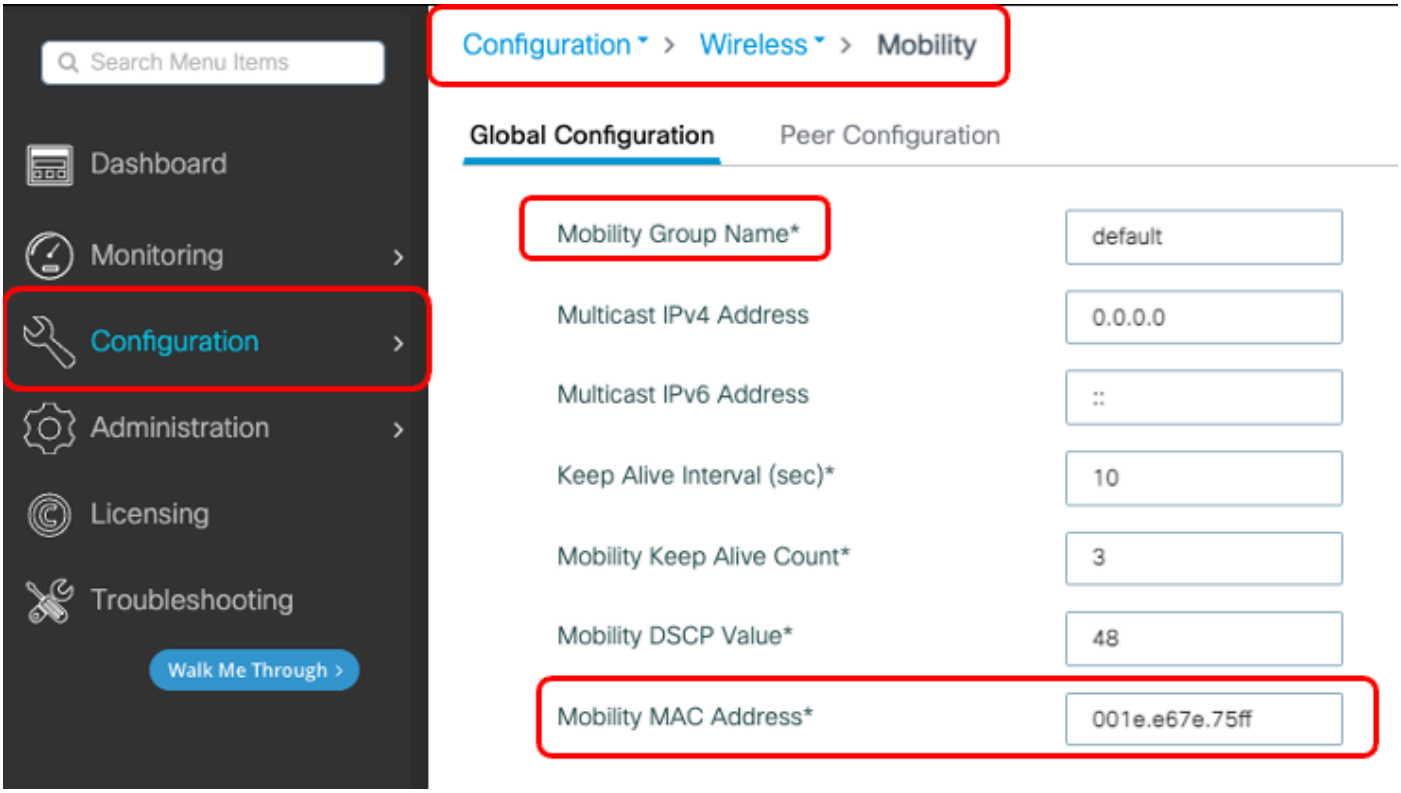
CLI에서 기본적으로 mobility mac은에 표시된 대로 0000.000.000입니다 **show run all | inc mobility mac-address**

9800이 **High Availability (HA) Stateful Switchover (SSO):**

컨피그레이션을 기본값으로 유지하고 섀시 MAC 주소를 사용하여 모빌리티 터널을 구성하는 경우 장애 조치가 발생할 때 활성 섀시 및 모빌리티 터널이 실패합니다.

따라서 C9800 HA 쌍에 대해 모빌리티 MAC 주소를 구성해야 합니다.

1단계: GUI에서 **Configuration > Wireless > Mobility > Global Configuration.**



CLI를 통해:

```
# config t
# wireless mobility mac-address  <AAAA.BBBB.CCCC>
# wireless mobility group name <mobility-group-name>
```

1단계. 9800 WLC의 모빌리티 컨피그레이션을 수집합니다.

두 9800 WLC의 경우 **Configuration > Wireless > Mobility > Global Configuration** Cisco의 IT 조직이 **Mobility Group Name** 및 **Mobility MAC Address.**

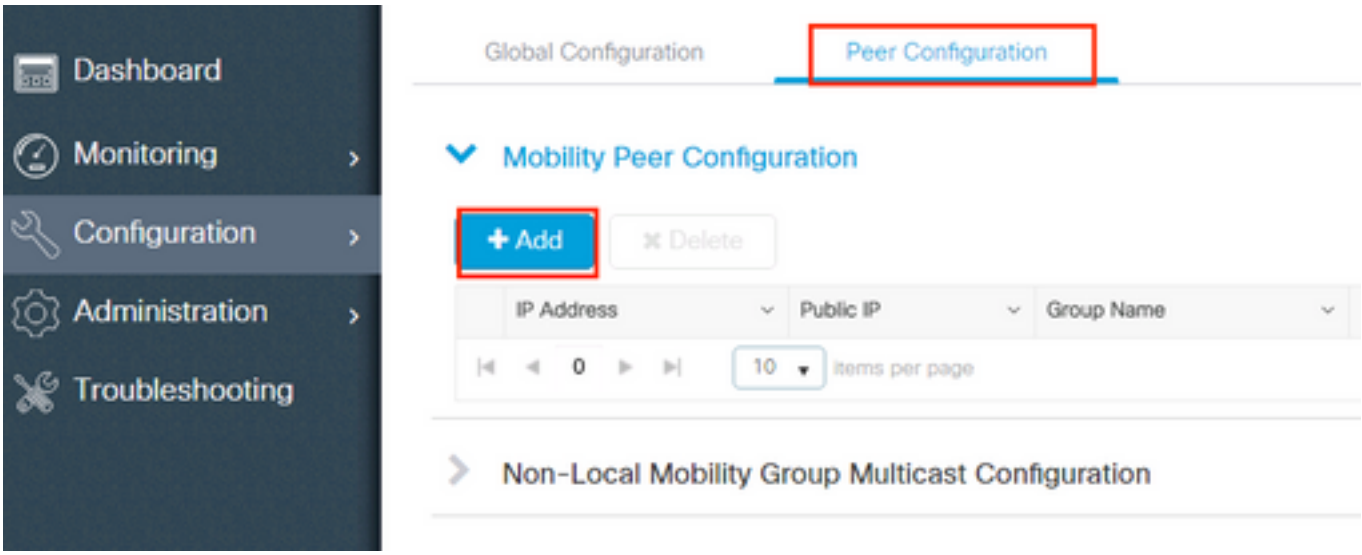
CLI를 통해:

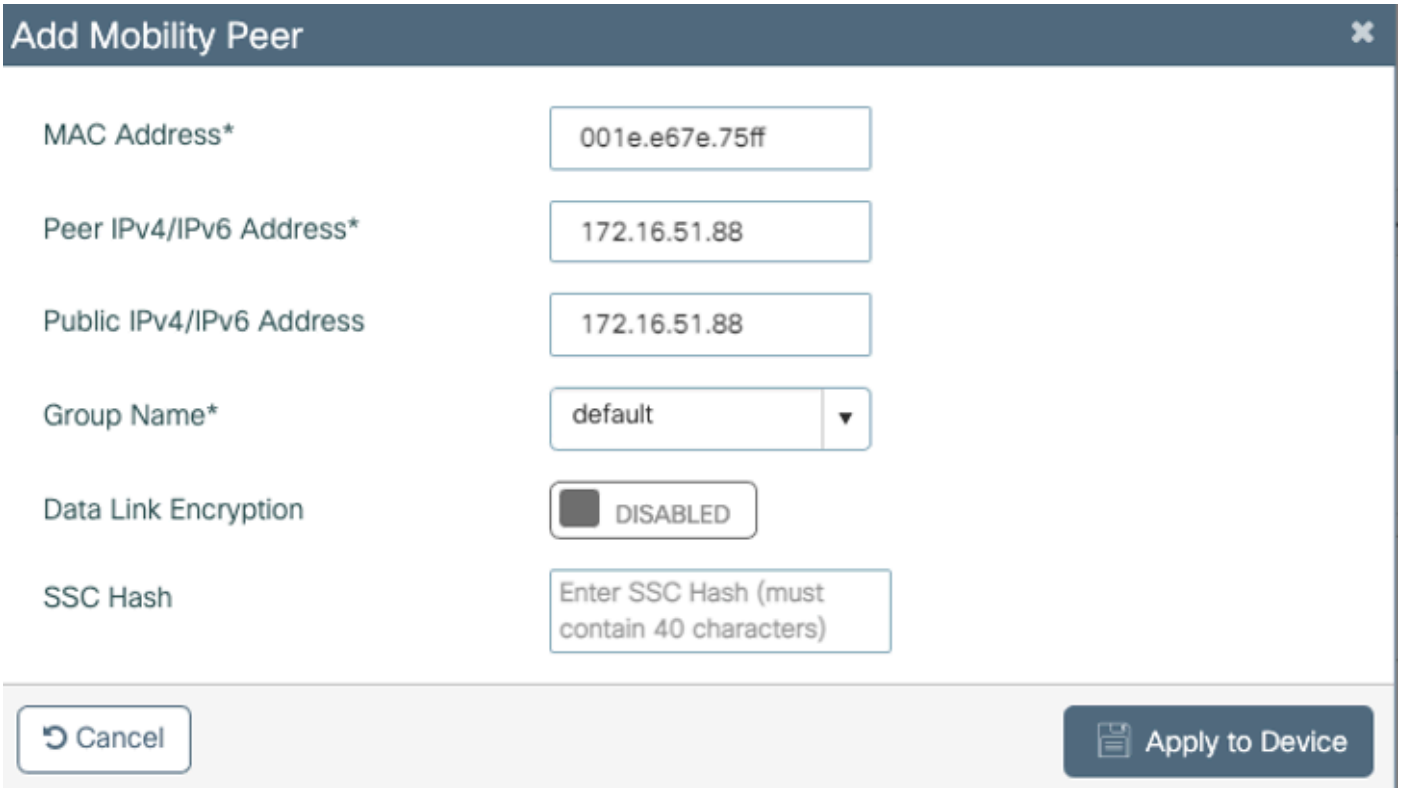
```
#show wireless mobility summary

Mobility Summary

Wireless Management VLAN: 2652
Wireless Management IP Address: 172.16.51.88
Wireless Management IPv6 Address:
Mobility Control Message DSCP Value: 48
Mobility Keepalive Interval/Count: 10/3
Mobility Group Name: default
Mobility Multicast Ipv4 address: 0.0.0.0
Mobility Multicast Ipv6 address: ::
Mobility MAC Address: 001e.e67e.75ff
Mobility Domain Identifier: 0x34ac
```

2단계. 피어 컨피그레이션 추가

탐색 **Configuration > Wireless > Mobility > Peer Configuration** 피어 컨트롤러 정보를 입력합니다. 두 9800 WLC에 대해 동일한 작업을 수행합니다.

GUI를 통해 다음을 수행합니다.





CLI를 통해:

```
# config t
# wireless mobility group member mac-address <peer-mac-address> ip <peer-ip-address> group
<group-name> [ data-link-encryption ]
```

참고: 선택적으로 데이터 링크 암호화를 활성화할 수 있습니다.

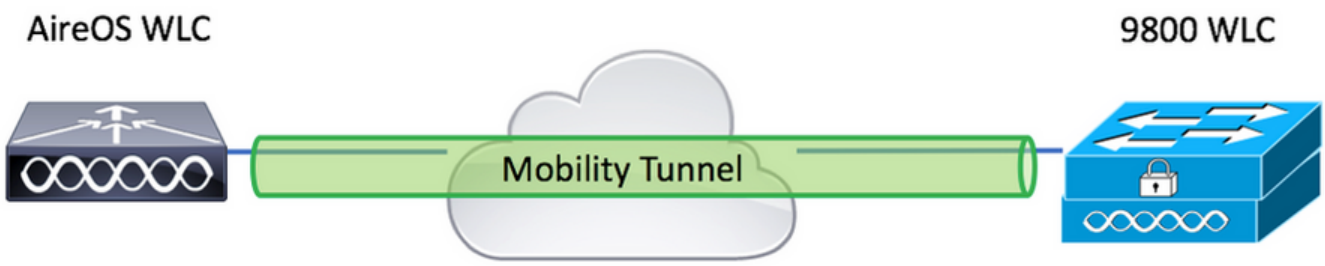
AireOS WLC와 9800-CL 컨트롤러 간의 모빌리티 터널

이 시나리오는 **brownfield** 구축 또는 컨트롤러 마이그레이션 중에 AireOS 컨트롤러에서 제어하는 AP(액세스 포인트) 영역과 9800에서 제어하는 또 다른 AP로 네트워크를 분할합니다.

AP는 물리적 또는 RF 영역별로 컨트롤러 간에 분산되므로, 클라이언트는 이동할 때만 컨트롤러 간에 로밍하는 것이 좋습니다.

회피 **salt and pepper** 구축. 선택적으로, 이 모빌리티 토폴로지는 **guest anchor** 여기서 9800은 외부 컨트롤러로 작동하고 AireOS는 앵커 컨트롤러로 작동합니다.

네트워크 다이어그램



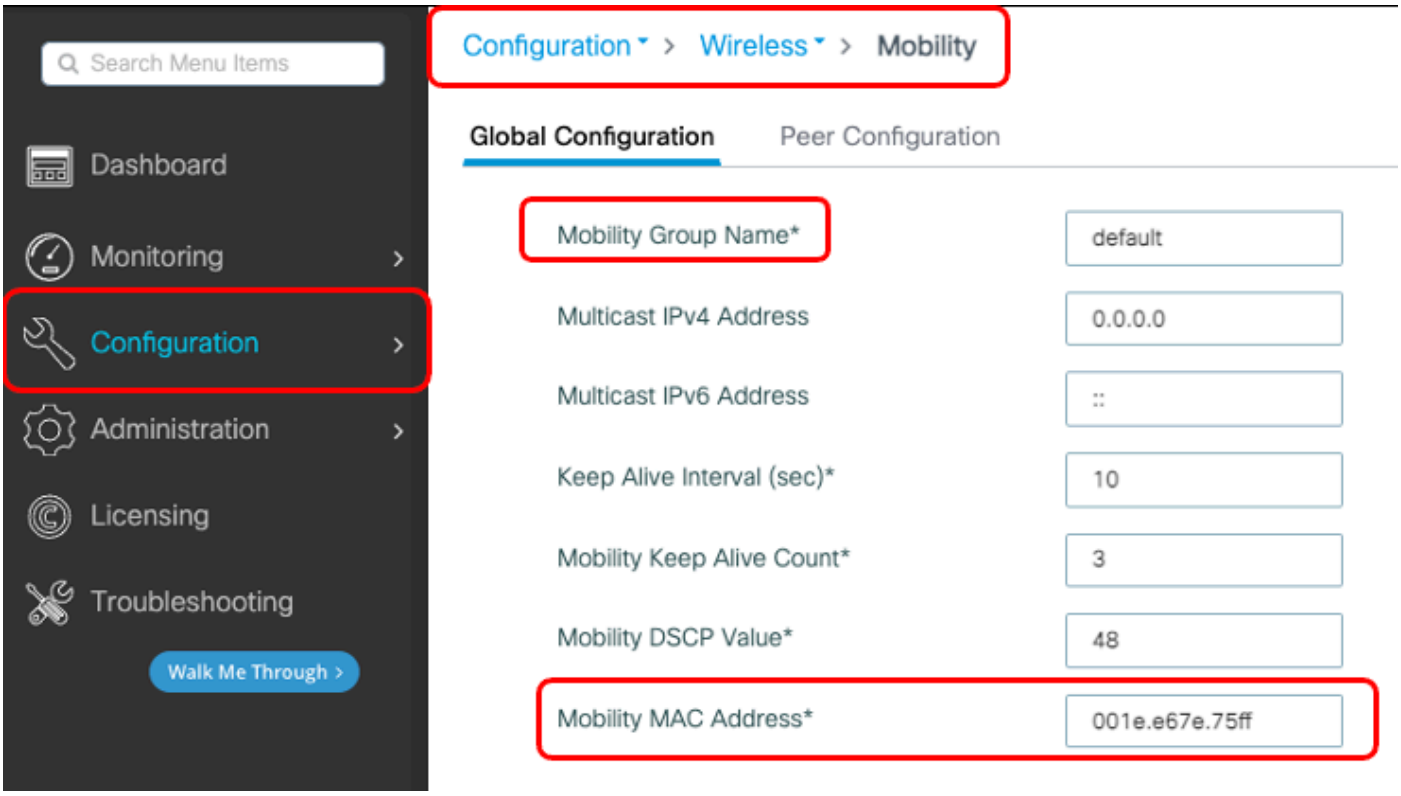
AireOS WLC 구성

9800 컨트롤러가 **High Availability**, 모빌리티 MAC 주소를 구성했는지 확인합니다.

1단계. 9800 WLC 모빌리티 정보를 수집합니다.

GUI를 통해 다음을 수행합니다.

탐색 **Configuration > Wireless > Mobility > Global Configuration** Cisco의 IT 조직이 **Mobility Group Name** 및 **Mobility MAC Address.**



CLI를 통해:

```
#show wireless mobility summary

Mobility Summary

Wireless Management VLAN: 2652
Wireless Management IP Address: 172.16.51.88
Wireless Management IPv6 Address:
Mobility Control Message DSCP Value: 48
Mobility Keepalive Interval/Count: 10/3
Mobility Group Name: default
Mobility Multicast Ipv4 address: 0.0.0.0
Mobility Multicast Ipv6 address: ::
Mobility MAC Address: 001e.e67e.75ff
Mobility Domain Identifier: 0x34ac
```

2단계. 9800 WLC에서 해시 값 수집

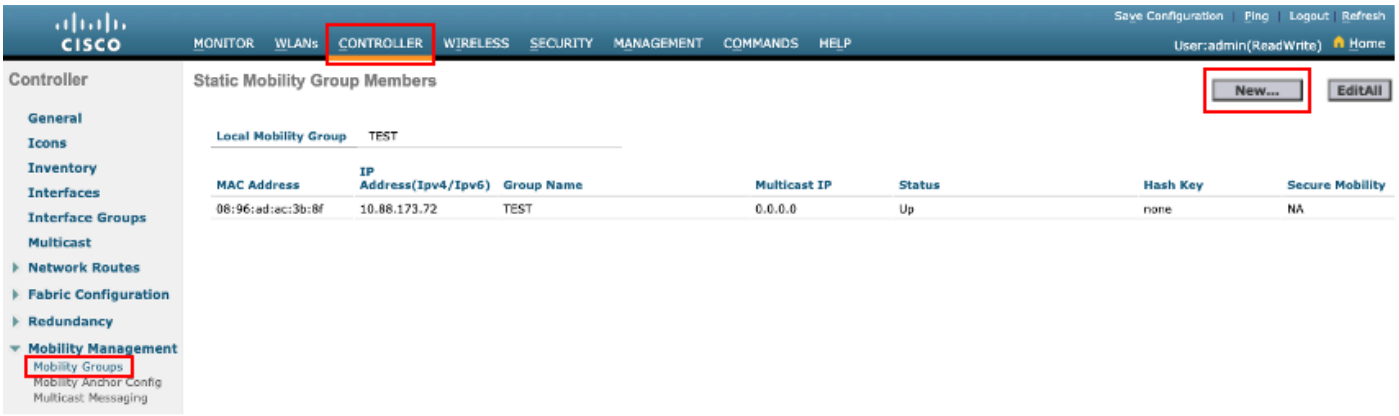
```
# show wireless management trustpoint
Trustpoint Name : Jay-9800_WLC_TP
```

```
Certificate Info : Available
Certificate Type : SSC
Certificate Hash : d7bde0898799dbfeffd4859108727d3372d3a63d
Private key Info : Available
FIPS suitability : Not Applicable
```

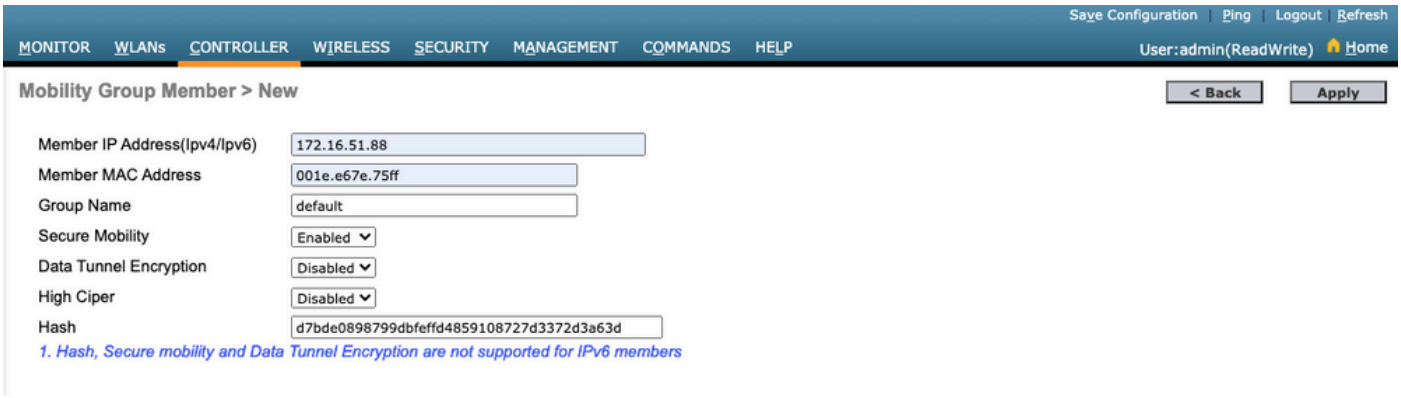
3단계. 9800 WLC 정보를 AireOS WLC에 추가합니다.

GUI를 통해 다음을 수행합니다.

탐색 **CONTROLLER > Mobility Management > Mobility Groups > New.**



값을 입력하고 **Apply.**



> **참고**: 해시는 9800에서 C9800-CL과 같은 자체 서명 인증서를 사용하는 경우에만 필요합니다. 하드웨어 어플라이언스에는 SUDI 인증서가 있으며 해시(예: 9800-40, 9800-L 등)가 필요하지 않습니다.

CLI를 통해:

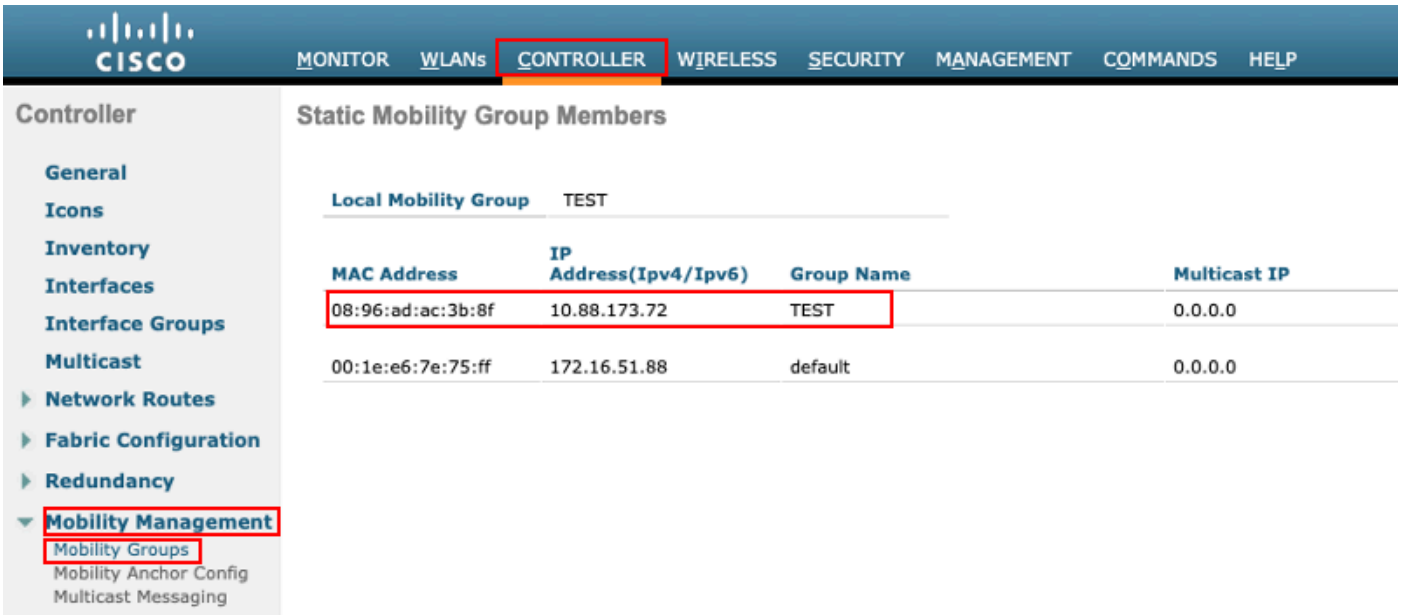
```
>config mobility group member add <9800 mac-address> <9800 WLC-IP> <group-name> encrypt enable
>config mobility group member hash <9800 WLC-IP> <9800 WLC-Hash>
>config mobility group member data-dtls <9800 mac-address> disable
```

9800 WLC 컨피그레이션

1단계. AireOS 모빌리티 정보를 수집합니다.

GUI를 통해 다음을 수행합니다.

AireOS GUI에 로그인하여 **CONTROLLER > Mobility Management > Mobility Groups** MAC 주소, IP 주소 및 그룹 이름을 기록해 둡니다.



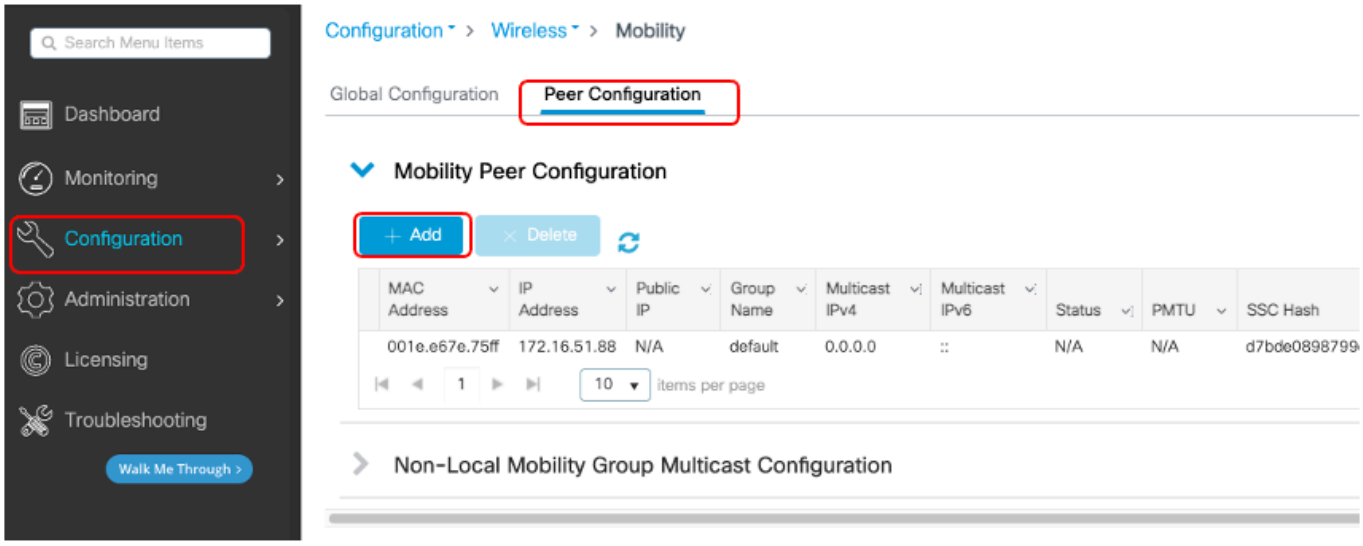
CLI를 통해:

```
>show mobility summary

Mobility Protocol Port.......................... 16666
Default Mobility Domain......................... TEST
Multicast Mode ................................. Disabled
Mobility Domain ID for 802.11r.................. 0x6ef9
Mobility Keepalive Interval..................... 10
Mobility Keepalive Count........................ 3
Mobility Group Members Configured............... 2
Mobility Control Message DSCP Value............. 48

Controllers configured in the Mobility Group

 MAC Address        IP Address         Group Name                  Multicast IP
Status
08:96:ad:ac:3b:8f   10.88.173.72         TEST                        0.0.0.0
Up
```

2단계. 9800 WLC에 AireOS WLC 정보를 추가합니다.

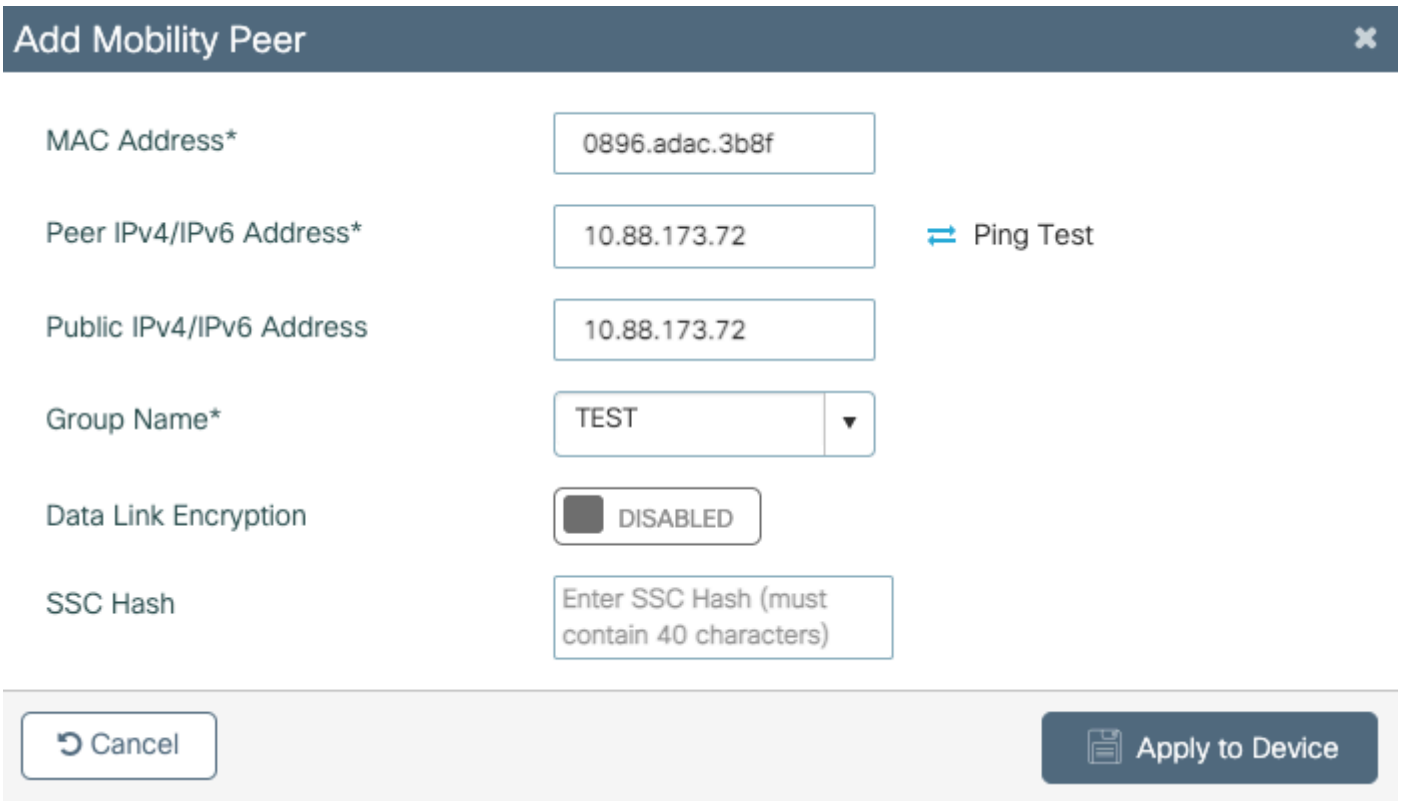
GUI를 통해 다음을 수행합니다.

탐색 **Configuration > Wireless > Mobility > Peer Configuration > Add**



AireOS WLC 정보를 입력 합니다.

> **참고**: 9800 WLC에서는 컨트롤 플레인 암호화가 항상 활성화되어 있으므로 AireOS 측에서 보안 모빌리티를 활성화해야 합니다. 그러나 데이터 링크 암호화는 선택 사항입니다. 9800 쪽에서 활성화한 경우 AireOS에서 다음을 사용하여 활성화하십시오. **config mobility group member data-dtls enable**



CLI를 통해:

```
# config t
# wireless mobility group member mac-address <peer-mac-address> ip <ip-address> group <group-name>
```

다음을 확인합니다.

구성이 올바르게 작동하는지 확인하려면 이 섹션을 활용하십시오.

AireOS WLC 확인

```
>show mobility summary

Mobility Protocol Port..... 16666
Default Mobility Domain..... TEST
Multicast Mode ..... Disabled
Mobility Domain ID for 802.11r..... 0x6ef9
Mobility Keepalive Interval..... 10
Mobility Keepalive Count..... 3
Mobility Group Members Configured..... 2
Mobility Control Message DSCP Value..... 48

Controllers configured in the Mobility Group
 MAC Address        IP Address                                    Group Name
Multicast IP                                  Status
 00:1e:e6:7e:75:ff  172.16.51.88                                   default
0.0.0.0                                       Up
 08:96:ad:ac:3b:8f  10.88.173.72                                    TEST
0.0.0.0                                       Up
```

Catalyst 9800 WLC 확인

```
#show wireless mobility summary
Mobility Summary

Wireless Management VLAN: 2652
Wireless Management IP Address: 172.16.51.88
Mobility Control Message DSCP Value: 48
Mobility Keepalive Interval/Count: 10/3
Mobility Group Name: mb-kcg
Mobility Multicast Ipv4 address: 0.0.0.0
Mobility Multicast Ipv6 address: ::
Mobility MAC Address: 001e.e67e.75ff

Controllers configured in the Mobility Domain:

 IP               Public Ip         Group Name                      Multicast IPv4    Multicast
IPv6                               Status                       PMTU
-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
172.16.51.88       N/A               default                          0.0.0.0           ::
N/A                          N/A
10.88.173.72      10.88.173.72        TEST                            0.0.0.0           ::
Up                           1385
```

문제 해결

이 섹션에서는 컨피그레이션 트러블슈팅에 사용되는 정보를 제공합니다.

모빌리티 터널 구현 문제를 해결하려면 다음 명령을 사용하여 프로세스를 디버깅합니다.

아이레OS WLC

1단계. 모빌리티 디버그를 활성화합니다.

```
debug mobility handoff enable
debug mobility error enable
debug mobility dtls error enable
debug mobility dtls event enable
debug mobility pmtu-discovery enable
debug mobility config enable
debug mobility directory enable
```

2단계. 컨피그레이션 재생 및 출력 확인

AirOS WLC에서 성공적인 모빌리티 터널 생성의 예.

```
*capwapPingSocketTask: Feb 07 09:53:38.507: Client initiating connection on 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.507: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.508: Received DTLS packet from mobility peer 172.16.0.21
bytes: 48
*capwapPingSocketTask: Feb 07 09:53:38.508: mm_dtls2_process_data_rcv_msg:1207 rcvBufLen 48
clr_pkt_len 2048 peer ac100015
*capwapPingSocketTask: Feb 07 09:53:38.508: Record    : type=22, epoch=0, seq=0
*capwapPingSocketTask: Feb 07 09:53:38.508:    Hndshk : type=3, len=23 seq=0, frag_off=0,
frag_len=23
*capwapPingSocketTask: Feb 07 09:53:38.508: Handshake in progress for link 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.508: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.508: DTLS consumed packet from mobility peer 172.16.0.21
bytes: 48
!
!<--output-omited-->
!
*capwapPingSocketTask: Feb 07 09:53:38.511: dtls2_cert_verify_callback: Forcing Certificate
validation as success
*capwapPingSocketTask: Feb 07 09:53:38.511: Peer certificate verified.
*capwapPingSocketTask: Feb 07 09:53:38.511: Handshake in progress for link 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.511: Nothing to send on link 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.511: DTLS consumed packet from mobility peer 172.16.0.21
bytes: 503
*capwapPingSocketTask: Feb 07 09:53:38.511: Received DTLS packet from mobility peer 172.16.0.21
bytes: 56
*capwapPingSocketTask: Feb 07 09:53:38.511: mm_dtls2_process_data_rcv_msg:1207 rcvBufLen 56
clr_pkt_len 2048 peer ac100015
*capwapPingSocketTask: Feb 07 09:53:38.511: Record    : type=22, epoch=0, seq=6
*capwapPingSocketTask: Feb 07 09:53:38.511:    Hndshk : type=13, len=6 seq=3, frag_off=0,
frag_len=6
*capwapPingSocketTask: Feb 07 09:53:38.523: Handshake in progress for link 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.523: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.523: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.523: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.523: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.523: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.524: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.524: Sending packet to 172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.524: DTLS consumed packet from mobility peer 172.16.0.21
bytes: 56
*capwapPingSocketTask: Feb 07 09:53:38.527: Received DTLS packet from mobility peer 172.16.0.21
bytes: 91
*capwapPingSocketTask: Feb 07 09:53:38.527: mm_dtls2_process_data_rcv_msg:1207 rcvBufLen 91
```

```
clr_pkt_len 2048 peer ac100015
*capwapPingSocketTask: Feb 07 09:53:38.527: Record    : type=20, epoch=0, seq=8
*capwapPingSocketTask: Feb 07 09:53:38.527: Connection established for link 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.527: ciperspec 1
*capwapPingSocketTask: Feb 07 09:53:38.527: Nothing to send on link 172.16.0.5:16667 <->
172.16.0.21:16667
*capwapPingSocketTask: Feb 07 09:53:38.527: DTLS consumed packet from mobility peer 172.16.0.21
bytes: 91
*mmMobility: Feb 07 09:53:38.527: DTLS Action Result message received
*mmMobility: Feb 07 09:53:38.527:  Key plumb succeeded
*mmMobility: Feb 07 09:53:38.527: mm_dtls2_callback: Connection established with
172.16.0.21:16667
*mmMobility: Feb 07 09:53:38.527: mm_dtls2_db_status_up:895 Connections status up for entry
172.16.0.21:16667
*mmMobility: Feb 07 09:53:38.527: mm_dtls2_callback: DTLS Connection established with
172.16.0.21:16667, Sending update msg to mobility HB
```

Catalyst 9800 WLC

기본적으로 9800 컨트롤러는 특별한 디버그 절차 없이 프로세스 정보를 지속적으로 로깅합니다.

문제 해결을 위해 컨트롤러에 연결하고 무선 구성 요소와 연결된 로그를 검색하기만 하면 됩니다.

로그는 며칠에 걸쳐 있을 수 있습니다. 이는 컨트롤러의 사용 빈도에 따라 달라집니다.

분석을 간소화하려면 시간 범위 또는 마지막 시간(기본 시간은 10분으로 설정됨) 동안 로그를 가져오고 IP 또는 MAC 주소로 필터링할 수 있습니다.

1단계. 문제가 발생했을 때까지의 시간에 로그를 추적할 수 있도록 컨트롤러 시간의 현재 시간을 확인합니다.

```
# show clock
```

2단계. 문제와 관련될 수 있는 Cisco IOS 레벨의 정보가 있는 경우 컨트롤러 로그를 수집합니다.

```
# show logging
```

3단계. 특정 주소에 대한 상시 작동 알림 레벨 추적을 수집합니다. 모바일 피어 IP 또는 MAC를 사용하여 필터링할 수 있습니다.

```
# show logging profile wireless filter ipv4 to-file bootflash:ra-AAAA.BBBB.CCCC.txt
```

이 명령은 최근 10분 동안 로그를 생성하며, 이 시간을 명령으로 조정할 수 있습니다 **show logging profile wireless last 1 hour filter mac AAAA.BBBB.CCCC to-file bootflash:ra-AAAA.BBBB.CCCC.txt**.

세션에 내용을 표시하거나 파일을 외부 TFTP 서버에 복사할 수 있습니다.

```
# more bootflash:always-on-<FILENAME.txt>

or

# copy bootflash:always-on-<FILENAME.txt> tftp://a.b.c.d/path/always-on-<FILENAME.txt>
```

무선 활성 추적

Always-on 로그가 터널 컨피그레이션 중에 트리거된 문제를 알기에 충분한 정보를 제공하지 않는 경우 조건부 디버깅 및 캡처를 활성화할 수 있습니다 **Radio Active (RA)** 추적 - 더 자세한 프로세스 활동을 제공합니다.

1단계. 이미 활성화된 디버그 조건이 없는지 확인합니다.

```
# show debugging
IOSXE Conditional Debug Configs:

Conditional Debug Global State: Stop


IOSXE Packet Tracing Configs:




Packet Infra debugs:

Ip Address                                              Port
------------------------------------------------------|----------
```

모니터링할 주소와 관련이 없는 조건이 표시되면 비활성화합니다.

특정 주소를 제거하려면

```
# no debug platform condition feature wireless { mac <aaaa.bbbb.cccc> | ip <a.b.c.d> }
```

모든 조건을 제거하려면(권장 방법):

```
# clear platform condition all
```

2단계. 모니터링할 주소의 디버그 조건을 추가합니다.

```
# debug platform condition feature wireless ip <a.b.c.d>
```

> **참고**: 둘 이상의 모빌리티 피어를 동시에 모니터링하려면 **debug platform condition feature wireless mac** MAC 주소당 명령.

3단계. 9800 WLC에서 지정된 주소 활동의 모니터링을 시작하도록 합니다.

```
# debug platform condition start
```

> **참고**: 나중에 수집하기 위해 모든 것이 내부적으로 버퍼링되므로 모빌리티 활동의 출력이 표시되지 않습니다.

4단계. 모니터링할 문제나 동작을 재현합니다.

5단계. 디버그를 중지합니다.

```
# debug platform condition stop
```

6단계. 주소 활동의 출력을 수집합니다.

```
# show logging profile wireless filter ipv4 to-file bootflash:ra-AAAA.BBBB.CCCC.txt
```

이 명령은 지난 10분 동안 로그를 생성합니다. 명령 **show logging profile wireless last 1 hour filter mac AAAA.BBB.CCCC to-file bootflash:ra-AAAA.BBB.CCCC.txt**를 사용하여 이 시간을 조정할 수 있습니다.

다음 중 하나를 복사하거나 **FILENAME.txt** 외부 서버에 연결하거나 화면에 출력을 직접 표시합니다.

파일을 외부 서버에 복사:

```
# copy bootflash:FILENAME.txt tftp://a.b.c.d/ra-FILENAME.txt
```

콘텐츠 표시:

```
# more bootflash:ra-FILENAME.txt
```

7단계. 여전히 실패 원인을 찾을 수 없는 경우 로그의 내부 레벨을 수집합니다.

(클라이언트를 다시 디버깅할 필요는 없습니다. 내부적으로 이미 저장된 로그를 사용하되, 더 광범위한 로그를 수집합니다.)

```
# show logging profile wireless internal filter ipv4 to-file bootflash:raInternal-
AAAA.BBBB.CCCC.txt
```

다음 중 하나를 복사하거나 **FILENAME.txt** 외부 서버에 연결하거나 화면에 출력을 직접 표시합니다.

파일을 외부 서버에 복사:

```
# copy bootflash:FILENAME.txt tftp://a.b.c.d/ra-FILENAME.txt
```

콘텐츠 표시:

```
# more bootflash:ra-FILENAME.txt
```

8단계. 디버그 조건을 제거합니다.

```
# clear platform condition all
```

참고: 문제 해결 세션 후 항상 디버그 상태를 제거 합니다.

9800 WLC에서 성공적인 모빌리티 터널 생성의 예.

```
2021/09/28 10:20:50.497612 {mobilityd_R0-0}{1}: [errmsg] [26516]: (info): %MM_NODE_LOG-6-
MEMBER_ADDED: Adding Mobility member (IP: IP: 172.16.55.28: default)
2021/09/28 10:20:52.595483 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_data of XID (0) to (ipv4: 172.16.55.28 )
```

```
2021/09/28 10:20:52.595610 {mobilityd_R0-0}{1}: [mm-pmtu] [26516]: (debug): Peer IP:
172.16.55.28 PMTU size is 1385 and calculated additional header length is 148
2021/09/28 10:20:52.595628 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_ctrl_req of XID (80578) to (ipv4: 172.16.55.28 )
2021/09/28 10:20:52.595686 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive data packet misssed, total missed packet = 1
2021/09/28 10:20:52.595694 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive ctrl packet misssed, total missed packet = 1
2021/09/28 10:21:02.596500 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_data of XID (0) to (ipv4: 172.16.55.28 )
2021/09/28 10:21:02.596598 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive data packet misssed, total missed packet = 2
2021/09/28 10:21:02.598898 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
001e.e68c.5dff Received keepalive_data, sub type: 0 of XID (0) from (ipv4: 172.16.55.28 )
2021/09/28 10:21:12.597912 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_data of XID (0) to (ipv4: 172.16.55.28 )
2021/09/28 10:21:12.598009 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 Data link set state to UP (was DOWN)
2021/09/28 10:21:12.598361 {mobilityd_R0-0}{1}: [errmsg] [26516]: (note): %MM_NODE_LOG-5-
KEEP_ALIVE: Mobility Data tunnel to peer IP: 172.16.55.28 changed state to UP

! !<--output-omited--> !

2021/09/28 10:21:22.604098 {mobilityd_R0-0}{1}: [ewlc-infra-evq] [26516]: (debug): DTLS record
type: 22, handshake
2021/09/28 10:21:22.604099 {mobilityd_R0-0}{1}: [ewlc-infra-evq] [26516]: (info): DTLS client
hello
2021/09/28 10:21:22.611477 {mobilityd_R0-0}{1}: [ewlc-infra-evq] [26516]: (debug): DTLS record
type: 22, handshake
2021/09/28 10:21:22.611555 {mobilityd_R0-0}{1}: [ewlc-infra-evq] [26516]: (debug): DTLS record
type: 22, handshake
2021/09/28 10:21:22.611608 {mobilityd_R0-0}{1}: [ewlc-infra-evq] [26516]: (debug): DTLS record
type: 22, handshake
2021/09/28 10:21:22.611679 {mobilityd_R0-0}{1}: [ewlc-infra-evq] [26516]: (debug): DTLS record
type: 22, handshake
2021/09/28 10:21:22.611933 {mobilityd_R0-0}{1}: [mm-dtls] [26516]: (note): Peer IP: 172.16.55.28
Port: 16666, Local IP: 172.16.51.88 Port: 16666 DTLS_SSC_HASH_VERIFY_CB: SSC hash validation
success
2021/09/28 10:21:22.612163 {mobilityd_R0-0}{1}: [ewlc-dtls-sessmgr] [26516]: (info): Remote
Host: 172.16.55.28[16666] Completed cert verification, status:CERT_VALIDATE_SUCCESS

! !<--output-omited--> !

2021/09/28 10:21:52.603200 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 Control link set state to UP (was DOWN)
2021/09/28 10:21:52.604109 {mobilityd_R0-0}{1}: [errmsg] [26516]: (note): %MM_NODE_LOG-5-
KEEP_ALIVE: Mobility Control tunnel to peer IP: 172.16.55.28 changed state to UP
```

내장형 패킷 캡처

대부분의 경우 WLC 간에 교환되는 패킷을 확인하는 데 매우 유용합니다. 특히 다음 기능으로 캡처를 필터링하는 것이 좋습니다 **Access Control Lists (ACLs)** 캡처할 수 있습니다.

CLI의 임베디드 캡처를 위한 컨피그레이션 템플릿입니다.

1단계. 필터 ACL을 생성합니다.

```
conf t
ip access-list extended <ACL_NAME>
10 permit ip host <WLC_IP_ADDR> host <PEER_WLC_IP_ADDR>
20 permit ip host <PEER_WLC_IP_ADDR>host <WLC_IP_ADDR>
```

```
end
```

2단계. 캡처 매개변수를 정의합니다.

```
monitor capture <CAPTURE_NAME> access-list <ACL_NAME> buffer size 10 control-plane both
interface <INTERFACE_NAME> both limit duration 300
```

참고: INTERFACE_NAME 매개변수에 대한 관리 인터페이스를 선택합니다.

3단계. 캡처를 시작합니다.

```
monitor capture <CAPTURE_NAME> start
```

4단계. 캡처를 중지합니다.

```
monitor capture <CAPTURE_NAME> stop
```

5단계. Troubleshooting(트러블슈팅) **> Packet Capture on GUI(GUI에서 패킷 캡처)로 이동하여 패킷 캡처 파일을 수집합니다.**

일반적인 문제 해결 시나리오

다음 예는 9800개의 WLC 사이에 형성된 터널로 구성됩니다.

연결 문제로 인한 제어 및 데이터 경로 중단

사용 **Always-On-Logs** 및 **Embedded packet captures** 문제 해결을 위한 추가 정보를 제공하려면

```
2021/09/28 09:54:22.490625 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_ctrl_req of XID (80552) to (ipv4: 172.16.55.28 )
2021/09/28 09:54:22.490652 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive data packet misssed, total missed packet = 29
2021/09/28 09:54:22.490657 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive ctrl packet misssed, total missed packet = 10
2021/09/28 09:54:32.491952 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_data of XID (0) to (ipv4: 172.16.55.28 )
2021/09/28 09:54:32.492127 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive data packet misssed, total missed packet = 30
```

패킷 캡처는 동작을 확인하는 데 유용합니다.

```
 90 2021-09-28 12:33:52.924939 172.16.51.88   172.16.55.28   116 Mobi-Control - PingReq[Malformed Packet]
 91 2021-09-28 12:34:02.925946 172.16.51.88   172.16.55.28   172 Mobi-Data Keep-Alive - Mobility CAPWAP Ping Request
 92 2021-09-28 12:34:12.925946 172.16.51.88   172.16.55.28   172 Mobi-Data Keep-Alive - Mobility CAPWAP Ping Request
 93 2021-09-28 12:34:22.927945 172.16.51.88   172.16.55.28   172 Mobi-Data Keep-Alive - Mobility CAPWAP Ping Request
 94 2021-09-28 12:34:22.927945 172.16.51.88   172.16.55.28   116 Mobi-Control - PingReq[Malformed Packet]
 95 2021-09-28 12:34:32.927945 172.16.51.88   172.16.55.28   172 Mobi-Data Keep-Alive - Mobility CAPWAP Ping Request
 96 2021-09-28 12:34:42.929944 172.16.51.88   172.16.55.28   172 Mobi-Data Keep-Alive - Mobility CAPWAP Ping Request
 97 2021-09-28 12:34:52.930951 172.16.51.88   172.16.55.28   172 Mobi-Data Keep-Alive - Mobility CAPWAP Ping Request
```

debug 및 WLC 모두 Control 또는 Data ping에 대한 응답이 없음을 확인합니다. 일반적인 시나리오에서는 IP 연결이 허용되지만 포트 16666 또는 16667은 네트워크 전반에서 통신할 수 없음을 보여줍니다.

WLC 간 컨피그레이션 불일치

이 경우 WLC 간의 모든 포트에 대한 연결을 확인했지만 keepalive miss가 계속 표시됩니다.

사용 **Always-On-Logs** 및 **Embedded packet captures** 문제 해결을 위한 추가 정보를 제공하려면

```
2021/09/28 11:34:22.927477 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_data of XID (0) to (ipv4: 172.16.55.28 )
2021/09/28 11:34:22.928025 {mobilityd_R0-0}{1}: [mm-pmtu] [26516]: (debug): Peer IP:
172.16.55.28 PMTU size is 1385 and calculated additional header length is 148
2021/09/28 11:34:22.928043 {mobilityd_R0-0}{1}: [mm-client] [26516]: (debug): MAC:
0000.0000.0000 Sending keepalive_ctrl_req of XID (80704) to (ipv4: 172.16.55.28 )
2021/09/28 11:34:22.928077 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive data packet misssed, total missed packet = 8
2021/09/28 11:34:22.928083 {mobilityd_R0-0}{1}: [mm-keepalive] [26516]: (note): Peer IP:
172.16.55.28 keepalive ctrl packet misssed, total missed packet = 3
```

피어 172.16.55.28의 내부 로그는 컨피그레이션 불일치를 확인하는 데 도움이 됩니다

```
2021/09/28 17:33:22.963 {mobilityd_R0-0}{1}: [mm-keepalive] [27081]: (ERR): Peer IP:
172.16.51.88 Failed to validate endpoint: Invalid argument
2021/09/28 17:33:22.963 {mobilityd_R0-0}{1}: [errmsg] [27081]: (ERR): %MM_NODE_LOG-3-
PING_DROPPED: Drop data ping from IP: 172.16.51.88. Failed to validate endpoint
```

일반 구성 불일치 포함: 잘못된 그룹 이름, 불일치 **Data Link Encryption** 잘못된 모빌리티 mac 주소입니다.

그룹 불일치 로그:

```
2021/09/28 17:33:22.963 {mobilityd_R0-0}{1}: [errmsg] [27081]: (ERR): %MM_INFRA_LOG-3-
MSG_PROC_FAILED_GROUP_NAME_HASH: Pkt group name hash: 82FE070E6E9A37A543CEBED96DB0388F Peer
group name hash: 3018E2A00F10176849AC824E0190AC86 Failed to validate endpoint. reason: Group
name hash mismatch.
```

MAC 주소 불일치 로그:

```
2021/09/28 19:09:33.455 {mobilityd_R0-0}{1}: [errmsg] [27081]: (ERR): %MM_INFRA_LOG-3-
MSG_PROC_FAILED_MAC_ADDR: Pkt MAC: 001e.e67e.75fa Peer MAC: 001e.e67e.75ff Failed to validate
endpoint. reason: MAC address mismatch.
```

DTLS 핸드셰이크 문제

이러한 종류의 문제는 WLC 간의 DTLS 터널 설정과 관련이 있습니다. 데이터 경로가 UP이지만 제어 경로가 남아 있을 수 있습니다 **DOWN**.

사용 **Always-On-Logs** 및 **Embedded packet captures** 문제 해결을 위한 추가 정보를 제공하려면

```
2021/09/28 19:30:23.534 {mobilityd_R0-0}{1}: [mm-msg] [27081]: (ERR): Peer IP: 172.16.51.88
Port: 16666 DTLS_MSG: DTLS message process failed. Error: Invalid argument
2021/09/28 19:30:23.534 {mobilityd_R0-0}{1}: [errmsg] [27081]: (warn): %MM_NODE_LOG-4-
DTLS_HANDSHAKE_FAIL: Mobility DTLS Ctrl handshake failed for 172.16.51.88 HB is down, need to
re-initiate DTLS handshake
2021/09/28 19:30:23.534 {mobilityd_R0-0}{1}: [ewlc-capwapmsg-sess] [27081]: (ERR): Source
IP:172.16.51.88[16666], DTLS message process failed. length:52
```

Use **show wireless management trustpoint** 및 **show crypto pki trustpoints commands** 인증서 정보를 확인합니다.

HA SSO 시나리오

고가용성 SSO 쌍에 컨트롤러가 있는 경우 알아야 할 중요한 사항이 있습니다. 모빌리티 MAC 주소는 기본적으로 구성되지 않으며, 장애 조치가 발생할 경우 모빌리티 터널이 중단될 수 있습니다.

show **wireless mobility summary(무선 모빌리티 요약** 표시)는 현재 사용 중인 모빌리티 MAC을 제공하지만 반드시 구성할 필요는 없습니다. 컨피그레이션에 show run으로 구성된 mobility MAC이 있는지 **확인 | i 모빌리티**

실행 중인 컨피그레이션에서 모빌리티 mac이 구성되지 않은 경우, 스탠바이 WLC로 장애 조치할 때 변경되며, 이로 인해 모빌리티 터널이 실패합니다.

간단한 해결 방법은 Configuration(컨피그레이션) **> Wireless(무선) > Mobility** web UI(**모빌리티** 웹 UI) 페이지로 이동하여 apply(적용)를 **누르십시오**. 그러면 현재 모빌리티 MAC이 컨피그레이션에 저장됩니다. 그러면 MAC는 장애 조치 시 동일하게 유지되며 모빌리티 터널이 보존됩니다.

이 문제는 주로 명령줄을 통해 모빌리티 컨피그레이션을 수행하고 모빌리티 MAC 주소를 구성하는 것을 잊은 경우에 발생합니다. 설정을 적용하면 웹 UI에서 모빌리티 MAC 주소를 자동으로 저장합니다.

관련 정보

- Catalyst 9800에서 WLAN Anchor Mobility 기능 구성
- 기술 지원 및 문서 – Cisco Systems

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.