Introduction

This document describes the different types of wireless roaming and fast-secure roaming methods available for IEEE 802.11 Wireless LANs (WLANs) supported on the Cisco Unified Wireless Network (CUWN).
The document does not provide all of the specifics about how each method works or how they are configured. The main purpose of this document is to describe the differences between the various techniques available, their advantages and limitations, and the frames-exchange on each method. Examples of WLAN Controller (WLC) debugs are provided, and wireless packet captures are used in order to analyze and explain the events that occur for each roaming method described.

**Prerequisites**

**Requirements**

Cisco recommends that you have knowledge of these topics:

- IEEE 802.11 WLAN Fundamentals
- IEEE 802.11 WLAN Security
- IEEE 802.1X/EAP Basics

**Components Used**

The information in this document is based on Cisco WLAN Controller Software Version 7.4, but most of the debug outputs and behaviors described might apply to any software version that supports the methods discussed. Specifics of all the methods explained here remain the same on later Cisco WLAN Controller codes (up to version 8.3 by the time this article was updated).

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, make sure that you understand the potential impact of any command.

**Background Information**

Before a description of the different fast-secure roaming methods available for WLANs is given, it is important to understand how the WLAN association process works, and how a regular roaming event occurs when there is no security configured on the Service Set Identifier (SSID).

When an 802.11 wireless client connects to an Access Point (AP), before it begins to pass traffic (wireless data frames), it first must pass the basic 802.11 Open System authentication process. Then, the association process must be completed. Think of the Open System authentication process as "connecting the cable" on the AP that the client selects. This is very important to understand, because it is always the wireless client that selects which AP is preferred, and bases the decision on multiple factors that vary between vendors. This is why the client begins this process by sending the Authentication frame to the selected AP, as shown later in this document. The AP cannot request that you establish a connection.

Once the Open System authentication process is completed successfully with a response from the AP ("cable connected"), the association process essentially finishes the 802.11 Layer 2 (L2) negotiation that establishes the link between the client and the AP. The AP assigns an association ID to the client if the connection is successful, and prepares it in order to pass traffic or perform a higher-level security method if configured on the SSID. The Open System authentication process consists of two management frames as well as the association process. Authentication and Association frames are wireless management frames, not data frames, which are basically the ones used for the connection process with the AP.
Here is a capture of the wireless frames over-the-air for this process:

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>SSID</th>
<th>Protocol</th>
<th>Channel frequency</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0:00:00:00</td>
<td>Aironet_b7:ab:5c</td>
<td>Cisco_f0:68:d0</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 authentication, SNI=244, FMS=0, Flags...</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0:00:00:01</td>
<td>Cisco_f0:68:d0</td>
<td>Aironet_b7:ab:5c</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 Association Request, SNI=244, FMS=0, Flags...</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0:00:00:02</td>
<td>Aironet_b7:ab:5c</td>
<td>Cisco_f0:68:d0</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 Association Response, SNI=271, FMS=0, Flags...</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0:00:00:03</td>
<td>Cisco_f0:68:d0</td>
<td>Aironet_b7:ab:5c</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 Association Request, SNI=244, FMS=0, Flags...</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0:00:00:04</td>
<td>Cisco_f0:68:d0</td>
<td>Aironet_b7:ab:5c</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 Association Response, SNI=271, FMS=0, Flags...</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0:00:00:05</td>
<td>Cisco_f0:68:d0</td>
<td>Aironet_b7:ab:5c</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 Association Request, SNI=244, FMS=0, Flags...</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0:00:00:06</td>
<td>Cisco_f0:68:d0</td>
<td>Aironet_b7:ab:5c</td>
<td>h4:78:ac:f0:68:d0</td>
<td>802.11</td>
<td>2462 Association Response, SNI=271, FMS=0, Flags...</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If you want to learn about 802.11 wireless sniffing, and about the filters/colors used on Wireshark for the captures that appear in this document, visit the Cisco Support Community post called [802.11 Sniffer Capture Analysis](#).

The wireless client begins with the Authentication frame, and the AP replies with another Authentication frame. The client then sends the Association Request frame, and the AP finishes in a reply with the Association Response frame. As shown from the DHCP packets, once the 802.11 Open System authentication and association processes are passed, the client begins to pass data frames. In this case, there is no security method configured on the SSID, so the client immediately begins to send data frames (in this case DHCP) that are not encrypted.

As shown later in this document, if security is enabled on the SSID, there are higher-level authentication and encryption handshake frames for the specific security method, just after the Association Response and before sending any client traffic data frames, such as DHCP, Address Resolution Protocol (ARP), and applications packets, which are encrypted. Data frames can only be sent until the client is fully authenticated, and the encryption keys are negotiated, based on the security method configured.

Based on the previous capture, here are the messages that you see in the outputs of the WLC `debug client` command when the wireless client begins a new association to the WLAN:

*apfMsConnTask_0: Jun 21 18:55:14.221: 00:40:96:b7:ab:5c
Association received from mobile on BSSID 84:78:ac:f0:68:d0

`!!-- This is the Association Request from the wireless client to the selected AP.`

*apfMsConnTask_0: Jun 21 18:55:14.222: 00:40:96:b7:ab:5c
Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d0
(status 0) ApVap1Id 1 Slot 0

`!!-- This is the Association Response from the AP to the client.`

**Note:** The WLC debug used for the outputs shown in this document is the `debug client` command, and the examples only show some relevant messages, not the entire output. For more details about this debug command, reference the document called [Understanding Debug Client on Wireless LAN Controllers (WLCs)](#).

These messages show the Association Request and Response frames; the initial Authentication frames are not logged at the WLC because this handshake happens quickly at the AP-level on the CUWN.
What information appears when the client roams? The client always exchanges four management frames upon establishment of a connection to an AP, which is due to either client establishment of association, or a roaming event. The client has only one connection established to only one AP at a time. The only difference in the frame exchange between a new connection to the WLAN infrastructure and a roaming event is that the Association frames of a roaming event are called Reassociation frames, which indicate that the client is actually roaming from another AP with no attempts to establish a new association to the WLAN. These frames can contain different elements that are used in order to negotiate the roaming event; this depends on the setup, but those details are out of the scope of this document.

Here is an example of the frame exchange:

<table>
<thead>
<tr>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>SSID</th>
<th>Protocol</th>
<th>Channel</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:40:96:b7:ab:5c</td>
<td></td>
<td>84:78:ac:f0:2a:90</td>
<td>802.11</td>
<td></td>
<td>Authentication, SM=2011, FH=0, Flags=</td>
</tr>
<tr>
<td>2</td>
<td>00:16:08:10:2a:90</td>
<td></td>
<td>84:78:ac:f0:2a:90</td>
<td>802.11</td>
<td></td>
<td>Authentication, SM=2010, FH=0, Flags=</td>
</tr>
<tr>
<td>3</td>
<td>00:22:16:10:2a:90</td>
<td></td>
<td>84:78:ac:f0:2a:90</td>
<td>802.11</td>
<td></td>
<td>Authentication, SM=2011, FH=0, Flags=</td>
</tr>
<tr>
<td>4</td>
<td>00:40:96:b7:ab:5c</td>
<td></td>
<td>84:78:ac:f0:2a:90</td>
<td>802.11</td>
<td></td>
<td>Authentication, SM=2011, FH=0, Flags=</td>
</tr>
<tr>
<td>5</td>
<td>4.529164:10:2a:90</td>
<td></td>
<td>84:78:ac:f0:2a:90</td>
<td>802.11</td>
<td></td>
<td>Authentication, SM=2011, FH=0, Flags=</td>
</tr>
<tr>
<td>5</td>
<td>4.529164:10:2a:90</td>
<td></td>
<td>84:78:ac:f0:2a:90</td>
<td>802.11</td>
<td></td>
<td>Authentication, SM=2011, FH=0, Flags=</td>
</tr>
</tbody>
</table>

These messages appear in the debug output:

*apfMsConnTask_2: Jun 21 19:02:19.709: 00:40:96:b7:ab:5c
  Reassociation received from mobile on BSSID 84:78:ac:f0:2a:90
  !--- This is the Reassociation Request from the wireless client
     to the selected AP.

*apfMsConnTask_2: Jun 21 19:02:19.710: 00:40:96:b7:ab:5c
  Sending Assoc Response to station on BSSID 84:78:ac:f0:2a:90
  (status 0) ApVapId 1 Slot 0
  !--- This is the Reassociation Response from the AP to the client.

As shown, the client successfully performs a roaming event after the Reassociation Request to the new AP is sent, and receives the Reassociation Response from the AP. Since the client already has an IP address, the first data frames are for ARP packets.

If you expect a roaming event, but the client sends an Association Request instead of a Reassociation Request (which you can confirm from some captures and debugs similar to those explained earlier in this document), then the client is not really roaming. The client begins a new association to the WLAN as if a disconnection took place, and tries to reconnect from scratch. This can happen for multiple reasons, such as when a client moves away from the coverage areas and then finds an AP with enough signal quality to start an association, but it normally indicates a client issue where the client does not initiate a roaming event due to drivers, firmware, or software issues.

**Note:** You can check with the wireless client vendor in order to determine the cause of the issue.

**Roaming with Higher-Level Security**

When the SSID is configured with L2 higher-level security on top of basic 802.11 Open System authentication, then more frames are required for the initial association and when roaming. The two most-common security methods standardized and implemented for 802.11 WLANs are described in this document:
- **WPA/WPA2-PSK (Pre-Shared Key)** - authentication of clients with a Pre-shared-Key.
- **WPA/WPA2-EAP (Extensible Authentication Protocol)** - authentication of clients with an 802.1X/EAP method in order to validate more secure credentials through the use of an Authentication Server, such as certificates, username and password, and tokens.

It is important to know that, even though these two methods (PSK and EAP) authenticate/validate the clients in different ways, both use basically the same WPA/WPA2 rules for the key management process. Whether the security is WPA/WPA2-PSK or WPA/WPA2-EAP, the process known as the WPA/WPA2 4-Way handshake begins the key negotiation between the WLC/AP and the client with a Master Session Key (MSK) as the original key material once the client is validated with the specific authentication method used.

Here is a summary of the process:

1. An MSK is derived from the EAP authentication phase when 802.1X/EAP security is used, or from the PSK when WPA/WPA2-PSK is used as the security method.
2. From this MSK, the client and WLC/AP derive the Pairwise Master Key (PMK), and the WLC/AP generates a Group Master Key (GMK).
3. Once these two Master Keys are ready, the client and the WLC/AP initiate the WPA/WPA2 4-Way handshake (which is illustrated later in this document with some screen captures and debugging) with the Master Keys as the seeds for negotiation of the actual encryption keys.
4. Those final encryption keys are known as the Pairwise Transient Key (PTK) and the Group Transient Key (GTK). The PTK is derived from the PMK and used in order to encrypt unicast frames with the client. The Group Transient Key (GTK) is derived from the GMK, and is used in order to encrypt multicast/broadcast on this specific SSID/AP.

**WPA/WPA2-PSK**

When WPA-PSK or WPA2-PSK is performed via Temporal Key Integrity Protocol (TKIP) or Advanced Encryption Standard (AES) for the encryption, the client must go through the process known as the WPA 4-Way handshake for both the initial association and also when roaming. As previously explained, this is basically the key management process used in order for WPA/WPA2 to derive the encryption keys. However, when PSK is performed, it is also used in order to verify that the client has a valid Pre-Shared Key to join the WLAN. This capture shows the initial association process when WPA or WPA2 with PSK is performed:

As shown, after the 802.11 Open System authentication and association process, there are four EAPOL frames from the WPA 4-Way handshake, which are initiated by the AP with message-1, and finished by the client with message-4. After a successful handshake, the client begins to pass data frames (such as DHCP), which in this case are encrypted with the keys derived from the 4-Way handshake (this is why you cannot see the actual content and type of traffic from the wireless captures).

**Note:** EAPOL frames are used in order to transport all of the key management frames and
802.1X/EAP Authentication frames over-the-air between the AP and the client; they are transmitted as wireless data frames.

These messages appear in the debug outputs:

*apfMsConnTask_0: Jun 21 19:30:05.172: 00:40:96:b7:ab:5c
Association received from mobile on BSSID 84:78:ac:f0:68:d1

*apfMsConnTask_0: Jun 21 19:30:05.173: 00:40:96:b7:ab:5c
Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d1
(state 0) ApVapId 2 Slot 0
--- The Association handshake is finished.

*dot1xMsgTask: Jun 21 19:30:05.178: 00:40:96:b7:ab:5c
Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c
state INITPMK (message 1), replay counter
00.00.00.00.00.00.00.00
--- Message-1 of the WPA/WPA2 4-Way handshake is sent from the WLC/AP to the client.

*Dot1x_NW_MsgTask_4: Jun 21 19:30:05.289: 00:40:96:b7:ab:5c
Received EAPOL-Key from mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 19:30:05.289: 00:40:96:b7:ab:5c
Received EAPOL-key in PTK_START state (message 2)
from mobile 00:40:96:b7:ab:5c
--- Message-2 of the WPA/WPA2 4-Way handshake is successfully received from the client.

*Dot1x_NW_MsgTask_4: Jun 21 19:30:05.290: 00:40:96:b7:ab:5c
Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c
state PTKINITNEGOTIATING (message 3), replay counter
00.00.00.00.00.00.00.01
--- Message-3 of the WPA/WPA2 4-Way handshake is sent from the WLC/AP to the client.

*Dot1x_NW_MsgTask_4: Jun 21 19:30:05.309: 00:40:96:b7:ab:5c
Received EAPOL-Key from mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 19:30:05.310: 00:40:96:b7:ab:5c
Received EAPOL-key in PTKINITNEGOTIATING state (message 4)
from mobile 00:40:96:b7:ab:5c
--- Message-4 (final message) of the WPA/WPA2 4-Way handshake is successfully received from the client, which confirms the installation of the derived keys. They can now be used in order to encrypt data frames with current AP.

When roaming, the client basically follows the same frame exchange, where the WPA 4-Way handshake is required in order to derive new encryption keys with the new AP. This is due to security reasons established by the standard, and the fact that the new AP does not know the original keys. The only difference is that there are Reassociation frames instead of Association frames, as shown in this capture:

You see the same messages in the debug outputs, but the first packet from the client is a Reassociation instead of an Association, as shown and explained previously.
WPA/WPA2-EAP

When an 802.1X/EAP method is used in order to authenticate the clients on a secure SSID, there are even more frames required before the client begins to pass traffic. These extra frames are used in order to authenticate the client credentials, and dependent upon the EAP method, there might be between four and twenty frames. These come after the Association/Reassociation, but before the WPA/WPA2 4-Way handshake, because the authentication phase derives the MSK used as the seed for the final encryption key generation in the key management process (4-Way handshake).

This wireless capture shows an example of the frames exchanged over the air between the AP and the wireless client upon initial association when WPA with PEAPv0/EAP-MSCHAPv2 is performed:

Sometimes this exchange shows more or less frames, which depends on multiple factors, such as the EAP method, retransmissions due to problems, client behavior (such as the two Identity Requests in this example, because the client sends an EAPOL START after the AP sends the first Identity Request), or if the client already exchanged the certificate with the server. Whenever the SSID is configured for an 802.1X/EAP method, there are more frames (for the authentication), and hence, it requires more time before the client begins to send data frames.

Here is a summary of the debug messages:

```
*apfMsConnTask_0: Jun 21 23:41:19.092: 00:40:96:b7:ab:5c
  Association received from mobile on BSSID 84:78:acc:0:68:d8
*apfMsConnTask_0: Jun 21 23:41:19.094: 00:40:96:b7:ab:5c
  Sending Assoc Response to station on BSSID 84:78:acc:0:68:d8
  (status 0) ApVapId 9 Slot 0

--- The Association handshake is finished.
```

```
*dot1xMsgTask: Jun 21 23:41:19.098: 00:40:96:b7:ab:5c
```

 sometimes this exchange shows more or less frames, which depends on multiple factors, such as the EAP method, retransmissions due to problems, client behavior (such as the two Identity Requests in this example, because the client sends an EAPOL START after the AP sends the first Identity Request), or if the client already exchanged the certificate with the server. Whenever the SSID is configured for an 802.1X/EAP method, there are more frames (for the authentication), and hence, it requires more time before the client begins to send data frames.

Here is a summary of the debug messages:

```
*afMscConnTask_0: Jun 21 23:41:19.092: 00:40:96:b7:ab:5c
  Association received from mobile on BSSID 84:78:acc:0:68:d8
*afMscConnTask_0: Jun 21 23:41:19.094: 00:40:96:b7:ab:5c
  Sending Assoc Response to station on BSSID 84:78:acc:0:68:d8
  (status 0) ApVapId 9 Slot 0

--- The Association handshake is finished.
```

```
*dot1xMsgTask: Jun 21 23:41:19.098: 00:40:96:b7:ab:5c
```
Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c
(EAP Id 1)

!--- The EAP Identity Request is sent to the client once it is associated in order to begin the higher-level authentication process. This informs the client that an identity to start this type of 802.1X/EAP authentication must be provided.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.226: 00:40:96:b7:ab:5c
  Received EAPOLSTART from mobile 00:40:96:b7:ab:5c

!--- The wireless client decides to start the EAP authentication process, and informs the AP with an EAPOL START data frame.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.227: 00:40:96:b7:ab:5c
  Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c
  (EAP Id 2)

!--- WLC/AP sends another EAP Identity Request to the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.235: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.235: 00:40:96:b7:ab:5c
  Received Identity Response (count=2) from mobile 00:40:96:b7:ab:5c

!--- The client responds with an EAP Identity Response on an EAPOL frame.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.301: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.301: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 3)

!--- Once the WLC/AP sends the client response to the Authentication Server on a RADIUS Access-Request packet, the server responds with a RADIUS Access-Challenge in order to officially start the EAP negotiation, handshake, and authentication with the client (sometimes with mutual authentication, dependent upon the EAP method). This response received by the WLC/AP is sent to the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.347: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.347: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 4, EAP Type 25)

!--- The client responds with an EAP Response on an EAPOL frame, which is sent to the Authentication Server on a RADIUS Access-Request packet. The server responds with another RADIUS Access-Challenge. This process continues, dependent upon the EAP method (the exchange of certificates when used, the building of TLS tunnels, validation of client credentials, client validation of server identity when applicable). Hence, the next few messages are basically the same on the WLC/AP side, as this acts as a "proxy" between the client and the Authentication Server exchanges.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.375: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.375: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 4)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.375: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.375: 00:40:96:b7:ab:5c
  Received EAP Response from mobile 00:40:96:b7:ab:5c
  (EAP Id 4, EAP Type 25)
Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 5)

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 5, EAP Type 25)

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 6)

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 6, EAP Type 25)

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 7)

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 7, EAP Type 25)

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 8)

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 8, EAP Type 25)
The authentication finishes and is successful for this client, so the RADIUS Server sends a RADIUS Access-Accept to the WLC/AP. This RADIUS Access-Accept comes with the special attributes that are assigned to this client (if any are configured on the Authentication Server for this client). This Access-Accept also comes with the MSK derived with the client in the EAP authentication process, so the WLC/AP installs it in order to initiate the WPA/WPA2 4-Way handshake with the wireless client.

The accept/pass of the authentication is sent to the client as
an EAP-Success message.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.473: 00:40:96:b7:ab:5c
Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c
state INITPMK (message 1), replay counter
00.00.00.00.00.00.00.00

!--- Message-1 of the WPA/WPA2 4-Way handshake is sent from the
WLC/AP to the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.481: 00:40:96:b7:ab:5c
Received EAPOL-Key from mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.481: 00:40:96:b7:ab:5c
Received EAPOL-key in PTK_START state (message 2)
from mobile 00:40:96:b7:ab:5c

!--- Message-2 of the WPA/WPA-2 4-Way handshake is successfully
received from the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.481: 00:40:96:b7:ab:5c
Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c
state PTKININITNEGOTIATING (message 3), replay counter
00.00.00.00.00.00.00.01

!--- Message-3 of the WPA/WPA2 4-Way handshake is sent from the
WLC/AP to the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.487: 00:40:96:b7:ab:5c
Received EAPOL-Key from mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.487: 00:40:96:b7:ab:5c
Received EAPOL-key in PTKINITNEGOTIATING state (message 4)
from mobile 00:40:96:b7:ab:5c

!--- Message-4 (final message) of the WPA/WPA2 4-Way handshake
is successfully received from the client, which confirms the
installation of the derived keys. They can now be used in
order to encrypt data frames with the current AP.

When the wireless client performs a regular roaming here (the normal behavior, without
implementation of a fast-secure roaming method), the client must go through the exact same
process and perform a full authentication against the Authentication Server, as shown in the
captures. The only difference is that the client uses a Reassociation Request in order to inform the
new AP that it is actually roaming from another AP, but the client still has to go through full
validation and new key generation:

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>RSSI</th>
<th>Protocol</th>
<th>Channel/frequency</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.000221</td>
<td>Cisco F0:2a:198</td>
<td>Aironet b7:ab:5c</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Authentication, Sn-96, PN-O, Flags:---------c</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.000857</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Reassociation Request, Sn-96, PN-O Flags:---------c</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.024409</td>
<td>Cisco F0:2a:198</td>
<td>Aironet b7:ab:5c</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Request, Identity</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.025712</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Start</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.025530</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Response, Identity</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.025627</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Request, Protected EAP (EAP-PEAP)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.025811</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Client Hello</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.025776</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.025828</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.025836</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.025956</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.025964</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.026015</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.026029</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.026038</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.026047</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.026056</td>
<td>Aironet b7:ab:5c</td>
<td>Cisco F0:2a:198</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 Change Cipher Spec, Encrypted Handshake Message</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.125583</td>
<td>Cisco F0:2a:198</td>
<td>Aironet b7:ab:5c</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 QoS Data, Sn-76, PN-O, Flags:---------c</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.125592</td>
<td>Aironet b7:ab:5c</td>
<td>Broadcast</td>
<td>8478:ac:0f:12:19:98</td>
<td>802.11</td>
<td>2437 QoS Data, Sn-76, PN-O, Flags:---------c</td>
<td></td>
</tr>
</tbody>
</table>

As shown, even when there are less frames than in the initial authentication (which is caused by
multiple factors, as mentioned before), when the client roams to a new AP, the EAP authentication
and the WPA key management processes must still be completed in order to continue to pass
data frames (even if traffic was actively sent before roaming). Therefore, if the client has an active
application that is sensitive to delays (such as voice-traffic applications, or applications that are
sensitive to timeouts), then the user can perceive problems when roaming, such as audio gaps or
application disconnects. This depends on how long the process takes in order for the client to continue to send/receive data frames. This delay might be longer, dependent upon: the RF environment, the amount of clients, the round-trip time between the WLC and LAPs and with the Authentication Server, and other reasons.

Here is a summary of the debug messages for this roaming event (basically the same as the previous ones, so these messages are not described further):

*apfMsConnTask_0: Jun 21 23:41:19.092: 00:40:96:b7:ab:5c
  Association received from mobile on BSSID 84:78:ac:f0:68:d8
*apfMsConnTask_0: Jun 21 23:41:19.094: 00:40:96:b7:ab:5c
  Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d8
      (status 0) ApVapId 9 Slot 0
  !--- The Association handshake is finished.

*dot1xMsgTask: Jun 21 23:41:19.098: 00:40:96:b7:ab:5c
  Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c
      (EAP Id 1)
  !--- The EAP Identity Request is sent to the client once it is associated in order to begin the higher-level authentication process. This informs the client that an identity to start this type of 802.1X/EAP authentication must be provided.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.226: 00:40:96:b7:ab:5c
  Received EAPOL START from mobile 00:40:96:b7:ab:5c
  !--- The wireless client decides to start the EAP authentication process, and informs the AP with an EAPOL START data frame.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.227: 00:40:96:b7:ab:5c
  Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c
      (EAP Id 2)
  !--- WLC/AP sends another EAP Identity Request to the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.235: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.235: 00:40:96:b7:ab:5c
  Received Identity Response (count=2) from mobile 00:40:96:b7:ab:5c
  !--- The client responds with an EAP Identity Response on an EAPOL frame.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.301: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.301: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
      (EAP Id 3)
  !--- Once the WLC/AP sends the client response to the Authentication Server on a RADIUS Access-Request packet, the server responds with a RADIUS Access-Challenge in order to officially start the EAP negotiation, handshake, and authentication with the client (sometimes with mutual authentication, dependent upon the EAP method). This response received by the WLC/AP is sent to the client.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.344: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c
*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.344: 00:40:96:b7:ab:5c
  Received EAP Response from mobile 00:40:96:b7:ab:5c
      (EAP Id 3, EAP Type 25)
  !--- The client responds with an EAP Response on an EAPOL frame, which is sent to the Authentication Server on a RADIUS Access-Request packet. The server responds with another RADIUS Access-Challenge. This process continues, dependent upon the EAP method (the exchange
of certificates when used, the building of TLS tunnels, validation of client credentials, client validation of server identity when applicable). Hence, the next few messages are basically the same on the WLC/AP side, as this acts as a “proxy” between the client and the Authentication Server exchanges.

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.347: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.347: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 4)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.375: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.375: 00:40:96:b7:ab:5c
  Received EAP Response from mobile 00:40:96:b7:ab:5c
  (EAP Id 4, EAP Type 25)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.377: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.377: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 5)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.403: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.403: 00:40:96:b7:ab:5c
  Received EAP Response from mobile 00:40:96:b7:ab:5c
  (EAP Id 5, EAP Type 25)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.421: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.421: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 7)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.425: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.425: 00:40:96:b7:ab:5c
  Received EAP Response from mobile 00:40:96:b7:ab:5c
  (EAP Id 7, EAP Type 25)

*Dot1x_NW_MsgTask_4: Jun 21 23:41:19.427: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c
Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 8)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 8, EAP Type 25)

Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 9)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 9, EAP Type 25)

Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 10)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 10, EAP Type 25)

Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 11)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 11, EAP Type 25)

Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 13)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c
The authentication finishes and is successful for this client, so the RADIUS Server sends a RADIUS Access-Accept to the WLC/AP. This RADIUS Access-Accept comes with the special attributes that are assigned to this client (if any are configured on the Authentication Server for this client). This Access-Accept also comes with the MSK derived with the client in the EAP authentication process, so the WLC/AP installs it in order to initiate the WPA/WPA2 4-Way handshake with the wireless client.

The accept/pass of the authentication is sent to the client as an EAP-Success message.

Message-1 of the WPA/WPA2 4-Way handshake is sent from the WLC/AP to the client.

Message-2 of the WPA/WPA2 4-Way handshake is successfully received from the client.

Message-3 of the WPA/WPA2 4-Way handshake is sent from the WLC/AP to the client.

Message-4 (final message) of the WPA/WPA2 4-Way handshake is successfully received from the client, which confirms the installation of the derived keys. They can now be used in order to encrypt data frames with the current AP.

This is the way that 802.1X/EAP and the WPA/WPA2 security framework work. In order to prevent the application/service impact on delays from a regular roaming event, multiple fast-secure roaming methods are developed and implemented by the WiFi industry in order to accelerate the roaming process when security is used on the WLAN/SSID. The clients face some latency when they continue to pass traffic while roaming around between APs via deployment of high-level security on the WLAN. This is due to the EAP authentication and key-management frame exchanges required by the security setup, as previously explained.

It is important to understand that fast-secure roaming is just the term used by the industry in reference to the implementation of a method/scheme that accelerates the roaming process when
security is configured on the WLAN. The different fast-secure roaming methods/schemes that are available for WLANs, and are supported by the CUWN, are explained in the next section.

**Fast-Secure Roaming with CCKM**

Cisco Centralized Key Management (CCKM) is the first fast-secure roaming method developed and implemented on enterprise WLANs, created by Cisco as the solution used in order to mitigate the delays explained thus far, when 802.1X/EAP security is used on the WLAN. As this is a Cisco proprietary protocol, it is only supported by Cisco WLAN infrastructure devices and wireless clients (from multiple vendors) that are Cisco Compatible Extension (CCX)-compatible for CCKM.

CCKM can be implemented with all of the different encryption methods available for WLANs, to include: WEP, TKIP, and AES. It is also supported with most of the 802.1X/EAP authentication methods used for WLANs, dependent upon the CCX version supported by the devices.

**Note:** For an overview on the feature content supported by the different versions of the CCX specification (including EAP methods supported), reference the [CCX Versions and Features document](#), and verify the exact CCX version that is supported by your wireless clients (if they are CCX-compatible), so that you can confirm if the security method you desire to use with CCKM can be implemented.

This wireless capture provides an example of the frames exchanged upon initial association when you perform CCKM with TKIP as the encryption, and PEAPv0/EAP-MSCHAPv2 as the 802.1X/EAP method. This is basically the same exchange as if WPA/TKIP with PEAPv0/EAP-MSCHAPv2 is performed, but this time CCKM between the client and the infrastructure is negotiated so that they use different key hierarchy and cache methods in order to perform Fast Secure Roaming when the client must roam:
Here is a summary of the debug messages (with some EAP exchanges removed in order to reduce the output):

*apfMsConnTask_0: Jun 25 15:41:41.507: 00:40:96:b7:ab:5c
  Association received from mobile on BSSID 84:78:ac:f0:68:d3
--- This is the Association Request from the client.

*apfMsConnTask_0: Jun 25 15:41:41.507: 00:40:96:b7:ab:5c
  Processing WPA IE type 221, length 22 for mobile
  00:40:96:b7:ab:5c
*apfMsConnTask_0: Jun 25 15:41:41.507: 00:40:96:b7:ab:5c
  CCKM: Mobile is using CCKM
--- The WLC/AP finds an Information Element that claims CCKM
  support on the Association request that is sent from the client.

*apfMsConnTask_0: Jun 25 15:41:41.507: 00:40:96:b7:ab:5c
  Setting active key cache index 8 ---> 8
--- This is the key cache index for this client, which is set temporally.

*apfMsConnTask_0: Jun 25 15:41:41.508: 00:40:96:b7:ab:5c
  Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d3
  (status 0) ApVapId 4 Slot 0
--- The Association Response is sent to the client.

*dot1xMsgTask: Jun 25 15:41:41.513: 00:40:96:b7:ab:5c
  Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c
  (EAP Id 1)
--- An EAP Identity Request is sent to the client once it is
  associated in order to begin the higher-level authentication
  process. This informs the client that an identity to start
  this type of 802.1X/EAP authentication must be provided.
  Further EAP messages are not described, as they are basically
  the same as the ones previously-explained.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.536: 00:40:96:b7:ab:5c
  Received EAPOL START from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.536: 00:40:96:b7:ab:5c
  Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c
  (EAP Id 2)

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.546: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.546: 00:40:96:b7:ab:5c
  Received EAP Response packet with mismatching id
  (currentid=2, eapid=1) from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.550: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.550: 00:40:96:b7:ab:5c
  Received Identity Response (count=2) from mobile
  00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.555: 00:40:96:b7:ab:5c
  Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.555: 00:40:96:b7:ab:5c
  Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c
  (EAP Id 3)
*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.594: 00:40:96:b7:ab:5c
  Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.594: 00:40:96:b7:ab:5c
  Received EAP Response from mobile 00:40:96:b7:ab:5c
  (EAP Id 3, EAP Type 25)

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.840: 00:40:96:b7:ab:5c
  Processing Access-Accept for mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.841: 00:40:96:b7:ab:5c
  Creating a PKC PMKID Cache entry for station 00:40:96:b7:ab:5c
  (RSN 0)

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.841: 00:40:96:b7:ab:5c
  Setting active key cache index 8 ---> 8

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.841: 00:40:96:b7:ab:5c
  Setting active key cache index 8 ---> 0

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.841: 00:40:96:b7:ab:5c
  CCKM: Create a global PMK cache entry

--- WLC creates a global PMK cache entry for this client, which is for CCKM in this case.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.841: 00:40:96:b7:ab:5c
  Sending EAP-Success to mobile 00:40:96:b7:ab:5c
  (EAP Id 13)

--- The client is informed of the successful EAP authentication.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.841: 00:40:96:b7:ab:5c
  Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c
  INITPMK(message 1), replay counter 00.00.00.00.00.00.00.00

--- Message-1 of the initial 4-Way handshake is sent from the WLC/AP to the client.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.858: 00:40:96:b7:ab:5c
  Received EAPOL-Key from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.858: 00:40:96:b7:ab:5c
  Received EAPOL-key in PTK_START state (message 2) from mobile
  00:40:96:b7:ab:5c

--- Message-2 of the initial 4-Way handshake is received successfully from the client.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.858: 00:40:96:b7:ab:5c
  CCKM: Sending cache add

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.858: 00:40:96:b7:ab:5c
  CCKM: Sending CCKM PMK (Version_1) information to mobility group

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.858: 00:40:96:b7:ab:5c
  CCKM: Sending CCKM PMK (Version_2) information to mobility group

--- The CCKM PMK cache entry for this client is shared with the WLCs on the mobility group.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.858: 00:40:96:b7:ab:5c
  Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c
  state PTKINITNEGOTIATING (message 3), replay counter
  00.00.00.00.00.00.00.01

--- Message-3 of the initial 4-Way handshake is sent from the WLC/AP to the client.

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.866: 00:40:96:b7:ab:5c
  Received EAPOL-Key from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 25 15:41:41.866: 00:40:96:b7:ab:5c
  Received EAPOL-key in PTKINITNEGOTIATING state (message 4) from mobile
  00:40:96:b7:ab:5c

--- Message-4 (final message) of this initial 4-Way handshake is received successfully from the client, which confirms the
installation of the derived keys. They can now be used in order to encrypt data frames with the current AP.

With CCKM, the initial association to the WLAN is similar to the regular WPA/WPA2, where an MSK (also known here as the Network Session Key (NSK)) is mutually derived with the client and the RADIUS Server. This master key is sent from the server to the WLC after a successful authentication, and is cached as the basis for derivation of all subsequent keys for the lifetime of the client association with this WLAN. From here, the WLC and the client derive the seed information that is used for fast-secure roaming based on CCKM, going through a 4-Way handshake similar to that of WPA/WPA2, in order to derive the unicast (PTK) and multicast/broadcast (GTK) encryption keys with the first AP.

The big difference is noticed when roaming. In this case, the CCKM client sends a single Reassociation Request frame to the AP/WLC (including an MIC and a sequentially incrementing Random Number), and provides enough information (including the new AP MAC address -BSSID-) in order to derive the new PTK. With this Reassociation Request, the WLC and new AP also have enough information in order to derive the new PTK, so they simply respond with a Reassociation Response. The client can now continue to pass traffic, as shown in this capture:

Here is a summary of the WLC debugs for this roaming event:

*apfMsConnTask_2: Jun 25 15:43:33.749: 00:40:96:b7:ab:5c CCKM: Received REASSOC REQ IE
*apfMsConnTask_2: Jun 25 15:43:33.749: 00:40:96:b7:ab:5c Reassociation received from mobile on BSSID 84:78:ac:f0:2a:93
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c Processing WPA IE type 221, length 22 for mobile 00:40:96:b7:ab:5c
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c CCKM: Mobile is using CCKM
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c Setting active key cache index 0 ---> 8
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c CCKM: Processing RBASSOC REQ IE
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c CCKM: using HMAC MD5 to compute MIC
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c WLC computes the MIC used for this CCKM fast-roaming exchange.
*apfMsConnTask_2: Jun 25 15:43:33.750: 00:40:96:b7:ab:5c CCKM: Received a valid REASSOC REQ IE
*apfMsConnTask_2: Jun 25 15:43:33.751: 00:40:96:b7:ab:5c CCKM: Initializing PMK cache entry with a new PTK
*apfMsConnTask_2: Jun 25 15:43:33.751: 00:40:96:b7:ab:5c CCKM: ReASSOC REQ IE
*apfMsConnTask_2: Jun 25 15:43:33.751: 00:40:96:b7:ab:5c CCKM: Received a valid REASSOC REQ IE

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>BSSID</th>
<th>Protocol</th>
<th>Channel</th>
<th>frequency</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2237</td>
<td>Authentication SN-2714, FW-0, Flags-.........</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2437</td>
<td>Reassociation Request, SN-2714, FW-0, Flags</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2237</td>
<td>Initialization SN-2714, FW-0, Flags-.........</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2437</td>
<td>Reassociation Response, SN-2714, FW-0, Flags</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2437</td>
<td>QoS Data, SN-2714, FW-0, Flags-.........</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2437</td>
<td>QoS Data, SN-2714, FW-0, Flags-.........</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.0000000</td>
<td>Airfront_B7:ab:5c</td>
<td>Cisco_f0:2a:93</td>
<td>84:78:ac:f0:2a:93</td>
<td>802.11</td>
<td>2437</td>
<td>QoS Data, SN-2714, FW-0, Flags-.........</td>
<td></td>
</tr>
</tbody>
</table>

--- The Reassociation Request is received from the client, which provides the CCKM information needed in order to derive the new keys with a fast-secure roam.

--- WLC computes the MIC used for this CCKM fast-roaming exchange.

--- The new PTK is derived.
As shown, fast-secure roaming is performed while avoiding the EAP authentication frames and even more 4-Way handshakes, because the new encryption keys are still derived, but based on the CCKM negotiation scheme. This is completed with the roaming Reassociation frames and the information previously-cached by the client and the WLC.

### FlexConnect with CCKM

- Central Authentication is supported. This includes Local and Central data switching. The APs must be part of the same FlexConnect Group.
- Flex Local Authentication is supported. In connected mode, the cache will be distributed from the AP to the controller and then to the rest of the APs in the FlexConnect Group.
- Standalone mode is supported. If the cache is already present on the AP (due to previous distribution), fast roam shall work. New authentication in standalone mode does not support fast-secure roaming.

### Pros with CCKM

- CCKM is the fastest fast-secure roaming method mostly deployed on enterprise WLANs. Clients do not need to go over a key management handshake in order to derive new keys when a move between APs takes place, and are never again required to perform a full 802.1X/EAP authentication with new APs during the client lifetime on this WLAN.
- CCKM supports all of the encryption methods available within the 802.11 standard (WEP, TKIP, and AES), in addition to some legacy Cisco proprietary methods still used on legacy clients.
Cons with CCKM

- CCKM is a Cisco proprietary method, which limits the implementation and support to Cisco WLAN infrastructure and CCX wireless clients.
- CCX Version 5 is not widely adopted, so CCKM with WPA2/AES is not supported by many CCX wireless clients (mainly because most of them already support CCKM with WPA/TKIP, which is still very secure).

Fast-Secure Roaming with PMKID Caching / Sticky Key Caching

Pairwise Master Key ID (PMKID) caching, or **Sticky Key Caching (SKC)**, is the first fast-secure roaming method suggested by the IEEE 802.11 standard within the 802.11i security amendment, where the main purpose is to standardize a high level of security for WLANs. This fast-secure roaming technique was added as an optional method for WPA2 devices in order to improve roaming when this security was implemented.

This is possible because, every time a client is fully EAP-authenticated, the client and Authentication Server derive an MSK, which is used in order to derive the PMK. This is used as the seed for the WPA2 4-Way handshake in order to derive the final unicast encryption key (PTK) that is used for the session (until the client roams to another AP or the session expires); hence, this method prevents the EAP authentication phase when roaming because it reutilizes the original PMK cached by the client and the AP. The client only has to go through the WPA2 4-Way handshake in order to derive new encryption keys.

This method is not widely deployed as the recommended 802.11 standard fast-secure roaming method mainly due to these reasons:

- This method is optional and is not supported by all WPA2 devices, because the purpose of the 802.11i amendment does not concern fast-secure roaming, and the IEEE was already working on another amendment to standardize fast-secure roaming for WLANs (802.11r, which is covered later in this document).
- This method has a big limitation on its implementation: Wireless clients can only perform fast-secure roaming when roaming back to an AP where they had previously authenticated/connected.

With this method, the initial association to any AP is just like a regular first-time authentication to the WLAN, where the entire 802.1X/EAP authentication against the Authentication Server and the 4-Way handshake for key generation must happen before the client is able to send data frames, as shown in this screen capture:
The debugs reveal the same EAP authentication frame exchange as the rest of the methods upon initial authentication to the WLAN, with some outputs added in regards to the key caching techniques used here. These debug outputs are cut in order to show mainly the new information, not the entire EAP frame exchange, because basically the same information is exchanged every time for authentication of the client against the Authentication Server. This is demonstrated thus far, and correlated with the EAP authentication frames shown in the packet captures, so most of the EAP messages are removed from the debug outputs for simplicity:

*apfMsConnTask_0: Jun 22 00:23:15.097: ec:85:2f:15:39:32
Association received from mobile on BSSID 84:78:ac:0f:68:d2
--- This is the Association Request from the client.

*apfMsConnTask_0: Jun 22 00:23:15.097: ec:85:2f:15:39:32
Processing RSN IE type 48, length 20 for mobile ec:85:2f:15:39:32
--- The WLC/AP finds an Information Element that claims PMKID Caching support on the Association request that is sent from the client.

*apfMsConnTask_0: Jun 22 00:23:15.098: ec:85:2f:15:39:32
Received RSN IE with 0 PMKIDs from mobile ec:85:2f:15:39:32
--- Since this is an initial association, the Association Request comes without any PMKID.

*apfMsConnTask_0: Jun 22 00:23:15.098: ec:85:2f:15:39:32
Setting active key cache index 8 ---> 8

*apfMsConnTask_0: Jun 22 00:23:15.099: ec:85:2f:15:39:32
Sending Assoc Response to station on BSSID 84:78:ac:0f:68:d2
--- The Association Response is sent to the client.

*dot1xMsgTask: Jun 22 00:23:15.103: ec:85:2f:15:39:32
Sending EAP-Request/Identity to mobile ec:85:2f:15:39:32
(EAP Id 1)
Dot1x_NW_MsgTask_2: Jun 22 00:23:15.118: ec:85:2f:15:39:32
Received EAPOL EAPPKT from mobile ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.118: ec:85:2f:15:39:32
Received Identity Response (count=1) from mobile ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.126: ec:85:2f:15:39:32
Processing Access-Challenge for mobile ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.126: ec:85:2f:15:39:32
Sending EAP Request from AAA to mobile ec:85:2f:15:39:32
(EAP Id 2)

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.146: ec:85:2f:15:39:32
Received EAPOL EAPPKT from mobile ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.146: ec:85:2f:15:39:32
Received EAP Response from mobile ec:85:2f:15:39:32
(EAP Id 2, EAP Type 25)

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274: ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274: ec:85:2f:15:39:32
Creating a PKC PMKID Cache entry for station ec:85:2f:15:39:32
(RSN 2)

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274: ec:85:2f:15:39:32
Setting active key cache index 8 ---> 8

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274: ec:85:2f:15:39:32
Setting active key cache index 8 ---> 0

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274: ec:85:2f:15:39:32
Adding BSSID 84:78:ac:f0:68:d2 to PMKID cache at index 0
for station ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274:
New PMKID: (16)

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274:
[0000] c9 4d 0d 97 03 aa a9 0f 1b c8 33 73 01 f1 18 f5

WLC creates a PMK cache entry for this client, which is
used for SKC in this case, so the PMKID is computed with
the AP MAC address (BSSID 84:78:ac:f0:68:d2).

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.274: ec:85:2f:15:39:32
Sending EAP-Success to mobile ec:85:2f:15:39:32
(EAP Id 12)

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.275:
Including PMKID in M1 (16)

The hashed PMKID is included on the Message-1 of the
WPA/WPA2 4-Way handshake.

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.275:
[0000] c9 4d 0d 97 03 aa a9 0f 1b c8 33 73 01 f1 18 f5

This is the hashed PMKID.

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.275: ec:85:2f:15:39:32
Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32
state INITPMK (message 1), replay counter
00.00.00.00.00.00.00.00

Message-1 of the WPA/WPA2 4-Way handshake is sent from
the WLC/AP to the client.

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.284: ec:85:2f:15:39:32
Received EAPOL-Key from mobile ec:85:2f:15:39:32

Dot1x_NW_MsgTask_2: Jun 22 00:23:15.284: ec:85:2f:15:39:32
Received EAPOL-key in PTK_START state (message 2) from mobile ec:85:2f:15:39:32

--- Message-2 of the WPA/WPA-2 4-Way handshake is successfully received from the client.

*Dot1x_NW_MsgTask_2: Jun 22 00:23:15.284: ec:85:2f:15:39:32
PMK: Sending cache add

*Dot1x_NW_MsgTask_2: Jun 22 00:23:15.285: ec:85:2f:15:39:32
Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state PTKINITNEGOTIATING (message 3), replay counter 00.00.00.00.00.00.01

--- Message-3 of the WPA/WPA2 4-Way handshake is sent from the WLC/AP to the client.

*Dot1x_NW_MsgTask_2: Jun 22 00:23:15.291: ec:85:2f:15:39:32
Received EAPOL-Key from mobile ec:85:2f:15:39:32
*Dot1x_NW_MsgTask_2: Jun 22 00:23:15.291: ec:85:2f:15:39:32
Received EAPOL-key in PTKINITNEGOTIATING state (message 4) from mobile ec:85:2f:15:39:32

--- Message-4 (final message) of this initial WPA/WPA2 4-Way handshake is successfully received from the client, which confirms the installation of the derived keys. They can now be used in order to encrypt data frames with the current AP.

With this method, the AP and wireless client cache the PMKs of the secure associations already established. Therefore, if the wireless client roams to a new AP where it has never associated, the client must perform a full EAP authentication again, as shown in this wireless capture where the client roams to a new AP:

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>BSS Id</th>
<th>Protocol</th>
<th>Channel frequency</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>Apple_10:39:32</td>
<td>Cisco_05:31:92</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.002310</td>
<td>Cisco_05:31:92</td>
<td>Apple_10:39:32</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.007754</td>
<td>Apple_15:39:32</td>
<td>Cisco_09:2a:92</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.007138</td>
<td>Cisco_00:2a:92</td>
<td>Apple_15:39:32</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
</tbody>
</table>

With this method, the AP and wireless client cache the PMKs of the secure associations already established. Therefore, if the wireless client roams to a new AP where it has never associated, the client must perform a full EAP authentication again, as shown in this wireless capture where the client roams to a new AP:

However, if the wireless client roams back to an AP where a previous association/authentication took place, then the client sends a Reassociation Request frame that lists multiple PMKIDs, which informs the AP of the PMKs cached from all of the APs where the client has previously authenticated. Therefore, since the client is roaming back to an AP that also has a PMK cached for this client, then the client does not need to reauthenticate through EAP in order to derive a new PMK. The client simply goes through the WPA2 4-Way handshake in order to derive the new transient encryption keys:

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>BSS Id</th>
<th>Protocol</th>
<th>Channel frequency</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>Cisco_05:31:92</td>
<td>Apple_15:39:32</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.002154</td>
<td>Apple_15:39:32</td>
<td>Cisco_05:31:92</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.002393</td>
<td>Cisco_05:31:92</td>
<td>Apple_15:39:32</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.004531</td>
<td>Cisco_05:31:92</td>
<td>Apple_15:39:32</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.005072</td>
<td>Apple_15:39:32</td>
<td>Cisco_05:31:92</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.005263</td>
<td>Cisco_05:31:92</td>
<td>Apple_15:39:32</td>
<td>E47:8ac:01:2a:92</td>
<td>802.11</td>
<td>Authentication, SN=462, PMK, Flags-</td>
<td></td>
</tr>
</tbody>
</table>

Note: This capture does not show the first 802.11 Open System authentication frame from
the client, but this is not due to the method implemented, as this frame is always required. The reason is that this specific frame is not captured by the adapter or wireless packet capture software used in order to sniff the over-the-air frames for this example, but it is left like this on the example for educational purposes. Be aware that there is a possibility that this might happen when you perform over-the-air packet captures; some frames might be missed by the capture, but are actually exchanged between the client and the AP. Otherwise, the roaming never starts on this example.

Here is a summary of the WLC debugs for this fast-secure roaming method:

*apfMsConnTask_0: Jun 22 00:26:40.787: ec:85:2f:15:39:32
  Reassociation received from mobile on BSSID 84:78:ac:f0:68:d2
  !--- This is the Reassociation Request from the client.

*apfMsConnTask_0: Jun 22 00:26:40.787: ec:85:2f:15:39:32
  Processing RSN IE type 48, length 38 for mobile ec:85:2f:15:39:32
  !--- The WLC/AP finds an Information Element that claims PMKID Caching support on the Association request that is sent from the client.

*apfMsConnTask_0: Jun 22 00:26:40.787: ec:85:2f:15:39:32
  Received RSN IE with 1 PMKIDs from mobile ec:85:2f:15:39:32
  !--- The Reassociation Request from the client comes with one PMKID.

*apfMsConnTask_0: Jun 22 00:26:40.787:
  Received PMKID: (16)
  *apfMsConnTask_0: Jun 22 00:26:40.788:
  [0000] c9 4d 0d 97 03 aa a9 0f 1b c8 33 73 01 f1 18 f5
  !--- This is the PMKID that is received.

*apfMsConnTask_0: Jun 22 00:26:40.788: ec:85:2f:15:39:32
  Searching for PMKID in MSCB PMKID cache for mobile ec:85:2f:15:39:32
  !--- WLC searches for a matching PMKID on the database.

*apfMsConnTask_0: Jun 22 00:26:40.788: ec:85:2f:15:39:32
  Found a cache entry for BSSID 84:78:ac:f0:68:d2 in PMKID cache at index 0 of station ec:85:2f:15:39:32

*apfMsConnTask_0: Jun 22 00:26:40.788: ec:85:2f:15:39:32
  Found a valid PMKID in the MSCB PMKID cache for mobile ec:85:2f:15:39:32
  !--- The WLC validates the PMKID provided by the client, and confirms that it has a valid PMK cache for this client-and-AP pair.

*apfMsConnTask_0: Jun 22 00:26:40.788: ec:85:2f:15:39:32
  Setting active key cache index 1 ---> 0

*apfMsConnTask_0: Jun 22 00:26:40.788: ec:85:2f:15:39:32
  Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d2(status 0) ApVapId 3 Slot 0
  !--- The Reassociation Response is sent to the client, which
validates the fast-roam with SKC.

*dot1xMsgTask: Jun 22 00:26:40.795: ec:85:2f:15:39:32
Initiating RSN with existing PMK to mobile
ec:85:2f:15:39:32

--- WLC initiates a Robust Secure Network association with
this client-and-AP pair based on the cached PMK found.

Hence, EAP is avoided as per the next message.

*dot1xMsgTask: Jun 22 00:26:40.795: ec:85:2f:15:39:32
Skipping EAP-Success to mobile ec:85:2f:15:39:32

*dot1xMsgTask: Jun 22 00:26:40.795: ec:85:2f:15:39:32
Found an cache entry for BSSID 84:78:ac:f0:68:d2 in
PMKID cache at index 0 of station ec:85:2f:15:39:32

*dot1xMsgTask: Jun 22 00:26:40.795: Including PMKID in M1(16)

--- The hashed PMKID is included on the Message-1 of the
WPA/WPA2 4-Way handshake.

*dot1xMsgTask: Jun 22 00:26:40.795:
[0000] c9 4d 0d 97 03 aa a9 0f 1b c8 33 73 01 f1 18 f5

--- The PMKID is hashed. The next messages are the same
WPA/WPA2 4-Way handshake messages described thus far
that are used in order to finish the encryption keys
generation/installation.

*dot1xMsgTask: Jun 22 00:26:40.795: ec:85:2f:15:39:32
Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state
INITPMK (message 1), replay counter 00.00.00.00.00.00.00.00

*Dot1x_NW_MsgTask_2: Jun 22 00:26:40.811: ec:85:2f:15:39:32
Received EAPOL-Key from mobile ec:85:2f:15:39:32

*Dot1x_NW_MsgTask_2: Jun 22 00:26:40.812: ec:85:2f:15:39:32
Received EAPOL-key in PTK_START state (message 2) from mobile
ec:85:2f:15:39:32

*Dot1x_NW_MsgTask_2: Jun 22 00:26:40.812: ec:85:2f:15:39:32
PMK: Sending cache add

*Dot1x_NW_MsgTask_2: Jun 22 00:26:40.812: ec:85:2f:15:39:32
Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state
PTKINITNEGOTIATING (message 3), replay counter
00.00.00.00.00.00.00.00.01

*Dot1x_NW_MsgTask_2: Jun 22 00:26:40.820: ec:85:2f:15:39:32
Received EAPOL-Key from mobile ec:85:2f:15:39:32

*Dot1x_NW_MsgTask_2: Jun 22 00:26:40.820: ec:85:2f:15:39:32
Received EAPOL-key in PTKINITNEGOTIATING state (message 4)
from mobile ec:85:2f:15:39:32

**FlexConnect with PMKID Caching / Sticky Key Caching**

- When you use this method on a FlexConnect setup, it could work and the behavior might seem similar to what was previously explained if you use Central Authentication back to the WLC (with either central or local switching); however, this SKC method is not supported on FlexConnect.
- This method is only officially supported on CUWN with Local mode APs, not on FlexConnect or other modes.
Pros with PMKID Caching / Sticky Key Caching

This method can be implemented locally by autonomous-independent APs, without the need of a centralized device to manage the cached keys.

Cons with PMKID Caching / Sticky Key Caching

- As mentioned earlier in this document, the main limitation of this method is that the client can only perform fast-secure roaming when roaming back to an AP where it previously associated/authenticated. If roaming to a new AP, the client must complete the full EAP authentication again.
- The wireless client and APs must remember all of the PMKs derived on every new authentication, so this feature is normally limited to a certain amount of PMKs that are cached. Since this limit is not clearly defined by the standard, the vendors can define different limits on their SKC implementations. For example, the Cisco WLAN Controllers can currently cache the PMKs from a client for up to eight APs. If a client roams to more than eight APs per session, the oldest APs are removed from the cache list in order to store the newly-cached entries.
- This method is optional and is still not supported by many WPA2 devices; therefore, this method is not widely adopted and deployed.
- SKC is not supported when you perform intercontroller roaming, which occurs when you move between APs managed by different WLCs, even if they are on the same mobility group.

Fast-Secure Roaming with Opportunistic Key Caching

Opportunistic Key Caching (OKC), also known as Proactive Key Caching (PKC) (this term is explained in greater detail in a note that follows), is basically an enhancement of the WPA2 PMKID caching method described previously, which is why it is also named Proactive/Opportunistic PMKID Caching. Hence, it is important to note that this is not a fast-secure roaming method defined by the 802.11 standard and is not supported by many devices, but just like PMKID caching, it works with WPA2-EAP.

This technique allows the wireless client and the WLAN infrastructure to cache only one PMK for the lifetime of the client association with this WLAN (derived from the MSK after the initial 802.1X/EAP authentication with the Authentication Server), even when roaming between multiple APs, as they all share the original PMK that is used as the seed on all WPA2 4-way handshakes. This is still required, just as it is in SKC, in order to generate new encryption keys every time the client reassociates with the APs. For the APs to share this one original PMK from the client session, they must all be under some sort of administrative control, with a centralized device that caches and distributes the original PMK for all of the APs. This is similar to the CUWN, where the WLC performs this job for all of the LAPs under its control, and uses the mobility groups in order to handle this PMK between multiple WLCs; therefore, this is a limitation on autonomous AP environments.

With this method, just as in PMKID Caching (SKC), the initial association to any AP is a regular first-time authentication to the WLAN, where you must complete the entire 802.1X/EAP authentication against the Authentication Server and the 4-Way handshake for key generation before you can send data frames. Here is a screen capture that illustrates this:
The debug outputs show basically the same EAP authentication frame exchange as the rest of the methods described in this document upon initial authentication to the WLAN (as shown in the captures), along with the addition of some outputs that concern the key caching techniques used by the WLC here. This debug output is also cut in order to show only the relevant information:

*apfMsConnTask_0: Jun 21 21:46:06.515: 00:40:96:b7:ab:5c
Association received from mobile on BSSID 84:78:ac:f0:68:d2
--- This is the Association Request from the client.

*apfMsConnTask_0: Jun 21 21:46:06.516: 00:40:96:b7:ab:5c
Processing RSN IE type 48, length 20 for mobile 00:40:96:b7:ab:5c
--- The WLC/AP finds an Information Element that claims
PMKID Caching support on the Association request that
is sent from the client.

*apfMsConnTask_0: Jun 21 21:46:06.516: 00:40:96:b7:ab:5c
Received RSN IE with 0 PMKIDs from mobile 00:40:96:b7:ab:5c
--- Since this is an initial association, the Association
Request comes without any PMKID.

*apfMsConnTask_0: Jun 21 21:46:06.516: 00:40:96:b7:ab:5c
Setting active key cache index 0 ----> 8

*apfMsConnTask_0: Jun 21 21:46:06.516: 00:40:96:b7:ab:5c
Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d2 (status 0) ApVapId 3 Slot
--- The Association Response is sent to the client.

*dot1xMsgTask: Jun 21 21:46:06.522: 00:40:96:b7:ab:5
Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c (EAP Id 1)

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.614: 00:40:96:b7:ab:5c
Received EAPOL Data, SN=2437, FN=0, Flags...
Sending EAP-Request/Identity to mobile 00:40:96:b7:ab:5c (EAP Id 2)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

Received Identity Response (count=2) from mobile 00:40:96:b7:ab:5c

Processing Access-Challenge for mobile 00:40:96:b7:ab:5c

Sending EAP Request from AAA to mobile 00:40:96:b7:ab:5c (EAP Id 3)

Received EAPOL EAPPKT from mobile 00:40:96:b7:ab:5c

Received EAP Response from mobile 00:40:96:b7:ab:5c (EAP Id 3, EAP Type 25)

Processing Access-Accept for mobile 00:40:96:b7:ab:5c

Creating a PKC PMKID Cache entry for station 00:40:96:b7:ab:5c (RSN 2)

Setting active key cache index 8 ---> 8

Setting active key cache index 8 ---> 0

Adding BSSID 84:78:ac:f0:68:d2 to PMKID cache at index 0 for station 00:40:96:b7:ab:5

WLC creates a PMK cache entry for this client, which is used for OKC in this case, so the PMKID is computed with the AP MAC address (BSSID 84:78:ac:f0:68:d2).

In the WPA/WPA2 4-Way handshake, the hashed PMKID is included on the Message-1 of the WPA/WPA2 4-Way handshake.

Sending EAP-Success to mobile 00:40:96:b7:ab:5c (EAP Id 13)

Found an cache entry for BSSID 84:78:ac:f0:68:d2 in PMKID cache at index 0 of station 00:40:96:b7:ab:5

--- The PMK cache entry for this client is shared with the WLCs on the mobility group.

--- The hashed PMKID is included on the Message-1 of the WPA/WPA2 4-Way handshake.
--- This is the hashed PMKID. The next messages are the same WPA/WPA2 4-Way handshake messages described thus far that are used in order to finish the encryption keys generation/installation.

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.844: 00:40:96:b7:ab:5c
  Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c state INITPMK (message 1), replay counter 00.00.00.00.00.00.00.00

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.865: 00:40:96:b7:ab:5c
  Received EAPOL-Key from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.865: 00:40:96:b7:ab:5c
  Received EAPOL-key in PTK_START state (message 2)
  from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.865: 00:40:96:b7:ab:5c
  PMK: Sending cache add

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.865: 00:40:96:b7:ab:5c
  Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c state PTKINITNEGOTIATING (message 3), replay counter 00.00.00.00.00.00.00.01

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.890: 00:40:96:b7:ab:5c
  Received EAPOL-Key from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 21:46:06.890: 00:40:96:b7:ab:5c
  Received EAPOL-key in PTKINITNEGOTIATING state (message 4)
  from mobile 00:40:96:b7:ab:5c

With this method, the wireless client and the WLC (for all the managed APs) cache the one original PMK of the secure association that is initially established. Basically, every time the wireless client connects to a specific AP, a PMKID is hashed based on: the client MAC address, the AP MAC address (BSSID of the WLAN), and the PMK derived with that AP. Therefore, since OKC caches the same original PMK for all of the APs and the specific client, when this client (re)associates to another AP, the only value that changes in order to hash the new PMKID is the new AP MAC address.

When the client initiates roaming to a new AP and sends the Reassociation Request frame, it adds the PMKID on the WPA2 RSN Information Element if it wants to inform the AP that a cached PMK is used for fast-secure roaming; as it already knows the MAC address of the BSSID (AP) for where it roams, then the client simply hashes the new PMKID that is used on this Reassociation Request. When the AP receives this request from the client, it also hashes the PMKID with the values that it already has (the cached PMK, the client MAC address, and its own AP MAC address), and responds with the successful Reassociation Response that confirms the PMKIDs matched. The cached PMK can be used as the seed that starts a WPA2 4-Way handshake in order to derive the new encryption keys (skipping EAP):
In this capture, the Reassociation Request frame from the client is selected and expanded so that you can see more details of the frame. The MAC address information and also the Robust Security Network (RSN, as per 802.11i – WPA2) Information Element, where information about the WPA2 settings used for this association is shown (highlighted is the PMKID obtained from the hashed formula).

Here is a summary of WLC debugs for this fast-secure roaming method with OKC:

```
*apfMsConnTask_2: Jun 21 21:48:50.562: 00:40:96:b7:ab:5c
Reassociation received from mobile on BSSID 84:78:ac:02:a9:92

--- This is the Reassociation Request from the client.

*apfMsConnTask_2: Jun 21 21:48:50.563: 00:40:96:b7:ab:5c
Processing RSN IE type 48, length 38 for mobile 00:40:96:b7:ab:5c

--- The WLC/AP finds and Information Element that claims PMKID Caching support on the Association request that is sent from the client.

*apfMsConnTask_2: Jun 21 21:48:50.563: 00:40:96:b7:ab:5c
Received RSN IE with 1 PMKIDs from mobile 00:40:96:b7:ab:5c

--- The Reassociation Request from the client comes with one PMKID.

*apfMsConnTask_2: Jun 21 21:48:50.563:
Received PMKID: (16)

*apfMsConnTask_2: Jun 21 21:48:50.563:
```
As the client has never authenticated with this new AP, the WLC cannot find a valid PMKID to match the one provided by the client. However, since the client performs OKC and not SKC (as per the following messages), the WLC computes a new PMKID based on the information gathered (the cached PMK, the client MAC address, and the new AP MAC address).

The new PMKID is computed and validated to match the one provided by the client, which is also computed with the same information. Hence, the fast-secure roam is possible.
Initiating RSN with existing PMK to mobile 00:40:96:b7:ab:5c

--- WLC initiates a Robust Secure Network association with this client-and AP pair with the cached PMK found. Hence, EAP is avoided, as per the next message.

*dot1xMsgTask: Jun 21 21:48:50.570: 00:40:96:b7:ab:5c
Skipping EAP-Success to mobile 00:40:96:b7:ab:5c

*dot1xMsgTask: Jun 21 21:48:50.570: 00:40:96:b7:ab:5c
Found an entry for BSSID 84:78:ac:f0:2a:92 in PMKID cache at index 0 of station 00:40:96:b7:ab:5c

*dot1xMsgTask: Jun 21 21:48:50.570:
Including PMKID in M1 (16)

--- The hashed PMKID is included on the Message-1 of the WPA/WPA2 4-Way handshake.

*dot1xMsgTask: Jun 21 21:48:50.570: 00:40:96:b7:ab:5c
Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c state INITPMK (message 1), replay counter 00.00.00.00.00.00.00.00

*Dot1x_NW_MsgTask_4: Jun 21 21:48:50.589: 00:40:96:b7:ab:5c
Received EAPOL-Key from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 21:48:50.589: 00:40:96:b7:ab:5c
Received EAPOL-key in PTK_START state (message 2) from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 21:48:50.590: 00:40:96:b7:ab:5c
PMK: Sending cache add

*Dot1x_NW_MsgTask_4: Jun 21 21:48:50.590: 00:40:96:b7:ab:5c
Sending EAPOL-Key Message to mobile 00:40:96:b7:ab:5c state PTKINITNEGOTIATING (message 3), replay counter 00.00.00.00.00.00.00.01

*Dot1x_NW_MsgTask_4: Jun 21 21:48:50.610: 00:40:96:b7:ab:5c
Received EAPOL-Key from mobile 00:40:96:b7:ab:5c

*Dot1x_NW_MsgTask_4: Jun 21 21:48:50.610: 00:40:96:b7:ab:5c
Received EAPOL-key in PTKINITNEGOTIATING state (message 4) from mobile 00:40:96:b7:ab:5c

As shown at the beginning of the debugs, the PMKID must be computed after the Reassociation Request from the client is received. This is needed in order to validate the PMKID and confirm that the cached PMK is used with the WPA2 4-Way handshake to derive the encryption keys and finish the fast-secure roaming. Do not confuse the CCKM entries on the debugs; this is not used in order to perform CCKM, but OKC, as previously explained. CCKM here is simply a name used by the WLC for those outputs, such as the name of a function that handles the values in order to compute the PMKID.

**FlexConnect with Opportunistic Key Caching**

- Central Authentication is supported. This includes Local and Central data switching. If the AP
is part of the same FlexConnect group, fast-secure roam is by the AP, else fast-secure roam is by the controller.

**Note:** This setup might work if the APs are not on the same FlexConnect Group, but this is not a recommended or supported setup.

- Flex Local Authentication is supported. In connected mode, the cache will be distributed from the AP to the controller and then to the rest of the APs in the FlexConnect Group.
- Standalone mode is supported. If the cache is already present on the AP (due to previous distribution), fast-secure roam shall work. New authentication in standalone mode does not support fast-secure roaming.

**Pros with Opportunistic Key Caching**

- The wireless client and the WLAN infrastructure do not need to remember multiple PMKIDs, but simply cache the one original PMK from the initial authentication to the WLAN. Then you must rehash the proper PMKID (used on the Reassociation Request) required with each AP secure association in order to validate the fast-secure roaming.
- Here, the wireless client performs fast-secure roaming to a new AP on the same WLAN/SSID, even if it never associated with that AP (not the case in SKC). As long as the client performs the initial 802.1X/EAP authentication with one AP managed by the centralized deployment that handles the PMK cache for all of the APs for where the client roams, then no more full authentications are required for the rest of the client lifetime on this WLAN.

**Cons with Opportunistic Key Caching**

- This method is only deployed on a centralized environment where all of the APs are under some sort of administrative control (such as a WLAN Controller) that is responsible for caching and sharing the one original PMK from the client session. Therefore, this is a limitation on autonomous AP environments.
- The techniques that are applied in this method are not suggested or described on the 802.11 standard, so the support varies widely from one device to another. Nevertheless, this is still the method that was more adopted while waiting for 802.11r.

**Note about the term “Proactive Key Caching”**

Proactive Key Caching (or PKC) has been known as OKC (Opportunistic Key Caching), and the two terms have been used interchangeably when describing the same method explained here. However, this was just a term that was used by Airspace in 2001 for an old key caching method, which was then used by the 802.11i standard as the basis for “Preauthentication” (another Fast Secure Roaming method briefly explained below). PKC is not Preauthentication or OKC (Opportunistic Key Caching), but when you hear or read about PKC, the reference is basically to OKC, and not to Preauthentication.

**Fast-Secure Roaming with Preauthentication**

This method is also suggested by the IEEE 802.11 standard within the 802.11i security amendment, so it also works with WPA2, but it is the only Fast Secure Roaming method that is not supported by Cisco WLAN infrastructures. For this reason, it is explained only briefly here and without outputs.
With Preauthentication, the wireless clients can authenticate with multiple APs at a time while associated with the current AP. When this occurs, the client sends the EAP authentication frames to the current AP where it is connected, but it is destined to the other AP(s) where the client wants to perform Preauthentication (neighbor APs that are possible candidates for roaming). The current AP sends these frames to the target AP(s) over the distribution system. The new AP performs full authentication against the RADIUS server for this client, so an entire new EAP authentication handshake is completed, and this new AP acts as the Authenticator.

The idea is to perform authentication and derive PMK with the neighbor APs before the client actually roams to them, so when it is time to roam, the client is already authenticated and with a PMK already cached for this new AP-to-client secure association, so they only need to perform the 4-Way Handshake and experience a fast roam after the client sends its initial Reassociation request.

Here is a capture from an AP’s beacon that shows the RSN IE field that advertises support for Preauthentication (this one is from a Cisco AP, where it is confirmed that Preauthentication is not supported):

**Pros with Preauthentication**

There is one PMK for each AP-to-client secure association, which could be considered a security advantage in case an AP is compromised and the keys become stolen (cannot be used with other APs). However, this security advantage is handled by the WLAN infrastructure in different ways on other methods.

**Cons with Preauthentication**

- Because there is one PMK per AP, the clients have a limit on the amount of APs that can be pre-authenticated.
• Every time a client performs preauthentication with a new AP, there is a complete EAP authentication exchange, which means more load on the network and on the Authentication Server.
• Most wireless clients do not support this method, as this was never highly adopted (OKC was more adopted).

Fast-Secure Roaming with 802.11r

The fast-secure roaming technique based on the 802.11r amendment (officially named Fast BSS Transition by the 802.11 standard, and known as FT) is the first method officially ratified (on 2008) by the IEEE for the 802.11 standard as the solution to perform fast transitions between APs (Basic Service Sets or BSSs), which clearly defines the key hierarchy that is used when you handle and cache keys on a WLAN. However, its adoption has been slow, mainly due to the other solutions already available when fast transitions were actually required, such as with VoWLAN implementations when used with one of the methods previously explained in this document. There are only a few devices that currently support some of the FT options (by 2013).

This technique is more complex to explain than the other methods, as it introduces new concepts and multiple layers of PMKs that are cached on different devices (each device with a different role), and provides even more options for fast-secure roaming. Therefore, a brief summary is provided about this method and the way it is implemented with each option available.

802.11r is different from SKC and OKC, primarily because of these reasons:

- Handshake messaging (PMKID, ANonce, and SNonce exchange, for example) happens in 802.11 Authentication frames or in Action frames instead of Reassociation frames. Unlike PMKID caching methods, the separate 4-Way handshake phase, which is carried after the (re)association message exchange, is avoided. The key handshake with the new AP begins before the client fully roams/reassociates with this new AP.
- It provides two methods for the fast-roaming handshake: over the AIR, and over the Distribution System (DS).
- 802.11r has more layers of key hierarchy.
- As this protocol avoids the 4-Way handshake for the key management when a client roams (generates new encryption keys -PTK and GTK- without the need of this handshake), it can also be applied for WPA2 setups with a PSK, and not only when 802.1X/EAP is used for the authentication. This accelerates the roaming even more for these setups, where no EAP or 4-Way handshake exchanges occur.

With this method, the wireless client performs just one initial authentication against the WLAN infrastructure when a connection is established to the first AP, and performs fast-secure roaming while roaming between APs of the same FT mobility domain.

This is one of the new concepts, which basically refers to the APs that use the same SSID (known as an Extended Service Set or ESS) and handle the same FT keys. This is similar to the other methods explained thus far. The way the APs handle the FT mobility domain keys is normally based on a centralized setup, such as the WLC or mobility groups; however, this method can also be implemented on autonomous AP environments.

Here is a summary of the key hierarchy:

- An MSK is still derived on the client supplicant and the Authentication Server from the initial
802.1X/EAP authentication phase (transferred from the Authentication Server to the Authenticator (WLC) once the authentication is successful). This MSK, like in the other methods, is used as the seed for the FT key hierarchy. When you use WPA2-PSK instead of an EAP authentication method, the PSK is basically this MSK.

- A Pairwise Master Key R0 (PMK-R0) is derived from the MSK, which is the first-level key of the FT key hierarchy. The key holders for this PMK-R0 are the WLC and the client.
- A second-level key, called a Pairwise Master Key R1 (PMK-R1), is derived from the PMK-R0, and the key holders are the client and the APs managed by the WLC that holds the PMK-R0.
- The third and final level key of the FT key hierarchy is the PTK, which is the final key used in order to encrypt the 802.11 unicast data frames (similar to the other methods that use WPA/TKIP or WPA2/AES). This PTK is derived on FT from the PMK-R1, and the key holders are the client and the APs managed by the WLC.

**Note:** Dependent upon the WLAN vendor and the implementation setups (such as Autonomous APs, FlexConnect, or Mesh), the WLAN infrastructure might transfer and handle the keys in a different way. It might even change the roles of the key holders, but since that is out of the scope of this document, the examples based on the key hierarchy summary given previously are the next focus. The differences are actually not that relevant to understand the process, unless you actually need to analyze in-depth the infrastructure devices (and their code) in order to discover a software issue.

**Fast BSS Transition Over-the-Air**

With this method, the first association to any AP is a regular first-time authentication to the WLAN, where the entire 802.1X/EAP authentication against the Authentication Server and the 4-Way handshake for key generation must occur before data frames are sent, as shown in this screen capture:
The main differences are:

- The Authentication Key Management negotiation is slightly different than regular WPA/WPA2, so some extra information is used in order to perform this negotiation when the association to a WLAN infrastructure that supports FT occurs. As shown in the capture, the Association Request frame from the client is selected and the AKM field of the RNS Information Element is highlighted in order to show that this client wants to perform FT over 802.1X/EAP.

- Also shown is the Mobility Domain Information Element (part of FT), where the FT Capability and Policy field indicates if the Fast BSS Transition is completed Over-the-Air or Over-the-DS when fast roaming (indicating Over-the-Air in this capture).

- Another information element is also added (Fast BSS Transition or FT IE, which is described later in this document) with information that is required in order to perform the FT authentication sequence when FT roaming.

- The key generation is different due to the key hierarchy, so even though the FT 4-Way handshake looks similar to the WPA/WPA2 4-Way handshake, it is actually slightly different in content.

The debugs show basically the same EAP authentication frame exchange as the rest of the methods upon initial authentication to the WLAN (as noticed from the captures), but some outputs that concern the key caching techniques used by the WLC are added; thus, this debug output is cut in order to show only the relevant information:
--- This is the Association request from the client.

  Marking this mobile as TGr capable.

--- WLC recognizes that the client is 802.11r-capable.

  Processing RSN IE type 48, length 20 for mobile ec:85:2f:15:39:32

--- The WLC/AP finds an Information Element that claims FT support on the Association request that is sent from the client.

*apfMsConnTask_0: Jun 27 19:25:23.427:
  Sending assoc-resp station:ec:85:2f:15:39:32
  AP:84:78:ac:f0:68:d0-00 thread:144be808

*apfMsConnTask_0: Jun 27 19:25:23.427:
  Adding MDIE, ID is:0xaaf0

  Including FT Mobility Domain IE (length 5) in Initial assoc Resp to mobile

  Sending R0KH-ID as:-84.30.6.-3

  Sending R1KH-ID as 3c:ce:73:d8:02:00

  Including FT IE (length 98) in Initial Assoc Resp to mobile

  Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d6 (status 0) ApVapId 7 Slot 0

--- The Association Response is sent to the client once the FT information is computed (as per the previous messages), so this is included in the response.

  Sending EAP-Request/Identity to mobile ec:85:2f:15:39:32 (EAP Id 1)

--- EAP begins, and follows the same exchange explained so far.

  Got action frame from this client.

  Received EAPOL EAPPKT from mobile ec:85:2f:15:39:32

  Received Identity Response (count=1) from mobile ec:85:2f:15:39:32

  Processing Access-Challenge for mobile ec:85:2f:15:39:32

  Sending EAP Request from AAA to mobile ec:85:2f:15:39:32 (EAP Id 2)

  Received EAPOL EAPPKT from mobile ec:85:2f:15:39:32

  Received EAP Response from mobile ec:85:2f:15:39:32 (EAP Id 2, EAP Type 25)

--- The client is validated/authenticated by the RADIUS Server.

Creating a PKC PMKID Cache entry for station
ec:85:2f:15:39:32 (RSN 2)
Resetting MSCB PMK Cache Entry 0 for station
ec:85:2f:15:39:32
Setting active key cache index 8 ---> 8
Setting active key cache index 8 ---> 0
Adding BSSID 84:78:ac:f0:68:d6 to PMKID cache at index 0
for station ec:85:2f:15:39:32
*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.628: New PMKID: (16)
*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.628: [0000] 52 b8 8f cf 50 a7 90 98 2b ba d6 20 79 e4 cd f9
Created PMK Cache Entry for TGr AKM:802.1x ec:85:2f:15:39:32

--- WLC creates a PMK cache entry for this client, which is
used for FT with 802.1X in this case, so the PMKID is
computed with the AP MAC address (BSSID 84:78:ac:f0:68:d6).

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.629:
RIKH-ID:3:00::30:2:02:00
MSK Len:48
pmkValidTime:1807

--- The ROKH-ID and R1KH-ID are defined, as well as the PMK
cache validity period.

PMK sent to mobility group

--- The FT PMK cache entry for this client is shared with the
WLCs on the mobility group.

Sending EAP-Success to mobile ec:85:2f:15:39:32 (EAP Id 12)

Found an cache entry for BSSID 84:78:ac:f0:68:d6 in PMKID
cache at index 0 of station ec:85:2f:15:39:32

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.630: Including PMKID in
MI (16)

--- The hashed PMKID is included on the Message-1 of the
initial FT 4-Way handshake.

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.630: [0000] 52 b8 8f cf 50 a7 90 98 2b ba d6 20 79 e4 cd f9

Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state
INITPMK (message 1), replay counter 00.00.00.00.00.00.00.0

--- Message-1 of the FT 4-Way handshake is sent from the
WLC/AP to the client.

Received EAPOL-Key from mobile ec:85:2f:15:39:32

Received EAPOL-key in PTK_START state (message 2) from
mobile ec:85:2f:15:39:32

--- Message-2 of the FT 4-Way handshake is received
successfully from the client.
Calculating PMKR0Name

--- The PMKR0Name is calculated.

Sending cache add

Adding MDIE, ID is:0xaaf0

Adding TIE for reassociation deadtime:20000 milliseconds

Adding TIE for R0Key-Data valid time :1807

Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state PTKINITNEGOITIATING (message 3), replay counter 00.00.00.00.00.00.00.01

--- After the MDIE, TIE for reassociation deadtime, and TIE for R0Key-Data valid time are calculated, the Message-3 of this FT 4-Way handshake is sent from the WLC/AP to the client with this information.

Received EAPOL-Key from mobile ec:85:2f:15:39:32

Received EAPOL-key in PTKINITNEGOITIATING state (message 4) from mobile ec:85:2f:15:39:32

--- Message-4 (final message) of this initial FT 4-Way handshake is received successfully from the client, which confirms the installation of the derived keys. They can now be used in order to encrypt data frames with the current AP.

**Note:** In order to debug this method and reach the extra 802.11r/FT outputs shown here, an additional debug is enabled along with the **debug client**, which is the **debug ft events enable**.

Here are the captures and debugging of an initial association to the WLAN when you perform FT with WPA2-PSK (instead of an 802.1X/EAP method), where the Association Response frame from the AP is selected in order to show the Fast BSS Transition Information Element (highlighted). Some of the key information needed in order to perform the FT 4-Way handshake is also shown:
Association received from mobile on BSSID 84:78:ac:f0:68:d6

Marking this mobile as TGr capable.

The WLC/AP finds an Information Element that claims FT support on the Association request that is sent from the client.

--- The WLC/AP finds an Information Element that claims FT support on the Association request that is sent from the client.
Sending Assoc Response to station on BSSID 84:78:ac:f0:68:d6
(status 0) ApVapId 7 Slot 0

--- The Association Response is sent to the client once the
FT information is computed (as per the previous messages),
so this is included in the response.

Sending EAP-Request/Identity to mobile ec:85:2f:15:39:32
(EAP Id 1)

--- EAP begins, and follows the same exchange explained so far.

Got action frame from this client.

Received EAPOL EAPPKT from mobile ec:85:2f:15:39:32

Received Identity Response (count=1) from mobile
ec:85:2f:15:39:32

Processing Access-Challenge for mobile ec:85:2f:15:39:32

Sending EAP Request from AAA to mobile ec:85:2f:15:39:32
(EAP Id 2)

Received EAPOL EAPPKT from mobile ec:85:2f:15:39:32

Received EAP Response from mobile ec:85:2f:15:39:32
(EAP Id 2, EAP Type 25)


--- The client is validated/authenticated by the RADIUS Server.

Creating a PKC PMKID Cache entry for station
ec:85:2f:15:39:32 (RSN 2)

Resetting MSCB PMK Cache Entry 0 for station
ec:85:2f:15:39:32

Setting active key cache index 8 ---> 8

Setting active key cache index 8 ---> 0

Adding BSSID 84:78:ac:f0:68:d6 to PMKID cache at index 0
for station ec:85:2f:15:39:32

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.628: New PMKID: (16)

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.628:
[0000] 52 b8 8f cf 50 a7 90 98 2b ba d6 20 79 e4 cd f9

Created PKM Cache Entry for TGr AKM:802.1x ec:85:2f:15:39:32

--- WLC creates a PKM cache entry for this client, which is
used for FT with 802.1X in this case, so the PMKID is
computed with the AP MAC address (BSSID 84:78:ac:f0:68:d6).

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.629:
R1KH-ID:3c:ce:73:d8:02:00  MSK Len:48 pmkValidTime:1807

--- The R0KH-ID and R1KH-ID are defined, as well as the PMK cache validity period.

PMK sent to mobility group

--- The FT PMK cache entry for this client is shared with the WLCs on the mobility group.

Sending EAP-Success to mobile ec:85:2f:15:39:32 (EAP Id 12)

Found an cache entry for BSSID 84:78:ac:f0:68:d6 in PMKID cache at index 0 of station ec:85:2f:15:39:32

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.630: Including PMKID in M1 (16)

--- The hashed PMKID is included on the Message-1 of the initial FT 4-Way handshake.

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.630:
[0000] 52 b8 8f cf 50 a7 90 98 2b ba d6 20 79 e4 cd f9

Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state INITPMK (message 1), replay counter 00.00.00.00.00.00.00.0

--- Message-1 of the FT 4-Way handshake is sent from the WLC/AP to the client.

Received EAPOL-Key from mobile ec:85:2f:15:39:32

Sending EAPOL-key in PTK_START state (message 2) from mobile ec:85:2f:15:39:32

--- Message-2 of the FT 4-Way handshake is received successfully from the client.

Calculating PMKR0Name

--- The PMKR0Name is calculated.

DOT11R: Sending cache add

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.639: Adding MDIE, ID is:0xaaf0

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.639: Adding TIE for reassociation deadtime:20000 milliseconds

*Dot1x_NW_MsgTask_2: Jun 27 19:25:23.639: Adding TIE for R0Key-Data valid time :1807

Sending EAPOL-Key Message to mobile ec:85:2f:15:39:32 state PTKINITNEGOTIATING (message 3), replay counter 00.00.00.00.00.00.00.01

--- After the MDIE, TIE for reassociation deadtime, and TIE for R0Key-Data valid time are calculated, the Message-3 of this FT 4-Way handshake is sent from the WLC/AP to the client with this information.

Received EAPOL-Key from mobile ec:85:2f:15:39:32

Received EAPOL-key in PTKINITNEGOTIATING state (message 4) from mobile ec:85:2f:15:39:32

--- Message-4 (final message) of this initial FT 4-Way handshake is received successfully from the client, which confirms the installation of the derived keys. They can now be used in order to encrypt data frames with the current AP.

With 802.11r, the initial association to the WLAN is the basis used in order to derive the base keys used by this technique, just as in the other fast-secure roaming methods. The main differences come when the client begins to roam; FT not only avoids 802.1X/EAP when this is used, but it actually performs a more efficient roaming method that combines the initial 802.11 Open System Authentication and Reassociation frames (which are always used and required when roaming between APs) in order to exchange FT information and derive new dynamic encryption keys in place of the 4-Way handshake.

The next capture shows the frames exchanged when a Fast BSS Transition Over-the-Air with 802.1X/EAP security is performed. The Open System Authentication frame from the client to the AP is selected in order to see the FT protocol Information Elements that are required to begin the FT key negotiation. This is used in order to derive the new PTK with the new AP (based on the PMK-R1). The field that shows the authentication algorithm is highlighted in order to show that this client does not perform a simple Open System Authentication, but a Fast BSS Transition:

Here are the debug outputs from the WLC when this FT roaming event occurs with 802.1X/EAP:

```
Doing preauth for this client over the Air
```
--- WLC begins FT fast-secure roaming over-the-Air with this client and performs a type of preauthentication, because the client asks for this with FT on the Authentication frame that is sent to the new AP over-the-Air (before the Reassociation Request).

   Doing local roaming for destination address 84:78:ac:f0:2a:96
--- WLC performs the local roaming event with the new AP to which the client roams.

   Got 1 AKMs in RSNIE
   RSNIE AKM matches with PMK cache entry :0x3
--- WLC receives one PMK from this client (known as AKM here), which matches the PMK cache entry hold for this client.

   Created a new preauth entry for AP:84:78:ac:f0:2a:96
*apfMsConnTask_2: Jun 27 19:25:48.751: Adding MDIE,
   ID is:0xaaf0
--- WLC creates a new preauth entry for this AP-and-Client pair, and adds the MDIE information.

*apfMsConnTask_2: Jun 27 19:25:48.763: Processing assoc-req
thread:144bef38
   Reassociation received from mobile on BSSID 84:78:ac:f0:2a:96
--- Once the client receives the Authentication frame reply from the WLC/AP, the Reassociation request is sent, which is received at the new AP to which the client roams.

   Marking this mobile as TGr capable.

   Processing RSN IE type 48, length 38 for mobile ec:85:2f:15:39:32
   Roaming succeed for this client.
--- WLC confirms that the FT fast-secure roaming is successful for this client.

*apfMsConnTask_2: Jun 27 19:25:48.765: Sending assoc-resp
thread:144bef38
*apfMsConnTask_2: Jun 27 19:25:48.766: Adding MDIE,
   ID is:0xaaf0
   Including FT Mobility Domain IE (length 5) in reassociation assoc Resp to mobile
   Sending Assoc Response to station on BSSID 84:78:ac:f0:2a:96 (status 0) ApVapId 7 Slot 0
--- The Reassociation response is sent to the client, which includes the FT Mobility Domain IE.

   Finishing FT roaming for mobile ec:85:2f:15:39:32
--- FT roaming finishes and EAP is skipped (as well as any other key management handshake), so the client is ready to pass encrypted data frames with the current AP.

Skipping EAP-Success to mobile ec:85:2f:15:39:32

Here is a capture that shows a Fast BSS Transition Over-the-Air with WPA2-PSK security, where the final Reassociation Response frame from the AP to the client is selected in order to show more details about this FT exchange:

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>BSSID</th>
<th>Protocol</th>
<th>Channel frequency</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000000</td>
<td>Apple_15:39:32</td>
<td>cisco_f0:2a:94</td>
<td>84:78:ac:f0:2a:94</td>
<td>802.11</td>
<td>2437</td>
<td>Auth</td>
</tr>
<tr>
<td>2</td>
<td>0.004548</td>
<td>cisco_f0:2a:94</td>
<td>Apple_15:39:32</td>
<td>84:78:ac:f0:2a:94</td>
<td>802.11</td>
<td>2437</td>
<td>Auth</td>
</tr>
<tr>
<td>3</td>
<td>0.009178</td>
<td>Apple_15:39:32</td>
<td>cisco_f0:2a:94</td>
<td>84:78:ac:f0:2a:94</td>
<td>802.11</td>
<td>2437 Reass</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.016183</td>
<td>cisco_f0:2a:94</td>
<td>Apple_15:39:32</td>
<td>84:78:ac:f0:2a:94</td>
<td>802.11</td>
<td>2437 Reass</td>
<td></td>
</tr>
</tbody>
</table>

---

IEEE 802.11 wireless LAN management frame

- **Tagged parameters (274 bytes):**
  - **Tag**: Supported Rates (1(B), 2(B), 5.5(B), 11(B), 6, 9, 12, 18, [Mbit/sec]
  - **Tag**: Extended Supported Rates (24, 36, 48, 54, [Mbit/sec]
  - **Tag**: HT Capabilities (802.11n D1.10)
  - **Tag**: HT Information (802.11n D1.10)
  - **Tag**: Vendor Specific: Microsoft: WMM/WME: Parameter Element
  - **Tag**: RSN Information
    - **Tag Number**: RSN Information (48)
    - **Tag length**: 38
    - **RSN Version**: 1
    - **Group Cipher Suite**: 00-0f-ac (Ieee8021) AES (CCM)
    - **Pairwise Cipher Suite Count**: 1
    - **Pairwise Cipher Suite List 00-0f-ac (Ieee8021) AES (CCM)
    - **Auth Key Management (AKM) Suite Count**: 1
    - **Auth Key Management (AKM) List 00-0f-ac (Ieee8021) FT using PSK
    - **RSN Capabilities**: 0x0028
    - **PMKID Count**: 1
    - **PMKID List**
      - **PMKID**: 7e370d965e054df50819b135f3e7c3424
  - **Tag**: Mobility Domain
    - **Tag Number**: Mobility Domain (54)
    - **Tag length**: 3
    - **Mobility Domain Identifier**: 0x00a
    - **FT Capability and Policy**: 0x00
    - **…. D. = Resource Request Protocol Capability**: 0x00
  - **Tag**: Fast BSS Transition
    - **Tag Number**: Fast BSS Transition (55)
    - **Tag length**: 133
    - **MIC Control**: 0x0000
    - **MIC**: 1d49a4b8d4d1b2f633e4695f9ee90b1256
    - **AP addresses**: f7e249a836e49a1b5e5e5f21b2a3b9b
    - **Subelement ID**: PMK-R1 key holder identifier (R1KH-ID) (1)
      - **Length**: 6
      - **PMK-R1 key holder identifier (R1KH-ID): 3ce73d80200
      - **Subelement ID**: PMK-R0 key holder identifier (R0KH-ID) (3)
      - **Length**: 4
      - **PMK-R0 key holder identifier (R0KH-ID): \254\100\036\006\375
      - **Subelement ID**: GTK subelement (2)
      - **Length**: 33
      - **Key Info**: 0x0002
      - **…. 10 = Key ID**: 2
      - **Key Length**: 0x10
      - **RSC**: 0000000000000000
      - **GTK**: 6487b855f7dc16749e3b3c487cb130d0fcf234a1be851
Here are the debug outputs when this FT roaming event occurs with PSK, which are similar to the ones when 802.1X/EAP is used:

Doing preauth for this client over the Air

!--- WLC begins FT fast-secure roaming over-the-Air with this client and performs a type of preauthentication, because the client asks for this with FT on the Authentication frame that is sent to the new AP over-the-Air (before the Reassociation Request).

Doing local roaming for destination address 84:78:ac:f0:2a:96

!--- WLC performs the local roaming event with the new AP to which the client roams.

Got 1 AKMs in RSNIE
RSNIE AKM matches with PMK cache entry :0x3

!--- WLC receives one PMK from this client (known as AKM here), which matches the PMK cache entry hold for this client.

Created a new preauth entry for AP:84:78:ac:f0:2a:96
*apfMsConnTask_2: Jun 27 19:25:48.751: Adding MDIE, ID is:0xaaf0

!--- WLC creates a new preauth entry for this AP-and-Client pair, and adds the MDIE information.

*apfMsConnTask_2: Jun 27 19:25:48.763: Processing assoc-req
thread:144bef38
Reassociation received from mobile on BSSID 84:78:ac:f0:2a:96

!--- Once the client receives the Authentication frame reply from the WLC/AP, the Reassociation request is sent, which is received at the new AP to which the client roams.

Marking this mobile as TGr capable.

Processing RSN IE type 48, length 38 for mobile ec:85:2f:15:39:32

Roaming succeed for this client.

!--- WLC confirms that the FT fast-secure roaming is successful for this client.

*apfMsConnTask_2: Jun 27 19:25:48.765: Sending assoc-resp
thread:144bef38
*apfMsConnTask_2: Jun 27 19:25:48.766: Adding MDIE, ID is:0xaaf0
Including FT Mobility Domain IE (length 5) in reassociation assoc Resp to mobile
Sending Assoc Response to station on BSSID 84:78:ac:f0:2a:96
(status 0) ApVapId 7 Slot 0

--- The Reassociation response is sent to the client, which includes the FT Mobility Domain IE.

Finishing FT roaming for mobile ec:85:2f:15:39:32

--- FT roaming finishes and EAP is skipped (as well as any other key management handshake), so the client is ready to pass encrypted data frames with the current AP.

Skipping EAP-Success to mobile ec:85:2f:15:39:32

As shown in the capture, once the Fast BSS Transition is negotiated upon initial association to the WLAN, the four frames that are used and required for roaming (Open System Authentication from the client, Open System Authentication from the AP, Reassociation Request, and Reassociation Response) are basically used as an FT 4-Way handshake in order to derive the new PTK (unicast encryption key) and GTK (multicast/broadcast encryption key).

This substitutes for the 4-Way handshake that normally occurs after these frames are exchanged, and the FT content and key negotiation on these frames is basically the same whether you use 802.1X/EAP or PSK as the security method. As shown in the capture, the AKM field is the main difference, which confirms if the client performs FT with PSK or 802.1X. Therefore, it is important to note that these four frames normally do not have this type of security information for key negotiation, but only when client FT roams if 802.11r is implemented and negotiated between the client and the WLAN infrastructure upon initial association.

**Fast BSS Transition Over-the-DS**

802.11r allows another implementation of Fast BSS Transition, where the FT roaming is initiated by the client with the new AP for which the client roams Over-the-DS (Distribution System), and not Over-the-Air. In this case, FT Action frames are used in order to initiate the key negotiation instead of the Open System Authentication frames.

Basically, once the client decides it might roam to a better AP, the client sends an FT Action Request frame to the original AP where it is currently connected before roaming. The client indicates the BSSID (MAC address) of the target AP where it wants to FT roam. The original AP forwards this FT Action Request frame to the target AP over the Distribution System (normally the wired infrastructure), and the target AP responds to the client with an FT Action Response frame (also over the DS, so it can finally send it over-the-air to the client). Once this FT Action frame exchange is successful, the client finishes the FT roaming; the client sends the Reassociation Request to the target AP (this time over-the-air), and receives a Reassociation Response from the new AP in order to confirm the roaming and final keys derivation.

In summary, there are four frames to negotiate Fast BSS Transition and derive new encryption keys, but here the Open System Authentication frames are substituted with the FT Action Request/Response frames, which are exchanged with the target AP over the Distribution System with the current AP. This method is also valid for both security methods 802.1X/EAP and PSK, all supported by the Cisco Wireless LAN Controllers; however, since this Over-the-DS transition is not supported and implemented by most of the wireless clients in the WiFi industry (and since the frame exchange and debug outputs are basically the same), examples are not provided in this document. Instead, this image is used in order to visualize the Fast BSS Transition Over-the-DS:
FlexConnect with 802.11r

- Central Authentication is supported. This includes Local and Central data switching. The APs must be part of the same FlexConnect Group.
- Local Authentication is not supported.
- Standalone mode is not supported.

Pros with 802.11r

- This method is the first that uses a key hierarchy clearly defined by the IEEE on the 802.11 standard as an amendment (802.11r), so the implementation of these FT techniques are more compatible between vendors and without different interpretations.
- 802.11r allows multiple techniques that are helpful, dependent on your needs (Over-the-Air and Over-the-DS, for 802.1x/EAP security and for PSK security).
- The wireless client performs fast-secure roaming to a new AP on the same WLAN/SSID, even if it never associated with that AP, and without the need to save multiple PMKIDs.
- This is the first fast-secure roaming method that allows faster roaming even with PSK security, and avoids the 4-Way handshake that is required when roaming between APs with WPA/WPA2 PSK. The main purpose of the fast-secure roaming methods is to avoid the 802.1X/EAP handshake when this security method is implemented; however, for PSK security the roaming event is accelerated even more with 802.11r by avoiding the 4-Way handshake.

Cons with 802.11r

- There are a few wireless client devices that actually support Fast BSS Transitions, and in
most cases, they do not support all of the techniques available on 802.11r.

- Because of the fact that these implementations are very young, there are not enough test results from real-production environments or enough debug results in order to address possible caveats that might appear.
- When you configure a WLAN/SSID in order to use any of the FT methods, then only wireless clients that support 802.11r are able to connect to this WLAN/SSID. The FT settings are not optional for the clients, so those wireless clients that do not support 802.11r must connect with a separate WLAN/SSID where FT is not configured at all.

Adaptive 802.11r

- Some legacy clients cannot associate with a WLAN/SSID that has 802.11r enabled even for "mixed mode" (which you hope you can have on the same SSID clients that support and that do not support 802.11r). This is when the driver of the client supplicant that is responsible for parsing the Robust Security Network Information Element (RSN IE) is old and not aware of the additional AKM suites in the IE. Due to this limitation, clients cannot send association requests to WLANs that advertise 802.11r support, and hence, you will probably need to configure one WLAN/SSID for 802.11r clients and a separate WLAN/SSID for clients that do not support 802.11r.
- In order to overcome this, Cisco Wireless LAN infrastructure introduced the Adaptive 802.11r feature. When FT mode is set to Adaptive at the WLAN level, WLAN advertises 802.11r Mobility Domain ID on an 802.11i-enabled WLAN. Some Apple iOS10 client devices identify the presence of MDIE on an 80211i/WPA2 WLAN and do a proprietary handshake in order to establish 802.11r association. Once the client completes successful 802.11r association, it will be able to do FT roaming as in a normal 802.11r enabled WLAN. The FT Adaptive is applicable only to selected Apple iOS10 (and later) devices. All other clients will continue to have 802.11i/WPA2 association on the WLAN, and perform the applicable FSR method as supported.
- More documentation about this new feature introduced for iOS10 devices to perform 802.11r on a WLAN/SSID where 802.11r is not truly enabled (so other non-802.11r clients will successfully connect), can be found in Enterprise Best Practices for iOS Devices on Cisco Wireless LAN.

Conclusions

- Keep in mind that the client is always the one that decides to roam to a specific AP, and the WLC/AP cannot decide this for the client. The roaming event is initiated by the wireless client once it considers it should roam.
- The WLC supports a combination of most or all of the FSR (Fast-Secure Roaming) methods together on the same WLAN/SSID. However, be aware that this normally does not work, as it depends highly on the client behavior (very different across different mobile devices) in order to support or even understand that which the WLC attempts to advertise as supported. Instead of achieving interoperability in just one SSID, there are normally more issues than the ones that are expected to be fixed, so this is not recommended. Deep testing with all possible clients to be used on this WLAN should be completed if this is really needed.
- It is very important to understand that fast-secure roaming methods are developed in order to accelerate the WLAN roaming process when you move between APs if the WLAN/SSID has
security enabled. When no security is in place, there is nothing to accelerate, as the client-AP simply exchanges the wireless management frames that are always required when roaming between APs before data frames are sent (Open System Authentication from the client, Open System Authentication from the AP, Reassociation Request, and Reassociation Response). Therefore, this cannot move any faster. If you encounter roaming issues without security, then there are no fast-roaming methods to improve roaming, only methods in order to confirm if the WLAN/SSID setup and design are appropriate for the wireless client stations to roam accordingly between the AP coverage cells.

- **802.11r/FT** is implemented with WPA2-PSK in order to accelerate roaming events with this security by avoiding the 4-Way handshake, as explained within the 802.11r section.

- All of the methods have their advantages and disadvantages, but in the end, you must always verify if the wireless client stations support the specific method that you want to implement, and if the Cisco WLAN infrastructure supports all of the methods available. Thus, you must select the best method that is actually supported by the wireless clients that connect to the specific WLAN/SSID. For example, in some deployments you might create a WLAN/SSID with CCKM for Cisco wireless IP Phones (which support WPA2/AES with CCKM, but not 802.11r), and then another WLAN/SSID with WPA2/AES via 802.11r/FT for wireless clients that support this Fast Secure Roaming method (or use OKC, if this is what is supported).

- If the wireless clients do not support any of the fast-secure roaming methods available, then you might need to accept the fact that those clients will always experiment the delays explained in this document when roaming between APs on a WLAN/SSID with 802.1X/EAP security (which can cause disruptions on the client apps/services).

- All methods, except SKC (WPA2 PMKID Caching), are supported for fast-secure roaming between APs managed by different WLCs (intercontroller roaming), as long as they are on the same mobility group.

- CUWN fully supports all different Fast-Secure Roaming methods covered in this article when using 802.1X/EAP authentication for WPA/WPA2. CUWN does not support Fast-Secure Roaming on methods that work with WPA2-RSN (CCKM, PMKID Caching/SKC, OKC/PKC) when using PSK (WPA2-Personal), where Fast-Roaming methods are mostly not needed. However, CUWN supports Fast-Secure Roaming in the case of WPA2-FT (802.11r) with PSK as also explained in this article.

**Related Information**

- [Cisco Fast Secure Roaming Application Note](#)
- [Technical Support & Documentation - Cisco Systems](#)