Technical Guide

OpenVPN

Feature Overview and Configuration Guide

Introduction

This guide describes AlliedWare Plus™ OpenVPN and its configuration.

AlliedWare Plus OpenVPN provides a seamless, secure, and easy means for employees to have access to the same resources whether they are inside or outside their company premises. Staff members have the ability to work securely from remote locations such as from home or when on business trips.

Products and software version that apply to this guide

This guide applies to AlliedWare Plus products that support OpenVPN, running version **5.4.5** or later.

To see whether a product supports OpenVPN, see the following documents:

- The product's Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com.

Feature support may change in later software versions. For the latest information, see the above documents.

The following features are supported since the following software versions:

- From 5.5.0-2.1 onwards, OpenVPN is supported on AR1050V
- From 5.5.1-0.1 onwards, you can set a minimum TLS version
- From 5.5.2-0.1 onwards, TLS-CRYPT is supported.
- From 5.5.2-1.1 onwards, 2FA is supported.
- From version 5.5.3-0.1 onwards Two-factor Authentication (2FA) for OpenVPN has been enhanced with the following features:

  - OpenVPN certificate authentication

  - 2FA self-registration

  - 2FA email One-Time Password (OTP)

Contents

What is OpenVPN?

AlliedWare Plus OpenVPN is a Secure Sockets Layer/Transport Layer Security (SSL/TLS)-based application used for creating a secure connection from a remote client to a central site. It establishes an encrypted and authenticated tunnel between the client and server and uses that tunnel for transporting traffic across intervening networks. AlliedWare Plus OpenVPN provides a full Data Link Layer access, proven standards-based SSL/TLS authentication and encryption, and implicit firewall/ NAT traversal.

AlliedWare Plus OpenVPN is built on a solid and industry-tested security foundation and is very easy to use. It offers you the flexibility to work in a variety of modes that are easy to understand and hard to make insecure.

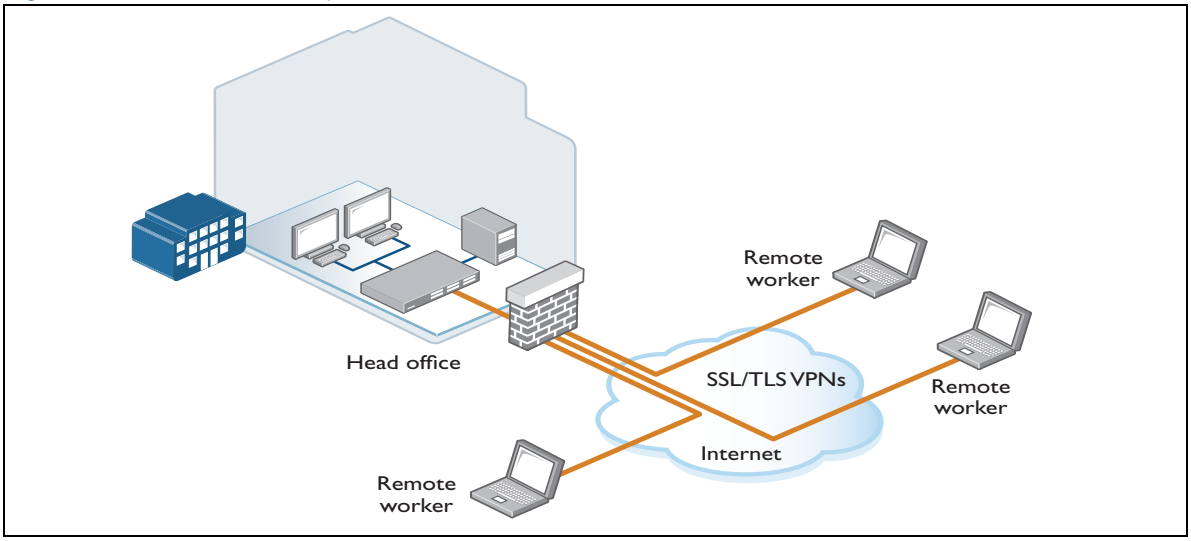
AlliedWare Plus OpenVPN provides the following key features:

- Protection of IPv4 and IPv6 traffic over TLS tunnel
- Configurable listening UDP port
- A maximum of 100 to 200 OpenVPN clients can concurrently connect, depending on the AlliedWare Plus firewall or router. See your product's Datasheet for more information.
- Group network access control based on 802.1Q tagged interfaces allows one or more clients to be associated with up to 64 VLANs.
- Server authentication using certificates, client authentication via RADIUS over IPv4/ IPv6.
- Virtual Tunnel Interface for OpenVPN tunnels
- Single OpenVPN tunnel interface
- IPv4 and IPv6 as a delivery protocol
- Support for:

  - TAP mode (Layer 2 based Ethernet TAP) and

  - TUN mode (Layer 3 based IP tunnel)

Figure 1: AlliedWare Plus OpenVPN



About OpenVPN TAP mode

The purpose of Terminal Access Point (TAP) mode is to enable the remote client to operate as though it were directly connected to the LAN that lies behind the OpenVPN server. TAP mode operates at the data link layer (Layer 2).

Effectively, the OpenVPN connection operates as though it were a virtual Network Interface Card (NIC) card in the client, connected to the LAN in behind the OpenVPN server. So, the OpenVPN connection operates like a Network Tap that sits on that central LAN. It transfers packets from that LAN over a tunnel to the remote client (and transports packets back from the remote client).

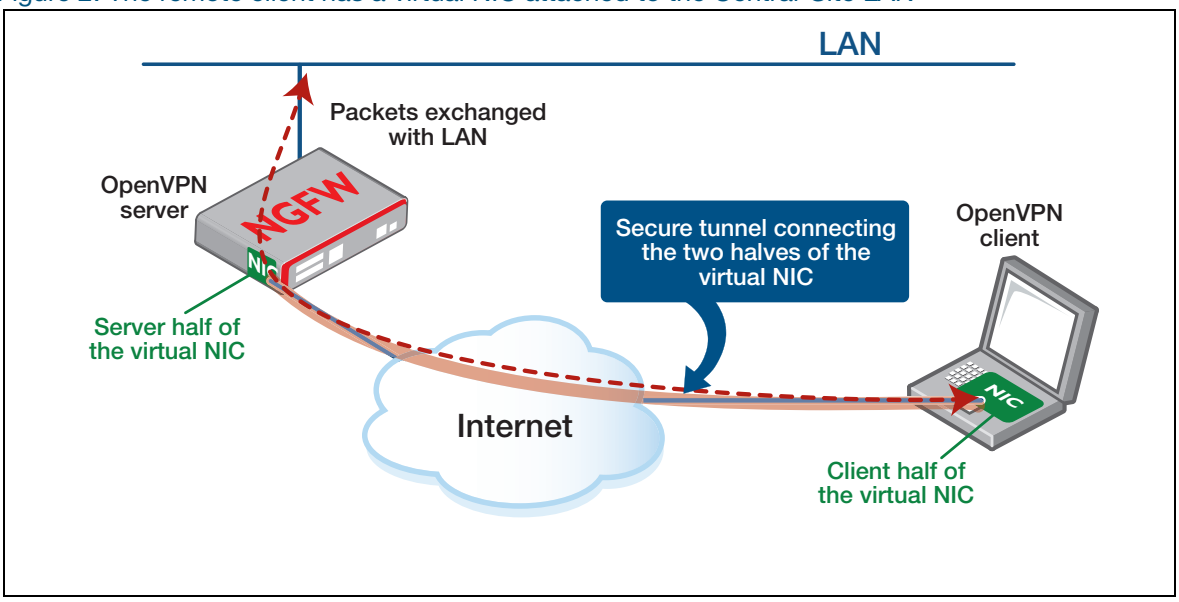
The full content of the Ethernet frames to/from the client are encapsulated in the tunnel and transported between the client and the server halves of the virtual NIC. Therefore, it appears to the client that it has received the frames directly off the central-site LAN, and appears to the central-site LAN that the client is directly connected to that LAN.

Within the OpenVPN server, the TAP appears as a Virtual Tunnel Interface (VTI) that carries **Layer 2** frames.

Note: Because the TAP encapsulates the full Ethernet frames, it can be used for transporting protocols other then IP, for example IPX or AppleTalk. It also means that any communication that relies on the exchange of broadcast or multicast packets will work seamlessly. In particular, the remote client will be able to obtain an IP address by DHCP from a DHCP server on the central LAN.

As well as enabling a remote client to appear to be connected to the central LAN, TAP mode can also be used to create a bridge connection to unite two LANs that are at separate locations.

Figure 2: The remote client has a virtual NIC attached to the Central-Site LAN



Because the TAP acts as a NIC attached to the central LAN, it will transport all that LAN's broadcast frames across the tunnel. This potentially adds extra traffic to the tunnel, transporting broadcasts that the remote client may not even be interested in. The TAP also adds the overhead of Ethernet headers on all packets transported over the VPN tunnel.

About OpenVPN TUN mode

TUN is also a virtual network device. TUN mode operates at the network layer (Layer 3), and creates a Virtual Tunnel Interface (VTI) that carries **Layer 3** packets. So, rather than encapsulating the full Ethernet frames, it takes the IP content of the frames, and routes that content via the tunnel.

You can also use TUN to:

- transport traffic that is destined for the VPN client
- transport only Layer 3 packets
- support VPN on mobile devices.

Note: TUN cannot be used in bridges, and broadcast traffic is not transported in TUN mode.

With TUN mode, the VPN connection appears as an IP interface on the remote client, and the AlliedWare Plus firewall or router acts as the Gateway for routing the VPN traffic to other networks.

In summary, TUN mode is like creating a secure tunnel for IP traffic between two devices or networks over the Internet, while TAP mode is like connecting two networks together as if they were on the same local network.

RADIUS attributes supported by OpenVPN

When RADIUS is used for client authentication, there are several attributes that can be configured on the RADIUS server for each user. These attributes provide a mechanism for shaping the remote user's network configuration when accessing the network via VPN so that they have a similar connectivity experience as they would have if directly connected to the central site LAN.

The following attributes are supported by OpenVPN:

ID	ATTRIBUTE	TYPE	SPECIFICATION	EXAMPLE	USAGE
1	User-Name	string	RFC2865	"foo"	Client username.
2	Password	string	RFC2865	"bar"	Client password.
6	Service-Type	integer	RFC2865	8 = Authenticate Only	OpenVPN requests login only to the RADIUS server.
8	Framed-IP-Address	ipaddr	RFC2865	10.10.10.50	IP address to be pushed to the client.
9	Framed-IP-Netmask	ipaddr	RFC2865	255.255.255.0	IP netmask to be pushed to the client
22	Framed-Route	string	RFC2865	"10.10.11.0/8 10.10.10.1 1"	Route to be pushed to the client.
MS-28	Microsoft-Primary-DNS-Server	ipaddr	RFC2548	10.10.10.1	Primary DNS to push to client. If multiple primary DNS servers are provided, only the first one will be used.
MS-29	Microsoft-Secondary-DNS-Server	ipaddr	RFC2548	10.10.10.2	Secondary DNS to push to client. If no primary address provided, this will be ignored.
97	Framed-IPv6-Prefix	ipv6prefix	RFC3162	"fc00:2::2/64"	IPv6 prefix to be pushed to the client.

ID	ATTRIBUTE	TYPE	SPECIFICATION	EXAMPLE	USAGE
169	DNS-Server-IPv6-Address	ipv6addr	RFC6911	"fc00:2::1“	IPv6 DNS address to be pushed to the client (without NH).
170	Route-IPv6-Information	ipv6prefix	RFC6911	"fc00:3::/64“	IPv6 route to be pushed to the client.
64	Tunnel-Type	integer	RFC3580	13 = VLAN	Client VLAN assignment. Tag the client traffic if 802.1Q tagging is configured (TAP mode only).
65	Tunnel-Medium-Type	integer	RFC3580	6 = 802	Client VLAN assignment. Tag the client traffic if 802.1Q tagging is configured (TAP mode only).
81	Tunnel-Private-Group-Id	string	RFC3580	"20" = VLANID 20	Client VLAN assignment. Tag the client traffic if 802.1Q tagging is configured (TAP mode only).

Configuring the AlliedWare Plus OpenVPN server

To configure the OpenVPN server on an AlliedWare Plus Firewall, you need to:

- Configure the RADIUS server and corresponding attributes in AlliedWare Plus for client authentication. Various attributes can be set on the RADIUS server for each user, influencing the remote user's network configuration during VPN access for a similar experience to a direct connection to the central site's LAN. Use an external or built-in internal RADIUS server with AlliedWare Plus.
- Configure the OpenVPN service and tunnel, see page 10.
- Generate certificates for OpenVPN clients. AlliedWare Plus supports external CA certificates or a built-in local CA. Optionally, use these certificates for the AlliedWare Plus server to authentication the clients, see page 12.
- Configure the OpenVPN client, see page 14.

Configure the RADIUS server

The configuration below establishes the RADIUS service to provide the user database and related attributes for OpenVPN client connection. There are two configuration options: Local and external.

Option 1: Configure the local user group and attributes

AlliedWare Plus firewalls and routers include a built-in local RADIUS server, which can provide the authentication service for OpenVPN. For more information about how to configure the local RADIUS server, see the Local RADIUS Server Feature Overview Guide.

1. **Enable the local RADIUS server**

```
awplus(config)# radius-server local
awplus(config-radsrv)# nas 127.0.0.1 key awplus-local-radius-server
awplus(config-radsrv)# server enable
```

2. **Configure the user group and attributes**

Create a user group as a container for the attributes so it can be assigned to user accounts. You may use the attribute command multiple times to configure multiple entries. For example, you can configure multiple entries of `attribute Framed-Route "<ip segment> <gateway>"` to allow OpenVPN to push multiple route entries to the client. For more information about OpenVPN attributes, see "RADIUS attributes supported by OpenVPN" on page 6.

```
awplus(config-radsrv)# group <group-name>
awplus(config-radsrv-group)# attribute <attribute-item>
awplus(config-radsrv-group)# exit
```

3. Configure users

Use the **user** command to create user accounts and assign them to the user group. Clients logging in to the OpenVPN server will get the attribute information from their user group. You can create multiple users and user groups for complex environments.

```
awplus(config-radsrv)# user <username> password <password> group <group-name>
awplus(config-radsrv)# exit
```

4. Configure local RADIUS

Specify the local RADIUS server host.

```
awplus(config)# radius-server host 127.0.0.1 key awplus-local-radius-server
awplus(config)# aaa authentication openvpn default group radius
```

Option 2: Configure an external RADIUS server

AlliedWare Plus can use an external RADIUS server, such as a FreeRADIUS server running in a Linux-based system.

1. Configure an external RADIUS server

```
awplus(config)#radius-server host {<hostname>| <ip-address>}
[auth-port <port-number>] [acct-port <port-number>] [timeout <seconds>]
[retransmit <retries>] [key <string>]

awplus(config)#aaa group server radius <radius-server-group>

awplus(config-sg)#server {<hostname>|<ip-address>} [auth-port <port-number>]
[acct-port <port-number>]

awplus(config-sg)#exit

awplus(config)#aaa authentication openvpn default group <radius-server-group>
```

For more information about how to use the external RADIUS server and to use the RADIUS server over TLS, see the RADIUS Feature Overview and Configuration Guide.

Configure the OpenVPN service and Virtual Tunnel Interface (VTI)

A Virtual Tunnel Interface (VTI) has similar characteristics to any other interface on the device. It does not directly map to any of the physical interfaces, but instead is actually the endpoint of a tunnel from another device. A VTI is often employed to isolate the actual tunnel from the traffic to be tunneled. It appears as a standard interface on the device, eliminating the need for complex ACLs or filters. This allows you to use standard routing protocols (or static routing) to guide the relevant traffic through the tunnel.

In this section, you will create a VTI for the OpenVPN router to be accessed by the client.

1. **Create the VTI and enter VTI configuration mode**

```
awplus(config)# interface tunnel<tunnel-id-number>
```

2. **Configure tunnel mode**

Enable the OpenVPN TAP or TUN mode. See "About OpenVPN TAP mode" on page 5 and "About OpenVPN TUN mode" on page 6.

```
awplus(config-if)# tunnel mode openvpn {tap|tun}
```

3. **Assign an IP address for the tunnel or bridge**

This step will create a routed VPN by adding an IP address and network mask to the tunnel. It internally adds a firewall rule to deny clients with IP addresses that don't belong to the tunnel subnet. You can use this IP address as the gateway for the ***Framed-Route*** attribute.

```
awplus(config-if)# ip address <ip-address/mask>
```

Or you can use the **bridge** command to create a virtual Ethernet bridge to bind the tunnel interface with a physical interface. The bridge will have two ports. One is the physical port that is connected to the LAN network, and the other is the tunnel interface that the virtual OpenVPN NIC will connect to.

```
awplus(config)# bridge <1-255>
awplus(config)# interface <physical-interface>
awplus(config-if)# bridge-group <bridge-id>
awplus(config-if)# exit
awplus(config)# interface tunnel<tunnel-id>
awplus(config-if)# bridge-group <bridge-id>
awplus(config-if)# exit
awplus(config)# interface br<bridge-id>
awplus(config-if)# ip address <ip-address/mask>
```

4. **Additional OpenVPN settings (optional)**

You can customize several OpenVPN settings in VTI configuration mode:

- Use the **tunnel openvpn authentication** command to configure the data channel authentication digest for the OpenVPN tunnel. The authentication digest can be set to use Secure Hash Standard with 160-bit digest size (sha1) or 256-bit digest size (sha256). You need to configure clients to use the same algorithms as configured on the server.
- Use the **tunnel openvpn cipher** command to configure the data channel encryption cipher for the OpenVPN tunnel. The encryption cipher can be set to use Advanced Encryption Standard symmetric key block cipher with 128-bit key (aes128) or 256-bit key (aes256). You need to configure clients to use the same algorithms as configured on the server.
- By default, client keys are renegotiated after 3600 seconds. You can change this by using the **tunnel openvpn expiry-seconds** command in the desired tunnel. You can disable the key renegotiate by setting the 'disable' parameter.
- If you intend to use greater than 100 concurrent tunnels on the AlliedWare Plus device, we recommend you change to data-usage-based renegotiation per VTI, instead of timer-based renegotiation. This means each VPN tunnel will independently renegotiate the key once it reaches the data limit. This prevents all tunnels from renegotiating at the same time. Use the **tunnel openvpn expiry-kbytes** command to configure the data-usage-based renegotiation.
- Use the **tunnel openvpn port** command to change the listening port for multiple concurrent tunnels. The default port is 1194.

Using 802.1Q VLAN tagging in TAP mode

You can use 802.1Q VLAN tagging on OpenVPN connections to keep traffic from different clients separate. The 802.1Q function only works in TAP mode.

When tagging/untagging is enabled, the OpenVPN server tags all client-originated packets as they exit from the tunnel, and untags all client-destined packets as they enter the tunnel. The VLAN ID (VID) inserted in the tag is the client VID obtained, if present, from the RADIUS server, or otherwise is the default VID. Untagged packets destined to the client are dropped. The VID is a number in the range 1 to 4094.

When VLAN tagging/untagging is disabled, OpenVPN server performs no tagging or untagging, and OpenVPN tunnels are then transparent to tagged packets.

1. **Configure 802.1q in tunnel interface**

```
awplus(config-if)# encapsulation dot1q <vid>
awplus(config-if)# tunnel openvpn tagging <vid>
```

2. **Set the VLAN tag in the local RADIUS group**

You also need to assign the VLAN tag to the local RADIUS server group, or configure the Tunnel-Type, Tunnel-Medium-Type, and Tunnel-Private-Group-Id in the external RADIUS server user database.

To assign it to the local RADIUS server group:

```
awplus(config-radsrv)# group <group-name>
awplus(config-radsrv-group)# vlan <vid>
```

3. **Tunnel sub-interface configuration**

After enabling tagging, you can give the tunnel sub-interface an IP address, to communicate with clients in the specified VLAN.

```
awplus(config)# interface tunnel<tunnel-id>.<vid>
awplus(config-if)# ip address <ip-address/mask>
```

Use a certificate to let clients verify the server's identity

Certificate authentication in OpenVPN is used to allow the clients to verify the identity of the server. The following steps describe how to export a certificate, so that the client can use it to verify the server. There are two options: use the local or a remote CA on the device.

Option 1: Using the local CA on the device

Your device includes a built-in local CA that can be enabled as the certificate issuer for OpenVPN server and clients. For more information about trustpoint configuration, see the PKI Feature Overview and Configuration Guide.

1. **Enable the local trustpoint**

```
awplus(config)# crypto pki trustpoint local
awplus(ca-config)# end
awplus# crypto pki enroll local
```

2. **Export the CA public certificate**

Use the **crypto pki export** command to export the CA public certificate as **pem** format, so that the OpenVPN client can use it to verify the certificate of the OpenVPN server. This command is entered from Privilege Exec mode and it does not appear in the device's running configuration. You can export the file to the flash file system or a TFTP server directory, or you can display its contents on the terminal.

```
awplus# crypto pki export local pem {<url>|terminal}
```

Option 2: Using a remote CA on the device

See the section “Creating a trustpoint based on an external Certificate Authority” in the PKI Feature Overview and Configuration Guide for step-by-step instructions.

The following steps show the minimum configuration required:

1. **Create the trustpoint**
2. **Authenticate it to the external CA**
3. **Create the Certificate Signing Request (CSR)**
4. **Export the CSR to the CA**
5. **Import the signed certificate back from the CA**

Note: When using a remote CA, client certificates are not generated on the AlliedWare Plus device.

Note that the trustpoint must contain a CA, server certificate, and server key. Following the procedure from the PKI Feature Overview and Configuration Guide will make sure of this.

Push entity routes to OpenVPN client

From AlliedWare Plus version 5.5.4-0.1 onwards, a device configured as an OpenVPN server can push route information based on specific entities. When an OpenVPN client establishes a connection, the IPv4 and IPv6 host addresses along with network subnets for that entity are pushed to the client. The client then routes matching traffic through the OpenVPN tunnel to the AlliedWare Plus device, using a configured IP as the gateway. Host addresses are sent as either a /32 IPv4 route or a /128 IPv6 route.

In the case of dynamic FQDN host addresses, when the OpenVPN client connects, all IP addresses known for that FQDN at that time will be pushed.

If the network subnets and host IPs change on the entity, then the client connections will not be updated. The clients will need to reconnect to get any new routes.

Entity network subnets that have an interface configured will not be used. This is because the interface restriction does not make sense for the OpenVPN client in this context.

Example configuration

```
zone openvpn_routes
 network internet
  host example
   ip address dynamic fqdn example.com
 network local
  ip subnet 172.16.0.0/16
  host example
    ip address 10.255.0.1
    ip address 10.255.0.64
!
interface tunnel0
 tunnel mode openvpn tap
 tunnel openvpn route openvpn_routes
 ip address 10.1.1.1/24
!
```

Note: DNS Relay cache and **ip domain-lookup via-relay** must be enabled for the **ip address dynamic fqdn *<domain-name>*** command to work. This is because DNS requests passing through the router are inspected for matching FQDNs. For more information about FQDN entities, see the: Application Awareness Feature Overview and Configuration Guide.

Configure the OpenVPN client

Create a .ovpn file

You can get an OpenVPN client from the official website openvpn.net, for Windows, Linux, MacOS, Android, and iOS. To setup an OpenVPN connection on the client, you need to create a ovpn file. A simple configuration file is shown below.

```
# Specify that this is a client
client

# Use the same network setting (tap or tun) as the Alliedware Plus server
dev tap

# Use either TCP or UDP as used by the Alliedware Plus server
proto udp

# Specify the hostname/IP address and port of the server.
# You can specify multiple remote entries for load balancing.
# You can omit the default port 1194.
remote 10.34.180.107 1194

# Specify a password as the client-side authentication method
auth-user-pass

# Use the same authentication algorithm as the server (SHA1 or SHA256)
auth SHA1

# Use the same cryptographic cipher as the server (AES-128-CBC or AES-256-CBC)
cipher AES-128-CBC

# Specify the CA certificate
ca C:/ca/cacert.pem

# Set the log file verbosity (0-7)
verb 3
```

Store the certificate

The above file has the path to the certificate: `ca C:/ca/cacert.pem`. Alternatively, you can store the contents of the .pem file directly in the .ovpn file.

Assign the client IP address

You need to assign the IP address for clients by specifying an attribute in the RADIUS server. The client IP address should be in the same network segment as the VPN tunnel or bridge address. You can use the **attribute Framed-Route** command multiple times, to push multiple route entries to the client. Use the RADIUS server attribute to assign an IP address, network, and route.

```
awplus(config-radsrv-group)# attribute Framed-IP-Address 192.168.1.2
awplus(config-radsrv-group)# attribute Framed-IP-Netmask 255.255.255.0
awplus(config-radsrv-group)# attribute Framed-Route "192.168.1.0/24
192.168.1.1"
```

You can also use DHCP to allocate an IP address automatically for clients, in TAP mode. The IP segment and default router address should match the IP address of the tunnel interface, bridge address, or 802.1q sub-interface address. The DHCP server can be located on the OpenVPN server device, in the same VLAN or as specified by the **ip helper-address** command for the VLAN or tunnel interface.

Use a DHCP server to assign an IP address, network, and route.

```
awplus(config)# ip dhcp pool <pool-name>
awplus(dhcp-config)# network <ip-address/mask>
awplus(dhcp-config)# range <lowest-address> <highest-address>
awplus(dhcp-config)# subnet-mask <mask>
awplus(dhcp-config)# default-router <router-address>
awplus(dhcp-config)# exit
awplus(config)# service dhcp-server
```

Set a minimum TLS version for OpenVPN

OpenVPN supports different TLS (Transport Layer Security) versions. This can be a security risk as the older TLS versions have known security flaws and were deprecated in 2020. From version 5.5.1-0.1 onwards, you can set a minimum TLS version for authenticating OpenVPN clients.

```
awplus(config)# interface <tunnel>
awplus(config-if)# tunnel openvpn tls-version-min {1.1|1.2|1.3}
```

For example, to set the minimum TLS version to 1.2, use the following commands:

```
awplus(config)# interface tunnel1
awplus(config-if)# tunnel openvpn tls-version-min 1.2
```

If you don’t specify a version, then the default TLS version of 1.0 is used.

Configure TLS Crypt on OpenVPN

From version 5.5.2-0.1 onwards, AlliedWare Plus firewalls and routers support TLS Crypt on OpenVPN. TLS Crypt uses a pre-shared key to secure the entire OpenVPN session from the first packet.

It provides several potential benefits:

- It prevents detection of the OpenVPN connection start, which is helpful in some situations when the OpenVPN protocol signature is detected and blocked.
- It prevents TLS denial of service attacks. DoS attacks are possible with TLS-Auth, where the attacker can open thousands of TLS connections simultaneously but not provide a valid certificate, jamming the available ports. With TLS Crypt the server would reject the connection up front.
- Data is encrypted twice, once by TLS Crypt and once by the TLS session.

To use TLS Crypt:

```
awplus(config)# interface <tunnel>
awplus(config-if)# tunnel openvpn tls-crypt <key-filename>
```

The key's filename starts with "flash:/" (e.g. flash:/openvpn.key). All clients and the server must share the same key file. TLS Crypt will automatically create the configured key file if it doesn't exist.

Monitor and troubleshoot your tunnels

Use the **show openvpn connections** command to check the OpenVPN user connection information:

```
awplus#show openvpn connections

Interface: tunnel0
                                        Rx       Tx
Username   Real Address                Bytes    Bytes     Connected Since
-------------------------------------------------------------------------------
test       ::ffff:10.33.22.15          2123465  1884560   Thu Apr  7 16:41:15
2021
```

Use the **show openvpn connections detail** command to see connection details:

```
awplus#show openvpn connections detail

Interface: tunnel0
 Username: user1
  Route:      192.168.20.0 255.255.255.0 192.168.10.2
  Address:    192.168.10.3 255.255.255.0
  DNS Server: 192.168.10.253
  DNS Server: 192.168.10.254
  VID:        20
 Username: user2
  VID:        100
```

Use the **show interface tunnel** command to check the status of OpenVPN interfaces:

```
awplus#show interface tunnel0

Interface tunnel0
  Link is UP, administrative state is UP
  Hardware is Tunnel
  IPv4 address 192.168.1.1/24 point-to-point 192.168.1.255
  index 21 metric 1 mtu 1419
  <UP,POINT-TO-POINT,RUNNING,MULTICAST>
  SNMP link-status traps: Disabled
  Bandwidth 1g
  Tunnel protocol/transport openvpn tun, listen port 1194
    cipher aes128, authentication sha1
    expiry-kbytes 0, expiry-seconds 3600
    tls-version-min 1.0
    tls-crypt Disabled
  Checksumming of packets disabled, DF bit set, path MTU discovery disabled
  Router Advertisement is disabled
  Router Advertisement default routes are accepted
  Router Advertisement prefix info is accepted
    input packets 0, bytes 0, dropped 0, multicast packets 0
    output packets 0, bytes 0, multicast packets 0, broadcast packets 0
    input average rate : 30 seconds 0 bps, 5 minutes 0 bps
    output average rate: 30 seconds 0 bps, 5 minutes 0 bps
  Time since last state change: 0 days 00:00:13
```

Two-Factor Authentication

Overview

Two-Factor Authentication (2FA) is a method of strengthening security by requiring a second method of authentication when connecting to an OpenVPN. The 2FA methods are in addition to a user's primary authentication method which uses their username and password.

The following acronyms are used in this section:

ACRONYM	DESCRIPTION
2FA	Two-Factor Authentication
CA	Certificate Authority
OTP	One-Time Password
HMAC	Hash-based Message Authentication Code. This is an event-based OTP where a counter is used.
HOTP	HMAC-based One-Time Password
TOTP	Time-based One-Time Password

Limitations

The following limitations apply for 2FA configuration:

- 2FA is only supported for OpenVPN connections.
- There is no support for the RADIUS server on the device to also provide verification for 2FA to external devices. 2FA is only supported for OpenVPN connections to the device itself, not for openVPN connections to other devices that are using the device as a RADIUS server.
- There is no support for 'grace periods' where a code is not needed for future connections for a fixed period of time after a successful verification.

The methods for configuring 2FA

Configure users with 2FA using one of these methods:

- "Use a mobile app" on page 19
- "2FA self-registration" on page 23
- "Use a mobile app" on page 19
- "Use a certificate" on page 28

Using these methods of 2FA provides an additional layer of security. 2FA verifies that the user and the service share a piece of private information in addition to the user's username and password. As this required token is constantly changing, it is also much more difficult for an attacker to guess.

Use a mobile app

what is it?

AlliedWare Plus has supported mobile applications since version 5.5.2-1.0. It requires a software-based authenticator that implements either the Time-based One-Time Password (TOTP) or HMAC-based One-Time Password (HOTP) algorithms. Users typically load these software authenticators, also known as authenticator apps, onto their mobile devices. Google Authenticator is a well-known implementation of such an app

You create a shared secret key for each user on your AlliedWare Plus device. A link or Quick Response (QR) code is provided to the user. They use this code to create an entry in their authenticator app. Each time the user authenticates they are required to enter the code shown in their authenticator app. This second verification step is in addition to their primary authentication method.

how does it work?

The process for 2FA using a mobile application is:

- The user's authenticator app and the AlliedWare Plus device have a shared secret key. The key is generated on the device and installed to the user's mobile phone using a QR code.
- When a user attempts to connect their OpenVPN client they are prompted for a username and password. If their username and password combination is correct, then a verification code is requested.
- The user checks their mobile authenticator app for the current code. This code is a hash of the current time-step (for TOTP) and makes use of the shared secret key stored in the app.
- The user then enters this code and the device checks if it would generate the same code for the current time-step. If it agrees, then authentication succeeds. By default, one code either side of the current time-step is checked. This is to account for delays in sending the code.
- If authentication succeeds, AlliedWare Plus stores the time-stamp as **used**. This prevents the same code from being used again and limits logins to once every 30 seconds. (This behaviour is configurable with the **2fa allow-reuse** command.)

how to configure it

Setting up authenticator-based 2FA on AlliedWare Plus is straightforward.

The basic steps are:

1. **Enable the 2FA service**

   Use the command **service 2fa** from Global Configuration mode.

2. **Set any global options, if non-default options are required**

   Use the commands **2fa issuer**, **2fa label**, **2fa hotp-window-size**, **2fa totp-window-size**, **2fa reset skew**, **2fa skew adjust** from Global Configuration mode.

3. **Create 2FA users or have them use 2FA self-registration**

   Use the command **2fa create users** or **2fa self-registration** ***<port-number>***.

4. **Load the shared secret keys into the user's authenticator apps**

   Use the command **2fa create user** from Privileged Executive mode. Users who are self-registering would do this from the web page.

5. **Add 2FA to the OpenVPN AAA method list.**

   Use the command **aaa authentication openvpn** from Global Configuration mode.

See "Example 5: Configuring basic two-factor authentication" on page 58 for more information about how to configure 2FA using a mobile app.

Once this is done, if a user is configured for 2FA, then they will require 2FA to connect to OpenVPN. There are also a number of global options that can be used to change certain behaviors of the feature, like whether to stop users with no 2FA configuration from connecting.

Configuration details

HOTP and TOTP

HOTP and TOTP are the two-supported modes for authenticator-based code verification. HOTP uses a counter as the input for the OTP hash function while TOTP uses the current time.

With HOTP both the mobile authenticator app and the AlliedWare Plus device start with a counter value of 1. When the device verifies a code as correct, it increments its internal counter. On the mobile app the counter is incremented each time the user hits the **refresh** button. This means if the user hits refresh too many times, the code displayed may be out of range of the codes that the device will accept. If this happens, the user needs to reset the counter in the mobile app (possibly by deleting and re-adding the entry) and hit **refresh**. Use the command **show 2fa user *<username>*** to display the number of times that is shown as the current HOTP counter value.

TOTP uses the current time divided by the time-step of 30 seconds as the input to the hash function. This means that as long as the time is synchronized between the mobile app and the AlliedWare Plus device, they will be working with the same input to the hash function.

allow code reuse

By default, TOTP codes can only be used once. This limits logins to whenever a new code is generated (30s intervals). To allow the reuse of TOTP codes within the configured window size time-frame, enter the following commands:

```
awplus>enable
awplus# configure terminal
awplus(config)# 2fa allow-reuse
```

window size

The window size configuration affects the range of authenticator-based codes that will be checked. There are separate configuration commands to configure the window size for TOTP and HOTP users.

The TOTP window size is the number of codes checked, centered on the current time-step. For example, the default TOTP window size of 3 allows the device to accept the codes for the previous, current and next time-steps.

The following commands configure a window size of 5, meaning that in addition to the current code, the 2 codes before and 2 codes after are also checked:

```
awplus>enable
awplus# configure terminal
awplus(config)# 2fa totp-window-size 5
```

The HOTP window size is the number of codes checked, starting from the current stored counter value. If a code later in the window is detected, the stored counter is updated to that value. This helps keep the mobile authenticator app and device synchronized. For example, the default HOTP window size of 3 allows the device to accept the next 3 codes.

The following commands configure a window size of 10. This means that in addition to the current code, the 9 following codes are also checked:

```
awplus>enable
awplus# configure terminal
awplus(config)# 2fa hotp-window-size 10
```

scratch-codes

When a 2FA user is created, a number of **scratch codes** are created for that user. These are one-time emergency codes that can be used in place of a code generated by the authenticator app. Once they are used, they can not be used again. Use the command **2fa reset scratch-codes user *<username>*** command to regenerate scratch codes for a user.

time skew adjustment

TOTP (time-based OTP) relies on the clock in the mobile authenticator app and the AlliedWare Plus device being synchronized. The authentication window size allows for some latitude in differences between the clocks, however if the time is wrong by a large amount, the user will not be able to successfully authenticate. This could happen, for example, due to an incorrect time zone configuration. In this case you can enable a mechanism to automatically adjust for time skew.

The time skew adjustment mechanism adds extra checking after an incorrect login. When an incorrect login occurs, the device will check a large number of codes either side of the current time. If it finds one that matches, it will record the **skew** of the code. If the user enters 3 incorrect (but different) codes in quick succession with the same time skew, the device will record this as an offset and automatically adjust which codes it checks for that user. To work, this requires 3 consecutive codes to be entered with no more than one time-step gap between them. The maximum amount of skew to check is configurable and defaults to 12.5 hours in either direction. Use the command **2fa skew adjust** to enable this feature.

To enable this feature, use the following commands:

```
awplus> enable
awplus# configure terminal
awplus(config)# 2fa skew-adjust
```

The maximum amount of skew to check is configured in 30s time-steps. By default it is 1500 time-steps or 12.5 hours. This can be set from 120 time-steps (1 hour) up to a maximum of 3000 time-steps (25 hours). The command to set the check to 25 hours is:

```
awplus(config)# 2fa max-skew 3000
```

The saved time skew offset for a user can be reset using the **2fa reset skew user <username>** command.

label and issuer

There are two global 2FA options that can be used to change how the entry for the key is displayed in the mobile authenticator app. These are the **label** and **issuer** options. These are purely cosmetic and have no impact on the verification codes generated. Setting a long string for these options will increase the size of the QR code. By default, the label is the hostname of the AlliedWare Plus device and the issuer is not set. For example, examine the following QR code link:

```
otpauth://totp/test@awplus?secret=7WXTHURCMT5HTA5JJUI3RN52LQ&issuer=Allied
```

The key is for the user **test**, the label is the default hostname **awplus**, and the issuer has been set as **Allied**.These strings are used by the mobile authenticator app to create the entry in its database.

Note: The label may also be used in the email template sent to email OTP users.

By default, the application-based key description is 'user@label'. The 'label' portion of this can be changed. The command to change the label to 'ATL VPN' is:

```
awplus(config)# 2fa label ATL VPN
```

The issuer field is not set by default. If it is set, it is also included in the QR code. The command to set the issuer to 'My Company' is:

```
awplus(config)# 2fa issuer My company
```

Run the command **show 2fa user <*username*> qr** to display user data after these changes will immediately reflect these changes in the OTP URL and QR code. This means that an existing user can remove their account in their mobile authenticator app and rescan the QR code to update the label and issuer value.

QR code display

A QR code is shown either when an authenticator-based user is created, or at a later stage using the **show 2fa user <*username*> qr** command. This QR code can be scanned by the user to add the account details to their mobile authenticator app.

QR codes display best in a color mode ANSI or UTF-8 terminal. Displaying the QR code with a terminal that is insufficiently wide, or doesn't have the right options enabled, may result in a QR code that is unscannable.

The QR code can be displayed in one of three different formats:

- **ansi:** this displays the code using ANSI block characters. These are black on grey and produce a rather large QR code.
- **utf8:** this displays the code using a UTF-8 mosaic. This produces a smaller code than ANSI and easier to work with. The examples in this guide use **utf8**.
- **link:** this creates a hyperlink for a QR code generator. Opening the link in a browser will display a QR code. As the string is passed to an online QR code generator (at Google), it may be a security concern for some installations.

2FA in Password

When adding 2FA to the AAA method list, it is possible to use the **2fa-in-password** parameter. This allows the user to enter their authentication code straight after their password in the password field of the OpenVPN client, for example **password654321**. This is useful if it is not possible to configure your client to accept a 2FA response. Use the command **aaa authentication openvpn** to configure this feature.

reject users not configured

You can deny authentication to users who have not been configured for 2FA with the command **2fa reject-unconfigured-users** from Global Configuration mode. To reject these users, enter the following commands:

```
awplus>enable
awplus# configure terminal
awplus(config)# 2fa reject-unconfigured-users
```

2FA self-registration

what is it?

Rather than the network administrator having to register each user and distribute their codes, the user can do it themselves. If self-registration for two-factor authentication is enabled, then a self-registration website will be hosted on the AlliedWare Plus device. The website allows unregistered users to register and obtain a QR code that they can use to save their secret key to an authenticator app on their mobile phone. Use the command **2fa self-registration port** to enable this feature and specify the HTTPS port.

Note: Users who self-register for 2FA will have time-based one time passwords (TOTP).

how to configure it

Assuming that the basic steps for setting up 2FA were followed, self-registration for 2FA on AlliedWare Plus can be set up using the following steps:

- choose a port for the self-registration website to be hosted on,
- if the website should be accessible externally, choose a port different to the secure HTTP port of the device,
- configure the firewall to allow HTTPS connections to that port,
- add the authentication server to the 2fa-registration AAA method list,
- and enable 2FA self-registration.

See "Example 6: Configuring users for two-factor authentication" on page 59 for more detail about configuring 2FA self-registration.

Use an email OTP

what is it?

From version 5.5.3-0.1 onwards, AlliedWare Plus additionally supports an email **One-Time Password** (OTP) implementation of 2FA. Email OTP uses the **Time-based One-Time Password** (TOTP) algorithm. The authentication server stores the email address of each user. RADIUS servers may instead store the domain name of the user's email address, and the device can combine it with the user's username to form their email address. Each time a user authenticates, they are required to enter a code sent to them via email. This second verification step is in addition to their primary authentication method.

how does it work?

The process for using 2FA email OTP is:

- When a user attempts to connect their OpenVPN client they are prompted for a username and password. If their username and password combination is correct, the device will then generate a password code and email it to the user.
- The user checks their email for the code then enters this code and the device checks if it is valid. If it agrees, then authentication succeeds. By default, a code is valid for 10 minutes. This is to account for delays in sending the code.
- If authentication succeeds, AlliedWare Plus stores the code as used. This prevents the same code from being used again, as each login attempt needs to generate a new code.

how to configure it

Setting up 2FA email OTP for all users on AlliedWare Plus is similar to using a mobile app as above. The main differences are:

- email OTP also needs to be enabled,
- 2FA users do not need to be created, because the email is retrieved from the authentication server. If a user’s email is not available on the authentication server, the user can be manually created on the AlliedWare Plus device with an email.
- self-registration would still create authenticator-based 2FA users, if it is enabled.

The AlliedWare Plus device can be configured to automatically retrieve the email of users to send OTP codes. This means that users do not need to be individually configured for 2FA. This requires the AAA server to contain the email addresses of the user.

Users who are manually configured for 2FA will be handled according to their manual configuration instead of retrieving their email from the authentication server and sending their OTP via email.

Note: Manually configured email users will be emailed an OTP at the configured email address, and not the address on the authentication server.

The following steps are required for this to work:

1. **Enable 2FA email OTP**

```
awplus>enable
awplus# configure terminal
awplus(config)# 2fa email-otp
```

2. **Configure the mail server**

Some of the prevalent commands are shown below. The **mail smtpserver tls [smtps|starttls]** command is particularly important. Without it the email codes sent out to users would be easy to intercept by an attacker.

```
awplus(config)# mail smtpserver 102.0.2.1
awplus(config)# mail smtpserver tls starttls
awplus(config)# mail smtpserver authentication plain username smtp_user password
pass
awplus(config)# mail from no_reply@company.com
```

3. **Configure the device with the email address**

The device needs to know where to get the email address from. In particular, the AAA server configuration needs to specify which attribute of the user will contain the email address. The configuration for LDAP and RADIUS servers are slightly different, so follow one of the following examples depending on which AAA server you are using.

Option 1: Collect email address from LDAP server

By default the ‘userPrincipalName’ attribute is used. To retrieve the user’s email address from the attribute ‘mail’ from the LDAP server ‘AD1’, use the commands:

```
ldap-server AD1
 email-attribute mail
```

Option 2: Collect email address from RADIUS server

By default the email address is not retrieved from a RADIUS server. The email address can either be retrieved completely from a RADIUS attribute, or constructed using the user's username combined with the value of a RADIUS attribute containing a domain name, for example, <***username***>**@**<***domain***>. The RADIUS attribute for the email or domain must be an ASCII text string attribute. This can be checked by using the **help** command in AlliedWare Plus to check whether the type of an attribute is ‘string’:

```
awplus# help radius-attribute
Standard Attributes:
  1     User-Name
  2     User-Password
  3     CHAP-Password
  4     NAS-IP-Address
...
awplus# help radius-attribute framed-pool
Framed-Pool : string (Character string)
```

The RADIUS types are also listed here at ianna RADIUS Types. Some of these attributes are listed as the type ‘text’, which is also acceptable as an email or domain attribute.

If the RADIUS server defines the user’s email in the attribute ‘User-Name’, then use the command:

```
awplus(config)# 2fa radius-email-attribute email User-Name
```

If the RADIUS server defines the user's email domain name in the attribute ‘Framed-Pool’, then use the command:

```
awplus(config)# 2fa radius-email-attribute domain Framed-Pool
```

Option 3: Configure the 2FA email address on local RADIUS server

The local RADIUS server can also be used on AlliedWare Plus to register a 2FA email address for users. The corresponding RADIUS attribute(s) must be configured on the device with the **2fa radius-email-attribute** command as shown above. Then, the following commands should be used to configure 2FA for the RADIUS groups.

To assign an email address in the ‘User-Name’ attribute, use the following commands:

```
radius-server local
 server enable
 group g-example1
  attribute User-Name example1@abc.com
 group g-example2
  attribute User-Name example2@def.com
 user test1 password pass1 group g-example1
 user test2 password pass2 group g-example2
```

To assign the domain name of a user's email address in the ‘Framed-Pool’ attribute, use the following commands:

```
radius-server local
 server enable
 group abc.com
  attribute Framed-Pool abc.com
 user user1 password pass3 group abc.com
 user user2 password pass4 group abc.com
```

Configuration details

email template

When a user is sent an email by the device for email OTP, the device uses a template to generate the email. The template can be configured to use a particular format. The configured template must show the OTP, and otherwise can also contain the user's username, email address, OTP expiry time, and/or the configured label.

2FA uses an email template when sending an email to manually configured email OTP users and automatically emailed users.

The default email template is this:

```
Subject: %%LABEL%% 2FA Email OTP code

Verification code: %%OTP%%

The verification code will expire in %%EXPIRY_TIME%% minutes.
This is an automated message, please do not reply.
```

A subject line with an empty line immediately following it is required. There are some words surrounded by %% signs. These are replaced by the specified details when the email is sent. There are five different options for these, each of which can be used multiple times if needed:

ATTRIBUTE	DESCRIPTION
%%OTP%%	The OTP code that the user can use to log in. This must be included in the template somewhere.
%%USERNAME%%	The username of the user who is attempting to connect.
%%EMAIL_ADDR%%	The email address of the user, which the email will be sent to.
%%EXPIRY_TIME%%	The number of minutes that the OTP will be valid for.
%%LABEL%%	A configurable label that is set by the 2FA configuration, which is also displayed in application-based 2FA.

To configure an email template, first the new email template needs to be saved to a file on the flash memory of the device. An example of the contents of the file is this:

```
Subject: %%LABEL%% 2FA Email OTP code for %%USERNAME%%

Verification code: %%OTP%%

The verification code will expire in %%EXPIRY_TIME%% minutes.
This is an automated message, please do not reply.

This email was intended for %%EMAIL_ADDRESS%%.
Send by %%LABEL%%
```

If the file was saved to flash:/email_template.txt, then the following command would set the email template that would be used for 2FA:

```
awplus(config)# 2fa email-template email_template.txt
```

If the file changes, the template will not be updated until this command is run again, or when the 2FA service is restarted.

email OTP expiry time

The expiry time of email OTPs can be configured from 1 minute up to 1440 minutes. (1 day) By default, email OTPs expire after 10 minutes. The command to configure an email OTP expiry time of 90 minutes is:

```
awplus(config)# 2fa email-expiry-time 90
```

label The label option can be configured for the email template that is sent to users. This label is the same label that is displayed to application-based 2FA users. The default email template contains the label in the subject line, which is <***label***> 2FA Email OTP code. If unconfigured, the label is set to the hostname of the AlliedWare Plus device. The command to change the label to 'ATL VPN' is:

```
awplus(config)# 2fa label ATL VPN
```

Changing this will also affect the label on application-based 2FA users.

reject users not configured Email OTP cannot be enabled at the same time as rejecting unconfigured users since both settings determine how unconfigured users are authenticated.

Use a certificate

what is it? From version 5.5.3-0.1 onwards, the OpenVPN server can also use certificates to verify the identity of the clients, as an addition to the other methods. This method of authentication adds an extra layer of security on top of the existing username and password.

how does it work? For a client to connect to the OpenVPN server, certificates need to be provided which have been signed by the same CA that the server is using. The only way to generate such a client certificate requires using the CA's private key. Therefore as long as the server trusts the CA, it can trust a client with a valid certificate.

It is possible to configure which CA the OpenVPN Server is using, supporting both an external CA or a locally self-signed CA that exists on the device. When using an external CA the client certificates are generated and distributed externally. However, if a local self-signed CA is used, the certificates must be generated on the AlliedWare Plus device and then exported for distribution.

how to configure it

Using a local CA

In this case, the AlliedWare Plus device acts as the **Certificate Authority** (CA) and is responsible for generating the client certificates needed for the end users to connect. The following steps tell you how to generate and use client certificates using the local trustpoint:

1. **On the AlliedWare Plus server, create the client certificates**

Enroll each user with the trustpoint you configured in "Use a certificate to let clients verify the server's identity" on page 12. Enrolling a user generates a certificate and key for them, using the private key from the local CA stored on the device. Their username will become the Common Name (CN) on their certificate. For example, add local user 'clientA1':

```
awplus#crypto pki enroll local user clientA1
Enter an export passphrase, or "abort" to cancel.
****
Enter the export passphrase again.

****
Generating a user private key for "clientA1"...
Successfully enrolled user "clientA1".
The PKCS#12 file is ready to export.
```

Do this for each client that needs to connect. For example, to add local user ‘clientA2’ and ‘clientA3’:

```
awplus# crypto pki enroll local user clientA2
...
awplus# crypto pki enroll local user clientA3
```

2. **On the AlliedWare Plus device, export the client certificates**

When the device reboots it will clear all enrolled users and delete the certificate files off the device, so make sure to export them and store them on a separate secure device. For example:

```
awplus# crypto pki export local pkcs12 clientA1 tftp://192.0.2.1/clientA1.p12
Copying...
Successful operation
awplus# crypto pki export local pkcs12 clientA2 tftp://192.9.2.1/clientA2.p12
...
awplus# crypto pki export local pkcs12 clientA3 tftp://192.9.2.1/clientA3.p12
...
```

The PKCS#12 file exported contains the local CA, the client certificate and the client key that is bundled into a single **.p12** file.

3. **On the AlliedWare Plus device, enable client certificate authentication on the tunnel**

This is the same process for a trustpoint using a local self-signed CA, or an external CA. This step makes the tunnel check certificates. This trustpoint needs to be the one you enrolled the users on. For example:

```
awplus# configure terminal
awplus(config)# interface tunnel1
awplus(config-if)# tunnel openvpn verify-client-certificate trustpoint local
```

4. **On the AlliedWare Plus device, optionally turn off common name checking**

You can do this if you want to have all users using one certificate, however, doing that is less secure.

```
awplus(config-if)# no tunnel openvpn verify-client-certificate strict-common-
name-check
```

5. **On a separate PC, create the client .ovpn file**

Samples of **.ovpn** file templates can be found on the OpenVPN website. These sample templates typically include explanations of the various **.ovpn** file configuration options and advice on default settings.

Storing the certificates inline in the .ovpn file

Each client will need their own unique **.ovpn** configuration file. The clients need access to the following three items in order to connect:

- The CA certificate
- Their client certificate
- The private key for their client certificate

From a PC, use (for example) OpenSSL to extract these three items from the **.p12** file you exported from the AlliedWare Plus device. If you are using OpenSSL on a Linux PC, you can see the commands to separate these items by using the OpenSSL help:

```
openssl pkcs12 -help
```

These three items can be stored inline within the **.ovpn** configuration file. Add the following tags for the three different items:

```
<ca>
</ca>
<cert>
</cert>
<key>
</key>
```

Between these tags, paste the three items. Each of these should start with a ----- BEGIN ----- tag and finish with an ----- END ----- tag.

For example:

```
# rest of file omitted

# The CA certificate
<ca>
-----BEGIN CERTIFICATE-----
MIICpjCCAY4CCQD/WyXBOT2bmjANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDDApP
...
3fgeDe6t1tpzbQ==
-----END CERTIFICATE-----
</ca>

# The client's certificate
<cert>
-----BEGIN CERTIFICATE-----
MIICpDCCAYwCAQMwDQYJKoZIhvcNAQELBQAwFTETMBEGA1UEAwwKT3BlblZQTi1D
...
0UmayaINKnw=
-----END CERTIFICATE-----
</cert>

# The client's key
<key>
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAsSq4DqHc4+z2Tpz1hsx9MizIUmtTukIY0PCOxSIaME3uXO/F
...
X56ZryeBtbJdsVNgL0dq1LWCe46W8PqaeE6K20mWLwnq8FHY3Oyk
-----END RSA PRIVATE KEY-----
</key>
```

If you are editing a previously-used file, delete any existing certificate configuration, for example, any line saying CA c:/ca/cacert.pem.

Using a separate certificate file

It is also possible to provide the **.p12** file to the client instead of storing it inline within the **.ovpn** configuration file. In that case, in the **.ovpn** file, add a line for **pkcs12** with the path to the file. For example, **pkcs12 <path>/all-clients.p12**.

Note: Note that you may need to create one .opvn file for each different client operating system, depending on where in your system you store the files. Otherwise, the file paths in the .opvn file may not work for different devices.

1. **Load the edited .ovpn file onto the client**

Once the user loads this **.ovpn** configuration file into their OpenVPN client, they will be able to connect as usual.

Show commands to monitor your configuration

The **show 2fa** command shows the state of the 2FA service. It includes the number of users and all configurable options.

```
awplus#show 2fa
Two-Factor Authentication Information:

Settings:
  Allow TOTP code reuse:          No
  Reject users with no config:    No
  Allow self-registration:        Yes
  Self-registration port:         443
  TOTP window size:               3
  HOTP window size:               3
  Attempt Skew Adjustment:        No
  Issuer:                         Allied
  Label:                          Unset (using hostname)
  Debug:                          Disabled
  Email OTP status:               Enabled (Mail server configured with TLS)
  Email OTP expiry time:          90 minutes
  Email template:                 email_template.txt

Number of configured users:       5
```

The **show 2fa users** command displays an alphabetical list of users, the type of each user, and the last time they successfully authenticated using 2FA.

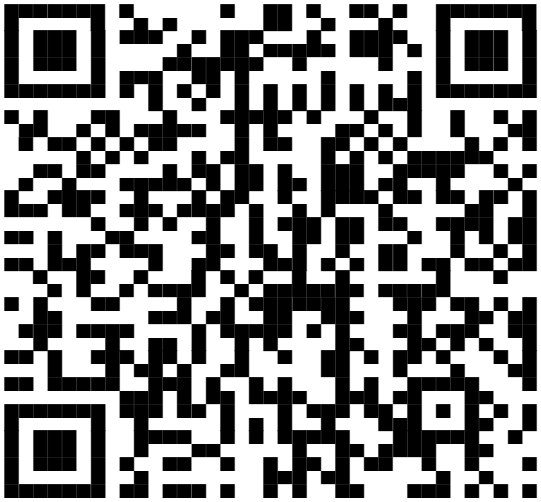
```
awplus#show 2fa users
Two-Factor Authentication users:

Username                                    Mode      Last OTP Login
-------------------------------------------------------------------------------
test1                                       TOTP      -
test2                                       Email     -
test3                                       Skip-2FA  -
user1                                       TOTP      30 Jun 2022 16:49:54 NZST
user2                                       TOTP      -
```

To see the data and optional QR code for a single authenticator-based user, use the **show 2fa user <username> qr utf8** command.

```
awplus#show 2fa user test qr utf8
Two-Factor Authentication information for user:

Username:         test
Last OTP login:   30 Jun 2022 16:49:54 NZST
Secret:           RIVS33ZJCBQUU7WJMDHXJKUDYU
Mode:             TOTP
OTP URL:          otpauth://totp/test@awplus?secret=RIV...&issuer=Allied
Scratch codes:
  44857801
  39444714 (Used 2022-06-30T04:48:52Z)
  48774624
  89142923 (Used 2022-06-30T04:50:39Z)
  52234486
```



The **show 2fa user <username>** command can also be used to verify the configuration of other types of users. If the user ‘test2’ is configured as an email user, then the output of the command would be as follows:

```
awplus# show 2fa user test2
Two-Factor Authentication information for user:

Username:         test2
Mode:             Email
Email:            test2@xyz.com
```

If the user test3 is configured to skip 2FA, then the output would be like this:

```
awplus# show 2fa user test3
Two-Factor Authentication information for user:

Username:         test3
Mode:             Skip-2FA
Skip 2FA:         True
```

The **show 2fa email-template** command shows the currently set email template.

```
awplus#show 2fa email-template
Subject: %%LABEL%% 2FA Email OTP code for %%USERNAME%%

Verification code: %%OTP%%

The verification code will expire in %%EXPIRY_TIME%% minutes.
This is an automated message, please do not reply.

This email was intended for %%EMAIL_ADDRESS%%.
Send by %%LABEL%%
```

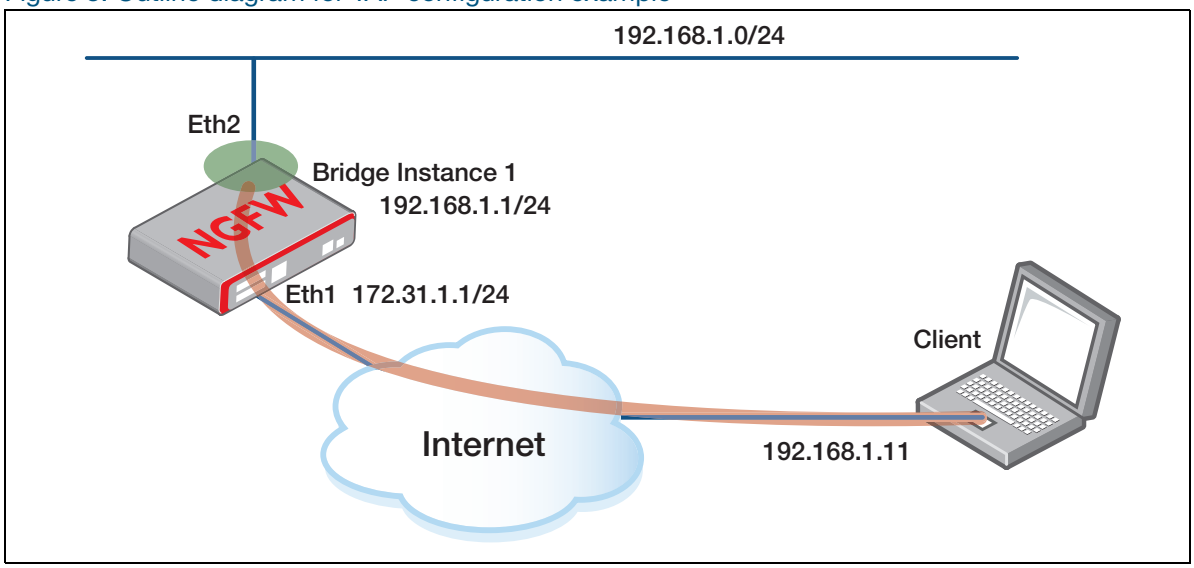
AlliedWare Plus OpenVPN configuration examples

OpenVPN supports remote access from multiple operating systems and mobile devices, which means you can have remote access to the company internal network. For more information about how to configure OpenVPN on the client device, visit OpenVPN.

The following examples show how to configure both OpenVPN TAP service and TUN service.

Example 1: Configuring OpenVPN TAP service

Figure 3: Outline diagram for TAP configuration example



1. **Configure local RADIUS server for OpenVPN TAP mode**

```
awplus# configure terminal
```

- Specify a local RADIUS server host (IP address 127.0.0.1 indicates that the device itself is operating as the RADIUS server) and set parameters for the server.

```
awplus(config)# radius-server host 127.0.0.1 key awplus-local-radius-server
awplus(config)# aaa authentication openvpn default group radius
```

- Enter the local RADIUS server configuration mode.

```
awplus(config)# radius-server local
```

- Configure the client user group and configure the client IP address. Note that this step is optional for configuring OpenVPN TAP mode.

```
awplus(config-radsrv)# group client
```

- Configure the client user IP address. If you want to support more client users, you need to create a group for each client user. Note that if you want to configure the client IP address/mask with the RADIUS server, then this step is required. If you don't want to configure the client IP address/mask with the RADIUS server, then this step is not required and you can configure the client IP address via DHCP.

```
awplus(config-radsrv-group)# attribute Framed-IP-Address 192.168.1.11
```

- Configure the IP subnet mask of the tunnel interface. Note that if you want to configure the client IP address/mask with the RADIUS server, then this step is required. If you don't want to configure the client IP address/mask with the RADIUS server, then this step is not required and you can configure the client IP address via DHCP.

```
awplus(config-radsrv-group)# attribute Framed-IP-Netmask 255.255.255.0
```

- **Optional:** Configure the route for packets routing from network 192.168.0.0/16 to the remote network through the tunnel with 192.168.1.1 being the IP address of the remote tunnel interface.

```
awplus(config-radsrv-group)# attribute Framed-Route "192.168.0.0/16
192.168.1.1"
```

- Return to the local RADIUS server configuration mode.

```
awplus(config-radsrv-group)# exit
```

- Add the NAS with an IP address to the list of clients that may send authentication requests to the local RADIUS server. In this case, the NAS is the device itself, so the NAS address is 127.0.0.1.

```
awplus(config-radsrv)# nas 127.0.0.1 key awplus-local-radius-server
```

- Add a user to the RADIUS server database and specify the user name and password.

```
awplus(config-radsrv)# user remote password very_secret group client
```

- Enable the local RADIUS server.

```
awplus(config-radsrv)# server enable
```

2. **Configure server authentication**

- Declare local CA (Certificate Authority) as the trust point that the system uses.

```
awplus(config)# crypto pki trustpoint local
```

- Obtain a system certificate from local CA.

```
awplus# crypto pki enroll local
```

- Export this CA public certificate, so the VPN client can use it to verify the Computer Certificate of the VPN router. This generates a file named **cacert.pem** on the flash file system (this file will be used in this example). This command is entered from Privilege Exec mode, and so does not appear in the device running configuration. This certificate can also optionally be exported to the CLI terminal window directly, or to an external TFTP server directory.

```
awplus# crypto pki export local pem cacert.pem
```

3. **Configure the interface connecting the device to the Internet**

```
awplus(config)# interface eth1
awplus(config-if)# ip address 172.31.1.1/24
```

4. **Enable OpenVPN TAP service**

- Create a virtual tunnel interface (VTI) for the OpenVPN router to be accessed by the client.

```
awplus(config-if)# interface tunnel1
awplus(config-if)# tunnel mode openvpn tap
awplus(config-if)# exit
```

5. **Connect OpenVPN clients to the LAN**

- Create a virtual Ethernet bridge to connect the VPN clients to the LAN.

```
awplus(config)# bridge 1
```

This newly created bridge will have two ports. One is the physical port eth2 that is connected to the LAN network. The other is the tunnel interface where the virtual OpenVPN TAP NIC will connect to.

- Assign eth2 and tunnel2 to the bridge.

```
awplus(config)# interface eth2
awplus(config-if)# bridge-group 1
awplus(config)# interface tunnel2
awplus(config-if)# bridge-group 1
```

6. **Enable other traffic to be routed to the Internet**

Not all the traffic that enters the eth2 interface of the AlliedWare Plus firewall or router is destined to go to the OpenVPN clients. Much of the traffic is destined to simply be routed to the Internet. So, we need a method to route traffic out of the bridge instance and deliver it through eth1 to the Internet.

- To do this, we need to configure an IP address on the bridge instance.

```
awplus(config)# interface br1
awplus(config-if)# ip address 192.168.1.1/24
```

The logic for routing packets from the LAN to the Internet is as follows:

- When packets enter the AlliedWare Plus firewall or router via interface ETH2, they are deemed to have entered Bridge Instance 1.
- If the destination IP address of the packet is not within the subnet of Bridge Instance 1 (192.168.1.0/24), then the packet needs to be routed out of the bridge instance.

Configuring OpenVPN client for TAP service

Various OpenVPN clients are accessible across multiple platforms. The commonality among these clients is their reliance on a .ovpn file. After creating the .ovpn file, configuring the client usually involves loading the file. This particular file was tested with OpenVPN 2.3 but is expected to function with OpenVPN 2.1 or newer clients.

OpenVPN TAP mode client **.ovpn** config file

```
#Configure for client mode
client

#The server requires the client to provide a username/password for authentication.
auth-user-pass

#Require encryption
cipher AES-128-CBC

#Configure for TAP mode
dev tap
proto udp

#The address of the OpenVPN router to connect to
remote 172.31.1.1
```

Then paste the content of the **cacert.pem** flash file generated previously at "Configure server authentication" on page 36, into the **.ovpn** config file, with a header and footer around it, as shown below.

Samples of **.ovpn** file templates can be found on the OpenVPN website. These sample templates typically include explanations of the various **.ovpn** file configuration options, advice on default settings, and also show locations of where to paste the **cacert.pem** content.

Contents of the **cacert.pem** file

```
-----BEGIN CERTIFICATE-----
MIICXDCCAcWgAwIBAgIBADANBgkqhkiG9w0BAQUFADA0MRcwFQYDVQQKEw5BbGxp
ZWQtVGVsZXNpczEZMBcGA1UEAxMQQWxsaWVkd2FyZVBsdXNDQTAeFw0xNTA2MTkx
MDQxMzZaFw0zNTA2MTQxMDQxMzZaMDQxFzAVBgNVBAoTDkFsbGllZC1UZWxlc2lz
MRkwFwYDVQQDExBBbGxpZWR3YXJlUGx1c0NBMIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQCywR9mBFrxnIVw577vtmvui5RnTBjLMJWkqF6eOhvt1Rzw9H8SZWPI
zjLs+Z21b5Nsc8YjB0kLcISu3lfZrxtIPlkEm81U8mImHJnTBAmkHUZi5fBJbH12
KG7rUZ/Zxq591+vatwabyRiDIPEeis/aa1wEFm03uBc21NMsQYENiwIDAQABo34w
fDAMBgNVHRMEBTADAQH/MCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRl
ZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUXJbOiME3bd2KhOLH8D25nt7xQ6UwHwYD
VR0jBBgwFoAUXJbOiME3bd2KhOLH8D25nt7xQ6UwDQYJKoZIhvcNAQEFBQADgYEA
YZG8DWKYj8rYkkvbZ1/ZhsLzubAH1ggUueS1eCzanh9zi5o92+pwOTPgi/g3JnDe
cpw06L6kvgTItZxwa32whO2RjUzzhCLegc9DKyEixmAZ6bUd29Idsn1DpGHEgPXT
QgSTQrb4HcOAlbrG7Eh/IlRN1ByE4QAUNaP6N84nZwo=
-----END CERTIFICATE-----
```

This **.ovpn** file can be used by all clients. The individual clients use the username and password to authenticate themselves. OpenVPN client applications typically prompt the user to enter the username and password when connecting, and store this for subsequent connections.

The following show command can be used to display the start and end dates of the certificates on the device:

Certificate expiry information

```
awplus#show crypto pki certificates local

--------------------
Trustpoint "local" Certificate Chain
--------------------
Self-signed root certificate
   Subject     : /O=Allied Telesis, Inc./CN=AlliedWarePlusCAA05050G152000036
   Issuer      : /O=Allied Telesis, Inc./CN=AlliedWarePlusCAA05050G152000036
   Valid From  : Jun 11 10:41:50 2016 GMT
   Valid To    : Jun  9 10:41:50 2026 GMT
   Fingerprint : 5C6A616A 3A28699A D24156E5 505E4CE7 2B109D0D
```

Example 2: Configuring OpenVPN TUN service

Server

1. **Configure the local RADIUS server for OpenVPN TUN mode**

```
awplus# configure terminal
```

- Declare local CA (Certificate Authority) as the trust point that the system uses.

```
awplus(config)# crypto pki trustpoint local
```

- Obtain a system certificate from local CA.

```
awplus# crypto pki enroll local
```

- Enter the local RADIUS server configuration mode.

```
awplus(config)# radius-server local
```

- Configure client user group and configure client IP address.

```
awplus(config-radsrv)# group client
```

- Configure client user IP address. If you want to support more client users, you need to create a group for each client user.

```
awplus(config-radsrv-group)# attribute Framed-IP-Address 192.168.2.11
```

- Configure IP subnet mask of the tunnel interface.

```
awplus(config-radsrv-group)# attribute Framed-IP-Netmask 255.255.255.0
```

- **Optional**: Configure the route for packets routing from network 192.168.0.0/16 to the remote network through the tunnel with 192.168.2.1 being the IP address of the remote tunnel interface.

```
awplus(config-radsrv-group)# attribute Framed-Route "192.168.0.0/16
192.168.2.1"
```

- Return to the local RADIUS server configuration mode.

```
awplus(config-radsrv-group)# exit
```

- Add the NAS with an IP address to the list of clients that may send authentication requests to the local RADIUS server. In this case, the NAS is the switch itself, so the NAS address is 127.0.0.1.

```
awplus(config-radsrv)# nas 127.0.0.1 key awplus-local-radius-server
```

- Add a user to the RADIUS server database and specify the user name and password.

```
awplus(config-radsrv)# user remote password very_secret group client
```

- Enable local RADIUS server.

```
awplus(config-radsrv)# server enable
awplus(config-radsrv)# exit
```

2. **Configure the OpenVPN to authenticate using RADIUS**

- Specify a local RADIUS server host (IP address 127.0.0.1 indicates that the switch itself is operating as the RADIUS server) and set parameters for the server.

```
awplus(config)# radius-server host 127.0.0.1 key awplus-local-radius-
server
awplus(config)# aaa authentication openvpn default group radius
```

3. **Configure tunnel interface**

- Create a tunnel interface.

```
awplus(config-if)# interface tunnel20
awplus(config-if)# tunnel mode openvpn tun
```

- Configure an IP address for the tunnel interface.

```
awplus(config-if)# ip address 192.168.2.1/24
```

4. **Configure other interfaces**

```
awplus(config)# interface eth1
awplus(config-if)# ip address 172.31.1.1/24
awplus(config)# interface eth2
awplus(config-if)# ip address 192.168.1.1/24
```

Client

1. **Configure OpenVPN client for TUN service**

Several OpenVPN clients are available for many platforms. Most have in common that they rely on a **.ovpn-file**. Once the .ovpn file is created client configuration is typically a matter of loading the file.

- Each OpenVPN client logs in using a unique user name and password, and an Open VPN client can be configured to use the following .ovpn config file and associated CA certificate.
- Some OpenVPN clients are able to reference the CA certificate file directly without having to paste in the certificate information into the .ovpn file template as below.

This file was tested with OpenVPN 2.3 but should work with OpenVPN 2.1 or newer clients.

OpenVPN TUN mode client **.ovpn** config file

```
remote 172.31.1.1 1194 udp
pull
tls-client
cipher AES-128-CBC
auth SHA1
tls-cipher TLS-DHE-RSA-WITH-AES-256-CBC-SHA
auth-user-pass
ca cacert.pem
dev-type tun
topology subnet
port 1194
verb 7
```

2. Create the file **cacert.pem**

- Create the file **cacert.pem** referred to in the **.ovpn** file on the AlliedWare Plus firewall or router, by using the command **crypto pki export local pem cacert.pem**.

Example **export** command

```
awplus#crypto pki export local pem cacert.pem
Copying...
Successful operation
```

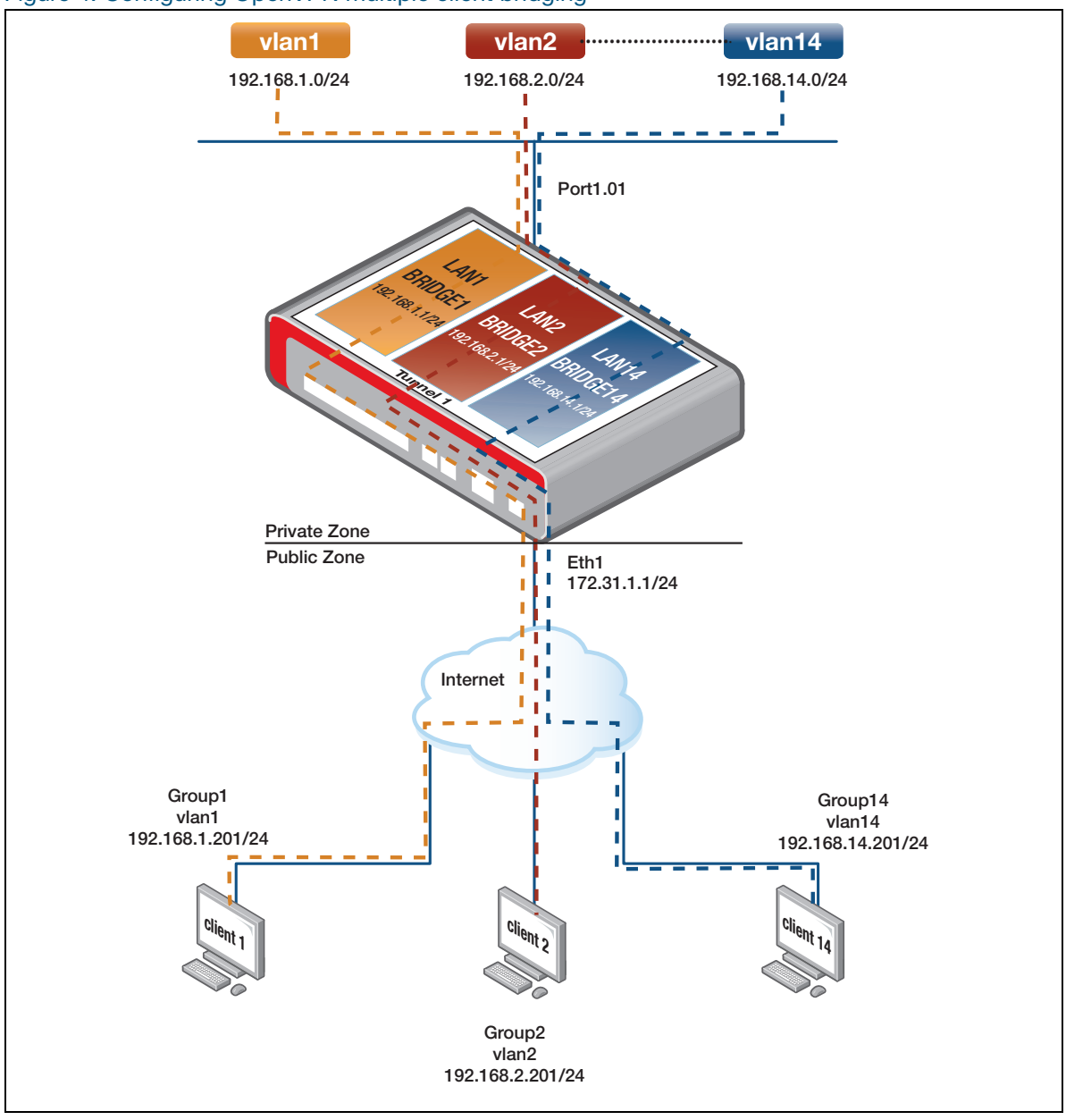
Then, copy this file off the AlliedWare Plus firewall or router and onto the client device.

Example 3: Configuring OpenVPN multiple client bridging

The following example consists of multiple (14) remote OpenVPN clients connecting into a central AlliedWare Plus firewall or router.

In this example, the AlliedWare Plus device is configured to bridge each incoming client VPN connection to a specific VLAN. Each bridged network is configured to be in its own unique firewall network entity inside the private zone. Traffic between each firewall network entity is blocked. Firewall rules are configured to allow each client to access only the specific VLAN that they are a member of.

Figure 4: Configuring OpenVPN multiple client bridging



AlliedWare Plus firewall or router configuration

1. **Local radius server and CA configuration**

- Configure the local radius server.
- Declare local CA (Certificate Authority) as the trust point that the system uses, and obtain a system certificate from local CA.

Local radius server and CA configuration

```
radius-server host 127.0.0.1 key radius
!
aaa authentication openvpn default group radius
!
crypto pki trustpoint local
!
crypto pki enroll local
radius-server local
 server enable
 nas 127.0.0.1 key radius
```

2. **Client user group database configuration**

- Configure a client user group database, with each group entry consisting of an IP address and subnet mask. An 802.1q VLAN tag is associated with each group, and each client is allocated an IP address and subnet mask for the specific network that they need to access.

Client user group database configuration

```
group group1
 vlan 1
 attribute Framed-IP-Address 192.168.1.201
 attribute Framed-IP-Netmask 255.255.255.0
group group2
 vlan 2
 attribute Framed-IP-Address 192.168.2.201
 attribute Framed-IP-Netmask 255.255.255.0
group group3
 vlan 3
 attribute Framed-IP-Address 192.168.3.201
 attribute Framed-IP-Netmask 255.255.255.0
group group4
 vlan 4
 attribute Framed-IP-Address 192.168.4.201
 attribute Framed-IP-Netmask 255.255.255.0
group group5
 vlan 5
 attribute Framed-IP-Address 192.168.5.201
 attribute Framed-IP-Netmask 255.255.255.0
group group6
 vlan 6
 attribute Framed-IP-Address 192.168.6.201
 attribute Framed-IP-Netmask 255.255.255.0
group group7
 vlan 7
 attribute Framed-IP-Address 192.168.7.201
 attribute Framed-IP-Netmask 255.255.255.0
group group8
 vlan 8
 attribute Framed-IP-Address 192.168.8.201
 attribute Framed-IP-Netmask 255.255.255.0
```

```
configuration continued...
```

```
group group9
 vlan 9
attribute Framed-IP-Address 192.168.9.201
 attribute Framed-IP-Netmask 255.255.255.0
group group10
 vlan 10
 attribute Framed-IP-Address 192.168.10.201
 attribute Framed-IP-Netmask 255.255.255.0
group group11
 vlan 11
 attribute Framed-IP-Address 192.168.11.201
 attribute Framed-IP-Netmask 255.255.255.0
group group12
 vlan 12
 attribute Framed-IP-Address 192.168.12.201
 attribute Framed-IP-Netmask 255.255.255.0
group group13
 vlan 13
 attribute Framed-IP-Address 192.168.13.201
 attribute Framed-IP-Netmask 255.255.255.0
group group14
 vlan 14
 attribute Framed-IP-Address 192.168.14.201
 attribute Framed-IP-Netmask 255.255.255.0
```

3. **Radius server user database authentication configuration**

- Configure the Radius Server database to authenticate each incoming OpenVPN client connection, and associate each client to their appropriate client user group.

Radius server user database authentication configuration

```
user user1 encrypted password <password1> group group1
user user2 encrypted password <password2> group group2
user user3 encrypted password <password3> group group3
user user4 encrypted password <password4> group group4
user user5 encrypted password <password5> group group5
user user6 encrypted password <password6> group group6
user user7 encrypted password <password7> group group7
user user8 encrypted password <password8> group group8
user user9 encrypted password <password9> group group9
user user10 encrypted password <password10> group group10
user user11 encrypted password <password11> group group11
user user12 encrypted password <password12> group group12
user user13 encrypted password <password13> group group13
user user14 encrypted password <password14> group group14
```

4. **Firewall zone and network entity configuration**

- Configure a named private firewall zone entity and an associated network entity for each subnet, and configure a public zone entity for the WAN connection to the Internet via eth1.

Firewall zone and network entity configuration

```
zone private
 network lan1
  ip subnet 192.168.1.0/24!
 network lan2
  ip subnet 192.168.2.0/24
 network lan3
  ip subnet 192.168.3.0/24
 network lan4
  ip subnet 192.168.4.0/24
 network lan5
  ip subnet 192.168.5.0/24
 network lan6
  ip subnet 192.168.6.0/24
 network lan7
  ip subnet 192.168.7.0/24
 network lan8
  ip subnet 192.168.8.0/24
 network lan9
  ip subnet 192.168.9.0/24
 network lan10
  ip subnet 192.168.10.0/24
 network lan11
  ip subnet 192.168.11.0/24
 network lan12
  ip subnet 192.168.12.0/24
 network lan13
  ip subnet 192.168.13.0/24
 network lan14
  ip subnet 192.168.14.0/24
!
zone public
 network all
  ip subnet 0.0.0.0/0 interface eth1
 network interface
  ip subnet 172.31.1.0/24
 host router
  ip address 172.31.1.1
```

5. **Firewall OpenVPN application configuration**

- Configure application for OpenVPN to be passed by the firewall.

Firewall OpenVPN application configuration

```
application openvpn
 protocol udp
 dport 1194
```

6. **Firewall rules configuration**

When the firewall is enabled, all traffic is then blocked by default, so firewall rules need to configured to allow specific application traffic to pass through the firewall.

- Configure a firewall rule to allow traffic from the private firewall zone to access the public internet.
- Configure firewall rules to allow traffic from each OpenVPN client to access their specific named LAN network entity.
- Configure a firewall rule to allow incoming OpenVPN traffic to pass through the public interface.

Firewall rules configuration

```
firewall
 rule 100 permit any from private to public
 rule 110 permit any from private.lan1 to private.lan1
 rule 210 permit any from private.lan2 to private.lan2
 rule 310 permit any from private.lan3 to private.lan3
 rule 410 permit any from private.lan4 to private.lan4
 rule 510 permit any from private.lan5 to private.lan5
 rule 610 permit any from private.lan6 to private.lan6
 rule 710 permit any from private.lan7 to private.lan7
 rule 810 permit any from private.lan8 to private.lan8
 rule 910 permit any from private.lan9 to private.lan9
 rule 1010 permit any from private.lan10 to private.lan10
 rule 1110 permit any from private.lan11 to private.lan11
 rule 1210 permit any from private.lan12 to private.lan12
 rule 1310 permit any from private.lan13 to private.lan13
 rule 1410 permit any from private.lan14 to private.lan14
 rule 1500 permit openvpn from public to public.interface.router
protect
```

7. **Firewall NAT rules configuration**

- Configure a firewall NAT masquerade rule to translate the source IP address of all traffic originating from the private zone destined to the internet using the public IP address of eth1 WAN.

Firewall NAT rules configuration

```
nat
 rule 100 masq any from private to public
enable
```

8. **VLAN database configuration**

- Configure the VLAN database.

VLAN database configuration

```
vlan database
 vlan 2-14 state enable
```

9. **Switchport configuration**

- In this example, switchport 1.0.1 is configured to be an 802.1q trunked member of the VLANs that OpenVPN clients will access. This port could be connected to a separate Layer2 access switch. In this example, the native VLAN has been removed from the switchport, so that only 802.1q VLAN tagged frames are accepted.

Switchport configuration

```
interface port1.0.1
 switchport mode trunk
 switchport trunk allowed vlan add 1-14
 switchport trunk native vlan none
```

10. **WAN interface configuration**

- Configure public interface ethernet1 with the static ip address allocated by the Internet Service Provider.

WAN interface configuration

```
interface eth1
 ip address 172.31.1.1/24
```

11. **VTI and VTI sub interfaces configuration**

- Configure Virtual Tunnel Interface (VTI) in OpenVPN Tap mode, and configure a series of sub interfaces with associated 802.1q VLAN ID encapsulation. OpenVPN is configured to use port number 1194.
- Each incoming VPN data stream is decrypted. The resulting Ethernet frames contain a source IP address and subnet mask that is matched against a specific client user group database entry. The 802.1q VLAN tag configured in the matching client user group entry is inserted into the decrypted Ethernet frames.
- This allows incoming decrypted 802.1q tagged Ethernet data streams to be forwarded to the appropriate VTI sub interface based on the matching 802.1q VLAN tags.

VTI and VTI sub interfaces configuration

```
interface tunnel1
 encapsulation dot1q 1
 encapsulation dot1q 2
 encapsulation dot1q 3
 encapsulation dot1q 4
 encapsulation dot1q 5
 encapsulation dot1q 6
 encapsulation dot1q 7
 encapsulation dot1q 8
 encapsulation dot1q 9
 encapsulation dot1q 10
 encapsulation dot1q 11
 encapsulation dot1q 12
 encapsulation dot1q 13
 encapsulation dot1q 14
 tunnel openvpn tagging 1
 tunnel openvpn port 1194
 tunnel mode openvpn tap
```

12. **Bridge configuration**

- Configure bridges instances to allow each OpenVPN client to connect to their own unique bridged network.

Bridge configuration

```
bridge 1
bridge 2
bridge 3
bridge 4
bridge 5
bridge 6
bridge 7
bridge 8
bridge 9
bridge 10
bridge 11
bridge 12
bridge 13
bridge 14
```

13. **Bridge IP address configuration**

- Each bridge instance is configured with an IP address within the network that each client connects to.

Bridge IP address configuration

```
interface br1
 ip address 192.168.1.1/24
!
interface br2
 ip address 192.168.2.1/24
!
interface br3
 ip address 192.168.3.1/24
!
interface br4
 ip address 192.168.4.1/24
!
interface br5
 ip address 192.168.5.1/24
!
interface br6
 ip address 192.168.6.1/24
!
interface br7
 ip address 192.168.7.1/24
!
interface br8
 ip address 192.168.8.1/24
!
interface br9
 ip address 192.168.9.1/24
!
interface br10
 ip address 192.168.10.1/24

!
interface br11
 ip address 192.168.11.1/24
!
interface br12
 ip address 192.168.12.1/24
!
interface br13
 ip address 192.168.13.1/24
!
interface br14
 ip address 192.168.14.1/24
```

14. **Association of VLANs with bridges configuration**

- Associate each VLAN with a bridge instance.

Association of VLANs with bridges configuration

```
interface vlan1
 bridge-group 1
!
interface vlan2
 bridge-group 2
!
interface vlan3
 bridge-group 3
!
interface vlan4
 bridge-group 4
!
interface vlan5
 bridge-group 5
!
interface vlan6
 bridge-group 6
!
interface vlan7
 bridge-group 7
!
interface vlan8
 bridge-group 8
!
interface vlan9
 bridge-group 9
!
interface vlan10
 bridge-group 10
!
interface vlan11
 bridge-group 11
!
interface vlan12
 bridge-group 12
!
interface vlan13
 bridge-group 13
!
interface vlan14
 bridge-group 14
```

15. **Association of VTI sub-interfaces with bridges configuration**

- Each VTI sub interface is linked to the appropriate bridge group.
- This final step ensures each incoming OpenVPN client connection is bridged to the appropriate VLAN interface, allowing each client to access their respective networks.

Association of VTI sub interfaces with bridges

```
interface tunnel1.1
 bridge-group 1
!
interface tunnel1.2
 bridge-group 2
!
interface tunnel1.3
 bridge-group 3
!
interface tunnel1.4
 bridge-group 4
!
interface tunnel1.5
 bridge-group 5
!
interface tunnel1.6
 bridge-group 6
!
interface tunnel1.7
 bridge-group 7
!
interface tunnel1.8
 bridge-group 8
!
interface tunnel1.9
 bridge-group 9
!
interface tunnel1.10
 bridge-group 10
!
interface tunnel1.11
 bridge-group 11
!
interface tunnel1.12
 bridge-group 12
!
interface tunnel1.13
 bridge-group 13
!
interface tunnel1.14
 bridge-group 14
```

Configuring OpenVPN client for Bridge TAP service

- Each OpenVPN client logs in using a unique user name and password, and an Open VPN client can be configured to use the following **.ovpn** config file and associated CA certificate.
- Some OpenVPN clients are able to reference the CA certificate file directly without having to paste in the certificate information into the .ovpn file template as below.

Example **.ovpn** config file

```
# tun.ovpn
client
auth-user-pass
cipher AES-128-CBC
dev tap
proto udp
remote 172.31.1.1
ca c:/users/support/cacert.pem
verb 7
```

- Create the file **cacert.pem** referred to in the **.ovpn** file on the AlliedWare Plus firewall or router, by using the command **crypto pki export local pem cacert.pem**.

Example **crypto pki export local pem cacert.pem** command

```
awplus#crypto pki export local pem cacert.pem
Copying...
Successful operation
```

Then, copy this file off the AlliedWare Plus firewall or router and onto the client device.

Example 4: Configuring OpenVPN with Client Certificate authentication

The following steps show how to configure OpenVPN client certificate authentication:

1. **Configure local RADIUS server for OpenVPN TAP mode**

```
awplus# configure terminal
```

- Specify a local RADIUS server host (IP address 127.0.0.1 indicates that the device itself is operating as the RADIUS server) and set parameters for the server.

```
awplus(config)# radius-server host 127.0.0.1 key awplus-local-radius-server
awplus(config)# aaa authentication openvpn default group radius
```

- Enter the local RADIUS server configuration mode.

```
awplus(config)# radius-server local
```

- Configure the client user group and configure the client IP address. Note that this step is optional for configuring OpenVPN TAP mode.

```
awplus(config-radsrv)# group client
```

- Configure the client user IP address. If you want to support more client users, you need to create a group for each client user. Note that if you want to configure the client IP address/mask with the RADIUS server, then this step is required. If you don't want to configure the client IP address/mask with the RADIUS server, then this step is not required and you can configure the client IP address via DHCP.

```
awplus(config-radsrv-group)# attribute Framed-IP-Address 198.51.100.11
```

- Configure the IP subnet mask of the tunnel interface. Note that if you want to configure the client IP address/mask with the RADIUS server, then this step is required. If you don't want to configure the client IP address/mask with the RADIUS server, then this step is not required and you can configure the client IP address via DHCP.

```
awplus(config-radsrv-group)# attribute Framed-IP-Netmask 255.255.255.0
```

- **Optional:** Configure the route for packets routing from network 198.51.100.0/16 to the remote network through the tunnel with 198.51.100.1 being the IP address of the remote tunnel interface.

```
awplus(config-radsrv-group)# attribute Framed-Route "198.51.0.0/16
198.51.100.1"
```

- Return to the local RADIUS server configuration mode.

```
awplus(config-radsrv-group)# exit
```

- Add the NAS with an IP address to the list of clients that may send authentication requests to the local RADIUS server. In this case, the NAS is the device itself, so the NAS address is 127.0.0.1.

```
awplus(config-radsrv)# nas 127.0.0.1 key awplus-local-radius-server
```

- Add a user to the RADIUS server database and specify the user name and password.

```
awplus(config-radsrv)# user remote password very_secret group client
```

- Enable the local RADIUS server.

```
awplus(config-radsrv)# server enable
```

2. **Configure client certificate authentication**

- Declare local CA (Certificate Authority) as the trust point that the system uses.

```
awplus(config)# crypto pki trustpoint local
```

- Obtain a system certificate from the local CA.

```
awplus# crypto pki enroll local
```

- Enroll the user to generate their client certificate, repeat this step for each user.

Example **crypto pki enroll** command

```
awplus#crypto pki enroll local user remote
Enter an export passphrase, or "abort" to cancel.

Enter the export passphrase again.

Generating a user private key for "remote"...
Successfully enrolled user "remote".
The PKCS#12 file is ready to export.
```

- Export the client certificate

Example **crypto pki export pkcs12** command

```
awplus#crypto pki export local pkcs12 remote tftp://192.0.2.1/remote.p12
Copying...
Successful operation
```

Save this file as you will need it when creating the client configuration file.

3. **Configure the interface connecting the device to the Internet**

Example to configure interface eth1 with IP address

```
awplus(config)#interface eth1
awplus(config-if)#ip address 172.31.1.1/24
```

4. **Enable OpenVPN TAP service**

- Create a virtual tunnel interface (VTI) for the OpenVPN router to be accessed by the client.

Example to create a VTI for the OpenVPN router

```
awplus(config-if)#interface tunnel1
awplus(config-if)#tunnel mode openvpn tap
awplus(config-if)#exit
```

5. **Connect OpenVPN clients to the LAN**

- Create a virtual Ethernet bridge to connect the VPN clients to the LAN.

```
awplus(config)# bridge 1
```

This newly created bridge will have two ports. One is the physical port eth2 that is connected to the LAN network. The other is the tunnel interface where the virtual OpenVPN TAP NIC will connect to.

- Assign eth2 and tunnel2 to the bridge.

Example to assign interfaces to the bridge

```
awplus(config)#interface eth2
awplus(config-if)#bridge-group 1
awplus(config)#interface tunnel1
awplus(config-if)#bridge-group 1
```

6. **Enable other traffic to be routed to the Internet**

Not all the traffic that enters the eth2 interface of the AlliedWare Plus firewall or router is destined to go to the OpenVPN clients. Much of the traffic is destined to simply be routed to the Internet. So, we need a method to route traffic out of the bridge instance and deliver it through eth1 to the Internet.

- To do this, configure an IP address on the bridge instance.

Example to configure an IP address on the bridge

```
awplus(config)#interface br1
awplus(config-if)# ip address 198.51.100.1/24
```

The logic for routing packets from the LAN to the Internet is as follows:

- When packets enter the AlliedWare Plus firewall or router via interface eth2, they are deemed to have entered Bridge Instance 1.
- If the destination IP address of the packet is not within the subnet of Bridge Instance 1 (192.168.1.0/24), then the packet needs to be routed out of the bridge instance.

7. **Enable OpenVPN Client-Certificate Authentication**

- Enter the interface configuration mode for the VTI.

```
awplus(config)# interface tunnel1
```

- Enable the Client-Certificate Authentication by selecting the trustpoint configured in "Configure client certificate authentication" on page 53.

```
awplus(config)# tunnel openvpn verify-client-certificate trustpoint
local
```

8. **Configure OpenVPN client for TAP service using Client-Certificate Authentication**

Several OpenVPN clients are available for many platforms. Most platforms have in common that they rely on an **.ovpn** file. Once the **.ovpn** file is created, client configuration is typically a matter of loading the file.

Use OpenSSL to extract the CA, client certificate and client key from the remote**.p12** file you exported previously. Enter the password you used when enrolling the user in "Configure client certificate authentication" on page 53.

In this example, the output is printed straight to the terminal. It is also possible to save it into three different files. See "Storing the certificates inline in the .ovpn file" on page 30 for information about using **openssl pkcs12 -help**.

Example configuring OpenVPN client for TAP service using Client-Certificate Authentication with output to the terminal

```
$ openssl pkcs12 -in remote.p12 -nodes
Enter Import Password:
Bag Attributes
    friendlyName: remote
    localKeyID: 05 F6 A1 1B 4E 2B 71 7D 1E 71 D8 65 BA BB E5 50 11 80 8E 51
subject=O = AlliedWare Plus, CN = remote

issuer=O = "Allied Telesis, Inc.", CN = AlliedWarePlusCAA05050G152000022

-----BEGIN CERTIFICATE-----
MIIDaTCCAlGgAwIBAgIJANZ2fYn7eV7+MA0GCSqGSIb3DQEBCwUAMEoxHTAbBgNV
D1RVKUF8ra37r/BUpNEPXvhUuWjAYe0Re7jpPU3fBEVsOi8gF5B4ZgyzXriiZqXW
gvKeTk5QbGoeih6NFkmQfuVSEWKOZ8x1vuYqop7v12PT9DVJDzjbRT7D6BJsciYv
KgnJQfpTUDpooA9pQs99VQHry8qn6uPBnvom8C+dwqiFm35GCUOBJ1LS4G+YSjO8
...
dALDs20SfUPegV57Hw==
-----END CERTIFICATE-----
Bag Attributes: <No Attributes>
subject=O = "Allied Telesis, Inc.", CN = AlliedWarePlusCAA05050G152000022

issuer=O = "Allied Telesis, Inc.", CN = AlliedWarePlusCAA05050G152000022

-----BEGIN CERTIFICATE-----
MIIDdDCCAlygAwIBAgIJALWqNifL5jplMA0GCSqGSIb3DQEBCwUAMEoxHTAbBgNV
Q6NVtGeUm5hmxDamqXeScxR8pJ5FCJmfI+lUQg7wRa8Qe1ltQnOoL9QJbhoxM++4
lstrdV4XzidGJDtmF9Q5sztPSeA7OYKdaZsK6/0xuHpNhV2GuRz/cW9LyNavWtxL
FNCJQxhovnro5z95Dgvickq0HzAM9kBqlQNkvHWYDaVOVtsk0JuwQDFH9G6SJkSq
...
A+oc8GS6uMQ5QJbDKenXfEoMs9Cg7+OM
-----END CERTIFICATE-----
Bag Attributes
    friendlyName: clientA1
    localKeyID: 05 F6 A1 1B 4E 2B 71 7D 1E 71 D8 65 BA BB E5 50 11 80 8E 51
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDhmn/hmlw73qTZ
TwwQCxdJJsNhxy6P8kT22gAlDjRhd01JP7J3IGbzfmqCCNVSLC4iH5ed8QKBgQCA
qM3gebEKg6VqnoZcOHnF6JPS1qF9rHFUDVnzhLlxdWeO3ZsZC1Phx5I6M5wxaAE+
eUjb8PLDUXM7wH46ahxn09ZA+e8U7f7yXqGfzo0p9z7TTkqNk/RyK9H+5vQAY08k
...
Y+Z+/8+mTJuwmHtYz/UWFCSM
-----END PRIVATE KEY-----
```

Copy the three different certificates including the -----BEGIN----- header and -----END----- footer into the appropriate inline tags in the **.ovpn** file. The first certificate with the line **CN = remote,** directly above this line is the client certificate. The CN should be the same as the clients username, which in this case is ‘remote’.

The following OpenVPN TAP mode client **.ovpn** file:

Example **.ovpn** file

```
#Configure for client mode
client
#The server requires the client to provide a username/password for
authentication.
auth-user-pass
#Require encryption
cipher AES-128-CBC
#Configure for TAP mode
dev tap
proto udp
#The address of the OpenVPN router to connect to
remote 172.31.1.1

#The CA certificate
<ca>
-----BEGIN CERTIFICATE-----
MIIDdDCCAlygAwIBAgIJALWqNifL5jplMA0GCSqGSIb3DQEBCwUAMEoxHTAbBgNV
Q6NVtGeUm5hmxDamqXeScxR8pJ5FCJmfI+lUQg7wRa8Qe1ltQnOoL9QJbhoxM++4
lstrdV4XzidGJDtmF9Q5sztPSeA7OYKdaZsK6/0xuHpNhV2GuRz/cW9LyNavWtxL
FNCJQxhovnro5z95Dgvickq0HzAM9kBqlQNkvHWYDaVOVtsk0JuwQDFH9G6SJkSq
...
A+oc8GS6uMQ5QJbDKenXfEoMs9Cg7+OM
-----END CERTIFICATE-----
</ca>

#The client certificate
<cert>
-----BEGIN CERTIFICATE-----
MIIDaTCCAlGgAwIBAgIJANZ2fYn7eV7+MA0GCSqGSIb3DQEBCwUAMEoxHTAbBgNV
D1RVKUF8ra37r/BUpNEPXvhUuWjAYe0Re7jpPU3fBEVsOi8gF5B4ZgyzXriiZqXW
gvKeTk5QbGoeih6NFkmQfuVSEWKOZ8x1vuYqop7v12PT9DVJDzjbRT7D6BJsciYv
KgnJQfpTUDpooA9pQs99VQHry8qn6uPBnvom8C+dwqiFm35GCUOBJ1LS4G+YSjO8
...
dALDs20SfUPegV57Hw==
-----END CERTIFICATE-----
</cert>

#The client key
<key>
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDhmn/hmlw73qTZ
TwwQCxdJJsNhxy6P8kT22gAlDjRhd01JP7J3IGbzfmqCCNVSLC4iH5ed8QKBgQCA
qM3gebEKg6VqnoZcOHnF6JPS1qF9rHFUDVnzhLlxdWeO3ZsZC1Phx5I6M5wxaAE+
eUjb8PLDUXM7wH46ahxn09ZA+e8U7f7yXqGfzo0p9z7TTkqNk/RyK9H+5vQAY08k
...
Y+Z+/8+mTJuwmHtYz/UWFCSM
-----END PRIVATE KEY-----
</key>
```

Samples of **.ovpn** file templates can be found on the OpenVPN website. These sample templates typically include explanations of the various **.ovpn** file configuration options and advice on default settings.

Each client will need their own unique **.ovpn** file. Once you have distributed the configuration file to the user, they can load it into their OpenVPN client application. When they connect, they will still be prompted to enter their username and password.

Example 5: Configuring basic two-factor authentication

The basic steps to get a single user setup and working with default two-factor authentication configuration are as follows:

1. **Start the 2FA service.**

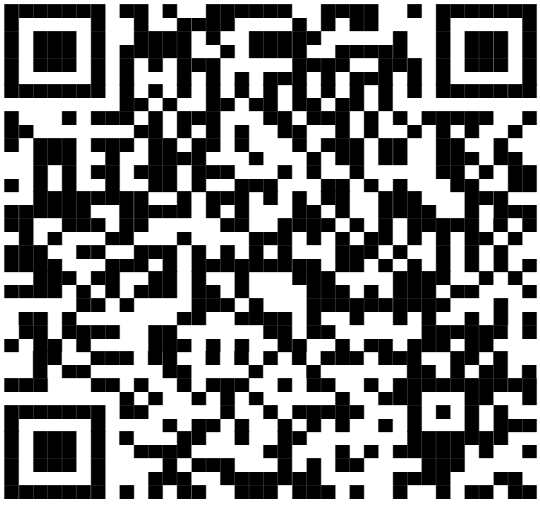
```
awplus>enable
awplus# configure terminal
awplus(config)# service 2fa
awplus(config)# exit
```

2. **Create the 2FA data for user ‘test’ and display a UTF-8 QR code to scan.**

This is done in privileged exec mode as the two-factor data is not stored in the running-configuration.

```
awplus#2fa create user test random-secret qr utf8
Two-Factor Authentication information for user:

Username:         test
Secret:           RIVS...
Mode:             TOTP
OTP URL:          otpauth://totp/test@awplus?secret=RIV...&issuer=Allied
Scratch codes:
  44857801
  39444714
  48774624
  89142923
  52234486
```



3. **Scan the QR code into the user's phone.**

If you need to do this step at a later stage, use the **show 2fa user test qr utf8** command to see the user’s QR code.

4. **Add two-factor authentication to the OpenVPN authentication method list.**

```
awplus# configure terminal
awplus(config)# aaa authentication openvpn default group radius 2fa
```

If you want the user to append the 2FA code to the password in the password field then use the following command:

```
awplus(config)# aaa authentication openvpn default group radius 2fa 2fa-in-
password
```

Stop 2FA

Stopping the 2FA service stops the 2FA configuration from showing in the running-configuration and prevents 2FA commands from working. This means, it will not be possible to view, create, or delete users with the service stopped. User data, however, is not deleted.

Additionally, the 2FA configuration is not reset until the device is rebooted, so the configuration will be restored if the service is restarted before the device is rebooted.

If the OpenVPN method list is configured with 2FA and the 2FA service is not running, two-factor authentication will be skipped and a critical message will be logged when a user connects.

Use the following commands to stop the 2FA service:

```
awplus>enable
awplus# configure terminal
awplus(config)# no service 2fa
```

Example 6: Configuring users for two-factor authentication

HOTP (HMAC-based One-Time Password)

HOTP mode users use an incrementing counter instead of the time for generating a code. The process for creating a HOTP user is the same as for creating a basic TOTP user, except for the addition of the **hotp** parameter in step 2.

In this example an HOTP user, ‘test2’, is created and a link to generate the QR code is displayed. (This link is instead of generating the QR code in the terminal window and can be used for HOTP or TOTP users.)

```
awplus#2fa create user test random-secret hotp qr link
Two-Factor Authentication information for user:

Username:         test2
Secret:           GEO...
Mode:             HOTP (counter: 1)
OTP URL:          otpauth://hotp/test2@awplus?secret=GEO...&issuer=Allied
Scratch codes:
  87118538
  41797033
  83083103
  49097462
  55084521

The following URL can be used to generate a QR code.
This results in the user key being sent to Google ser vers.
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://hotp/
test2@awplus%3Fsecret%3DGEO...%26issuer%3DAllied
```

Email

Email 2FA users will be emailed an OTP to the specified email, instead of using a code generated by an application with the secret key. The following example creates an email OTP user 'test1' with the email address test1@xyz.com.

```
awplus# 2fa create user test1 email test1@xyz.com
```

2FA self-registration

Users who self-register will be TOTP users with a random secret. To configure self-registration, first the service needs to be enabled using the following commands:

```
awplus(config)# service 2fa
awplus(config)# 2fa self-registration port <1-65535>
```

Note: The port should not be set to the same port as the HTTP server. It can be set to the same port as the HTTPS server.

Configure AAA authentication for 2FA self-registration as follows.

To use LDAP servers to authenticate the user for 2FA self-registration, use the command:

```
awplus(config)# aaa authentication 2fa-registration default group ldap
```

To use RADIUS servers to authenticate the user for 2FA self-registration, use the command:

```
awplus(config)# aaa authentication 2fa-registration default group radius
```

To use selected LDAP or RADIUS servers defined as a group ‘GRP1’ to authenticate the user for 2FA self-registration, use the command:

```
awplus(config)# aaa authentication 2fa-registration default group GRP1
```

How to self-register for OpenVPN 2FA

Unregistered users can visit the device web page and after successful authentication by username and password, they can self-register and use a TOTP Authenticator app to generate one-time passwords afterwards.

1. Visit the device URL to start 2FA self-registration.

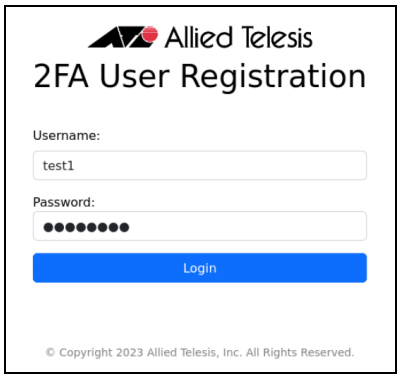
   For example, for the device 192.168.1.1, with the port set to 443, visit the web page https://192.168.1.1:1080/2fa-registration.

2. Enter username, password, and click **Login**.

3. On success, the web page shows the QR and scratch codes.

4. Scan the QR code from a TOTP Authenticator app.

5. Use the one-time password for an OpenVPN connection.



Register a user to skip 2FA

When 2FA is enabled on routers for OpenVPN, it is possible to allow some users to skip 2FA. Then those users do not need to generate one-time passwords from a TOTP authenticator app or email OTP code. The user configuration is stored similarly to 2FA users. For example, to allow the user ‘test1’ to skip 2FA, use the command:

```
awplus# 2fa create user test1 skip-2fa
```

Predefined secret

In some cases, it may be useful to add 2FA users with a user-defined secret rather than generating a random secret. For example, if the administrator would like the user to connect to multiple devices with the same code. This is only available for application-based 2FA users.

The following command creates an TOTP user, ‘test’, with the secret ‘MySecret’. The option to display the QR code has not been passed, so no QR code is displayed.

Note: Scratch codes are always randomly generated.

```
awplus#2fa create user test secret MySecret
Two-Factor Authentication information for user:

Username:         test
Secret:           MySecret
Mode:             TOTP
OTP URL:          otpauth://hotp/test@awplus?secret=MySecret&issuer=Allied
Scratch codes:
  44116392
  57816368
  58145620
  64911718
  39054267
```

Delete user

To delete the data for a 2FA user, (including TOTP, HOTP, manual email users or users configured to skip 2FA, use the command:

```
awplus# 2fa delete user test1
```