OpenVPN Configuration Guide, 17.2.0

November 2017

Supporting AT&T Vyatta Network Operating System

Contents

Copyright Statement

The training materials and other content provided herein for assistance in training on the Vyatta vRouter may have references to Brocade as the Vyatta vRouter was formerly a Brocade product prior to AT&T's acquisition of Vyatta. Brocade remains a separate company and is not affiliated to AT&T.

About This Guide

This guide describes how to configure OpenVPN on the AT&T Vyatta Network Operating System (referred to as a virtual router, vRouter, or router in the guide).

OpenVPN Overview

OpenVPN security mechanisms

This section provides an introduction to the security mechanisms and modes of operation for OpenVPN on the AT&T Vyatta vRouter.

- Preshared secret
- TLS

The security requirements for a virtual private network include authentication, confidentiality, and integrity. OpenVPN offers a choice of two different security mechanisms: preshared secret and Transport Layer Security (TLS).

> **Note:** Secure Sockets Layer (SSL) is the predecessor of TLS, and most current references to SSL are, in fact, references to TLS. Therefore, these terms are used interchangeably in this document.

Preshared secret

When preshared secret is used for security, OpenVPN works as follows:

1. The administrator uses the `generate openvpn key` command to generate a file that contains a certain number of random data bytes, that is, the secret to be used to provide security.
2. The administrator transfers the secret file to each of the two tunnel endpoints by using pre-established secure channels. For example, the file can be generated on one of the endpoints and then transferred to the other endpoint by using a secure file transfer protocol, such as SCP.
3. When the two endpoints need to establish the VPN tunnel, the OpenVPN process on the one endpoint authenticates the other endpoint. Authentication is based on the assumption that the preshared secret is known only to the other endpoint; that is, authentication is based on the assumption that if any host knows the shared secret, that host must be the other endpoint.
4. After the endpoints are authenticated, the OpenVPN process on each side derives a set of keys from the preshared secret. These keys are used for two purposes.

   Info:

   - Some keys are used in an encryption algorithm to encrypt the tunnel data. This encryption provides data confidentiality.
   - The others are used in a message authentication code (MAC) that uses a hash algorithm with the keys on the tunnel data. This code provides data integrity.

TLS

Transport Layer Security (TLS) is a cryptographic protocol that uses public key cryptography and does not require the two endpoints to have a preshared secret. OpenVPN uses TLS with X.509 certificates and requires public key infrastructure (PKI) to generate the certificates. When TLS is used, OpenVPN works as follows:

1. Using PKI, the administrator generates a certificate and the associated files for each endpoint. All certificates are “signed”# by the certificate authority (CA) of the PKI. The certificate for an endpoint contains many pieces of information, one of which is the name of the endpoint, which is stored in the Common Name field of the certificate.
2. The administrator transfers each certificate and the associated files to the corresponding endpoint by using a pre-established, secure channel (for example, SCP).
3. When two endpoints need to establish the VPN tunnel, one endpoint takes a passive role while the other endpoint must take an active role and initiate the TLS session with the passive endpoint.
4. After the active endpoint initiates the TLS session, the two sides authenticate each other by using their public-private key pairs and the public key of the CA, which is known to both endpoints.

5. After the two endpoints have authenticated each other, they establish a shared secret by using public-key cryptography. Each endpoint then derives a set of keys for the session. As for the preshared secret mechanism, these keys are then used for encryption and MAC on the tunnel data to provide data confidentiality and integrity. However, unlike the preshared secret mechanism, these keys are used only for the one session, and for this reason they are called “session keys.”#

Certificate generation and distribution using PKI involves numerous complex security issues, which are outside the scope of this document.

OpenVPN modes of operation

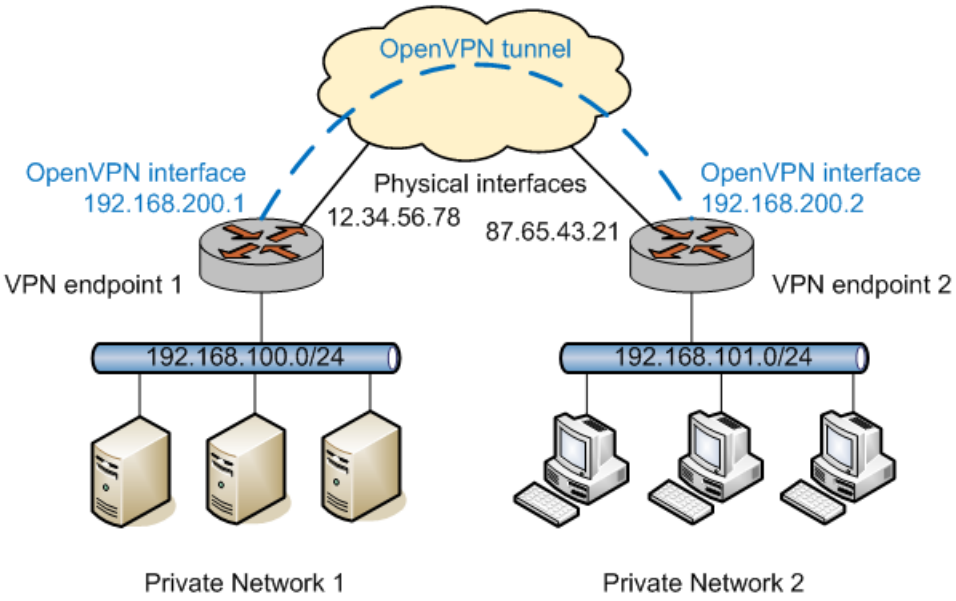
OpenVPN supports both site-to-site and remote access operation. In addition, client-side remote access support is available for accessing configuration information from an OpenVPN Access Server.

Note: If client-side access to OpenVPN Access Server is configured, all OpenVPN configuration parameters other than those used to connect to OpenVPN Access Server (that is, those within the `interfaces openvpn` *vtunx* **`remote-configuration`** command) are ignored.

Site-to-site operation

The following figure illustrates a simple site-to-site VPN operation. This operation could represent, for example, a connection between a branch office and a data center.

Figure 1: Site-to-site operation



At each of the two VPN tunnel endpoints, the OpenVPN process creates a routable “tunnel interface” and establishes a secure tunnel with the other endpoint. Subsequently, the two interfaces appear to be on the same network, although packets flowing between these two interfaces are actually processed and sent through the secure tunnel by the OpenVPN process.

Note that each endpoint has two relevant IP addresses.

- The tunnel IP address: This address is the virtual IP address (VIP) on each end of the tunnel. The tunnel IP address at each end of the tunnel must be on the same subnet. In the previous figure, the tunnel IP addresses of the two endpoints are 192.168.200.1 and 192.168.200.2.
- The physical IP address: This address is the IP address that is configured for the physical network interface over which the VPN tunnel is established. In the preceding figure, the physical IP addresses of the two endpoints are 12.34.56.78 and 87.65.43.21.

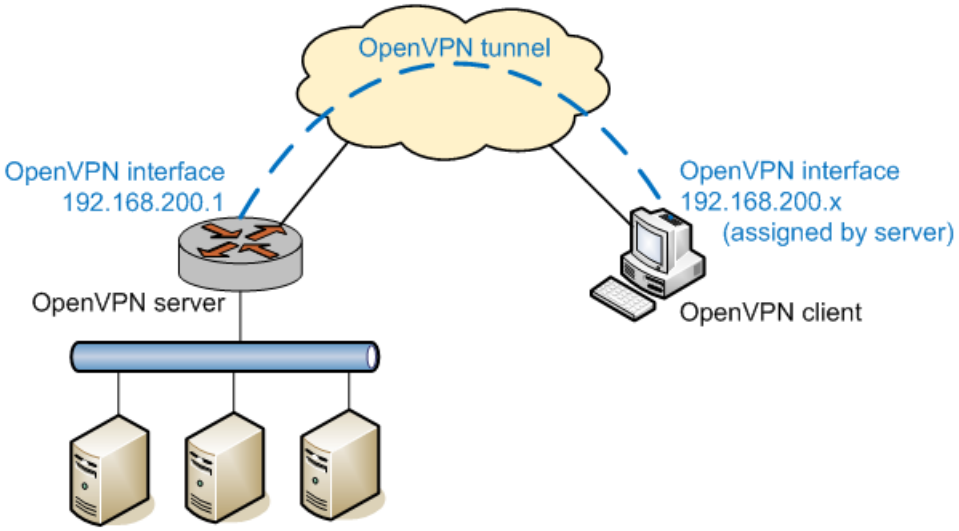
In most operations, the VPN tunnel transports traffic from different private subnets across the wide area network (WAN). In the preceding figure, each of the 192.168.100.0/24 and 192.168.101.0/24 private subnets is "behind" a VPN tunnel endpoint. Therefore, on each endpoint, you must add a static route that directs traffic to and from the remote private subnet through the tunnel interface.

In site-to-site mode, a single host can establish multiple OpenVPN tunnels, each of which may be to distinct sites. Even if all tunnels originate from a single physical interface, each tunnel is represented by a different tunnel interface IP address and operates independently.

Remote access operation

OpenVPN also supports remote access VPN that uses a client-server mode. In this mode, one OpenVPN endpoint acts as the server and all remote endpoints operate as clients, which connect to the OpenVPN server to establish VPN tunnels, so that each established client has an independent tunnel to the server. The following figure shows a simple remote access VPN setup.

Figure 2: Remote access operation



One major difference between site-to-site mode and remote access mode is that in remote access mode, all the VPN tunnels on the server side terminate at a single tunnel interface. A single termination point eliminates the need to set up separate tunnel interface IP addresses for each VPN tunnel. This single termination point is more convenient and operationally simpler for a remote access setup.

Another difference is that in remote access mode, the server-side OpenVPN process dynamically allocates all tunnel IP addresses from a configured subnet (192.168.200.0/24 in the example) instead of using fixed tunnel IP addresses for tunnel endpoints. Thus, when the OpenVPN process starts on the server, it creates the tunnel interface and assigns it an IP address from the subnet to the interface (for example, 192.168.200.1). Then, when a client establishes a VPN tunnel with the server, the server-side OpenVPN process also allocates the client an IP address from the same subnet (for example, 192.168.200.4) and the tunnel interface on the client adopts this address.

Client-Side access to OpenVPN access server

OpenVPN Access Server is a server that authenticates remote client access requests (either locally or through an authentication server) and provides OpenVPN tunnel configuration information to the requesting client. It also provides OpenVPN client software if the client requires it, although this is not required for AT&T Vyatta vRouter clients. The configuration information allows the client to then establish an OpenVPN tunnel and an OpenVPN server with minimal configuration on the client side.

The sequence of events is as follows:

1. An administrator configures OpenVPN Access Server for AT&T Vyatta vRouter client access and, potentially, configures a separate authentication server and OpenVPN server. The client requires only configuration information from the server. It does not require client software.

   Info:

   Note: It is possible for OpenVPN Access Server to act as the access server, authentication server, and OpenVPN server.

   Note: OpenVPN Access Server is not available from AT&T. It is available from OpenVPN Technologies, Inc. at http://openvpn.net.

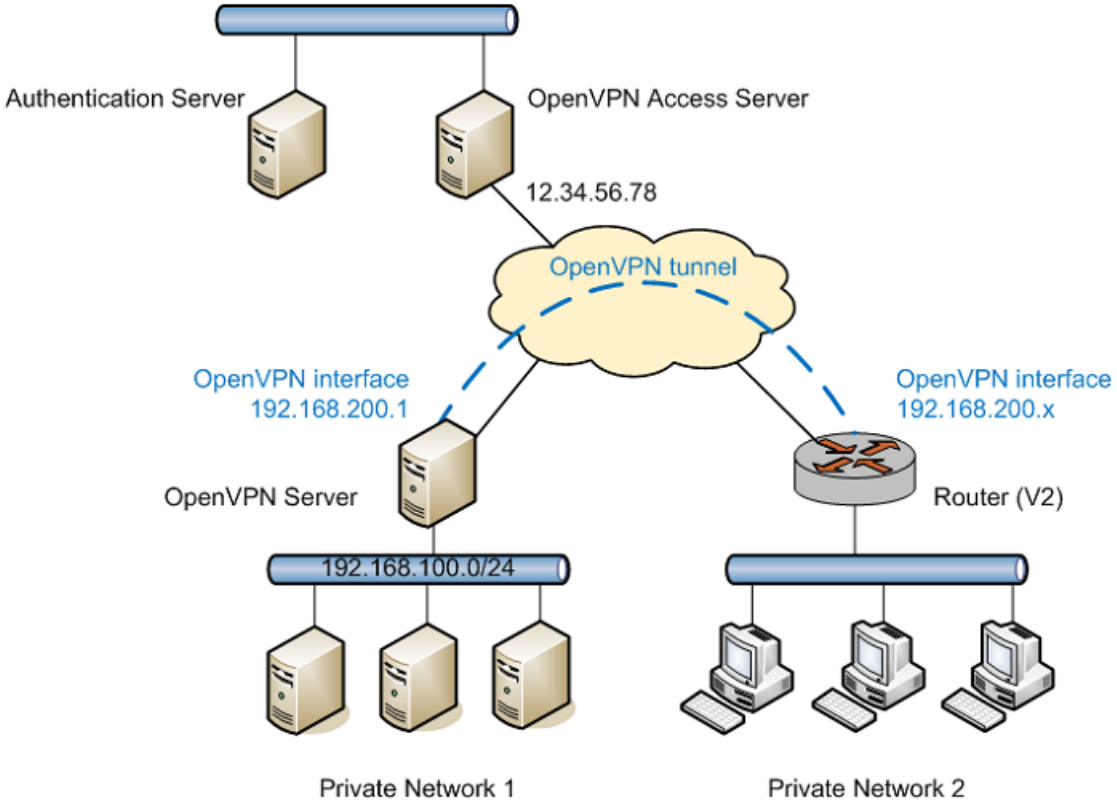
2. The AT&T Vyatta vRouter client accesses OpenVPN Access Server and provides a username and password.
3. OpenVPN Access Server authenticates the user, either acting as its own authentication server or by using an external authentication server, such as a RADIUS server.
4. After authentication, OpenVPN Access Server sends the AT&T Vyatta vRouter client device the configuration information needed to establish an OpenVPN tunnel with an OpenVPN server.
5. The AT&T Vyatta vRouter client then establishes an OpenVPN tunnel with the OpenVPN server specified in the downloaded configuration and is provided an IP address on the OpenVPN tunnel subnet.

Note: If the OpenVPN server is configured such that Autologin is enabled, then a tunnel username and tunnel password are not required; otherwise, they are required to establish the VPN tunnel.

The AT&T Vyatta vRouter has the OpenVPN client software preloaded and can use OpenVPN Access Server to obtain the information needed to establish an OpenVPN tunnel with an OpenVPN server. The only required configuration information is the IP address or host name of OpenVPN Access Server, a username and password for OpenVPN Access Server, and, potentially, the tunnel username and tunnel password for establishing the tunnel with the OpenVPN server.

The following figure shows an OpenVPN setup that uses OpenVPN Access Server, an authentication server, and an OpenVPN server.

Figure 3: Client#side access to OpenVPN access server



You can use the `show interfaces` command to show the assigned IP address on the client side of the OpenVPN tunnel.

OpenVPN Configuration

Basic usage scenarios

The following list describes basic scenarios of OpenVPN usage.

- Site-to-Site mode with preshared secret
- Site-to-Site mode with TLS
- Remote access mode
- OpenVPN clients on Windows Hosts
- Firewall configuration
- OpenVPN access server

Site-to-site mode with preshared secret

The following figure shows site-to-site VPN configured with preshared secret.

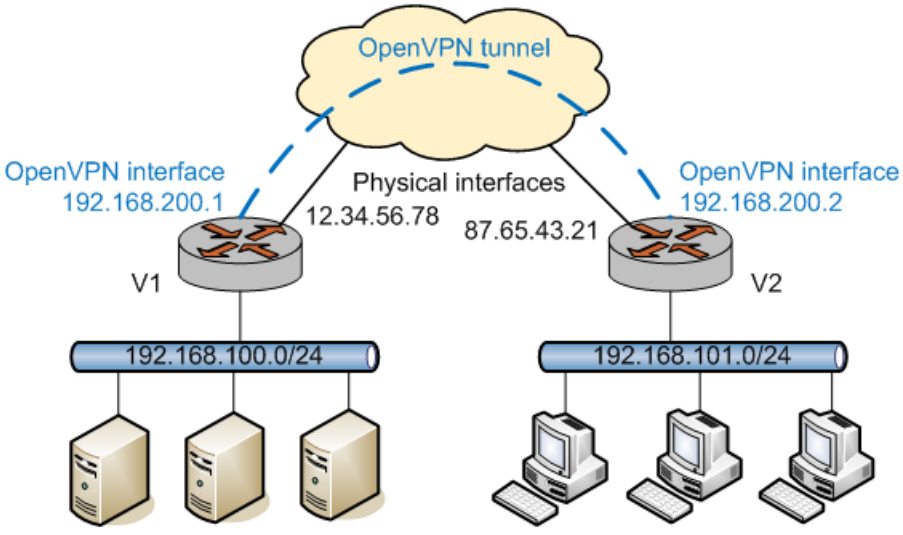
In this example:

- The physical IP addresses for V1 and V2 are 12.34.56.78 and 87.65.43.21, respectively.
- The tunnel IP addresses for V1 and V2 are 192.168.200.1 and 192.168.200.2, respectively.
- The subnet to be accessed from V1 (through V2 over the VPN) is 192.168.100.0/24.
- The subnet to be accessed on V2 (through V1 over the VPN) is 192.168.101.0/24.

To configure an OpenVPN tunnel, you create an interface of the openvpn type. The interface name is in the form of vtun*num*; for example, vtun0, vtun1, and so on.

In addition, you must add a static interface route to direct traffic for the remote subnet through the vtun0 tunnel interface. For information on setting up static routes, see AT&T Vyatta Network Operating System Basic Routing Configuration Guide.

Figure 4: Site-to-site VPN with preshared secret



To configure the V1 endpoint, perform the following steps in configuration mode.

Table 1: Site-to-site OpenVPN with preshared secret: V1 endpoint

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Set the tunnel IP address for the local endpoint.	`vyatta@V1# set interfaces openvpn vtun0 local-address 192.168.200.1`
Set the OpenVPN mode to site-to-site.	`vyatta@V1# set interfaces openvpn vtun0 mode site-to-site`
Set the tunnel IP address of the remote endpoint.	`vyatta@V1# set interfaces openvpn vtun0 remote-address 192.168.200.2`
Specify the physical IP address of the remote host.	`vyatta@V1# set interfaces openvpn vtun0 remote-host 87.65.43.21`
Specify the location of the file containing the preshared secret.	`vyatta@V1# set interfaces openvpn vtun0 shared-secret-key-file /config/auth/secret`
Commit the change.	`vyatta@V1# commit`

Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn vtun0`
	`local-address 192.168.200.1`
	`mode site-to-site`
	`remote-address 192.168.200.2`
	`remote-host 87.65.43.21`
	`shared-secret-key-file /config/auth/secret`

To configure a static route to access the remote subnet through the OpenVPN tunnel, perform the following steps in configuration mode.

Table 2: Site-to-site OpenVPN with preshared secret: V1 static route

Step	Command
Create the static route to access the remote subnet through the OpenVPN tunnel.	`vyatta@V1#set protocols static interface-route 192.168.101.0/24 next-hop-interface vtun0`
Commit the change.	`vyatta@V1# commit`

Show the static routing configuration.	`vyatta@V1# show protocols static`
	`interface-route 192.168.101.0/24 {`
	`next-hop-interface vtun0 {`
	`}`
	`}`

The V2 VPN endpoint is identical to the V1 endpoint, except that local and remote tunnel IP addresses are reversed. To configure the V2 endpoint, perform the following steps in configuration mode.

Table 3: Site-to-site OpenVPN with preshared secret: V2 endpoint

Step	Command
Create the vtun0 configuration node.	`vyatta@V2# set interfaces openvpn vtun0`
Set the tunnel IP address for the local endpoint.	`vyatta@V2# set interfaces openvpn vtun0 local-address 192.168.200.2`
Set the OpenVPN mode to site-to-site.	`vyatta@V2# set interfaces openvpn vtun0 mode site-to-site`
Set the tunnel IP address of the remote endpoint.	`vyatta@V2# set interfaces openvpn vtun0 remote-address 192.168.200.1`
Specify the physical IP address of the remote host.	`vyatta@V2#set interfaces openvpn vtun0 remote-host 12.34.56.78`
Specify the location of the file containing the preshared secret.	`vyatta@V2# set interfaces openvpn vtun0 shared-secret-key-file /config/auth/secret`
Commit the change.	`vyatta@V2# commit`

Show the OpenVPN configuration.	`vyatta@V2# show interfaces openvpn vtun0`
	`local-address 192.168.200.2`
	`mode site-to-site`
	`remote-address 192.168.200.1`
	`remote-host 12.34.56.78`
	`shared-secret-key-file /config/auth/secret`

Again, the shared secret file (created by generating the key with the `generate openvpn key` command on one system and then copying the key to the other system) must be the same on both endpoints (the path need not be the same, but the content must be). Note also that the **`remote-host`** option is required only on one of the endpoints; that is, the site-to-site tunnel can be established as long as even one endpoint has enough information to contact the other.

To configure a static route to access the remote subnet through the OpenVPN tunnel, perform the following steps in configuration mode.

Table 4: Site-to-site OpenVPN with preshared secret: V2 static route

Step	Command
Create the static route to access the remote subnet through the OpenVPN tunnel.	`vyatta@V2# set protocols static interface-route 192.168.100.0/24 next-hop-interface vtun0`
Commit the change.	`vyatta@V2# commit`

Step	Command

Show the static routing configuration.	`vyatta@V2# show protocols static`
	` interface-route 192.168.100.0/24 {`
	`  next-hop-interface vtun0 {`
	`  }`
	` }`

Site-to-site mode with TLS

When TLS is used in site-to-site mode, the AT&T Vyatta vRouter configuration is the same as described in the previous section, except that you must configure TLS-related options instead of the **`shared-secret-key-file`** option. As previously discussed, one endpoint takes the passive role and the other takes the active role.

Each endpoint must also have the following files, which are required for the TLS protocol.

- Certificate Authority (CA) certificate file: This file contains the certificate of the CA, which is used to validate the certificate of the other endpoint.
- Host certificate file: This file contains the certificate of the endpoint, which is presented to the other endpoint during the TLS negotiation.
- Host key file: This file contains the private key of the endpoint, which is kept secret from anybody else.
- Certificate revocation list (CRL) file: (Optional) This file contains a list of certificates that have been revoked, which prevent endpoints with these certificates from establishing a VPN tunnel.
- DH parameters file: (Only needed by the passive endpoint) This file contains Diffie Hellman parameters that are required only by the endpoint taking the passive role in the TLS negotiation.

More information about these files is available in the OpenVPN documentation.

The configuration that follows corresponds to the configuration for the example in the previous section. Assume that the necessary files have been generated and distributed to each endpoint and that V1 and V2 are passive and active, respectively.

To configure V1 for a site-to-site VPN with TLS, perform the following steps in configuration mode.

Table 5: V1 OpenVPN configuration: site-to-site with TLS

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Set the local IP address of the VPN tunnel.	`vyatta@V1# set interfaces openvpn vtun0` ` local-address 192.168.200.1`
Set the OpenVPN mode.	`vyatta@V1# set interfaces openvpn vtun0 mode` ` site-to-site`
Set the remote IP address of the VPN tunnel.	`vyatta@V1# set interfaces openvpn vtun0` ` remote-address 192.168.200.2`
Specify the physical IP address of the remote host.	`vyatta@V1# set interfaces openvpn vtun0` ` remote-host 87.65.43.21`
Set the role of this endpoint.	`vyatta@V1# set interfaces openvpn vtun0 tls` ` role passive`

Step	Command
Specify the location of the CA certificate file.	`vyatta@V1# set interfaces openvpn vtun0 tls ca-cert-file /config/auth/ca.crt`
Specify the location of the host certificate file.	`vyatta@V1# set interfaces openvpn vtun0 tls cert-file /config/auth/V1.crt`
Specify the location of the CRL parameters file.	`vyatta@V1# set interfaces openvpn vtun0 tls crl-file /config/auth/crl.pem`
Specify the location of the DH file.	`vyatta@V1# set interfaces openvpn vtun0 tls dh-file /config/auth/dh1024.pem`
Specify the location of the host key file.	`vyatta@V1# set interfaces openvpn vtun0 tls key-file /config/auth/V1.key`
Commit the change.	`vyatta@V1# commit`

Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn vtun0`
	` local-address 192.168.200.1`
	` mode site-to-site`
	` remote-address 192.168.200.2`
	` remote-host 87.65.43.21`
	` tls {`
	`  role passive`
	`  ca-cert-file /config/auth/ca.crt`
	`  cert-file /config/auth/V1.crt`
	`  crl-file /config/auth/cr1.pem`
	`  dh-file /config/auth/dh1024.pem`
	`  key-file /config/auth/V1.key`
	` }`

Note that the configuration is the same as in the previous section except that the **shared-secret-key-file** option has been replaced by **tls** options. The V1 endpoint takes the passive role, so the **dh-file** option is required. The **crl-file** option is also specified in this example.

To configure V2 for a site-to-site VPN with TLS, perform the following steps in configuration mode.

Table 6: V2 OpenVPN configuration: site-to-site with TLS

Step	Command
Create the vtun0 configuration node.	`vyatta@V2# set interfaces openvpn vtun0`
Set the local IP address of the VPN tunnel.	`vyatta@V2# set interfaces openvpn vtun0 local-address 192.168.200.2`
Set the OpenVPN mode.	`vyatta@V2# set interfaces openvpn vtun0 mode site-to-site`

Step	Command
Set the remote IP address of the VPN tunnel.	`vyatta@V2# set interfaces openvpn vtun0 remote-address 192.168.200.1`
Specify the physical IP address of the remote host.	`vyatta@V2# set interfaces openvpn vtun0 remote-host 12.34.56.78`
Set the role of this endpoint.	`vyatta@V2# set interfaces openvpn vtun0 tls role active`
Specify the location of the CA certificate file.	`vyatta@V2#set interfaces openvpn vtun0 tls ca-cert-file /config/auth/ca.crt`
Specify the location of the host certificate file.	`vyatta@V2# set interfaces openvpn vtun0 tls cert-file /config/auth/V2.crt`
Specify the location of the host key file.	`vyatta@V2# set interfaces openvpn vtun0 tls key-file /config/auth/V2.key`
Commit the change.	`vyatta@V2# commit`

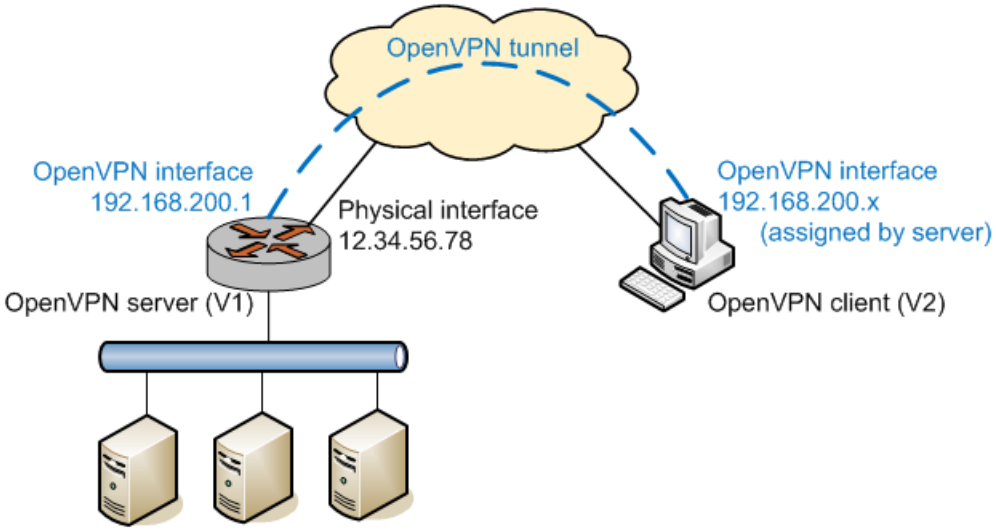
Show the OpenVPN configuration.	`vyatta@V2# show interfaces openvpn vtun0`
	`local-address 192.168.200.2`
	`mode site-to-site`
	`remote-address 192.168.200.1`
	`remote-host 12.34.56.78`
	`tls {`
	`role active`
	`ca-cert-file /config/auth/ca.crt`
	`cert-file /config/auth/V2.crt`
	`key-file /config/auth/V2.key`
	`}`

The configuration is the same as in the previous example except that the **tls** option is specified; the **crl-file** option is not specified; and, because the V2 endpoint takes the active role, the **dh-file** option is not needed.

Remote access mode

The following figure illustrates a typical remote access VPN setup in which one OpenVPN endpoint acts as the server. Remote users run OpenVPN as clients to connect to the server and establish VPN tunnels.

Figure 5: Remote access mode



Note that OpenVPN requires TLS in remote access mode, and the server takes the passive role while the clients are active. Therefore, it is not necessary to specify the **`tls role`** option when operating in this mode. In the preceding figure, assuming that V1 is the server and V2 is a client, the configuration for V1 is shown below.

To configure V1 for remote access with TLS, perform the following steps in configuration mode. The example has the following characteristics.

- The **`mode`** option specifies that this endpoint operates in server mode.
- The **`server subnet`** option indicates that the tunnel IP address of the client is allocated from the 192.168.200.0/24 subnet and that the tunnel IP address of the server (that is, the address of vtun0 on the server) is 192.168.200.1.
- The **`remote-host`** option is not set because the clients are actively contacting the server.

Table 7: V1 OpenVPN configuration: remote access with TLS (server)

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Set the OpenVPN mode.	`vyatta@V1# set interfaces openvpn vtun0 mode server`
Set the subnet for the OpenVPN tunnel.	`vyatta@V1# set interfaces openvpn vtun0 server subnet 192.168.200.0/24`
Specify the location of the CA certificate file.	`vyatta@V1# set interfaces openvpn vtun0 tls ca-cert-file /config/auth/ca.crt`
Specify the location of the host certificate file.	`vyatta@V1# set interfaces openvpn vtun0 tls cert-file /config/auth/V1.crt`
Specify the location of the CRL parameters file.	`vyatta@V1# set interfaces openvpn vtun0 tls crl-file /config/auth/crl.pem`

Step	Command
Specify the location of the DH file.	`vyatta@V1# set interfaces openvpn vtun0 tls dh-file /config/auth/dh1024.pem`
Specify the location of the host key file.	`vyatta@V1# set interfaces openvpn vtun0 tls key-file /config/auth/V1.key`
Commit the change.	`vyatta@V1# commit`

Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn vtun0`
	`mode server`
	`server {`
	`  subnet 192.168.200.0/24`
	`}`
	`tls {`
	`  ca-cert-file /config/auth/ca.crt`
	`  cert-file /config/auth/V1.crt`
	`  crl-file /config/auth/cr1.pem`
	`  dh-file /config/auth/dh1024.pem`
	`  key-file /config/auth/V1.key`
	`}`

To configure V2 for remote access with TLS, perform the following steps in configuration mode. This example has the following characteristics.

- V2 is in client mode and so it needs to actively contact the server; therefore, the **`remote-host`** option is needed to indicate the location of the server.
- When the tunnel is established, the tunnel IP address of V2 (that is, the address of vtun0 on V2) is assigned by V1 from the 192.168.200.0/24 subnet.

Table 8: V2 OpenVPN configuration: remote access with TLS (client)

Step	Command
Create the vtun0 configuration node.	`vyatta@V2# set interfaces openvpn vtun0`
Set the OpenVPN mode.	`vyatta@V2# set interfaces openvpn vtun0 mode client`
Specify the physical IP address of the remote host.	`vyatta@V2# set interfaces openvpn vtun0 remote-host 12.34.56.78`
Specify the location of the CA certificate file.	`vyatta@V2# set interfaces openvpn vtun0 tls ca-cert-file /config/auth/ca.crt`
Specify the location of the host certificate file.	`vyatta@V2# set interfaces openvpn vtun0 tls cert-file /config/auth/V2.crt`
Specify the location of the host key file.	`vyatta@V2# set interfaces openvpn vtun0 tls key-file /config/auth/V2.key`

Step	Command
Commit the change.	`vyatta@V2# commit`

Show the OpenVPN configuration.	`vyatta@V2# show interfaces openvpn vtun0`
	` mode client`
	` remote-host 12.34.56.78`
	` tls {`
	`  ca-cert-file /config/auth/ca.crt`
	`  cert-file /config/auth/V2.crt`
	`  key-file /config/auth/V2.key`
	` }`

OpenVPN clients on windows hosts

As mentioned earlier, OpenVPN is different from and cannot interoperate with the “SSL VPN” solutions on the market, and therefore OpenVPN must be installed on all VPN hosts. In a remote access VPN setup, many remote users will need to connect to the OpenVPN server from hosts that run Windows. To set up the OpenVPN client on a Windows machine, download and install the OpenVPN Windows Installer package from the OpenVPN Web at (http://openvpn.net/index.php/downloads.html).

After installation, the OpenVPN client is either run from the Windows command line or controlled by the OpenVPN GUI. Using the setup from the previous section as an example, if the V2 client is a Windows host, the OpenVPN client is run from the command line by issuing the command shown in the following example, using the address, certificate, and key information for your site.

Running OpenVPN from the command line

```
openvpn --dev tun --client --remote ip-address
 --ca ca-cert-filename
 --cert endpoint-cert-filename
 --key endpoint-key-filename
```

This command establishes a VPN tunnel with the V1 OpenVPN server shown in Table 2 *(page 21)*. Note that the referenced files must be in the same directory from which this command is issued. Otherwise, full paths should be used for the files.

Alternatively, to control the OpenVPN client by using the OpenVPN GUI, you must create a control file. The file must be named with the .ovpn extension, for example, vyatta.ovpn. A configuration file that corresponds to the preceding command line would look as shown in the following example (with corresponding changes for your site information).

OpenVPN configuration file

```
dev tun
client
remote 12.34.56.78
ca ca.crt
cert V2.crt
key V2.key
```

Put the configuration file and the referenced files (certificates, etc.) into the OpenVPN configuration directory. This directory is usually `C:\Program files\OpenVPN\config`.

Start the OpenVPN GUI, which shows an icon in the notification area of the Windows taskbar. To establish the OpenVPN tunnel, right-click the icon and select **Connect** from the drop-down menu. If there are multiple .ovpn configuration files, the actions for each configuration appear in the drop-down menu of each file.

Firewall configuration

The firewall configuration for an OpenVPN tunnel interface is the same as the configuration for other types of interfaces. Here is an example.

To configure firewall on V1, perform the following steps in configuration mode.

Table 9: V1 OpenVPN firewall configuration

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Enter configuration commands.	`...`
Set the firewall rule for inbound traffic on the vtun0 interface.	`vyatta@V1# set interfaces openvpn vtun0 firewall in rules-in`
Enter configuration commands.	`...`
Commit the change.	`vyatta@V1# commit`

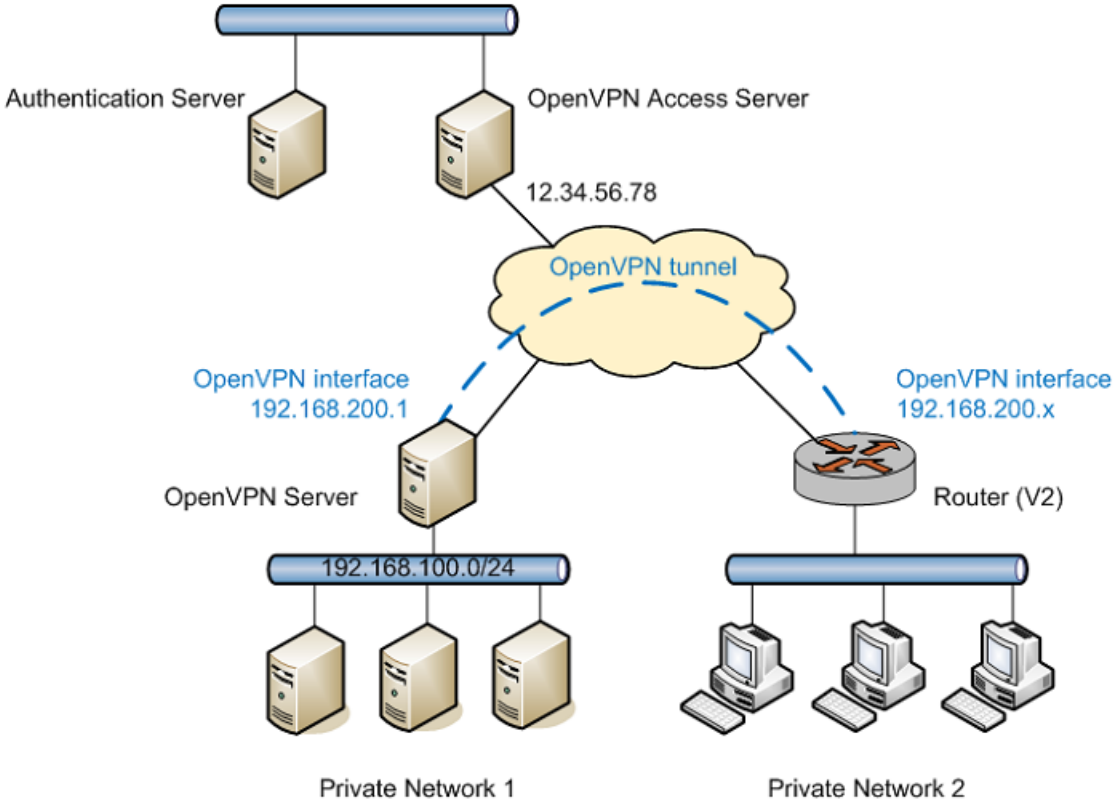
Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn`
	`openvpn vtun0 {`
	`  firewall {`
	`  in rules-in`
	`  }`
	`}`

For more information on configuring firewall for interfaces, see the firewall chapter in AT&T Vyatta Network Operating System Firewall Configuration Guide.

OpenVPN access server

Another OpenVPN scenario involves connecting to OpenVPN Access Server and using the configuration information it provides to establish an OpenVPN tunnel to an OpenVPN server. The configuration for this scenario is very simple because the OpenVPN Access Server provides all the necessary VPN configuration information to the connecting host (the AT&T Vyatta vRouter in this case). The following figure shows a configuration that uses OpenVPN Access Server.

Figure 6: Using OpenVPN access server to establish an OpenVPN tunnel



To configure V2 to establish an OpenVPN tunnel to an OpenVPN server using an OpenVPN Access Server as shown in the previous figure, perform the following steps in configuration mode.

Table 10: V2: Client-Side connection to OpenVPN access server (Autologin enabled)

Step	Command
Create the vtun0 configuration node.	`vyatta@V2# set interfaces openvpn vtun0`
Specify the OpenVPN Access Server IP address.	`vyatta@V2# set interfaces openvpn vtun0 remote-configuration server 12.34.56.78`
Specify the username to be authenticated by OpenVPN Access Server.	`vyatta@V2# set interfaces openvpn vtun0 remote-configuration username abcd`
Specify the password to be authenticated by OpenVPN Access Server.	`vyatta@V2# set interfaces openvpn vtun0 remote-configuration password efgh`
Commit the change.	`vyatta@V2# commit`

Step	Command

Show the configuration.	`vyatta@V2# show interfaces openvpn vtun0`
	` remote-configuration {`
	`  password efgh`
	`  server 12.34.56.78`
	`  username abcd`
	` }`

This example is valid for a scenario in which Autologin is enabled on the OpenVPN server for tunnel establishment. If Autologin is disabled, the following commands must be used to establish the tunnel:

- `interfaces openvpn` *vtunx* **`remote-configuration tunnel-username`** *username*
- `interfaces openvpn` *vtunx* **`remote-configuration tunnel-password`** *password*

To configure V2 to establish an OpenVPN tunnel to an OpenVPN server (with Autologin disabled) using OpenVPN Access Server as shown in the previous figure, perform the following steps in configuration mode.

Table 11: V2: client-side connection to OpenVPN access server (Autologin disabled)

Step	Command
Create the vtun0 configuration node.	`vyatta@V2# set interfaces openvpn vtun0`
Specify the OpenVPN Access Server IP address.	`vyatta@V2# set interfaces openvpn vtun0` ` remote-configuration server 12.34.56.78`
Specify the username to be authenticated by OpenVPN Access Server.	`vyatta@V2# set interfaces openvpn vtun0` ` remote-configuration username abcd`
Specify the password to be authenticated by OpenVPN Access Server.	`vyatta@V2# set interfaces openvpn vtun0` ` remote-configuration password efgh`
Specify the username required to establish the tunnel with the OpenVPN server.	`vyatta@V2# set interfaces openvpn vtun0` ` remote-configuration tunnel-username tun-un3`
Specify the password required to establish the tunnel with the OpenVPN server.	`vyatta@V2# set interfaces openvpn vtun0` ` remote-configuration tunnel-password tun-pwdxyz`
Commit the change.	`vyatta@V2# commit`

Show the configuration.	`vyatta@V2# show interfaces openvpn vtun0`
	` remote-configuration {`
	`  password efgh`
	`  server 12.34.56.78`
	`  tunnel-password tun-un3`
	`  tunnel-username tun-pwdxyz`
	`  username abcd`
	` }`

Advanced OpenVPN options

The previous section presented some basic OpenVPN scenarios and provided configuration steps for the AT&T Vyatta vRouter. This section presents a number of more-advanced concepts and configuration options that may be useful to administrators of more complex environments.

- Transport protocol (Site-to-Site, Client, Server)
- Cryptographic algorithms (Site-to-Site, Client, Server)
- Split tunneling (Site-to-Site, Client, Server)
- Broadcast network (Site-to-Site, Client, Server)
- Multiple remote endpoints (Client only)
- Remote access topology (Server only)
- Client-specific settings (Server only)

Transport protocol (site-to-site, client, server)

By default, OpenVPN uses User Datagram Protocol (UDP) as the underlying transport protocol. Because UDP is connectionless, either side can initiate the VPN tunnel by sending packets to UDP port 1194 (default) on the other endpoint. Alternatively, OpenVPN can also use TCP as the transport. However, if TCP is used, one endpoint must take a passive role (that is, it listens to incoming TCP connections), and the other endpoint must take an active role (that is, it initiates the TCP connection to the TCP port on the passive endpoint).

Each protocol has different advantages in this context. For example, using TCP is much less prone to firewall or NAT problems in networks between the two endpoints. However, when packet losses occur, the TCP retransmissions at the tunnel level may interfere with retransmissions from the individual TCP flows inside the VPN tunnel; therefore, using UDP can result in better performance.

The following example shows and describes the related configuration options.

Configuration options related to protocol type

```
interfaces {
    openvpn if_name {
        protocol protocol
        local-host local_host_ip
        local-port local_port
        remote-port remote_port
    }
}
```

- **protocol**: This argument is **udp**, **tcp-active**, or **tcp-passive**. If **protocol** is not specified or if it is specified as **udp**, then UDP is used. On the other hand, if TCP is used, note the following requirements.
  - As previously discussed, when TCP is used, one endpoint must be active and the other one passive.
  - On the **tcp-active** endpoint, the **remote-host** option must be set so that it can initiate the TCP connection.
  - On the **tcp-passive** endpoint, if the **remote-host** option is set, then only the specified host can initiate the TCP connection to this endpoint.
  - If TCP is used in remote access mode, the client must be **tcp-active** and the server must be **tcp-passive**.
  - When TCP combines with TLS, the active and passive roles for TCP and TLS should match. In other words, the **tcp-active** endpoint should also be active for TLS (similarly for passive). Note that this match is not an OpenVPN restriction, but it is enforced to avoid confusion.
- **local-host**: This argument is an IP address on any of the network interfaces on this endpoint. If **local-host** is set, the OpenVPN process accepts only sessions coming in on the particular IP address. This acceptance applies to both UDP and TCP. If **local-host** is not set, OpenVPN accepts incoming sessions on any interface. This argument can be used for any of the following:
  - The server endpoint in remote access mode
  - Either endpoint when UDP is used in site-to-site mode

- The **tcp-passive** endpoint when TCP is used in site-to-site mode

- **local-port**: This argument is the UDP or TCP port number on which OpenVPN accepts incoming sessions. If not set, OpenVPN accepts incoming sessions on the default port of 1194. This argument can be used for any of the following:
  - The server endpoint in remote access mode
  - Either endpoint when UDP is used in site-to-site mode
  - The **tcp-passive** endpoint when TCP is used in site-to-site mode
- **remote-port**: This argument is the UDP or TCP port number on the other endpoint to which OpenVPN initiates sessions. In other words, the other endpoint is accepting sessions on this port. If not set, OpenVPN initiates the session to the default port of 1194 on the remote endpoint. Note that, if set, the **remote-port** setting on one endpoint must match the **local-port** setting on the other, and conversely. This argument can be used for any of the following:
  - The client endpoint in remote access mode
  - Either endpoint when UDP is used in site-to-site mode
  - The **tcp-active** endpoint when TCP is used in site-to-site mode

Cryptographic algorithms (site-to-site, client, server)

As previously discussed, whichever security mechanism is used (preshared secret or TLS), after the VPN tunnel is established, the two endpoints apply an encryption algorithm and a hash algorithm on the tunneled VPN data to provide confidentiality and integrity. By default, the encryption and hash algorithms used by OpenVPN are Blowfish (with 128-bit keys) and SHA-1, respectively. This configuration should be reasonable in typical environments: the Blowfish algorithm performs well in software and has no known weakness, and SHA-1 is widely used and is part of the NIST Secure Hash Standard.

When a particular encryption or hash algorithm is required in an environment, the two configuration options shown in the following example can be used to specify the algorithm.

Configuration options related to security

```
interfaces {
    openvpn if_name{
        encryption algorithm
        hash algorithm
  }
}
```

- **encryption**: This argument is one of the following algorithms:
  - **des**: DES algorithm
  - **3des**: DES algorithm with triple encryption
  - **bf128**: Blowfish algorithm with 128-bit key
  - **bf256**: Blowfish algorithm with 256-bit key
  - **aes128**: AES algorithm with 128-bit key
  - **aes192**: AES algorithm with 192-bit key
  - **aes256**: AES algorithm with 256-bit key
- **hash**: This argument is one of the following hash algorithms:
  - **md5**: MD5 algorithm
  - **sha1**: SHA-1 algorithm
  - **sha256**: SHA-256 algorithm
  - **sha512**: SHA-512 algorithm

Split Tunneling (site-to-site, client, server)

When the OpenVPN tunnel is established between the two endpoints, by default, only the VPN traffic is routed through the tunnel. Other traffic, such as packets going to other places on the Internet, is still routed using

the normal default route, not through the VPN tunnel. This split tunneling occurs because two tunnels (are considered) to exist: the normal traffic route and the VPN tunnel.

On the one hand, split tunneling is very efficient because non-VPN traffic (for example, Internet traffic) travels through the normal route. In a remote access VPN setup, for example, split tunneling means that Internet traffic from a remote user travels to and from the user ISP directly without going to the VPN server, company network, firewall, and so on. On the other hand, bypassing these functions can be considered a security issue because, in such cases, the Internet traffic is not filtered or protected according to a company policy.

To disable split tunneling, use the configuration shown in the following example.

Configuration options related to split tunneling

```
interfaces {
    openvpn if_name {
        replace-default-route {
            local
        }
    }
}
```

- **replace-default-route**: This argument tells OpenVPN that the default route should be replaced by a route through the VPN tunnel, that is, split tunneling should be disabled. Note that, when set, this option has different effects depending on the OpenVPN mode in which the endpoint operates.
    - If the endpoint is in site-to-site mode or client mode, using **replace-default-route** replaces the default route on this endpoint with a route through VPN tunnel. In other words, it disables split tunneling on this endpoint.
    - If the endpoint is in server mode, using **replace-default-route** causes the clients connecting to this server to replace their default route. In other words, it disables split tunneling on the clients.
- **local**: This keyword under **replace-default-route** must be set if and only if the two tunnel endpoints are directly connected, i.e., on the same subnet.

Of course, because the OpenVPN tunnel interface is routable, static routes can be added, with or without split tunneling, to override the default behavior.

Broadcast network (site-to-site, client, server)

By default, an OpenVPN interface is configured as a “tun” device. A tun device is a virtual network interface that operates on Layer 3 (network layer) traffic, such as IP packets. There are cases in which the virtual interface needs to operate on Layer 2 (link layer) traffic. One example of this need is when subnets on each end of a tunnel must reside on the same subnet. In this case, the two subnets must be bridged across the tunnel. Bridging occurs on Layer 2. Another example is when a DHCP Relay resides on one side of a tunnel and the DHCP Server or DHCP clients reside on the other side. Clients must broadcast DHCP discovery messages and require a broadcast network to broadcast these messages. Because of this necessity, DHCP Relay requires that all interfaces to which it binds are broadcast interfaces.

A “tap” device is a virtual network interface that operates on Layer 2 (link layer) traffic and provides a broadcast network. A tap device is automatically configured by the system if the OpenVPN tunnel is to be used to bridge two subnets. If an OpenVPN tunnel is added to a bridge group then a tap device is implied and does not need to be configured explicitly. For cases that do not involve bridging, a tap device must be configured explicitly by using the `interfaces openvpn` *vtunx* **`device-type tap`** command.

Client and server configuration

To configure an OpenVPN client or server as a tap device, use the configuration shown in the following example.

Configuration options related to tap devices for client and server interfaces

```
interfaces {
    openvpn if_name{
        device-type
        tap
    }
```

```
}
```

- **device-type tap**: This argument tells OpenVPN that the tunnel is to be used as a tap device and operate on Layer 2 traffic. This configuration is required on both ends of the OpenVPN tunnel.

Site-to-site configuration

For site-to-site configurations, in addition to configuring the interface as tap device, you must also indicate the subnet mask for the local address that is specified. To configure an OpenVPN site-to-site interface as a tap device, use the configuration shown in the following example.

Configuration options related to tap devices for site-to-site interfaces

```
interfaces {
    openvpn if_name{
        device-type
            tap
        local-address ipv4 {
            subnet-mask mask
        }
    }
}
```

- **device-type tap**: This argument tells OpenVPN that the tunnel is to be used as a tap device and operate on Layer 2 traffic. This configuration is required on both ends of the OpenVPN tunnel.
- **local-address**: This argument is the IP address at the local end of the OpenVPN tunnel.
- **subnet-mask**: This argument is the subnet mask for **local-address** (for example, 255.255.255.0).

Multiple remote endpoints (client only)

In remote access mode, the **remote-host** argument must be specified on the client endpoints so that the clients can initiate the VPN sessions. In some environments, the administrator may want the clients to have a list of servers to provide some redundancy— if one of the servers fails, a client can try the next one. In the AT&T Vyatta vRouter, this server list can be configured by specifying multiple **remote-host** entries.

To configure multiple endpoints on V2, perform the following steps in configuration mode.

Table 12: V2 OpenVPN multiple endpoints configuration

Step	Command
Create the vtun0 configuration node.	`vyatta@V2# set interfaces openvpn vtun0`
Enter configuration commands.	`...`
Specify the physical IP address of the first remote host.	`vyatta@V2# set interfaces openvpn vtun0 remote-host 12.34.56.78`
Specify the physical IP address of the second remote host.	`vyatta@V2# set interfaces openvpn vtun0 remote-host 12.34.56.79`
Specify the physical IP address of the third remote host.	`vyatta@V2# set interfaces openvpn vtun0 remote-host 12.34.56.80`
Set the firewall rule for inbound traffic on the vtun0 interface.	`vyatta@V2# set interfaces openvpn vtun0 firewall in name rules-in`

Step	Command
Enter configuration commands.	`...`
Commit the change.	`vyatta@V2# commit`

Show the OpenVPN configuration.	`vyatta@V2# show interfaces openvpn vtun0`
	` ...`
	` remote-host 12.34.56.78`
	` remote-host 12.34.56.79`
	` remote-host 12.34.56.80`
	` ...`

When multiple entries are specified, a client starts from the beginning of the list and attempts to establish a VPN tunnel with the first remote host. If the first host does not work, the client tries the second one, and so on.

Note that multiple **remote-host** entries can also be specified in site-to-site mode. However, because the two endpoints are most likely fixed in this mode, such usage probably does not make sense in most cases.

Remote access topology (server only)

In remote access mode, two different remote access topologies can be configured by using the **topology** argument. The two different topologies are **subnet** and **point-to-point**, as shown in the following example.

Configuration options related to topology

```
interfaces {
    openvpn if_name{
        server {
            topology [subnet|point-to-point]
        }
    }
}
```

The **topology** argument primarily specifies how the tunnel interface is configured, how the addresses are allocated, and so on. At a high level, these topologies have the following implications.

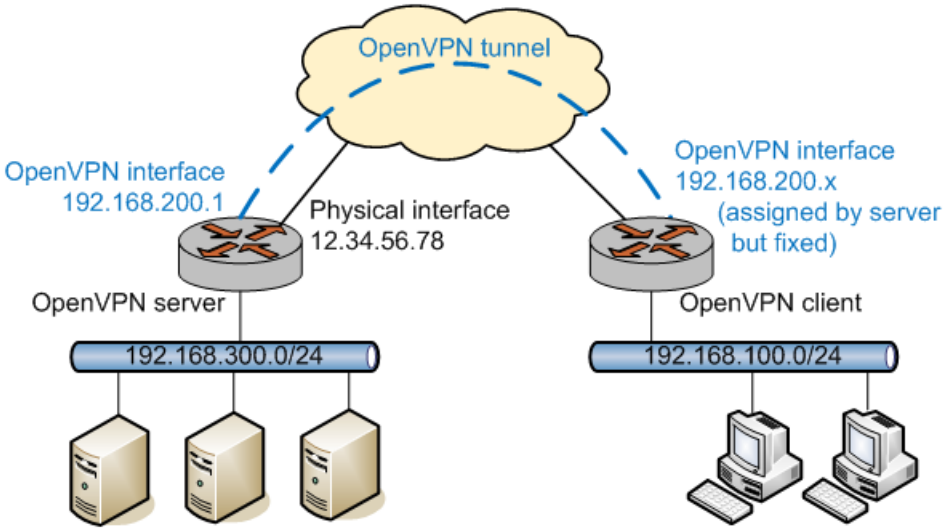
- **subnet**: This topology is compatible with OpenVPN clients on Windows hosts and is the default if **topology** is not used. Routing protocols that are configured to use a broadcast-style network are suited to this topology. However, this topology does not provide client isolation; that is, clients can reach one another.
- **point-to-point**: This topology is not compatible with Windows clients, and routing protocols using a broadcast-style network do not work with this topology. On the other hand, this topology provides client isolation.

Client-specific settings (server only)

In a typical remote access VPN setup, the clients are remote users—for example, users trying to access the company private network from home. Therefore, when a client establishes a VPN tunnel with the VPN server, it only needs to ensure that the client host itself can access the private network; so, it can use any tunnel IP address assigned by the server.

However, in some environments, the remote access mode is used to implement site-to-site functionality; that is, each client is in fact a site that establishes, in effect, a site-to-site tunnel with the server. The following figure illustrates this functionality.

Figure 7: Remote access mode



In such an environment, it may be useful to give a fixed IP address to each OpenVPN client. Furthermore, in such cases there may be a private network behind a client as well, and the OpenVPN server needs to determine that traffic destined to this private network should be routed to the particular client. Similarly, there may be networks behind the OpenVPN server that the client needs to access. In other words, these client-specific settings are tied to a particular client, and they can be configured by using the options that are shown in the following example and explained after the example.

Configuration options related to remote access mode

```
interfaces {
    openvpn if_name {
        server {
            client client_name {
               ip client_ip
               push-route ipv4net
               subnet client_subnet
            }
        }
    }
}
```

- **`client`**: This argument is the name for the client; this name corresponds to the common name specified in the certificate of the client. When a client initiates the VPN session, the server uses the name in the certificate to look up and apply client-specific settings (if any).
- **`ip`**: This argument is the fixed IP address that is assigned to the particular client.
- **`push-route`**: This argument is the network address of a network behind the OpenVPN server to which the client can route traffic. Multiple networks can be specified with multiple **`push-route`** configuration statements.
  `subnet`: This argument is the private subnet behind the particular client, and the OpenVPN process routes traffic destined to this subnet to the client. Multiple networks can be specified with multiple **`subnet`** configuration statements.

  Note that this setting only informs the OpenVPN server to which client the traffic for this subnet should be routed. However, before the OpenVPN server is in a position to make this decision, the traffic must be routed to the tunnel interface, so that it is processed by the OpenVPN server. For this reason, a static interface route must be added separately to direct traffic for this subnet to the tunnel interface.

In the preceding example, the V1 server can be configured with the client settings specific to the V2 client as follows (note that a static interface route is also needed for the subnet of the V2 client).

To configure this scenario, perform the following steps in configuration mode.

Table 13: V1 OpenVPN configuration: site-to-site with preshared secret

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Enter configuration commands.	`...`
Create the server configuration node.	`vyatta@V1# set interfaces openvpn vtun0 server`
Enter configuration commands.	`...`
Create the V2 client configuration node.	`vyatta@V1# set interfaces openvpn vtun0 server client V2`
Specify the IP address of the client.	`vyatta@V1# set interfaces openvpn vtun0 server client V2 ip 192.168.200.100`
Specify the subnet at the server that the client can access.	`vyatta@V1# set interfaces openvpn vtun0 server client V2 push-route 192.168.300.0/24`
Set the subnet at the client.	`vyatta@V1# set interfaces openvpn vtun0 server client V2 subnet 192.168.100.0/24`
Enter configuration commands.	`...`
Commit the change.	`vyatta@V1# commit`

Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn vtun0`
	`...`
	`server {`
	`  ...`
	`  client V2 {`
	`  ip 192.168.200.100`
	`  push-route 192.168.300.0/24`
	`  subnet 192.168.100.0/24`
	`  }`
	`  ...`
	`}`
	`...`

To configure the static interface route to access the remote subnet through the OpenVPN tunnel, perform the following steps in configuration mode.

Table 14: V1 static interface route configuration

Step	Command
Create the static interface route to access the remote subnet through the OpenVPN tunnel.	`vyatta@V1# set protocols static interface-route 192.168.100.0/24 next-hop-interface vtun0`
Commit the change.	`vyatta@V1# commit`

Show the static routing configuration.	`vyatta@V1# show protocols static`
	`interface-route 192.168.100.0/24 {`
	`next-hop-interface vtun0 {`
	`}`
	`}`

Using unsupported OpenVPN options

OpenVPN has over two hundred options, not all of which are feasible to support in the AT&T Vyatta vRouter. At the same time, the administrator of a particular environment might require OpenVPN options not supported by the AT&T Vyatta vRouter configuration. For these cases, the AT&T Vyatta vRouter provides the **openvpn-option** configuration attribute; this attribute allows any OpenVPN option to be specified, as shown in the following example.

The openvpn-option configuration attribute

```
interfaces {
    openvpn if_name{
        openvpn-option options
    }
}
```

The text of the **openvpn-option** attribute is passed directly (without any validation) to OpenVPN when OpenVPN is invoked, as if the text had been typed on the OpenVPN command line by the user. Therefore, multiple options can be entered together as shown in the following example.

To configure this example, perform the following steps in configuration mode.

Table 15: Entering multiple OpenVPN options using openvpn-option

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Enter configuration commands.	`...`
Set the desired OpenVPN options.	`vyatta@V1# set interfaces openvpn vtun0` `openvpn-option "--verb 5 --secret /config/auth/secret 1"`
Enter configuration commands.	`...`
Commit the change.	`vyatta@V1# commit`

Step	Command

Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn vtun0`
	`  ...`
	`  openvpn-option "--verb 5 --secret /config/auth/secret 1"`
	`  ...`

It is also possible to enter the commands separately as shown in the example that follows.

To configure this example, perform the following steps in configuration mode.

Table 16: Entering multiple OpenVPN options through multiple commands using openvpn-option

Step	Command
Create the vtun0 configuration node.	`vyatta@V1# set interfaces openvpn vtun0`
Enter configuration commands.	`...`
Set another desired OpenVPN option.	`vyatta@V1# set interfaces openvpn vtun0` `  openvpn-option "--secret /config/auth/secret 1"`
Set a desired OpenVPN option.	`vyatta@V1# set interfaces openvpn vtun0` `  openvpn-option "--verb 5"`
Enter configuration commands.	`...`
Commit the change.	`vyatta@V1# commit`

Show the OpenVPN configuration.	`vyatta@V1# show interfaces openvpn vtun0`
	`  ...`
	`  openvpn-option "--secret /config/auth/secret 1"`
	`  openvpn-option "--verb 5"`
	`  ...`

No validation is done on this setting; therefore, when using it, you should make sure that the specified OpenVPN options and their values (if any) are valid. Furthermore, because many OpenVPN options conflict with one another, you should also ensure that the specified options do not conflict with one another or with any other OpenVPN options that are configured through the AT&T Vyatta vRouter configuration. Finally, some OpenVPN options require coordination between the two endpoints (for example, the value must be 0 on one side and 1 on the other), and you must ensure such constraints are met.

AT&T SSL-VPN Client Bundler

Overview

AT&T Secure Sockets Layer-Virtual Private Network (SSL-VPN) Client Bundler enables the AT&T Vyatta vRouter to generate image bundles that facilitate the setup of SSL-VPN client connections. Bundles include the up-to-date SSL-VPN client configuration that is required to connect to the server, including the required Transport Layer Security (TLS) certificate authority (CA) certificate that is used by the server.

Supported operating systems

AT&T SSL-VPN Client Bundler supports the following operating systems as SSL-VPN clients:

- Windows: Windows Vista, Windows 7, and Windows 8
- Mac OS X: Supported by OpenVPN frontends that support generic OpenVPN configuration files
- Linux: Network Manager applets and OpenVPN CLI

Client bundles

All generated SSL-VPN client bundles are user independent. The bundles hold only SSL-VPN server instance-specific information. A new SSL-VPN client bundle is generated automatically whenever the SSL-VPN server configuration is changed and requires a new SSL-VPN client configuration. No client certificates for authentication are used at this stage—the authentication token username and password are used for authentication against the local service user database for the AT&T Vyatta vRouter or against a cooperating directory server (for example, Lightweight Directory Access Protocol (LDAP)).

Service-User web portal client bundle

SSL-VPN end users can obtain the generated bundles through the Service-User Web Portal of the AT&T Vyatta vRouter after the bundles have been configured. See Service-User web portal *(page 39)*.

For the Service-User Web Portal, the same credentials are required for authentication. The AT&T Vyatta vRouter administrator can obtain the bundles directly from the Vyatta file-system and distribute the bundles to the end users individually.

Windows client bundle

The SSL-VPN client bundle for Windows operating system includes the SSL-VPN client software and the dedicated SSL-VPN connection.

The AT&T SSL-VPN client bundle is distributed as part of the installation wizard and includes the following:

- Latest version of the SSL-VPN client software (a virtual network device driver)
- SSL-VPN client configuration file
- Required TLS CA certificate

The AT&T SSL-VPN client software includes an easy-to-use GUI for the end user to maintain the SSL-VPN connection. The installation wizard is distributed as an executable (.exe) file.

Linux client bundle

The Linux SSL-VPN client bundle is a .zip archive that holds an OpenVPN configuration file and the required TLS CA certificate of the server. The TLS CA certificate is intentionally not compatible with the configuration because some certificates have already been released and are currently being used.

The Network Manager applets for Linux require that the TLS CA certificate is a file that is separate from the image.

OS X client bundle

The OS X SSL-VPN client bundle is a generic OpenVPN-formatted configuration file (.ovpn) with the TLS CA certificate included. This file is imported onto the OS X operating system of OpenVPN clients and allows the user to use the imported VPN connection profile.

Administration of the client bundle

This section covers the setup and administration of the SSL-VPN client bundle on the AT&T Vyatta vRouter.

Generating the client bundle

The following example shows how configure the generation of the SSL-VPN client bundle after using the `interfaces openvpn` commands.

Table 17: Configuring the generation of the client bundle

Step	Command
Configure the OpenVPN tunnel interface for authentication. See Authentication of the client bundle *(page 38)*.	`vyatta@vyatta# set interfaces openvpn vtunX auth ...`
Configure the path to the file that contains the TLS CA certificate, which is part of the client bundle.	`vyatta@vyatta# set interfaces openvpn vtunX tls ca-cert-file filename_of_the_TLS_CA_certificate`
Configure the SSL-VPN server address to use for the client bundle.	`vyatta@vyatta# set interfaces openvpn vtunX local-host SSL-VPN_server_address`
Configure the SSL-VPN server port to use for the client bundle.	`vyatta@vyatta# set interfaces openvpn vtunX local-port SSL-VPN_server_port`
Configure the client certificate on the SSL-VPN server: client bundles do not use TLS client certificates for authentication—they are not required on the SSL-VPN connection.	`vyatta@vyatta# set interfaces openvpn vtunX client-cert-not-required`
Set a description for the name of the SSL-VPN endpoint.	`vyatta@vyatta# set interfaces openvpn vtunX description SSL-VPN_endpoint_name_for_end_user`

The **`client-cert-not-required`** keyword must be set to allow SSL-VPN clients to connect without a TLS client certificate that is specific to an end user. Even if client certificates were created, they are not included in any SSL-VPN client bundles.

The description serves as the identifier for various objects. In non-OpenVPN interfaces, the description serves as the network interface alias and is shown in the administration web interface that appears in the dashboard overview.

In the context of the SSL-VPN client bundle, the description is also used in the following cases:

- The Service-User Web Portal and is presented to the end user as the name of the SSL-VPN instance or endpoint
- Name of the SSL-VPN client as the profile name that is inside the AT&T SSL-VPN client
- Tunnelblick
- The Linux Network Manager applets
- File names of client bundles

Note: Tip: use an end-user friendly name to distinguish between potential different SSL-VPN endpoints or AT&T Vyatta vRouter instances, for example: ACME HQ, ACME EMEA, and so on. Setting the description to ACME HQ results in client bundle files, which the user has to download, with names like ACME HQ v1.exe, ACME HQ v1.zip, and so forth.

In addition to the mandatory settings, settings that are shown in the following example influence the configuration of the client bundle.

The following example shows how to configure additional settings for the client bundle.

Table 18: Configuring the generation of the client bundle

Step	Command
Configure the system for the hash algorithm.	`vyatta@vyatta# set interfaces openvpn vtunX hash hash_algorithm`
Configure the system for an encryption method.	`vyatta@vyatta# set interfaces openvpn vtunX encryption encryption_method`
Configure the system for a transport protocol.	`vyatta@vyatta# set interfaces openvpn vtunX protocol transport_protocol_to_use`

When optional settings or mandatory settings are changed, a new version of the SSL-VPN client bundles is generated during the next configuration commit.

To enable client bundle configuration, you must specify for which operating systems the bundles needs to be set.

By default, no client bundle is generated if no operating system is explicitly configured.

The following example shows how to create on commit all three operating systems client bundles.

Table 19: Configuring the operating systems

Step	Command
Configure OS X as the target operating system for which to create a client-bundle.	`vyatta@vyatta# set interfaces openvpn vtunX client-bundle osx`
Generate the client bundle, which consists of a standard OpenVPN-formatted configuration file.	`vyatta@vyatta# set interfaces openvpn vtunX client-bundle generic`
Configure Linux as the target operating system for which to create a client-bundle.	`vyatta@vyatta# set interfaces openvpn vtunX client-bundle linux`

Authentication of the client bundle

Authentication of SSL-VPN client bundle is accomplished through a username and password together as the authentication token without TLS client certificates. The authentication can be done against a set of AT&T Vyatta vRouter-maintained local service users or against central identity management systems like cooperating directory servers (for example, LDAP).

Management of local service users and authentication against a central identity management system is covered in Service User Management.

The SSL-VPN client authentication configuration only requires to reference to authentication profiles of central identify management systems or by referring local service users or groups of local service users.

The SSL-VPN client authentication configuration references the following:

- Authentication profiles of central identity management systems
- Local service users
- Groups of local service users

Any change to the service-user authentication, such as adding or removing a new local service user, or changing or adding an LDAP authentication profile, does not require a change to existing client bundles.

Because client bundles are independent of users, no such change requires a change to existing client setups. A change to service-user authentication does not require a restart of the SSL-VPN server, nor does it terminate the existing client connection.

Authentication methods can be combined for the same SSL-VPN instance to provide authentication against multiple LDAP servers and local service users. When one of these authentication resources grants access, the authorization of the SSL-VPN connection is granted and access to that SSL-VPN instance is permitted.

SSL-VPN access to a local service user

By default, no local service user is granted access to any SSL-VPN endpoint. Fine grained-access control can be granted by explicitly referring to which service user or group of service users is granted access.

The following example shows how to create the alice and bob local service users and grant access for them to the vtunX OpenVPN interface.

Table 20: Granting the alice and bob service-users access to the vtunx OpenVPN interface

Step	Command
Configure the alice user with a password.	`vyatta@vyatta# set resources service-users local user alice auth plaintext-password foo`
Configure the bob user with a password.	`vyatta@vyatta# set resources service-users local user bob auth plaintext-password bar`
Configure an interface for alice.	`vyatta@vyatta# set interfaces openvpn vtunX auth local user alice`
Configure an interface for bob.	`vyatta@vyatta# set interfaces openvpn vtunX auth local user bob`
Commit the configuration.	`vyatta@vyatta# commit`

This configuration allows the alice and bob service users to authenticate themselves by using their usernames and passwords when connecting with the SSL-VPN client bundles.

To refuse bob any further access to the vtunX OpenVPN interface, you must delete the service-user reference in the OpenVPN vtunX interface configuration:

```
vyatta@vyatta# delete interfaces openvpn vtunX auth local user bob
vyatta@vyatta# commit
```

Note: The preceding configuration change does not terminate the existing SSL-VPN session of user bob on vtunX, nor does it interrupt any other existing SSL-VPN client connection.

To grant access to the vtunX OpenVPN interface a group of multiple service users SSL-VPN, enter the following commands:

```
vyatta@vyatta# set resources service-users local group it-dep alice
vyatta@vyatta# set resources service-users local group it-dep bob
vyatta@vyatta# set interfaces openvpn vtunX auth local group it-dep
vyatta@vyatta# commit
```

The preceding configuration change assigns service users alice and bob to the it-dep group. All users of that group are granted access to the vtunX OpenVPN interface.

Note: A change to the membership of an individual user has immediate impact after the change is committed. An existing SSL-VPN connection for a service user who is dropped from a group that has been granted access is not terminated. The change just rejects any further authentication attempts to the vtunX OpenVPN service instance.

Granting SSL-VPN access to an LDAP the service user

LDAP authentication of an SSL-VPN connection requires a service-user LDAP authentication profile, which is configured in the following file:

`resources service-users ldap` *profilename*

Details on how to set up a service LDAP authentication profile are covered in Service User Management.

To enable SSL-VPN authentication against an LDAP service-user authentication profile with a profile name of example.com, the profile name just has to be referred to in the `openvpn vtunX auth` command for the interfaces, as shown here:

```
vyatta@vyatta# set resources service-users ldap example.com ....
vyatta@vyatta# set interfaces openvpn vtunX auth ldap example.com
vyatta@vyatta# commit
```

The preceding configuration change allows access to SSL-VPN for all users who can authenticate themselves with their LDAP credentials against the example.com LDAP profile.

Service-User web portal

Use the Service-User Web Portal to allow end users to obtain by themselves the SSL-VPN client bundles. The portal is available by default from the following public-interface address of the AT&T Vyatta vRouter:

https://*vRouter-public-IP* /service

The Service-User Web Portal is disabled by default and shows a message indicating that the service is not available.

To enable the Service-User Web Portal, enter the following commands:

```
vyatta@vyatta# set services https service-user
vyatta@vyatta# commit
```

All service users that are configured in the resource services-users file can authenticate themselves with their own credentials and passwords.

If all service users are granted access to individual SSL-PVN instances on the AT&T Vyatta vRouter, they are provided download links to SSL-VPN client bundles for each configured operating system.

Maintenance of SSL-VPN client bundles

This section covers the maintenance of SSL-VPN client bundles to provide reliable and secure SSL-VPN service to end users.

Changes to the following configuration options in `interfaces openvpn` **`vtunX`** cause a regeneration of all configured bundles:

- hash
- encryption
- tls ca-cert-file
- local-host
- local-port
- protocol
- description

This regeneration occurs to provide SSL-VPN client configuration that is always up to date. We recommend that an end user obtain the latest SSL-VPN client bundle to get SSL-VPN configuration changes.

The file name of each client bundle includes a suffix to identify the latest version of the client in this form: filename-v*versionnumber*.exe. For example: ACME HQ-v2.exe.

Note: Only the most recent version of a bundle is kept on the AT&T Vyatta vRouter persistent volume.

An SSL-VPN end user must manually obtain a new bundle. A bundle that is already installed or deployed is not automatically updated with the latest configuration.

SSL-VPN client software updates

The Linux client and the OS X client bundles contain only configuration files and TLS CA certificates, not actual code and binary files.

The SSL-VPN Client software in the Windows bundle might need to receive updates that are made available as regular AT&T Vyatta vRouter image releases. When an AT&T Vyatta vRouter release contains a new version of SSL-VPN Client, then all Windows bundles are regenerated with the latest version of the SSL-VPN Client software during the update process.

We recommend that you review the AT&T Vyatta vRouter release notes for each release for direction when the SSL-VPN client update includes critical fixes. For these fixes, advise your SSL-VPN end users to get the latest Windows OS client bundle. The installation wizard guides the users through the update process.

To keep the bundle versions for the operating systems synchronized, all bundles are regenerated when one platform requires regeneration. For example, if a new SSL-VPN client for the Windows OS becomes available, existing OS X and Linux bundles are also regenerated to have the same bundle version as the Windows OS version.

Deploying the SSL-VPN client bundle

To deploy an SSL-VPN client bundle to SSL-VPN end users, they can be obtained by the following methods:

- By the end users through the Service-User Web Portal
- By the AT&T Vyatta vRouter administrator through the AT&T Vyatta vRouter file-system

The set of files for the generated SSL-VPN client bundle on the AT&T Vyatta vRouter is located in the directory: `/config/auth/vpn/ssl-vpn/vtunX/`.

The files are located in the following directories by platform (operating system):

- Windows operating system: /config/auth/vpn/ssl-vpn/client-bundle/vtunX/windows/$description-v$version.exe
- Linux: /config/auth/vpn/ssl-vpn/vtunX/client-bundle/linux/$description-v$version.zip
- OS X: `/config/auth/vpn/ssl-vpn/vtunX/client-bundle/osx/$description-v$version.ovpn`
- Generic: `/config/auth/vpn/ssl-vpn/vtunX/client-bundle/generic/$description-v$version.ovpn`

Note: Those client-bundle directories are automatically generated and removed. Do not put any files or backups of bundles in these locations because the contents get deleted.

AT&T SSL-VPN client for Windows OS

The AT&T SSL-VPN Client for Windows operating system requires administrator privileges for installation because a device driver for a virtual network must be installed and a virtual network interface must be created.

An installation wizard is generated on demand with the latest SSL-VPN client configuration that is running on the AT&T Vyatta vRouter. Because the wizard is a Windows OS executable file, various Windows OS or browser security measures display messages to warn the end user about potential known malware risks.

Some Web browsers warn that downloaded Windows OS executable files are “not commonly downloaded and could be dangerous.” Those browsers often obtain the size and checksum of a file and send that information to the servers of the browser vendors on the Internet to get information about potential known malware. Because the installation wizard is generated on demand and is unique and not known to any of the Web browser vendors, their systems warn the user about the SSL-VPN client bundle for Windows OS as previously indicated.

The executable file for the installation wizard is not signed by the signing key for the publisher code because the wizard is generated on demand on the AT&T Vyatta vRouter. This generation causes a warning message, which indicates that the publisher is unknown and not trusted, in certain browsers when the end user is running the installation wizard on a Windows client machine.

Figure 8: Security warning message

The virtual network driver that is included is signed by the official AT&T Code Signing.

SSL-VPN client connection

This section is intended for the AT&T Vyatta vRouter administrator to which to refer the end user who wants to obtain and establish a connection through SSL-VPN.

Obtaining the SSL-VPN client bundle

Consult your IT administrator to get one of the following:

- A URL to the Service-Users Web Portal of the AT&T Vyatta vRouter SSL-VPN endpoint
  If you receive a URL to the Service-Users Web Portal, log in with credentials that are provided or from your cooperating account. Enter your credentials only when the URL is given to you by your IT administrator.

- A set of files that contains the required SSL-VPN client bundle
  If you receive the SSL-VPN client bundle as a set of files, continue with Installing the SSL-VPN client bundle *(page 42)*. Make sure to receive files only from your IT administrator and no other source. If you are in doubt, verify with your administrator that you have the correct files.

The URL to the Service-User Web Portal of the AT&T Vyatta vRouter might look like this:

https://*cooperative_address* /service

After you are logged in to the portal, the **SSL-VPN** section of the **Service-Users Web Portal** lists all granted SSL-VPN endpoints for your account.

Select an available SSL-VPN endpoint or location to which you want to connect and the correct SSL-VPN client bundle for the operating system on your computer.

Installing the SSL-VPN client bundle

Read the following section that applies to the operating system on which you want to use the SSL-VPN client bundle: Windows, OS X, or Linux.

Windows OS: AT&T SSL-VPN client

The SSL-VPN Client Bundle for Windows gets downloaded as an executable file. The bundle consists of the AT&T SSL-VPN Client software and the SSL-VPN client configuration required to connect to the SSL-VPN endpoint. The downloaded executables starts an installation wizard that guides you through the installation of the AT&T SSL-VPN client software. The installation wizard also handles the actual configuration for the AT&T SSL-VPN Client for your target SSL-VPN endpoint.

> **Note:** The AT&T SSL-VPN Client requires administration privileges to be installed. The installation does not start without these privileges.

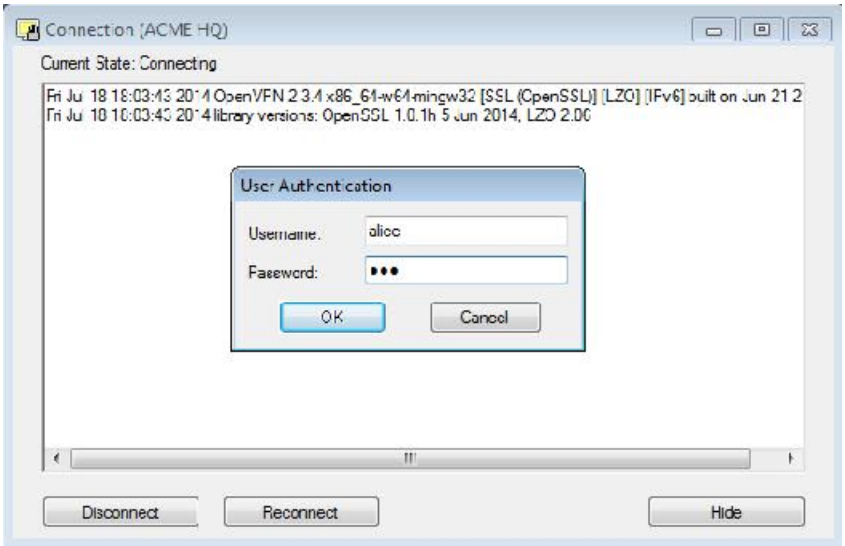
Consult your IT administrator if you do not have administration privileges and ask for further guidance.

Follow the instructions in the wizard and finish the installation and setup of the AT&T SSL-VPN client.

After installation, you can either select in the installation wizard to connect directly to the SSL-VPN or uncheck the **Connect to** box and close the installation wizard by clicking **Finish**.

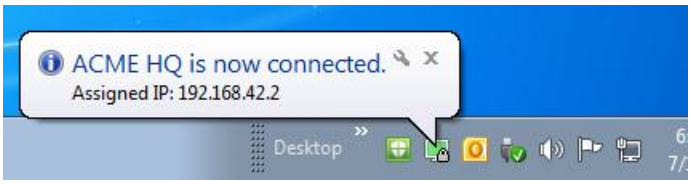
If the **Connect to** box is checked, the AT&T SSL-VPN client starts automatically and prompts you for a username and a password.

Figure 9: Service-Users web portal login page



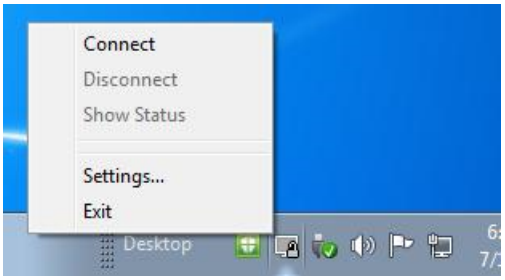
The username and password are either your cooperative credentials are provided to you by your IT administrator.

Figure 10: Connection confirmation pop-up message



After the AT&T SSL-VPN client is connected, it can be controlled through the tray menu by right-clicking on the tray icon.

Figure 11: AT&T SSL-VPN client tray menu



To quit the AT&T SSL-VPN client, use the tray menu and select **Exit**. After confirming the exit, the SSL-VPN connection terminates.

To connect again to the SSL-VPN endpoint or to start the AT&T SSL-VPN client after rebooting, either use the **Connect SSL-VPN to** shortcut on the desktop or navigate the following path in the Windows Start menu:

Start#All Programs#AT&T SSL-VPN Client

> **Note:** The installation wizard installs the AT&T SSL-VPN client persistently on your computer. So, the SSL-VPN client can be used across reboots and needs to be downloaded only once. Your IT administrator might ask you to obtain the latest version of the AT&T SSL-VPN client when changes are made to the SSL-VPN network configuration or maintenance updates are made to the AT&T SSL-VPN client software.

To uninstall the AT&T SSL-VPN client, go to the Windows Control Panel and select **Uninstall/Change**.

Linux platform

For Linux desktops, you must create a new VPN connection profile in Network Manager and import the provided SSL-VPN bundle configuration. See Network manager (Linux desktops) *(page 43)*.

For all other Linux variants, see OpenVPN CLI *(page 47)*.

Regardless of which version of the Linux desktop you are running, the SSL-VPN client bundle must be unpacked. The client bundle arrives as a .zip archive that, after extraction, holds the following two files:

- OpenVPN configuration (file suffix: .ovpn)
- TLS CA certificate (file suffix: .crt)

Network manager (Linux desktops)

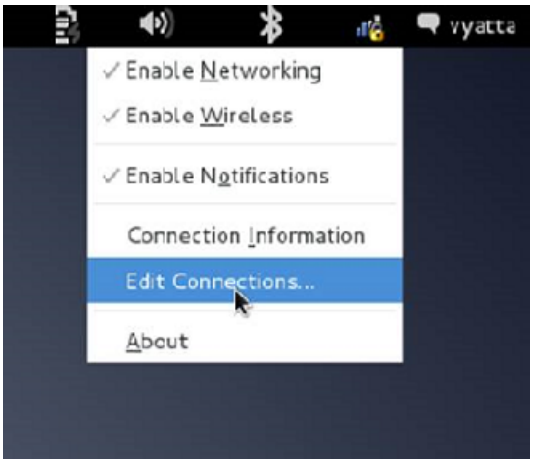
This section applies to Linux desktops, such as Gnome, KDE, and so on, which commonly use Network Manager applets to manage network connections and various types of VPN connections.

To make a VPN connection and import the configuration file, follow these steps:

1. Enter the tray menu of the Network Manager applet and select **Edit Connections**.

   Info:

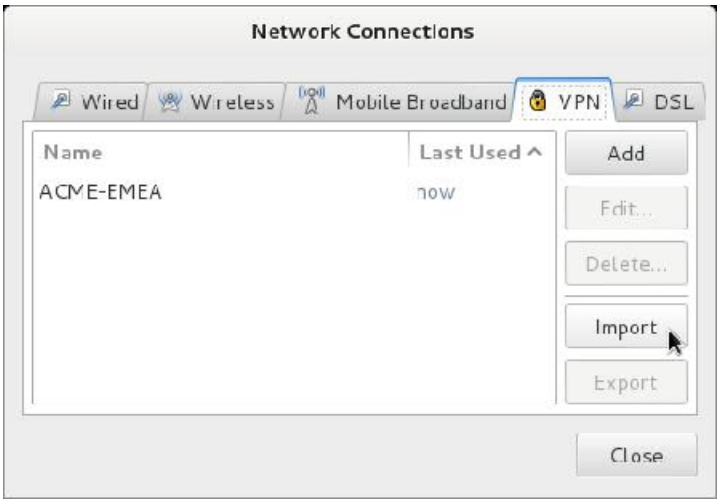
Figure 12: Editing connections... selection



2. Select the **VPN** tab in the **Network Connections** setup dialog box.
3. Select **Import**. In the file dialog, select the location of the unpacked Zip archive.

   Info:

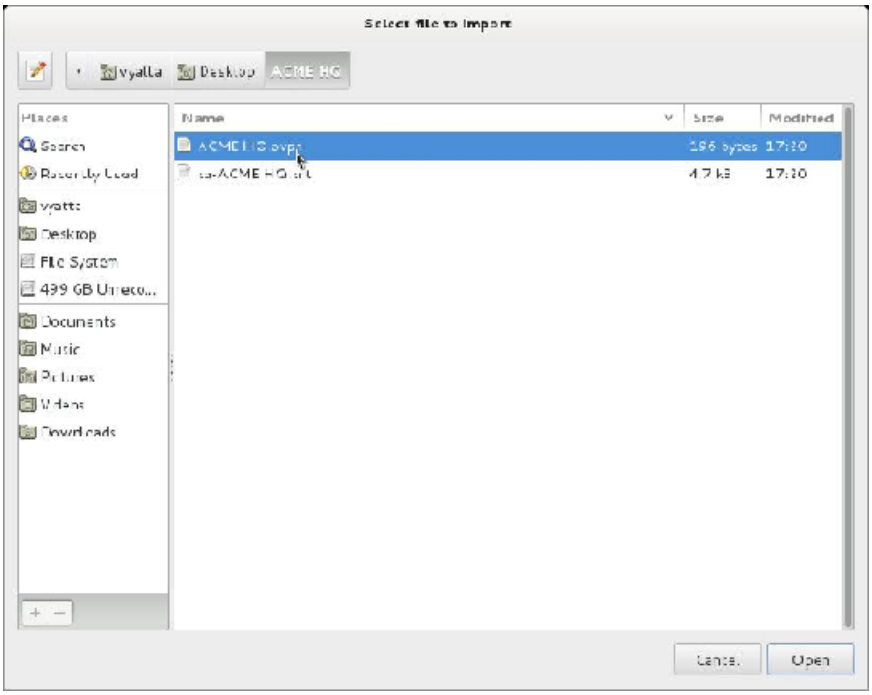
Figure 13: Network connections dialog box



4. Select the OpenVPN configuration file (.ovpn) as the file to import.

   Info:

Figure 14: Selecting the configuration file

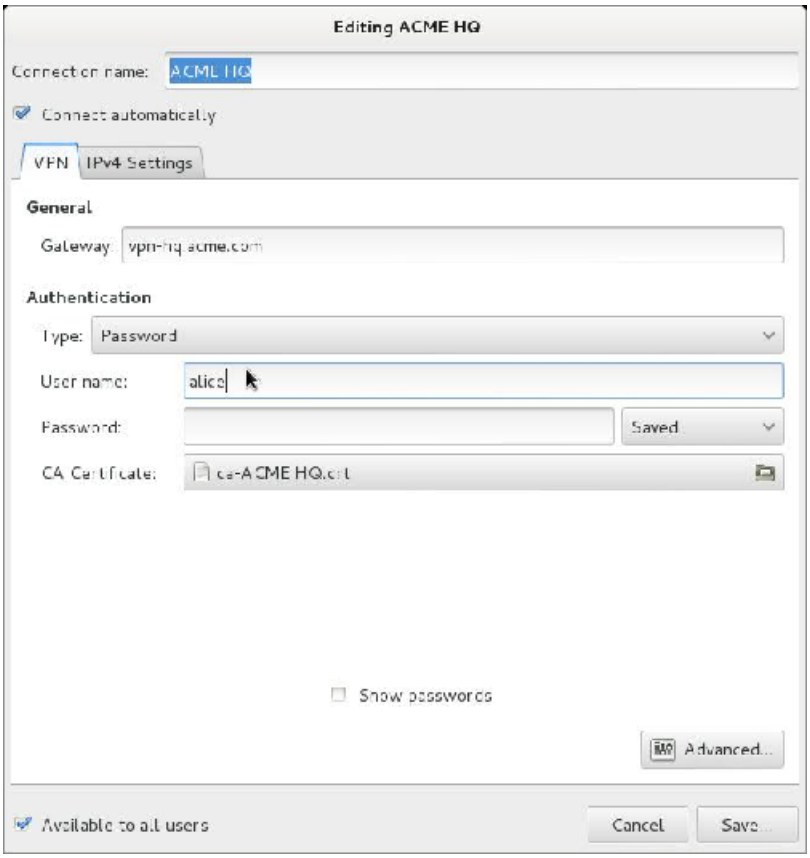


5. In the configuration dialog box after **User name**, enter your cooperating username for the SSL#VPN connection that is provided by your IT administrator or set your cooperating username.

   Info:

   The cooperating username indicates to users that they use their known cooperating credentials. In this example, the username is alice.

Figure 15: Entering login information



6. Verify that the CA certificate is set with the TLS CA certificate file (.crt) from the extracted .zip archive. All further settings were previously set by the imported profile.
7. Leave the **Network Connection** setup dialog box by saving the configuration.
8. Enter the tray menu of the Network Manager applet again and select **VPN Connections** and the recently imported SSL#VPN profile, which has the name of the SSL#VPN client bundle.

Info:

Figure 16: Selecting VPN connections and the SSL#VPN profile

Note: If the profile fails, verify that your system has the Network Manager OpenVPN plugin installed. Depending on your Linux desktop, you might need the following packages:

- network#manager#openvpn#gnome or NetworkManager#openvpn#gnome
- network#manager#openvpn#kde or NetworkManager#openvpn#kde4
- network#manager#openvpn

OpenVPN CLI

The SSL-VPN client bundle can also be used with the OpenVPN command line interface, which requires that you install the openvpn package.

Enter the unpackaged directory for the SSL-VPN client bundle and run the following command:

```
$ sudo openvpn SSL-VPN_client_bundle_name.ovpn
```

Note: Using this command requires root privileges.

This Open VPN command-line prompt requests the username and password for authentication. Use either the IT administrator-provided credentials or your cooperating credentials.

OS X platform

For OS X, the client bundle consists of a generic OpenVPN formatted configuration file for OpenVPN OS X clients. To make use of the SSL-VPN client bundle for OS X, you must first install an OpenVPN client application for OS X. There are various OS X OpenVPN clients available. All of those are compatible with the provided SSL-VPN client bundle configuration for OS X.

Configuring OpenVPN LDAP Authentication

Configuring LDAP authentication

To configure OpenVPN LDAP authentication, perform the following steps:

1. Configure service-user authentication through LDAP, as described in the “Service-User Management” chapter of AT&T Vyatta Network Operating System Basic System Configuration Guide.
2. Link an OpenVPN instance or interface to the LDAP service-user profile.

   Info:

   ```
   vyatta@vyatta# set interfaces openvpn vtunX auth ldap ldap-profile-name
   ```

Configuring LDAP authentication without client certificates

To perform OpenVPN LDAP authentication without client certificates, use the following command:

```
vyatta@vyatta# set interfaces openvpn vtunX client-cert-not-required
```

OpenVPN Commands

generate openvpn key <filename>

Generates a shared secret key that is contained in a file with the specified file name.

Syntax:
`generate openvpn key` *filename*

filename
A name for the file that contains the shared key.

Operational mode

Use this command to generate a shared secret key that is contained in a file with the specified file name. The key is required when the OpenVPN preshared secret mechanism is used. This command is available only to users with administrative privileges.

interfaces openvpn <vtunx>

Defines an OpenVPN interface.

Syntax:
`set interfaces openvpn` *vtunx*

Syntax:
`delete interfaces openvpn` *vtunx*

Syntax:
`show interfaces openvpn` *vtunx*

vtunx
Multi-node. The identifier for an OpenVPN interface. This identifier is vtun0 through **`vtun`***x*, where *x* is a non-negative integer.

You can define more than one OpenVPN interface by creating multiple `interfaces openvpn` configuration nodes.

Configuration mode

```
interfaces {
    openvpn vtunx {
    }
}
```

Use this command to define an OpenVPN interface.

Use the `set` form of this command to configure an interface.

Use the `delete` form of this command to remove an existing configuration for an interface.

Use the `show` form of this command to view the configuration for an interface.

interfaces openvpn <vtunx> description <description>

Provides a description for an OpenVPN interface.

Syntax:

```
set interfaces openvpn vtunx description description
```

Syntax:

```
delete interfaces openvpn vtunx description
```

Syntax:

```
show interfaces openvpn vtunx description
```

vtunx

The identifier for an OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

description

A brief description for the interface. If the description contains spaces, it must be enclosed in quotation marks.

Configuration mode

```
interfaces {
    openvpn vtunx {
        description description
    }
}
```

Use this command to provide a description for an OpenVPN interface.

Note: Committing configuration changes to this configuration node does not result in the OpenVPN process being restarted.

Use the `set` form of this command to provide a description for an interface.

Use the `delete` form of this command to remove the description for an interface.

Use the `show` form of this command to view the description for an interface.

interfaces openvpn <vtunx> device-type tap

Configures an OpenVPN interface as a tap device.

Syntax:

```
set interfaces openvpn vtunx device-type tap
```

Syntax:

```
delete interfaces openvpn vtunx device-type tap
```

Syntax:

```
show interfaces openvpn vtunx device-type
```

The OpenVPN interface is configured as a tun device.

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    openvpn vtunx {
        device-type {
            tap
        }
    }
```

```
}
```

Use this command to configure an OpenVPN interface as a tap device.

By default, an OpenVPN interface is configured as a tun device. A tun device is a virtual network interface that operates on Layer 3 (network layer) traffic, such as IP packets. A tap device is also a virtual network interface, but it operates on Layer 2 (link layer) traffic. A tap device is used in instances when two ends of an OpenVPN tunnel must be bridged or require a broadcast interface. A sample instance is DHCP Relay.

Note: Each end of an OpenVPN tunnel must have the same **device-type** configuration.

Use the `set` form of this command to configure an interface as a tap device.

Use the `delete` form of this command to return an interface to its default behavior, that is, the interface is configured as a tun device.

Use the `show` form of this command to view the **device-type** configuration.

interfaces openvpn <vtunx> disable

Disables an OpenVPN interface without discarding the configuration.

Syntax:
`set interfaces openvpn` *vtunx* **disable**

Syntax:
`delete interfaces openvpn` *vtunx* **disable**

Syntax:
`show interfaces openvpn` *vtunx*

The OpenVPN interface configuration is enabled.

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    openvpn vtunx {
        disable
    }
}
```

Use this command to disable an OpenVPN interface without discarding configuration. The interface can then be re-enabled at a later time without the need to redefine the configuration.

Use the `set` form of this command to disable an interface.

Use the `delete` form of this command to enable an interface.

Use the `show` form of this command to view the configuration of an interface.

interfaces openvpn <vtunx> encryption <algorithm>

Specifies the encryption algorithm to be used within an OpenVPN tunnel.

Syntax:
`set interfaces openvpn` *vtunx* **encryption** *algorithm*

Syntax:
`delete interfaces openvpn` *vtunx* **encryption**

Syntax:

```
show interfaces openvpn vtunx encryption
```

The encryption algorithm is the Blowfish algorithm with 128-bit key.

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

algorithm

The encryption algorithm that is used within the OpenVPN tunnel. The algorithm is one of the following:

3des: DES algorithm with triple encryption

aes128: AES algorithm with 128-bit key

aes192: AES algorithm with 192-bit key

aes256: AES algorithm with 256-bit key

bf128: Blowfish algorithm with 128-bit key

bf256: Blowfish algorithm with 256-bit key

des: DES algorithm

The default algorithm is **bf128**.

Configuration mode

```
interfaces {
    openvpn vtunx {
        encryption algorithm
    }
}
```

Use this command to specify the encryption algorithm that is used within an OpenVPN tunnel.

Use the `set` form of this command to specify the encryption algorithm that is used within a tunnel.

Use the `delete` form of this command to remove the encryption algorithm that is used within a tunnel and return to the default algorithm, which is the Blowfish algorithm with 128-bit key.

Use the `show` form of this command to view the encryption algorithm that is used within a tunnel.

interfaces openvpn <vtunx> firewall <action>

Applies a firewall rule set to an OpenVPN tunnel.

Syntax:

```
set interfaces openvpn vtunx firewall { in | out } name
```

Syntax:

```
delete interfaces openvpn vtunx firewall [ { in | out } name ]
```

Syntax:

```
show interfaces openvpn vtunx firewall
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

in

The firewall rule set is applied to inbound traffic.

out

The firewall rule set is applied to outbound traffic.

name

Rule set for forwarded packets on inbound interface.

Configuration mode

```
openvpn vtunx {
    firewall {
        in name
    }
}
```

Use the `set` form of this command to apply a firewall rule set to an OpenVPN tunnel.

Use the `delete` form of this command to remove a firewall rule set from an OpenVPN tunnel.

Use the `show` form of this command to display the firewall rule set applied to an OpenVPN tunnel.

interfaces openvpn <vtunx> hash <algorithm>

Specifies the hash algorithm to be used within an OpenVPN tunnel.

Syntax:
`set interfaces openvpn` *vtunx* **`hash`** *algorithm*

Syntax:
`delete interfaces openvpn` *vtunx* **`hash`**

Syntax:
`show interfaces openvpn` *vtunx* **`hash`**

The hash algorithm is SHA-1.

vtunx

The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

algorithm

The hash algorithm that is used within the OpenVPN tunnel. The algorithm is one of the following:

`md5`: MD5 algorithm

`sha1`: SHA-1 algorithm

`sha256`: SHA-256 algorithm

`sha512`: SHA-512 algorithm

The default algorithm is **`sha1`**.

Configuration mode

```
interfaces {
    openvpn vtunx {
        hash algorithm
    }
}
```

Use this command to specify the hash algorithm that is used within an OpenVPN tunnel.

Use the `set` form of this command to specify the hash algorithm that is used within a tunnel.

Use the `delete` form of this command to remove the hash algorithm that is used within a tunnel and return to the default algorithm, which is SHA-1.

Use the **`show`** form of this command to view the hash algorithm that is used within a tunnel.

interfaces openvpn <vtunx> ipv6 address

Assigns an IPv6 address to an OpenVPN interface.

Syntax:
`set interfaces openvpn` *vtunx* **`ipv6 address`** [**`autoconf`** | **`eui64`** *ipv6prefix* | **`link-local`** *ipv6-address*]

Syntax:
`delete interfaces openvpn` *vtunx* **`ipv6 address`** [**`autoconf`** | **`eui64`** *ipv6prefix* | **`link-local`** *ipv6-address*]

Syntax:
`show interfaces openvpn` *vtunx* **`ipv6 address`** [**`autoconf`** | **`eui64`**]

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

`autoconf`
Generates an IPv6 address using the Stateless Address Autoconfiguration (SLAAC) protocol. Generate this address if the interface is performing a host function rather than a router function. This address can be specified in addition to static IPv6, static IPv4, and IPv4 DHCP addresses on the interface.

ipv6prefix
A 64-bit IPv6 address prefix used to configure an IPv6 address in EUI-64 format. The system concatenates this prefix with a 64-bit EUI-64 address derived from the 48-bit MAC address of the interface.

`link-local` ***ipv6-address***
Specifies the 128-bit IPv6 address.

Configuration mode

```
interfaces {
    openvpn vtunx {
        ipv6 {
            address {
                autoconf
                eui64 ipv6prefix
                link-local ipv6address
            }
        }
    }
}
```

Use this command to assign an IPv6 address to an OpenVPN interface.

Employ the **`autoconf`** keyword to direct the system to autoconfigure the address, using the SLAAC protocol defined in RFC 4862. Alternatively, you can provide an EUI-64 IPv6 address prefix so that the system constructs the IPv6 address.

If you want the system to use SLAAC to acquire addresses on this interface, then, in addition to setting this parameter, you must also disable IPv6 forwarding. Disable forwarding either globally by using the `system ipv6 disable-forwarding` command or specifically on this interface by using the interfaces openvpn <vtunx> ipv6 disable-forwarding *(page 55)* command.

Use the `set` form of this command to assign an IPv6 address to an interface.

Use the `delete` form of this command to delete an IPv6 address from an interface.

Use the `show` form of this command to view settings of an IPv6 address configuration.

interfaces openvpn <vtunx> ipv6 disable

Disables IPv6 on an OpenVPN interface.

Syntax:
`set interfaces openvpn` *vtunx* **`ipv6 disable`**

Syntax:
`delete interfaces openvpn` *vtunx* **`ipv6 disable`**

Syntax:
`show interfaces openvpn` *vtunx* **`ipv6 disable`**

IPv6 is enabled on an OpenVPN interface

`vtunx`
The identifier of an OpenVPN interface. The identifier ranges from **`vtun0`** through **`vtun`***x*, where *x* is a nonnegative integer.

Configuration mode.

```
interfaces {
  openvpn vtunx {
    ipv6 {
      disable
    }
  }
}
```

By default, IPv6 is enabled on all interfaces. A global command exists which can disable IPv6, namely `set system ipv6 disable`, and this will take precedence over any of the existing per-interface based, IPv6 commands.

IPv6 Forwarding can be disabled via the `set interface openvpn vtunx ipv6 disable-forwarding` command, but note that IPv6 traffic can still be terminated on this interface.

IPv6 configuration can be totally disabled via the `set interface openvpn vtunx ipv6 disable` command.

Use the `set` form of this command to disable IPv6 on this interface.

Use the `delete` form of this command to enable IPv6 on this interface.

Use the `show` form of this command to display the current IPv6 configuration on this interface.

interfaces openvpn <vtunx> ipv6 disable-forwarding

Disables IPv6 packet forwarding on an OpenVPN interface.

Syntax:
`set interfaces` **`openvpn`** *vtunx* **`ipv6 disable-forwarding`**

Syntax:
`delete interfaces` **`openvpn`** *vtunx* **`ipv6 disable-forwarding`**

Syntax:
`show interfaces openvpn` *vtunx* **`ipv6 disable-forwarding`**

IPv6 packets are forwarded.

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    openvpn vtunx {
        ipv6 {
            disable-forwarding
```

```
        }
    }
}
```

Use this command to disable IPv6 packet forwarding on an OpenVPN interface.

You can also disable IPv6 forwarding globally, that is, for all interfaces, by using the `system ipv6 disable-forwarding` command.

Use the `set` form of this command to disable IPv6 packet forwarding on an interface.

Use the `delete` form of this command to enable IPv6 packet forwarding on an interface.

Use the `show` form of this command to view the configuration of IPv6 packet forwarding on an interface.

interfaces openvpn <vtunx> ipv6 dup-addr-detect-transmits <number>

Specifies the number of times to transmit Neighbor Solicitation (NS) packets as part of the Duplicate Address Detection (DAD) process.

Syntax:

```
set interfaces openvpn vtunx ipv6 dup-addr-detect-transmits number
```

Syntax:

```
delete interfaces openvpn vtunx ipv6 dup-addr-detect-transmits
```

Syntax:

```
show interfaces openvpn vtunx ipv6 dup-addr-detect-transmits
```

One NS packet is transmitted as part of the DAD process.

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

number

The number of times to transmit NS packets as part of the DAD process. The default number is 1.

Configuration mode

```
interfaces {
    openvpn vtunx {
        ipv6 {
            dup-addr-detect-transmits number
        }
    }
}
```

Use this command to specify the number of times to transmit NS packets as part of the DAD process.

Use the `set` form of this command to specify the number of times to transmit NS packets.

Use the `delete` form of this command to remove the number of times NS packets are transmitted and return to the default number of 1.

Use the `show` form of this command to view the number of times NS packets are transmitted.

interfaces openvpn <vtunx> ipv6 router-advert

Specifies the router advertisements (RA) to be sent from an OpenVPN interface.

Syntax:

```
set interfaces openvpn vtunx ipv6 router-advert [ cur-hop-limit limit ] [ default-lifetime lifetime ] [ default-preference preference ] [ link-mtu mtu ] [ managed-flag state ] [ max-interval interval ] [ min-interval interval ] [ other-config-flag state ] [ prefix ipv6net [ autonomous-flag state | on-link-flag state | preferred-lifetime lifetime | valid-lifetime lifetime ] ] [ reachable-time time ] [ retrans-timer time ] [ send-advert state ]
```

Syntax:

```
delete interfaces openvpn vtunx ipv6 router-advert [ cur-hop-limit ] [ default-lifetime ] [ default-preference ] [ link-mtu ] [ managed-flag ] [ max-interval ] [ min-interval ] [ other-config-flag ] [ prefix ipv6net [ autonomous-flag | on-link-flag | preferred-lifetime | valid-lifetime ] ] [ reachable-time ] [ retrans-timer ] [ send-advert ]
```

Syntax:

```
show interfaces openvpn vtunx ipv6 router-advert
```

Router advertisements are not sent on an interface.

vtunx
: The identifier for the OpenVPN interface. This may be **vtun0** to **vtun** *x*, where *x* is a non-negative integer.

cur-hope-limit ***limit***
: Limits the Hop Count field of the IP header for outgoing (unicast) IP packets. This limit is placed in the Hop Count field. The limit ranges from 0 through 255. The default limit is 64. A limit of 0 means that the limit is unspecified by the router.

default-lifetime ***lifetime***
: Specifies the lifetime, in seconds, associated with the default router. The lifetime either is 0, which indicates the router is not a default router, or it ranges from the interval specified in the **max-interval** argument through 9000 (18.2 hours). If not entered, the lifetime is three times the interval specified in the **max-interval** argument.

default-preference ***preference***
: Specifies the preference associated with the default router. The preference is one of the following:

  low: Makes the default router low preference.

  medium: Makes the default router medium preference. The default preference is **medium**.

  high: Makes the default router high preference.

link-mtu ***mtu***
: Specifies the MTU to be advertised for the link. The MTU either is 0 or it ranges from 1280 through the maximum MTU for the type of link, as defined in RFC 2464. The default MTU is 0, which means the MTU is not specified in the router advertisement message. That is because it is expected that the MTU is configured directly on the interface itself and not for routing advertisements. You can configure this option in cases where the link MTU is not well known.

  If the MTU entered here does not match the MTU configured on the interface, the system issues a warning but does not fail.

managed-flag ***state***
: Specifies whether to use the administered protocol for address autoconfiguration. The state is one of the following:

  true: Specifies that hosts use the administered (stateful) protocol for address autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration.

  false: Specifies that hosts use only stateless address autoconfiguration. The default state is **false**.

max-interval ***interval***
: Specifies the maximum time, in seconds, allowed between sending unsolicited multicast router advertisements from the interface. The interval ranges from 4 through 1800. The default interval is 600 (10 minutes).

min-interval ***interval***
: Specifies the minimum time, in seconds, allowed between sending unsolicited multicast router advertisements from the interface. The interval ranges from 3 through 0.75 times the interval specified

in the **max-interval** argument. The default interval is 0.33 times the interval specified in the **max-interval** argument.

other-config-flag ***state***

Specifies that the interface uses the administered (stateful) protocol for autoconfiguration of non-address information, as defined in RFC 4862. The state is one of the following:

true: Specifies that hosts use the administered protocol for autoconfiguration of non-address information.

false: Specifies that hosts use stateless autoconfiguration of non-address information. The default state is **false**.

prefix ***ipv6net***

Multi-node. Specifies an IPv6 prefix, in the format *ipv6-address/prefix*, to be advertised on the IPv6 interface.

You can define more than one IPv6 prefix by configuring multiple *prefix* configuration nodes.

autonomous-flag ***state***

Specifies whether the IPv6 prefix can be used for autonomous address configuration as defined in RFC 4862. The state is one of the following:

true: Specifies that the prefix can be used for autonomous address configuration. The default state is **true**.

false: Specifies that the prefix cannot be used for autonomous address configuration.

on-link-flag ***state***

Specifies whether the IPv6 prefix can be used for on-link determination, as defined in RFC 4862. The state is one of the following:

true: Specifies that the prefix can be used for on-link determination. The default state is **true**.

false: Specifies that the advertisement makes no statement about on-link or off-link properties of the prefix. For instance, the prefix might be used for address configuration with some addresses belonging to the prefix being on-link and others being off-link.

preferred-lifetime ***lifetime***

Specifies the lifetime, in seconds, that the addresses generated from the IPv6 prefix through Stateless Address Autoconfiguration (SLAAC) is to remain preferred, as defined in RFC 4862. The lifetime is with respect to the time the packet is sent. The lifetime ranges from 1 through 4294967296 plus the **infinity** keyword, which represents forever. (The actual value of **infinity** is a byte in which all bits are set to 1s: 0XFFFFFFFF.) The default lifetime is 604800 (7 days).

valid-lifetime ***lifetime***

Specifies the lifetime, in seconds, that the IPv6 prefix is valid for the purpose of on-link determination, as defined in RFC 4862. The lifetime is with respect to the time the packet is sent. The lifetime ranges from 1 through 4294967296 plus the **infinity** keyword, which represents forever. (The actual value of **infinity** is a byte in which all bits are set to 1s: 0XFFFFFFFF.) The default lifetime is 2592000 (30 days).

reachable-time ***time***

Specifies the length of time, in milliseconds, for which the system assumes a neighbor is reachable after having received a reachability confirmation. This time is used by address resolution and the Neighbor Unreachability Detection algorithm (see Section 7.3 of RFC 2461). The time ranges from 0 through 3600000, where 0 means the reachable time is not specified in the router advertisement message. The default time is 0.

retrans-timer ***time***

Specifies the length of time, in milliseconds, between retransmitted NS messages. This time is used by address resolution and the Neighbor Unreachability Detection algorithm (see Sections 7.2 and 7.3 of RFC 2461). The time ranges from 0 through 4294967295, where 0 means the retransmit time is not specified in the router advertisement message. The default time is 0.

send-advert ***state***

Specifies whether router advertisements are to be sent from this interface. The state is one of the following:

true: Sends router advertisements from this interface. The default state is **true**.

false: Does not send router advertisements from this interface. If this state is in effect, parameters in this configuration subtree are still used to configure the local implementation parameters.

Configuration mode

```
interfaces {
 openvpn vtunx {
  ipv6 {
   router-advert {
    cur-hop-limit limit
    default-lifetime lifetime
    default-preference preference
    link-mtu mtu
    managed-flag state
    max-interval interval
    min-interval interval
    other-config-flag state
    prefix ipv6net {
     autonomous-flag state
     on-link-flag state
     preferred-lifetime lifetime
     valid-lifetime lifetime
    }
    reachable-time time
    retrans-timer time
    send-advert state
   }
  }
 }
}
```

Use this command to specify the RA to be sent from an OpenVPN interface.

Router advertisements are sent by IPv6 routers to advertise their existence to hosts on the network. IPv6 hosts do not send router advertisements.

If the **router-advert** node of the configuration tree is missing, router advertisements are not sent. In addition, if IPv6 forwarding is disabled either globally (by using the `system ipv6 disable-forwarding` command) or on the interface (by using the `interfaces openvpn` *vtunx* **ipv6 disable-forwarding** command), RA are not sent.

Most RA parameters are required by either the Neighbor Discovery (ND) protocol or the Stateless Address Autoconfiguration (SLAAC) protocol. These parameters are used both locally for the IPv6 implementation and become part of the RA messages sent to hosts on the network so that they can be configured appropriately.

Use the `set` form of this command to create the **router-advert** configuration node and begin to send RA.

Use the `delete` form of this command to remove the **router-advert** configuration node and stop sending RA.

Use the `show` form of this command to view RA configuration.

interfaces openvpn <vtunx> local-address <ipv4>

Specifies the tunnel IP address on the local end of an OpenVPN tunnel.

Syntax:
`set interfaces openvpn` *vtunx* **local-address** *ipv4* [**subnet-mask** *mask*]

Syntax:
`delete interfaces openvpn` *vtunx* **local-address** *ipv4* [**subnet-mask**]

Syntax:
`show interfaces openvpn` *vtunx* **local-address** *ipv4* [**subnet-mask**]

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

ipv4

Mandatory. An IPv4 address.

subnet-mask ***mask***

A subnet mask for the network associated with the local address. A mask is required only when interfaces openvpn <vtunx> device-type tap *(page 50)* is set.

Configuration mode

```
interfaces {
    openvpn vtunx {
        local-address ipv4 {
            subnet-mask mask
        }
    }
}
```

Use this command to specify the tunnel IP address on the local end of the OpenVPN tunnel. Only a single address can be specified. This address is required for site-to-site mode OpenVPN tunnels but not for remote access mode tunnels. The subnet mask is required only when the interface is configured as a broadcast interface by using the `interfaces openvpn` *vtunx* **`device-type tap`** command.

Use the `set` form of this command to specify the tunnel IP address on the local end of a tunnel.

Use the `delete` form of this command to remove the tunnel IP address from the local end of a tunnel.

Use the `show` form of this command to view the tunnel IP address on the local end of a tunnel.

interfaces openvpn <vtunx> local-host <ipv4>

Specifies the local IP address to which connections are accepted.

Syntax:

`set interfaces openvpn` *vtunx* **`local-host`** *ipv4*

Syntax:

`delete interfaces openvpn` *vtunx* **`local-host`**

Syntax:

`show interfaces openvpn` *vtunx* **`local-host`**

vtunx

The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

ipv4

The IP address of the local physical interface. This address is the IP address on which connections are accepted. If not specified, then all connections are accepted.

Configuration mode

```
interfaces {
    openvpn vtunx {
        local-host ipv4
    }
}
```

Use this command to specify the local IP address to which connections are accepted. This address can be used as a server endpoint in a remote access mode tunnel or the TCP-passive endpoint when TCP is used in site-to-site mode. The address can be an IP address on any network interface on this endpoint. If this address is specified, the OpenVPN process accepts only sessions coming in on the particular IP address, and this acceptance applies to both UDP and TCP. If an address is not specified, OpenVPN accepts incoming sessions on any interface.

Use the `set` form of this command to specify the local IP address to which connections are accepted.

Use the `delete` form of this command to remove the local IP address to which connections are accepted.

Use the `show` form of this command to view the local IP address to which connections are accepted.

interfaces openvpn <vtunx> local-port <port>

Specifies a local UDP or TCP port on which incoming sessions are accepted.

Syntax:
`set interfaces openvpn` *vtunx* **`local-port`** *port*

Syntax:
`delete interfaces openvpn` *vtunx* **`local-port`**

Syntax:
`show interfaces openvpn` *vtunx* **`local-port`**

The default port number is 1194.

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

port
The number of a port on which incoming sessions are accepted. The default port number is 1194.

Configuration mode

```
interfaces {
    openvpn vtunx {
        local-port port
    }
}
```

Use this command to specify a local UDP or TCP port on which incoming sessions are accepted. This port can be used as a server endpoint in a remote access mode tunnel or the TCP-passive endpoint when TCP is used in site-to-site mode.

Use the `set` form of this command to specify a local port on which incoming sessions are accepted.

Use the `delete` form of this command to remove the local port on which incoming sessions are accepted.

Use the `show` form of this command to view the local port on which incoming sessions are accepted.

interfaces openvpn <vtunx> mode <mode>

Specifies the mode in which an OpenVPN interface operates.

Syntax:
`set interfaces openvpn` *vtunx* **`mode`** *mode*

Syntax:
`delete interfaces openvpn` *vtunx* **`mode`**

Syntax:
`show interfaces openvpn` *vtunx* **`mode`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

mode

Mandatory. The mode in which an OpenVPN interface operates. The mode is one of the following:

client: Defines the endpoint as the client in a remote access tunnel.

site-to-site: Defines the endpoint as one end of a site-to-site tunnel.

server: Defines the endpoint as the server in a remote access tunnel.

Configuration mode

```
interfaces {
    openvpn vtunx {
        mode mode
    }
}
```

Use this command to specify the mode in which an OpenVPN interface operates.

Use the `set` form of this command to specify the mode in which an interface operates.

Use the `delete` form of this command to remove the mode in which an interface operates.

Use the `show` form of this command to view the mode in which an interface operates.

interfaces openvpn <vtunx> openvpn-option <options>

Employs OpenVPN options that are not available with AT&T Vyatta vRouter OpenVPN commands.

Syntax:
`set interfaces openvpn` *vtunx* **`openvpn-option`** *options*

Syntax:
`delete interfaces openvpn` *vtunx* **`openvpn-option`**

Syntax:
`show interfaces openvpn` *vtunx* **`openvpn-option`**

vtunx

The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

options

Multi-node. A list of OpenVPN options to pass to the OpenVPN process.

You can define multiple OpenVPN option lists by creating multiple **`openvpn-option`** configuration nodes. Each must have a unique configuration.

Configuration mode

```
interfaces {
    openvpn vtunx {
        openvpn-option options
    }
}
```

Use this command to employ additional OpenVPN options that are not available with AT&T Vyatta vRouter OpenVPN commands. Because the OpenVPN process has over two hundred commands, only basic options are available with AT&T Vyatta vRouter commands. This command provides access to all options available in OpenVPN. For more information about OpenVPN, go to http://openvpn.net/.

Use the `set` form of this command to employ additional options.

Use the `delete` form of this command to remove additional options.

Use the `show` form of this command to view additional options.

interfaces openvpn <vtunx> protocol <protocol>

Specifies the OpenVPN communications protocol.

Syntax:
```
set interfaces openvpn vtunx protocol protocol
```

Syntax:
```
delete interfaces openvpn vtunx protocol
```

Syntax:
```
show interfaces openvpn vtunx protocol
```

The default protocol is UDP.

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

protocol
An OpenVPN communications protocol. The protocol is one of the following:

tcp-active: Defines the transport protocol as active TCP.

tcp-passive: Defines the transport protocol as passive TCP.

udp: Defines the transport protocol as UDP. UDP is the default protocol.

Configuration mode

```
interfaces {
    openvpn vtunx {
        protocol protocol
    }
}
```

Use this command to specify the OpenVPN communications protocol.

Use the `set` form of this command to specify the communications protocol.

Use the `delete` form of this command to remove the communications protocol.

Use the `show` form of this command to view the communications protocol.

interfaces openvpn <vtunx> remote-address <ipv4>

Specifies the IP address for the tunnel interface of the remote end of the OpenVPN tunnel.

Syntax:
```
set interfaces openvpn vtunx remote-address ipv4
```

Syntax:
```
delete interfaces openvpn vtunx remote-address
```

Syntax:
```
show interfaces openvpn vtunx remote-address
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

ipv4
Mandatory. The tunnel IP address on the remote end of the OpenVPN tunnel.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-address ipv4
    }
}
```

Use this command to specify the tunnel IP address on the remote end of the OpenVPN tunnel. Only a single address can be specified. This address is required for site-to-site mode OpenVPN tunnels but not for remote access mode tunnels.

Use the `set` form of this command to specify the tunnel IP address on the remote end of the tunnel.

Use the `delete` form of this command to remove the tunnel IP address from the remote end of the tunnel.

Use the `show` form of this command to view the tunnel IP address on the remote end of the tunnel.

interfaces openvpn <vtunx> remote-configuration password <password>

Specifies the password for client authentication by OpenVPN Access Server.

Syntax:
`set interfaces openvpn` *vtunx* **`remote-configuration password`** *password*

Syntax:
`delete interfaces openvpn` *vtunx* **`remote-configuration password`**

Syntax:
`show interfaces openvpn` *vtunx* **`remote-configuration password`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

password
A password to be used with a username for authentication by OpenVPN Access Server.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-configuration {
            password password
        }
    }
}
```

Use this command to specify a password that OpenVPN Access Server can use to authenticate a client. This password is used when the client initiates a connection with OpenVPN Access Server.

Use the `set` form of this command to specify a password.

Use the `delete` form of this command to remove a password.

Use the `show` form of this command to view a password.

interfaces openvpn <vtunx> remote-configuration server <address>

Specifies OpenVPN Access Server to which a client connects.

Syntax:
`set interfaces openvpn` *vtunx* **`remote-configuration server`** *address*

Syntax:
`delete interfaces openvpn` *vtunx* **`remote-configuration server`**

Syntax:
`show interfaces openvpn` *vtunx* **`remote-configuration server`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

address
The IP address (or hostname) of OpenVPN Access Server.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-configuration {
            server address
        }
    }
}
```

Use this command to specify the IP address or hostname of OpenVPN Access Server to which a client connects when establishing an OpenVPN tunnel.

Use the `set` form of this command to specify the IP address or hostname.

Use the `delete` form of this command to remove the IP address or hostname.

Use the `show` form of this command to view the IP address or hostname.

interfaces openvpn <vtunx> remote-configuration tunnel-password <password>

Specifies the password used to establish an OpenVPN tunnel with an OpenVPN server.

Syntax:
`set interfaces openvpn` *vtunx* **`remote-configuration tunnel-password`** *password*

Syntax:
`delete interfaces openvpn` *vtunx* **`remote-configuration tunnel-password`**

Syntax:
`show interfaces openvpn` *vtunx* **`remote-configuration tunnel-password`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

password
A password to be used with a username for tunnel establishment to an OpenVPN server.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-configuration {
            tunnel-password password
        }
    }
}
```

Use this command to specify a password used to establish an OpenVPN tunnel with an OpenVPN server. The password is required only if the OpenVPN server has Autologin disabled and you are using OpenVPN Access Server to provide information about OpenVPN tunnel configuration.

Use the `set` form of this command to specify a password.

Use the `delete` form of this command to remove a password.

Use the `show` form of this command to a view password.

interfaces openvpn <vtunx> remote-configuration tunnel-username <username>

Specifies a username used to establish an OpenVPN tunnel with an OpenVPN server.

Syntax:
`set interfaces openvpn` *vtunx* **`remote-configuration tunnel-username`** *username*

Syntax:
`delete interfaces openvpn` *vtunx* **`remote-configuration tunnel-username`**

Syntax:
`show interfaces openvpn` *vtunx* **`remote-configuration tunnel-username`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

username
A username to be used with a password for tunnel establishment to an OpenVPN server.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-configuration {
            tunnel-username username
        }
    }
}
```

Use this command to specify a username used to establish an OpenVPN tunnel with an OpenVPN server. The username is required only if the OpenVPN server has Autologin disabled and you are using OpenVPN Access Server to provide information about OpenVPN tunnel configuration.

Use the `set` form of this command to specify a username.

Use the `delete` form of this command to remove a username.

Use the `show` form of this command to view a username.

interfaces openvpn <vtunx> remote-configuration username <username>

Specifies a username for client authentication by OpenVPN Access Server.

Syntax:
```
set interfaces openvpn vtunx remote-configuration username username
```

Syntax:
```
delete interfaces openvpn vtunx remote-configuration username
```

Syntax:
```
show interfaces openvpn vtunx remote-configuration username
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

username
A username to be used with a password for authentication by OpenVPN Access Server.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-configuration {
            username username
        }
    }
}
```

Use this command to specify a username that OpenVPN Access Server can use to authenticate a client. This username is used when the client initiates a connection with OpenVPN Access Server.

Use the `set` form of this command to specify a username.

Use the `delete` form of this command to remove a username.

Use the `show` form of this command to view a username.

interfaces openvpn <vtunx> remote-host <hostname>

Specifies a remote IP address or hostname to which connections are made.

Syntax:
```
set interfaces openvpn vtunx remote-host hostname
```

Syntax:
```
delete interfaces openvpn vtunx remote-host
```

Syntax:
```
show interfaces openvpn vtunx remote-host
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

hostname
A remote IP address or hostname to which connections are made.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-host hostname
    }
}
```

Use this command to specify a remote IP address or hostname to which connections are made. This address or hostname is required by a client to specify a server endpoint in a remote access mode tunnel. It is also required by both sides in site-to-site mode.

Use the `set` form of this command to specify a remote IP address or hostname to which connections are made.

Use the `delete` form of this command to remove a remote IP address or hostname to which connections are made.

Use the `show` form of this command to view a remote IP address or hostname to which connections are made.

interfaces openvpn <vtunx> remote-port <port>

Specifies a port on which outgoing sessions are sent.

Syntax:

```
set interfaces openvpn vtunx remote-port port
```

Syntax:

```
delete interfaces openvpn vtunx remote-port
```

Syntax:

```
show interfaces openvpn vtunx remote-port
```

The default port number is 1194.

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

port

The number of a port on which outgoing sessions are sent. The default port number is 1194.

Configuration mode

```
interfaces {
    openvpn vtunx {
        remote-port port
    }
}
```

Use this command to specify a remote UDP or TCP port on which outgoing sessions are sent. This port can be used as a client endpoint in a remote access mode tunnel, either endpoint when UDP is used in site-to-site mode, or the TCP-active endpoint when TCP is used in site-to-site mode. Note that, if specified, the remote port setting on one endpoint must match the local port setting on the other, and conversely.

Use the `set` form of this command to specify a remote UDP or TCP port on which outgoing sessions are sent.

Use the `delete` form of this command to remove a remote UDP or TCP port on which outgoing sessions are sent.

Use the `show` form of this command to view a remote UDP or TCP port on which outgoing sessions are sent.

interfaces openvpn <vtunx> replace-default-route

Specifies that the default route should be through the OpenVPN tunnel.

Syntax:

```
set interfaces openvpn vtunx replace-default-route [ local ]
```

Syntax:

```
delete interfaces openvpn vtunx replace-default-route
```

Syntax:

```
show interfaces openvpn vtunx replace-default-route
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

local

Optional. This option must be set if and only if the two tunnel endpoints are directly connected, i.e., on the same subnet.

Configuration mode

```
interfaces {
    openvpn vtunx {
        replace-default-route {
            local
        }
    }
}
```

Use this command to tell OpenVPN that the default route should be replaced by a route through the VPN tunnel, i.e., split tunneling should be disabled. Note that, when set, this option has different effects depending on the OpenVPN mode in which the endpoint operates.

If the endpoint is in site-to-site mode or client mode, setting **replace-default-route** will replace the default route on this endpoint with a route through VPN tunnel. In other words, it disables split tunneling on this endpoint.

If the endpoint is in server mode, setting **replace-default-route** will cause the clients connecting to this server to replace their default route. In other words, it disables split tunneling on the clients.

Use the `set` form of this command to specify that the default route should be through the OpenVPN tunnel.

Use the `delete` form of this command to remove the configuration.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> server

Defines an OpenVPN server mode endpoint.

Syntax:

```
set interfaces openvpn vtunx server
```

Syntax:

```
delete interfaces openvpn vtunx server
```

Syntax:

```
show interfaces openvpn vtunx server
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
```

```
    openvpn vtunx {
        server {
        }
    }
}
```

Use this command to define an OpenVPN server mode endpoint.

Use the `set` form of this command to create the server mode configuration node.

Use the `delete` form of this command to remove the server mode configuration node.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> server client <client-name>

Defines a client site on the server in a remote access environment.

Syntax:
`set interfaces openvpn` *vtunx* **`server client`** *client-name*

Syntax:
`delete interfaces openvpn` *vtunx* **`server client`**

Syntax:
`show interfaces openvpn` *vtunx* **`server client`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

client-name
Mandatory. The “name” of the client. It corresponds to the “common name” contained in the client's certificate. When a client initiates the VPN session, the server uses the name in the certificate to look up and apply client-specific settings (if any).

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            client client-name {
            }
        }
    }
}
```

Use this command to define a client site on the server in a remote access environment.

> **Note:** Committing configuration changes to this configuration node does not result in the OpenVPN process being restarted. The configuration change will take effect the next time the client connects to the server.

Use the `set` form of this command to create the client configuration node.

Use the `delete` form of this command to remove the client configuration node.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> server client <client-name> disable

Specifies a client that will be disallowed from connecting to the OpenVPN server.

Syntax:
`set interfaces openvpn` *vtunx* **`server client`** *client-name* **`disable`**

Syntax:
`delete interfaces openvpn` *vtunx* **`server client`** *client-name* **`disable`**

Syntax:
`show interfaces openvpn` *vtunx* **`server client`** *client-name*

Clients are allowed to connect to the OpenVPN server.

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

client-name
Mandatory. The “name” of the client. It corresponds to the “common name” contained in the client's certificate.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            client client-name {
                disable
            }
        }
    }
}
```

Use this command to specify a client that will be disallowed from connecting to the OpenVPN server the next time it tries to connect (and any subsequent attempts). This will not affect the client’s current connection. The current connection can be reset using the `monitor interfaces openvpn` *vtunx* **`traffic`** command.

Note: If an OpenVPN client tries to connect to an OpenVPN server and is refused by the server, the client process will exit.

Use the `set` form of this command to specify a client that will be disallowed from connecting to the OpenVPN server on any subsequent attempts.

Use the `delete` form of this command to allow the client to connect to the OpenVPN server.

Note: An OpenVPN client needs to be restarted before it will initiate a new connection to the OpenVPN server.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> server client <client-name> ip <ipv4>

Specifies the IP address of a client in a remote access environment.

Syntax:
`set interfaces openvpn` *vtunx* **`server client`** *client-name* **`ip`** *ipv4*

Syntax:
`delete interfaces openvpn` *vtunx* **`server client`** *client-name* **`ip`**

Syntax:

```
show interfaces openvpn vtunx server client client-name ip
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

client-name
Mandatory. The “name” of the client. It corresponds to the “common name” contained in the client's certificate. When a client initiates the VPN session, the server uses the name in the certificate to look up and apply client-specific settings (if any).

ipv4
The IP address to be assigned to the client.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            client client-name {
                ip ipv4
            }
        }
    }
}
```

Use this command to specify the IP address to assign to the client in a remote access environment.

> **Note:** Committing configuration changes to this configuration node does not result in the OpenVPN process being restarted. The configuration change will take effect the next time the client connects to the server.

Use the `set` form of this command to specify the IP address to assign to the client in a remote access environment.

Use the `delete` form of this command to remove the IP address.

Use the `show` form of this command to view the IP address.

interfaces openvpn <vtunx> server client <client-name> push-route <ipv4net>

Specifies a route to be pushed to a client in a remote access environment.

Syntax:
```
set interfaces openvpn vtunx server client client-name push-route ipv4net
```

Syntax:
```
delete interfaces openvpn vtunx server client client-name push-route ipv4net
```

Syntax:
```
show interfaces openvpn vtunx server client client-name push-route
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

client-name
Mandatory. The “name” of the client. It corresponds to the “common name” contained in the client's certificate. When a client initiates the VPN session, the server uses the name in the certificate to look up and apply client-specific settings (if any).

ipv4net
Multi-node. The subnet to be made accessible to the OpenVPN client through the OpenVPN server.

You can define multiple subnets to push to clients by creating multiple **push-route** configuration nodes. Each must have a unique IPv4net address.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            client client-name {
                push-route ipv4net
            }
        }
    }
}
```

Use this command to specify a subnet that the client can access by routing packets through the server.

Note: Committing configuration changes to this configuration node does not result in the OpenVPN process being restarted. The configuration change will take effect the next time the client connects to the server. Use the `reset interfaces openvpn` *vtunx* command on the client to reset the connection.

Use the `set` form of this command to specify a route to be pushed to all clients.

Use the `delete` form of this command to remove the route configuration.

Use the `show` form of this command to view the route configuration.

interfaces openvpn <vtunx> server client <client-name> subnet <ipv4net>

Specifies a subnet at a client site in a remote access environment.

Syntax:
`set interfaces openvpn` *vtunx* **`server client`** *client-name* **`subnet`** *ipv4net*

Syntax:
`delete interfaces openvpn` *vtunx* **`server client`** *client-name* **`subnet`**

Syntax:
`show interfaces openvpn` *vtunx* **`server client`** *client-name* **`subnet`**

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

client-name
Mandatory. The “name” of the client. It corresponds to the “common name” contained in the client's certificate. When a client initiates the VPN session, the server uses the name in the certificate to look up and apply client-specific settings (if any).

ipv4net
Multi-node. A subnet at the client site.

You can define multiple subnet addresses by creating multiple **subnet** configuration nodes. Each must have a unique IPv4 network address.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            client client-name {
                subnet ipv4net
```

```
            }
        }
    }
}
```

Use this command to identify a subnet at a client site in a remote access environment.

Note: Committing configuration changes to this configuration node does not result in the OpenVPN process being restarted. The configuration change will take effect the next time the client connects to the server.

Use the `set` form of this command to specify the subnet.

Use the `delete` form of this command to remove the subnet configuration.

Use the `show` form of this command to view the subnet configuration.

interfaces openvpn <vtunx> server domain-name <domain-name>

Provides the domain name for OpenVPN clients.

Syntax:
`set interfaces openvpn` *vtunx* **`server domain-name`** *domain-name*

Syntax:
`delete interfaces openvpn` *vtunx* **`server domain-name`**

Syntax:
`show interfaces openvpn` *vtunx* **`server domain-name`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

domain-name
The domain name to be given to OpenVPN clients connected to this OpenVPN server. A domain name can include letters, numbers, hyphens (“-”), and one period (“.”). For example, “vyatta.com”.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            domain-name domain-name
        }
    }
}
```

Use this command to specify the domain name to be given to OpenVPN clients connected to this OpenVPN server.

Note: Certain applications on Windows clients (for example, “ipconfig”), refer to the domain name as the “Connection-specific DNS Suffix”.

Use the `set` form of this command to specify the domain name to be given to OpenVPN clients connected to this OpenVPN server.

Use the `delete` form of this command to remove the domain name configuration.

Use the `show` form of this command to view the domain name configuration.

interfaces openvpn <vtunx> server max-connections <number>

Specifies the maximum number of clients that can connect to the server in a remote access environment.

Syntax:
```
set interfaces openvpn vtunx server max-connections number
```

Syntax:
```
delete interfaces openvpn vtunx server max-connections
```

Syntax:
```
show interfaces openvpn vtunx server max-connections
```

The number of clients that can connect is either 1024 or the limit of system resources.

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

number
The maximum number of client connections that the server accepts. The number ranges from 1 through 1024. The default is 1024.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            max-connections number
        }
    }
}
```

Use this command to specify the maximum number of client connections that the server accepts. Once the limit is reached, any additional clients that attempt to connect to the server will be refused.

Use the `set` form of this command to specify the maximum number of clients that can connect to the server.

Use the `delete` form of this command to return to the default configuration.

Use the `show` form of this command to view the maximum number of client connections configured.

interfaces openvpn <vtunx> server name-server <ipv4>

Specifies a name server address to be pushed to clients in a remote access environment.

Syntax:
```
set interfaces openvpn vtunx server name-server ipv4
```

Syntax:
```
delete interfaces openvpn vtunx server name-server
```

Syntax:
```
show interfaces openvpn vtunx server name-server
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

ipv4
Multi-node. The IPv4 address of the name server to push to clients.

You can define multiple name server addresses to push to clients by creating multiple **name-server** configuration nodes. Each must have a unique IPv4 address.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            name-server ipv4
        }
    }
}
```

Use this command to specify an IPv4 address of a name server to be pushed to clients in OpenVPN remote access mode. This is supported by Windows clients. Other client types may not support this.

Use the `set` form of this command to specify an IPv4 address of a name server to be pushed to clients.

Use the `delete` form of this command to remove the name server configuration.

Use the `show` form of this command to view the name server configuration.

interfaces openvpn <vtunx> server push-route <ipv4net>

Specifies a route to be pushed to all clients in a remote access environment.

Syntax:
`set interfaces openvpn` *vtunx* **`server push-route`** *ipv4net*

Syntax:
`delete interfaces openvpn` *vtunx* **`server push-route`**

Syntax:
`show interfaces openvpn` *vtunx* **`server push-route`**

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

ipv4net
Multi-node. The subnet to be made accessible to the OpenVPN clients through the OpenVPN server.

You can define multiple subnets to push to clients by creating multiple **push-route** configuration nodes. Each must have a unique IPv4net address.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            push-route ipv4net
        }
    }
}
```

Use this command to specify a subnet that all clients can access by routing packets through the server. This route is pushed to all clients and the OpenVPN process is restarted.

Use the `set` form of this command to specify a route to be pushed to all clients.

Use the `delete` form of this command to remove the route configuration.

Use the `show` form of this command to view the route configuration.

interfaces openvpn <vtunx> server subnet <ipv4net>

Specifies the subnet from which client IP addresses are allocated.

Syntax:
`set interfaces openvpn` *vtunx* **`server subnet`** *ipv4net*

Syntax:
`delete interfaces openvpn` *vtunx* **`server subnet`**

Syntax:
`show interfaces openvpn` *vtunx* **`server subnet`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

ipv4net
The subnet from which client IP addresses are allocated. The prefix for the subnet must be /29 or smaller.

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            subnet ipv4net
        }
    }
}
```

This command is used on the server side of a remote access OpenVPN connection and specifies the subnet on which the remote clients will receive IP addresses.

Use this command to specify the subnet from which client IP addresses are allocated.

Use the `set` form of this command to specify the subnet.

Use the `delete` form of this command to remove the subnet configuration.

Use the `show` form of this command to view the subnet configuration.

interfaces openvpn <vtunx> server topology <topology>

Specifies the topology to use in a remote access environment.

Syntax:
`set interfaces openvpn` *vtunx* **`server topology`** *topology*

Syntax:
`delete interfaces openvpn` *vtunx* **`server topology`**

Syntax:
`show interfaces openvpn` *vtunx* **`server topology`**

The default is subnet.

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

topology
The topology used in remote access mode. Supported values are as follows:

point-to-point: This topology provides “client isolation” (that is, the clients cannot reach each other) but is not compatible with Windows clients, and routing protocols using a broadcast-style network would not work with this.

subnet: This topology is compatible with OpenVPN clients on Windows hosts and is the default if topology is not set. Routing protocols that are configured to use a broadcast-style network should work with this topology. However, this topology does not provide “client isolation” (that is, the clients can reach each other).

Configuration mode

```
interfaces {
    openvpn vtunx {
        server {
            topology topology
        }
    }
}
```

Use this command to specify the topology to use in a remote access environment.

Use the `set` form of this command to specify the topology.

Use the `delete` form of this command to remove the topology configuration.

Use the `show` form of this command to view the topology configuration.

interfaces openvpn <vtunx> shared-secret-key-file <filename>

Specifies the file containing a secret key shared with the remote end of the tunnel.

Syntax:

```
set interfaces openvpn vtunx shared-secret-key-file filename
```

Syntax:

```
delete interfaces openvpn vtunx shared-secret-key-file
```

Syntax:

```
show interfaces openvpn vtunx shared-secret-key-file
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

filename

The shared secret file. The file can be generated using the `generate openvpn key` operational command, and the other endpoint must have the same file for the preshared secret mechanism to work.

Configuration mode

```
interfaces {
    openvpn vtunx {
        shared-secret-key-file filename
    }
}
```

Use this command to specify the file containing a secret key shared with the remote end of the tunnel.

Certificate and key files are assumed to be in `/config/auth` unless an absolute path is specified.

Use the `set` form of this command to specify the file containing a secret key shared with the remote end of the tunnel.

Use the `delete` form of this command to remove the shared secret key file configuration.

Use the `show` form of this command to view the shared secret key file configuration.

interfaces openvpn <vtunx> tls

Defines a Transport Layer Security (TLS) configuration.

Syntax:
```
set interfaces openvpn vtunx tls
```

Syntax:
```
delete interfaces openvpn vtunx tls
```

Syntax:
```
show interfaces openvpn vtunx tls
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
        }
    }
}
```

Use this command to define a Transport Layer Security (TLS) configuration.

Use the `set` form of this command to create the TLS configuration node.

Use the `delete` form of this command to remove the TLS configuration node.

Use the `show` form of this command to view the TLS configuration.

interfaces openvpn <vtunx> tls ca-cert-file <filename>

Specifies the file containing the certificate authority's certificate.

Syntax:
```
set interfaces openvpn vtunx tls ca-cert-file filename
```

Syntax:
```
delete interfaces openvpn vtunx tls ca-cert-file
```

Syntax:
```
show interfaces openvpn vtunx tls ca-cert-file
```

vtunx
The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

filename
The file containing the certificate authority's certificate, which will be used to validate the other endpoint's certificate.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
            ca-cert-file filename
        }
    }
}
```

Use this command to specify the file containing the certificate authority's certificate.

Certificate and key files are assumed to be in `/config/auth` unless an absolute path is specified.

Use the `set` form of this command to specify the file containing the certificate authority's certificate.

Use the `delete` form of this command to remove the pointer to the file containing the certificate authority's certificate.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> tls cert-file <filename>

Specifies the file containing the endpoint's own certificate.

Syntax:
`set interfaces openvpn` *vtunx* **`tls cert-file`** *filename*

Syntax:
`delete interfaces openvpn` *vtunx* **`tls cert-file`**

Syntax:
`show interfaces openvpn` *vtunx* **`tls cert-file`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

filename
The file containing the endpoint's own certificate, which will be presented to the other endpoint during the TLS negotiation.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
            cert-file filename
        }
    }
}
```

Use this command to specify the file containing the endpoint's own certificate.

Certificate and key files are assumed to be in `/config/auth` unless an absolute path is specified.

Use the `set` form of this command to specify the file containing the endpoint's certificate.

Use the `delete` form of this command to remove the pointer to the file containing the endpoint's certificate.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> tls crl-file <filename>

Specifies the file containing a certificate revocation list.

Syntax:
`set interfaces openvpn` *vtunx* **`tls crl-file`** *filename*

Syntax:
`delete interfaces openvpn` *vtunx* **`tls crl-file`**

Syntax:
`show interfaces openvpn` *vtunx* **`tls crl-file`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

filename
A file containing a list of certificates that have been revoked, which will prevent endpoints with these certificates from establishing a VPN tunnel. Specifying this file in the TLS configuration is optional.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
            crl-file filename
        }
    }
}
```

Use this command to specify the file containing a certificate revocation list.

The file is assumed to be located in `/config/auth` unless an absolute path is specified.

Use the `set` form of this command to specify the file containing a certificate revocation list.

Use the `delete` form of this command to remove the pointer to the file containing a certificate revocation list.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> tls dh-file <filename>

Specifies the file containing Diffie Hellman parameters.

Syntax:
`set interfaces openvpn` *vtunx* **`tls dh-file`** *filename*

Syntax:
`delete interfaces openvpn` *vtunx* **`tls dh-file`**

Syntax:
`show interfaces openvpn` *vtunx* **`tls dh-file`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

filename
A file containing Diffie Hellman parameters that are required only by the endpoint taking the passive role in the TLS negotiation.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
```

```
            dh-file filename
        }
    }
}
```

Use this command to specify the file containing Diffie Hellman parameters.

The file is assumed to be in `/config/auth` unless an absolute path is specified.

Use the `set` form of this command to specify the file containing Diffie Hellman parameters.

Use the `delete` form of this command to remove the pointer to the file containing Diffie Hellman parameters.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> tls key-file <filename>

Specifies the file containing the endpoint's own private key.

Syntax:

```
set interfaces openvpn vtunx tls key-file filename
```

Syntax:

```
delete interfaces openvpn vtunx tls key-file
```

Syntax:

```
show interfaces openvpn vtunx tls key-file
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

filename

A file containing the endpoint's own private key, which is kept secret from everyone.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
            key-file filename
        }
    }
}
```

Use this command to specify the file containing the endpoint's own private key.

The file is assumed to be in `/config/auth` unless an absolute path is specified.

Use the `set` form of this command to specify the file containing the endpoint's own private key.

Use the `delete` form of this command to remove the pointer to the file containing the endpoint's own private key.

Use the `show` form of this command to view the configuration.

interfaces openvpn <vtunx> tls role <role>

Specifies the TLS role the endpoint will take.

Syntax:

```
set interfaces openvpn vtunx tls role role
```

Syntax:

```
delete interfaces openvpn vtunx tls
```

Syntax:
`show interfaces openvpn` *vtunx* **`tls`**

vtunx
The identifier for the OpenVPN interface. This may be **`vtun0`** to **`vtun`***x*, where *x* is a non-negative integer.

role
The TLS role that the endpoint will take. Supported values are as follows:

`active`: The endpoint takes the active role.

`passive`: The endpoint takes the passive role.

Configuration mode

```
interfaces {
    openvpn vtunx {
        tls {
            role role
        }
    }
}
```

Use this command to specify the TLS role the endpoint will take.

Use the `set` form of this command to specify the TLS role the endpoint will take.

Use the `delete` form of this command to remove the TLS role.

Use the `show` form of this command to view the configuration.

monitor interfaces openvpn <vtunx> traffic

Displays (captures) traffic on the OpenVPN interface.

Syntax:
`monitor interfaces openvpn` *vtunx* **`traffic`** [**`detail`** [**`filter`** *filter-name* | **`unlimited`** [**`filter`** *filter-name*]] | **`filter`** *filter-name* | **`save`** *filename* | **`unlimited`** [**`filter`** *filter-name*]]]

interface
The OpenVPN interface name.

`detail`
Provides detailed information about the monitored OpenVPN traffic.

`filter` ***filter-name***
Applies the specific PCAP (packet capture) filter to traffic.

`save` ***filename***
Saves the monitored traffic to the specified file.

`unlimited`
Monitors an unlimited amount of traffic.

Operational mode

```
monitor {
    interfaces {
        openvpn vtunx {
            traffic {
                detail filter filter-name
                detail unlimited filter filter-name
                filter filter-name
                save filename
                unlimited filter filter-name
            }
        }
```

```
    }
}
```

Use this command to capture traffic on the OpenVPN interface.

The following example shows the output of this command.

```
vyatta@vyatta# monitor interfaces openvpn vtun0 traffic
Capturing traffic on vtun0 ...
```

reset openvpn interface <vtunx>

Resets all tunnel connections on an OpenVPN interface.

Syntax:

```
reset openvpn interface vtunx
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun***x*, where *x* is a non-negative integer.

Operational mode

Use this command to reset all tunnel connections on an OpenVPN interface. In a site-to-site environment, the connection will be re-established after it is reset. This is also the case on the client side in a remote access environment. On the server side, in a remote access environment, all connections are dropped. The server will then wait for the clients to re-establish the connections.

Note: The OpenVPN process does not get restarted by this command, though all tunnel connections are reset.

reset openvpn interface <vtunx> client <client-name>

Resets a client connection.

Syntax:

```
reset openvpn interface vtunx client client-name
```

vtunx

The identifier for the OpenVPN interface. This may be **vtun0** to **vtun** *x*, where x is a non-negative integer.

client-name

The "name" of the client. It corresponds to the "common name" contained in the client's certificate. When a client initiates the VPN session, the server uses the name in the certificate to look up and apply client-specific settings (if any). This can be determined using the `show openvpn server status` command.

Operational mode

Use this command to reset the connection to a specific client. The connection to the client will be disconnected and the server will wait for the client to re-establish the connection.

Note: The OpenVPN process does not get restarted by this command.

show interfaces openvpn

Displays a status summary of all OpenVPN interfaces.

Syntax:

```
show interfaces openvpn
```

Operational mode

Use this command to display the high level status of all OpenVPN interfaces on the system.

The following example shows the output of this command.

```
vyatta@vyatta# show interfaces openvpn
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
vtun0            192.168.200.1/32                  u/u
```

show interfaces openvpn <interface>

Displays the detailed status of an OpenVPN interface.

Syntax:

```
show interfaces openvpn interface
```

interface

The OpenVPN interface name.

Operational mode

Use this command to display detailed status of an OpenVPN interface.

The following example shows the output of this command.

```
vyatta@vyatta# show interfaces openvpn vtun0
vtun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen
 100
    link/[65534]
    inet 192.168.200.1 peer 192.168.200.2/32 scope global vtun0

    RX:  bytes    packets     errors    dropped    overrun      mcast
          1216         16          0          0          0          0
    TX:  bytes    packets     errors    dropped    carrier collisions
             0          0          0          0          0          0
```

show interfaces openvpn <interface> brief

Displays the status summary of an OpenVPN interface.

Syntax:

```
show interfaces openvpn interface brief
```

interface

The OpenVPN interface name.

Operational mode

Use this command to display a status summary of an OpenVPN interface.

The following example shows the output of this command.

```
vyatta@vyatta# show interfaces openvpn vtun0 brief
```

```
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                     S/L  Description
---------        ----------                     ---  -----------
vtun0            192.168.200.1/32               u/u
```

show interfaces openvpn detail

Displays the detailed status of all OpenVPN interfaces on the system.

Syntax:

```
show interfaces openvpn detail
```

Operational mode

Use this command to display detailed status of all OpenVPN interfaces on the system.

The following example shows the output of this command.

```
vyatta@vyatta# show interfaces openvpn detail
vtun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen
 100
    link/[65534]
    inet 192.168.200.1 peer 192.168.200.2/32 scope global vtun0

    RX:  bytes    packets     errors    dropped    overrun      mcast
          1216         16          0          0          0          0
    TX:  bytes    packets     errors    dropped    carrier collisions
             0          0          0          0          0          0
```

show openvpn client status

Displays information on OpenVPN connections in client mode.

Syntax:

```
show openvpn client status
```

Operational mode

Use this command to display information on all OpenVPN client mode connections. This command is only available on a client-mode endpoint.

The following example shows the output of this command.

```
vyatta@vyatta:~$ show openvpn client status
OpenVPN client status on vtun2 [openvpn client]

Server CN       Remote IP       Tunnel IP       TX byte RX byte Connected Since
--------------- --------------- --------------- ------- ------- ---------------------
N/A             172.16.117.128  N/A                6.8K    8.1K N/A
```

show openvpn server status

Displays information on connected clients in server mode.

Syntax:

```
show openvpn server status
```

Operational mode

Use this command to display information on all connected clients. This command is only available on a server-mode endpoint. Also, note that the command output is not updated in real time. The time it was last updated is displayed.

The following example shows the output of this command.

```
vyatta@vyatta:~$ show openvpn server status
OpenVPN server status on vtun0 (last updated on Wed Oct 29 22:34:18 2008)

Client          Remote IP       Tunnel IP       TX byte RX byte Connected Since
--------------- --------------- --------------- ------- ------- ----------------
vclient1        192.168.252.3   192.168.200.4   16.0K   16.5K  Wed Oct 29 21:59:50 2008
```

show openvpn site-to-site status

Displays information on OpenVPN connections in site-to-site mode.

Syntax:

```
show openvpn site-to-site status
```

Operational mode

Use this command to display information on all connected sites. This command is only available on a site-to-site-mode endpoint.

The following example shows the output of this command.

```
vyatta@vyatta:~$ show openvpn site-to-site status
OpenVPN client status on vtun1 [openvpn with psk]

Remote CN       Remote IP       Tunnel IP       TX byte RX byte Connected Since
--------------- --------------- --------------- ------- ------- ---------------------
None (PSK)      192.168.74.160  192.168.2.2        8.9K    8.8K N/A

OpenVPN client status on vtun0 [openvpn with tls]

Remote CN       Remote IP       Tunnel IP       TX byte RX byte Connected Since
--------------- --------------- --------------- ------- ------- ---------------------
N/A             192.168.74.160  192.168.1.2       17.5K   15.1K N/A
```

Related commands

Commands for using other system features with OpenVPN interfaces are described in the following guides:

Related Commands Documented Elsewhere	
Bridging	Commands for configuring bridge groups on OpenVPN interfaces are described in AT&T Vyatta Network Operating System Bridging Configuration Guide.
Firewall	Commands for configuring firewall on OpenVPN interfaces are described in AT&T Vyatta Network Operating System Firewall Configuration Guide.

Related Commands Documented Elsewhere	
OSPF	Commands for configuring the Open Shortest Path First routing protocol on OpenVPN interfaces are described in AT&T Vyatta Network Operating System OSPF Configuration Guide.
Policy Based Routing	Commands for configuring Policy Based Routing on OpenVPN interfaces are described in AT&T Vyatta Network Operating System Policy-based Routing Configuration Guide.
QoS	Commands for configuring quality of service on OpenVPN interfaces are described in AT&T Vyatta Network Operating System QoS Configuration Guide.
RIP	Commands for configuring the Routing Information Protocol on OpenVPN interfaces are described in AT&T Vyatta Network Operating System RIP Configuration Guide.
RIPng	Commands for configuring RIPng on OpenVPN interfaces are described in AT&T Vyatta Network Operating System RIPng Configuration Guide.

List of Acronyms

Acronym	Description
ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
AH	Authentication Header
AMI	Amazon Machine Image
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
CCMP	AES in counter mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CLI	command-line interface
DDNS	dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DLCI	data-link connection identifier
DMI	desktop management interface
DMVPN	dynamic multipoint VPN
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
eBGP	external BGP
EBS	Amazon Elastic Block Storage
EC2	Amazon Elastic Compute Cloud
EGP	Exterior Gateway Protocol
ECMP	equal-cost multipath
ESP	Encapsulating Security Payload
FIB	Forwarding Information Base
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
HDLC	High-Level Data Link Control
I/O	Input/Output
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers

Acronym	Description
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IPS	Intrusion Protection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPOA	IP over ATM
IPsec	IP Security
IPv4	IP Version 4
IPv6	IP Version 6
ISAKMP	Internet Security Association and Key Management Protocol
ISM	Internet Standard Multicast
ISP	Internet Service Provider
KVM	Kernel-Based Virtual Machine
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	medium access control
mGRE	multipoint GRE
MIB	Management Information Base
MLD	Multicast Listener Discovery
MLPPP	multilink PPP
MRRU	maximum received reconstructed unit
MTU	maximum transmission unit
NAT	Network Address Translation
NBMA	Non-Broadcast Multi-Access
ND	Neighbor Discovery
NHRP	Next Hop Resolution Protocol
NIC	network interface card
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OSPFv2	OSPF Version 2
OSPFv3	OSPF Version 3
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PAT	Port Address Translation
PCI	peripheral component interconnect
PIM	Protocol Independent Multicast
PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPPoA	PPP over ATM

Acronym	Description
PPPoE	PPP over Ethernet
PPTP	Point-to-Point Tunneling Protocol
PTMU	Path Maximum Transfer Unit
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RHEL	Red Hat Enterprise Linux
RIB	Routing Information Base
RIP	Routing Information Protocol
RIPng	RIP next generation
RP	Rendezvous Point
RPF	Reverse Path Forwarding
RSA	Rivest, Shamir, and Adleman
Rx	receive
S3	Amazon Simple Storage Service
SLAAC	Stateless Address Auto-Configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SPT	Shortest Path Tree
SSH	Secure Shell
SSID	Service Set Identifier
SSM	Source-Specific Multicast
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
ToS	Type of Service
TSS	TCP Maximum Segment Size
Tx	transmit
UDP	User Datagram Protocol
VHD	virtual hard disk
vif	virtual interface
VLAN	virtual LAN
VPC	Amazon virtual private cloud
VPN	virtual private network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network
WAP	wireless access point
WPA	Wired Protected Access