Orbit OpenVPN Feature Guide

05-7231A01, Version B

Table of Contents

INTRODUCTION 4

Scope and Purpose 4

Abbreviations, Acronyms and Symbols 4

Applicable Documents 4

Normative 4

Informative 4

OVERVIEW 5

FEATURES 5

USE CASES 7

Decoding an OPENVPN Config File 7

DESCRIPTION 7

CONFIGURIng key-value: dev 7

CONFIGURIng key-values: remote, PROTO, PORT 8

CONFIGURIng key-values: cipher, auth, compress 8

CONFIGURIng TAG-values: CA, CERT, KEY 9

Connecting an openvpn client to an ORBIT openVPN server 11

DESCRIPTION 11

CONFIGURATION 11

Connecting an orbit openvpn client to an orbit openvpn server 14

DESCRIPTION 14

CONFIGURATION 14

Connecting an orbit openvpn client to an external openvpn server 17

DESCRIPTION 17

CONFIGURATION 17

configuring an orbit openvpn server for orbit openvpn clients 20

DESCRIPTION 20

CONFIGURATION 20

Table of Figures

Figure 1 Connecting an OpenVPN client to an Orbit OpenVPN server 11

Figure 2 Connecting an Orbit OpenVPN client to an Orbit OpenVPN server 14

Figure 3 Connecting an Orbit OpenVPN client to an external OpenVPN server 17

Figure 4 Configuring an Orbit OpenVPN server for Orbit OpenVPN clients 20

INTRODUCTION

SCOPE AND PURPOSE

This manual describes the typical use cases and configuration examples for the OpenVPN feature on the Orbit platform.

ABBREVIATIONS, ACRONYMS AND SYMBOLS

VPN: Virtual Private Network

APPLICABLE DOCUMENTS

The following documents form a part of this specification to the extent specified herein. Referenced documents are available from GE MDS or as an industry standard.

NORMATIVE

Ref	Title	Document Number	Location
RFC-KEYORDS	RFC 2119		http://datatracker.ietf.org/doc/rfc2119/

INFORMATIVE

Ref	Title	Document Number	Location

OVERVIEW

A virtual private network (VPN) is a system that lets you create secure point-to-point or site-to-site connections.

OpenVPN is a VPN system that is released under the GNU GPLv2 Open License. The OpenVPN daemon can be used to implement both the client and server side of the VPN deployment. OpenVPN uses the following to implement a VPN:

- OpenSSL to encrypt both data and control channels
- Transport layer security (TLS) for authentication
- TUN/TAP virtual network devices to establish communication

The OpenVPN feature on the Orbit allows the following:

- The ability to setup one OpenVPN server instance
    - This server instances allows two clients to connect to it
- The ability to setup two OpenVPN client instances
    - Each client instance can connect to one external OpenVPN server
- Allows for IPv4 only for the initial version
- Allows for TUN virtual network device support only for the initial version

This feature guide will provide guidance on utilizing OpenVPN on the Orbit for the following scenarios:

- Decoding the OpenVPN file format (*.ovpn or *.conf)
- Configuring an OpenVPN client on the Orbit for different use cases
- Configuring an OpenVPN server on the Orbit for different use cases

FEATURES

The following table describes the features supported for the OpenVPN functionality.

S.No.	Feature	Description
1	Creating a TUN virtual network device	Support for communications using the TUN (Layer 3) virtual network device
2	IPv4 support	IPv4 only support at present
3	TCP and UDP support	Support for OpenVPN using the TCP and UDP protocol
4	Encryption ciphers	The following encryption ciphers are supported at present: - AES-128-CBC (Default) - AES-192-CBC - AES-256-CBC
5	MAC ciphers	The following message authentication ciphers are supported at present: - SHA1 - SHA256 (Default) - SHA384 - SHA512
6	Compression	The following compression types are supported at present: - Disable (Default) - LZ4 - LZO

7	Server-side IP Subnets	Allow IPv4 routes/subnets behind the OpenVPN server that need to be made available to OpenVPN clients. This is configured on the server side.
8	Client-side IP Subnets	Allow IPv4 routes/subnets behind connected OpenVPN clients to be made available to the OpenVPN server and any other clients that connect to it. This is configured on the server side. At present client-to-client routing is not supported.

USE CASES

DECODING AN OPENVPN CONFIG FILE

DESCRIPTION

Some enterprise OpenVPN servers are setup to create an OpenVPN config file (*.ovpn on Windows, and *.conf on *nix) that can be loaded by the OpenVPN client to easily connect to the specified OpenVPN server.

At present the OpenVPN feature on the Orbit can not load and ingest these config files. The entries found in these configuration files needs to be manually configured via the CLI or WebUI for the OpenVPN client on the Orbit to connect to the external OpenVPN server.

The OpenVPN configuration file is a plain text file that is new line delimited, with inline key-value pairs as well as tags used to specify configuration options.

The following key-value pairs in the configuration file are supported:

- dev
- remote
- proto
- port
- cipher
- auth
- compress

The above key-value pairs can have values that are not supported by the OpenVPN feature on the Orbit.

The following tags in the configuration file are supported:

- ca
- cert
- key

CONFIGURING KEY-VALUE: DEV

The “dev” key-value pair in the OpenVPN config file indicates the type of virtual network interface device that the OpenVPN daemon is to use to establish a connection.

The OpenVPN feature on the Orbit only supports communications using TUN based virtual network devices. The following in the configuration file is supported by the OpenVPN feature on the Orbit:

```
dev tun
```

The OpenVPN feature on the Orbit will not support the configuration file if it has the following:

```
dev tap
```

This is because it is specifying the use of a TAP based virtual network device for establishing a VPN connection.

To setup the appropriate TUN virtual network device on the Orbit, enter the following config commands:

```
> config
% set interfaces interface TUN_CLIENT_1 type tuntap tuntap-config mode ip-over-tuntap
% set interfaces interface TUN_CLIENT_1 filter input IN_TRUSTED output OUT_TRUSTED
```

```
% set interfaces interface TUN_CLIENT_1 enabled true
% commit
```

CONFIGURING KEY-VALUES: REMOTE, PROTO, PORT

The “remote” key-value pair in the OpenVPN config file indicates the IP or URL of the OpenVPN server.

```
remote https://example.openvpn.com
remote 192.168.10.1
```

The above are both valid remote key-value pairs accepted by the OpenVPN feature on the Orbit.

The “proto” key-value pair in the OpenVPN config file indicates the protocol the OpenVPN connection should use.

```
proto tcp-client
proto udp
```

The above are both valid proto key-value pairs. The first configuration indicates using TCP and the second indicates using UDP. Only one proto key-value pair is allowed in each configuration file.

The “port” key-value pair in the OpenVPN config file indicates the port that the OpenVPN server is accepting connections on.

```
port 1194
```

The above specifies that the remote port 1194 is supposed to be used to establish a connection to the OpenVPN server. The port type is based on the protocol used. If TCP is used than the port number specifies a TCP port and if UDP is used than the port number specifies a UDP port.

It is possible to have multiple remote key-value pairs in the configuration file as shown below:

```
remote https://example.openvpn.com 1194 udp
remote 192.168.10.1 1192 tcp-client
```

The above specifies two remote server details that the OpenVPN client can attempt to connect to using different ports and protocols.

To setup the remote server information for the OpenVPN feature on the Orbit (client-side), enter the following config commands:

```
> config
% set services openvpn client TUN_CLIENT_1 server https://example.openvpn.com 1194
protocol udp
% set services openvpn client TUN_CLIENT_1 server 192.168.10.1 1192 protocol tcp
% commit
```

CONFIGURING KEY-VALUES: CIPHER, AUTH, COMPRESS

The “cipher” key-value pair in the OpenVPN config file indicates the encryption algorithm to use when communicating between the OpenVPN server and the OpenVPN client.

```
cipher AES-128-CBC
```

The OpenVPN feature on the Orbit only supports communications using only the following encryption algorithms:

- AES-128-CBC (Default)
- AES-192-CBC
- AES-256-CBC

The “auth” key-value pair in the OpenVPN config file indicates the MAC algorithm to use when communicating between the OpenVPN server and the OpenVPN client.

```
auth SHA256
```

The OpenVPN feature on the Orbit only supports communications using only the following encryption algorithms:

- SHA1
- SHA256 (Default)
- SHA384
- SHA512

The “compress” key-value pair in the OpenVPN config file indicates the compression to perform when communicating between the OpenVPN server and the OpenVPN client.

```
compress lzo
compress lz4
```

The above specifies two different compressions algorithms. Only one compress key-value pair is allowed in the configuration file. If it does not exist then the compression algorithm to use is decided by the OpenVPN server or no compression is performed.

To setup the encryption algorithm, MAC algorithm, and compression mode for the OpenVPN feature on the Orbit (client-side), enter the following config commands:

```
> config
% set services openvpn client TUN_CLIENT_1 ciphersuite encryption-algo aes128-cbc mac-
algo sha256-hmac
% set services openvpn client TUN_CLIENT_1 compression lz4
% commit
```

CONFIGURING TAG-VALUES: CA, CERT, KEY

The “ca” tag in the OpenVPN config file indicates the embedded contents of the certificate authority file. This is required for communicating between the OpenVPN server and the OpenVPN client.

```
<ca>
-----BEGIN CERTIFICATE-----
MIIE1jCCA76gAwIBAgIJAOMAQRbD8ADYMA0GCSqGSIb3DQEBCwUAMIGiMQswCQYD
…
xcPC3D4Gk0EW83PJorGi1+lPGNusEDO0xqlv2pLyQ07XVKWsYZo3AKQY
-----END CERTIFICATE-----
</ca>
```

The contents between the <ca> and </ca> need to be copied into to a separate file that then needs to be uploaded to the Orbit using the Orbit’s Certificate Manager. The CA can then be set on the Orbit (client-side), by entering the following config commands:

```
> config
% set services openvpn client TUN_CLIENT_1 pki ca-cert-id ovpn_ca_crt
% commit
```

The “cert” tag in the OpenVPN config file indicates the embedded contents of the certificate file. This is required for communicating between the OpenVPN server and the OpenVPN client.

```
<cert>
-----BEGIN CERTIFICATE-----
MIIE1jCCA76gAwIBAgIJAOMAQRbD8ADYMA0GCSqGSIb3DQEBCwUAMIGiMQswCQYD
…
xcPC3D4Gk0EW83PJorGi1+lPGNusEDO0xqlv2pLyQ07XVKWsYZo3AKQY
-----END CERTIFICATE-----
</cert>
```

The contents between the <cert> and </cert> need to be copied into to a separate file that then needs to be uploaded to the Orbit using the Orbit’s Certificate Manager. The certificate can then be set on the Orbit (client-side), by entering the following config commands:

```
> config
% set services openvpn client TUN_CLIENT_1 pki cert-id ovpn_client_1_crt
% commit
```

The “key” tag in the OpenVPN config file indicates the embedded contents of the key file. This is required for communicating between the OpenVPN server and the OpenVPN client.

```
<key>
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDnz0TCaFU1Dy4s
…
I0arr3xVhS/+VC2DwFQSScWp+uSAT32SG/NihcwUfxEf8F9vKsrIVtE8hZGdPCKe
1izxoc0xwmCSz9QWDkW3ax17
-----END PRIVATE KEY-----
</key>
```

The contents between the <key> and </key> need to be copied into to a separate file that then needs to be uploaded to the Orbit using the Orbit’s Certificate Manager.

The key can then be set on the Orbit (client-side), by entering the following config commands:

```
> config
% set services openvpn client TUN_CLIENT_1 pki key-id ovpn_client_1_key
% commit
root@orbit 16:05:47% show services openvpn server TUN_SERVER_1 | details
enabled          false;
protocol         udp;
port             1194;
tunnel-ip-subnet 10.8.2.0/24;
local-ip-subnets [ 192.168.2.64/26 ];
auth-type        pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_client_1_crt;
    key-id     ovpn_client_1_key;
    ca-cert-id ovpn_ca_bad_crt;
}
renegotiate-key  3600;
ciphersuite {
    encryption-algo aes128-cbc;
    mac-algo        sha256-hmac;
}
compression      disable;
```

CONNECTING AN OPENVPN CLIENT TO AN ORBIT OPENVPN SERVER

DESCRIPTION

This section describes connecting a generic OpenVPN client to an Orbit OpenVPN server. This will allow the local system to access the VPN set up by the Orbit's OpenVPN Server. This will also allow the local system's applications to connect to the resources and features that are exposed by the VPN.

A use case for this section would be as following:

- Having a user use his laptop to connect to a remote network while out of the office to perform diagnostic tasks without having to physically connect to the remote network.

The network layout for this is shown in the figure below.

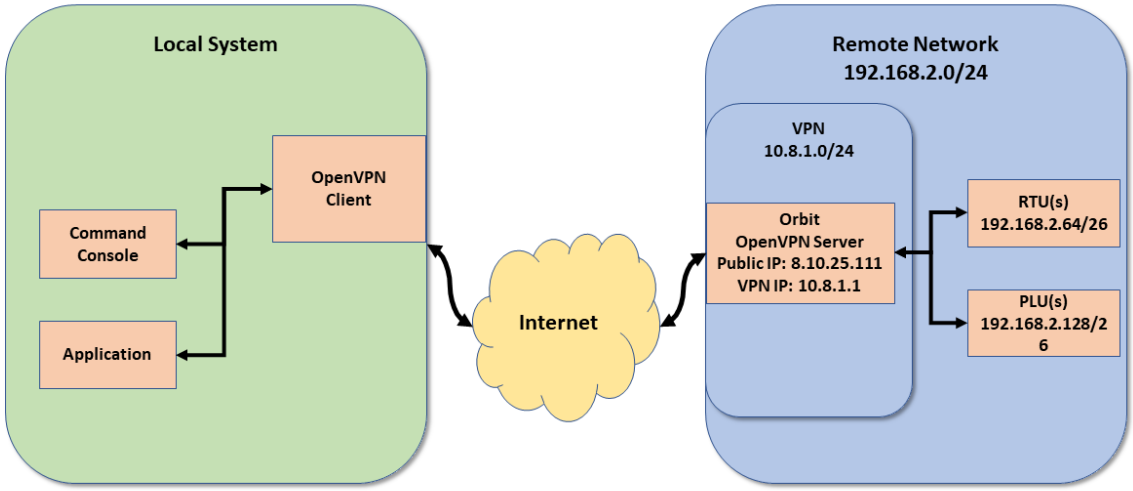

Figure 1 Connecting an OpenVPN client to an Orbit OpenVPN server

CONFIGURATION

The Orbit OpenVPN server for this section is configured as shown below:

```
% show services openvpn server TUN_SERVER_1 | details
enabled          enabled;
protocol         tcp;
port             1194;
tunnel-ip-subnet 10.8.1.0/24;
local-ip-subnets [ 192.168.2.64/26 192.168.2.128/26 ];
auth-type        pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_server_1_crt;
    key-id     ovpn_server_1_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key  3600;
ciphersuite {
    encryption-algo aes128-cbc;
    mac-algo        sha256-hmac;
}
compression      lz4;
```

There are two methods to configure the OpenVPN client on a local system. They are:

1. Using the GUI if the OpenVPN client has one. If you use this refer to the instructions for your OpenVPN client and set the values for the following fields:
    a. Server IP, for this configuration it is 8.10.25.111
    b. Server Port, for this configuration it is 1194
    c. Protocol, for this configuration it is TCP
    d. Encryption Algorithm, for this configuration it is AES-128-CBC
    e. MAC Algorithm, for this configuration it is SHA-256
    f. Compression, for this configuration it is LZ4
    g. CA Certificate, use the CA certificate file provided by the OpenVPN administrator
    h. Local Client Certificate, use the local client certificate file provided by the OpenVPN administrator
    i. Local Client Key, use the local client key file provided by the OpenVPN administrator
    j. NOTE: Certain other OpenVPN configurations when set will conflict with the above configurations and will not allow your local OpenVPN client to connect to the Orbit OpenVPN server
2. Using an OpenVPN configuration file

If your OpenVPN client requires you to provide a configuration file, you will need to create an empty ASCII text file and rename its file type to either:

- *.ovpn for Windows based systems
- *.conf for Linux based systems

Once the appropriate file type has been created, for this example the following will be inserted into the configuration file:

```
#
# Note: Lines beginning with the '#' symbol are not processed and act as a commentary
#

#
# Mandatory config specified for an openvpn client
#
client

#
# By default, do not bind to a specific local port, let the openvpn daemon pick a random
port for you
#
nobind
# Use this if you wish to specify a local port for the connection explicitly
#lport 7771

#
# Specify the openvpn server's external ip and port to connect to
# Update with your IP and port number
#
remote 8.10.25.111 1194

#
# Specify the protocol, choose one
#
proto tcp-client
#proto udp     # Default on Orbit

#
# Specify the tuntap virtual device type that the openvpn daemon should create for you
locally
#
dev tun
#dev tap       # Not supported by the Orbit at present
```

```
#
# Select and uncomment one of the encryption ciphers types to use by the openvpn daemon
#
cipher AES-128-CBC     # Default on Orbit
#cipher AES-192-CBC
#cipher AES-256-CBC

#
# Select and uncomment one of the MAC ciphers types to use by the openvpn daemon
#
#auth SHA1
auth SHA256              # Default on Orbit
#auth SHA384
#auth SHA512

#
# If compression is enabled select/uncomment the appropriate type
# If no compression is specified comment out both compress keywords
#
compress lz4
#compress lzo

#
# You can either specify the full or relative file path to the CA cert
# and the local cert/key pair on your system
#
#ca pki/ca.crt
#cert pki/client_1.crt
#key pki/client_1.key
#
# Or you can embed them in the configuration file as shown towards the end
#

#
# Diagnostic nfo
#
status debug.status 20        # Update the status file every 20 seconds
# Verbosity of the log file, 1-3 is normal, 4-8 is debug level verbosity
verb 6
#log debug_info.log

#
# Keep the OpenVPN client daemon alive
#
keepalive 10 30

#
# Insert the contents of your CA file in between the <ca> tags
#
<ca>
# -----BEGIN CERTIFICATE-----
# -----END CERTIFICATE-----
</ca>

#
# Insert the contents of your certificate file in between the <cert> tags
#
<cert>
# -----BEGIN CERTIFICATE-----
# -----END CERTIFICATE-----
</cert>

#
# Insert the contents of your key file in between the <key> tags
#
<key>
# -----BEGIN KEY-----
# -----END KEY-----
</key>
```

CONNECTING AN ORBIT OPENVPN CLIENT TO AN ORBIT OPENVPN SERVER

DESCRIPTION

This section describes connecting the Orbit's OpenVPN client to an Orbit OpenVPN server. This will allow the local network to be part of the VPN set up by the Orbit's OpenVPN Server. This will also allow the local network to connect to the resources and features that are exposed by the VPN.

A use case for this section would be as following:

- Having a process or system on a local network monitor the state of resources on a remote network and perform conditional tasks.

The network layout for this is shown in the figure below.

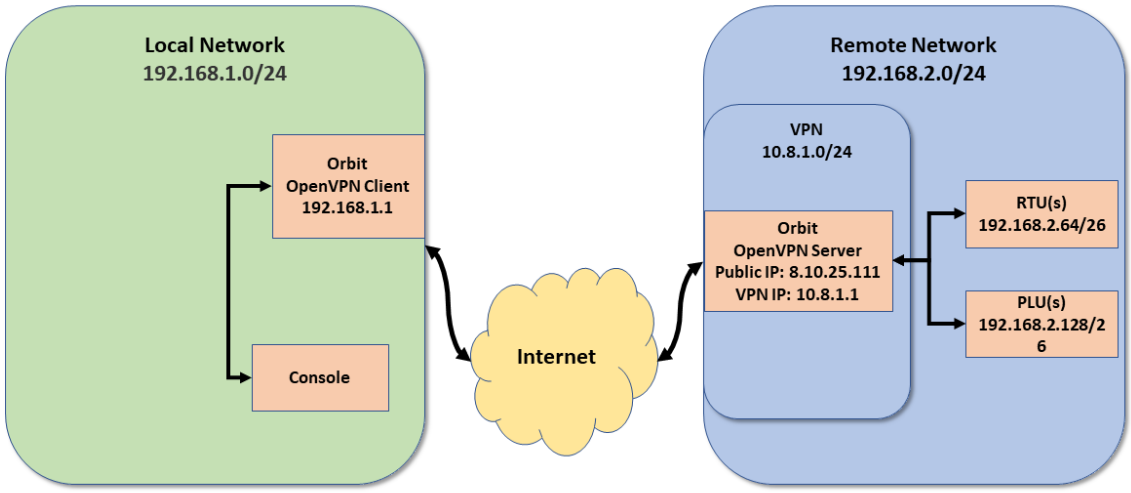

Figure 2 Connecting an Orbit OpenVPN client to an Orbit OpenVPN server

CONFIGURATION

The Orbit OpenVPN server for this section is configured as shown below:

```
% show services openvpn server TUN_SERVER_1 | details
enabled          enabled;
protocol         tcp;
port             1194;
tunnel-ip-subnet 10.8.1.0/24;
local-ip-subnets [ 192.168.2.64/26 192.168.2.128/26 ];
auth-type        pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_server_1_crt;
    key-id     ovpn_server_1_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key  3600;
ciphersuite {
    encryption-algo aes128-cbc;
    mac-algo        sha256-hmac;
}
compression      disable;
```

To configure the Orbit on the local network you will need access to the CLI of that Orbit.

The first step would be to create a TUNTAP interface on the Orbit for the OpenVPN client to use.

```
> configure
% set interfaces interface TUN_CLIENT_1 type tuntap
% set interfaces interface TUN_CLIENT_1 tuntap-config mode ip-over-tuntap
% set interfaces interface TUN_CLIENT_1 filter input IN_TRUSTED
% set interfaces interface TUN_CLIENT_1 filter output OUT_TRUSTED
% set interfaces interface TUN_CLIENT_1 enabled true
% commit
% show interfaces interface TUN_CLIENT_1 | details
type    tuntap;
enabled true;
tuntap-config {
    mode ip-over-tuntap;
}
filter {
    input  IN_TRUSTED;
    output OUT_TRUSTED;
}
```

The above configuration creates a TUNTAP interface named TUN_CLIENT_1 that uses the layer 3 TUN virtual network device that has been configured and enabled for use by the OpenVPN client on the Orbit.

The next step would be to create and configure an OpenVPN client on the Orbit that will connect to the OpenVPN server specified in this section.

```
> configure
% set services openvpn client TUN_CLIENT_1
% set services openvpn client TUN_CLIENT_1 server 8.10.25.111 1194 protocol tcp
% set services openvpn client TUN_CLIENT_1 auth-type pub-key
% set services openvpn client TUN_CLIENT_1 pki cert-type rsa
% set services openvpn client TUN_CLIENT_1 pki ca-cert-id ovpn_ca_crt
% set services openvpn client TUN_CLIENT_1 pki cert-id ovpn_client_1_crt
% set services openvpn client TUN_CLIENT_1 pki key-id ovpn_client_1_key
% set services openvpn client TUN_CLIENT_1 ciphersuite encryption-algo aes128-cbc
% set services openvpn client TUN_CLIENT_1 ciphersuite mac-algo sha256-hmac
% set services openvpn client TUN_CLIENT_1 compression disable
% set services openvpn client TUN_CLIENT_1 enabled true
% commit
% show services openvpn client TUN_CLIENT_1 | details
enabled                 true;
server 8.10.25.111 1194 {
    protocol tcp;
}
server-random-selection false;
auth-type               pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_client_1_crt;
    key-id     ovpn_client_1_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key         3600;
ciphersuite {
    encryption-algo aes128-cbc;
    mac-algo        sha256-hmac;
}
compression             disable;
% exit
```

NOTE: The above configuration assumes that the files [ovpn_ca_crt, ovpn_client_1_crt, ovpn_client_1_key] needed by OpenVPN's PKI (public key infrastructure) have already been generated and uploaded onto the Orbit's using the Certificate Management system.

To verify the status of the OpenVPN client on the Orbit the following commands can be used.

```
> show services openvpn client-state
services openvpn client-state TUN_CLIENT_1
 last-updated            "2021-11-08 15:34:42"
 last-state-message      "CONNECTED SUCCESS 10.8.1.2 8.10.25.111 1194 192.168.1.1 34348"
 device-bytes-received   0
 device-bytes-sent       0
 protocol-bytes-received 6114
```

```
protocol-bytes-sent     6034
auth-bytes-received     704
```

The above shows that the local Orbit's OpenVPN client has been connected to the Orbit OpenVPN server using the IP 8.10.25.111 on port 1194. It also shows that the Orbit's OpenVPN client has been assigned a VPN IP address of 10.8.1.2.

To verify that the appropriate routes for the VPN have been pushed onto the Orbit's OpenVPN client the following commands can be used.

```
> show routing-state routes outgoing-interface TUN_CLIENT_1
                        OUTGOING
DEST PREFIX       NEXT HOP  INTERFACE     SOURCE
-------------------------------------------------
10.8.1.0/24       -         TUN_CLIENT_1  kernel
192.168.2.64/26   10.8.1.1  TUN_CLIENT_1  kernel
192.168.2.128/26  10.8.1.1  TUN_CLIENT_1  kernel
> ping 192.168.2.64
PING 192.168.2.64 (192.168.2.64) 56(84) bytes of data.
64 bytes from 192.168.2.64: icmp_seq=1 ttl=64 time=0.403 ms
64 bytes from 192.168.2.64: icmp_seq=2 ttl=64 time=0.449 ms
64 bytes from 192.168.2.64: icmp_seq=3 ttl=64 time=0.453 ms
> traceroute 192.168.2.64
 1?: [LOCALHOST]                                        pmtu 1500
 1:  10.8.1.1                                             2.113ms
 1:  10.8.1.1                                             1.768ms
 2:  192.168.2.64                                         1.978ms reached
 2:  192.168.2.64                                         1.721ms reached
     Resume: pmtu 1500 hops 2 back 64
```

NOTE: For this configuration, the Orbit OpenVPN client will get assigned a VPN IPv4 address of 10.8.1.2 (since 10.8.1.1 is already assigned to the OpenVPN server). For the data to be transmitted from the external devices to devices within the VPN the appropriate routes need to be added on the OpenVPN server side. For this example, a static route set on the OpenVPN server's virtual interface will suffice.

CONNECTING AN ORBIT OPENVPN CLIENT TO AN EXTERNAL OPENVPN SERVER

DESCRIPTION

This section describes connecting the Orbit's OpenVPN client to an external enterprise level OpenVPN server. This will allow the local network to be part of the VPN set up by the enterprise OpenVPN server. This will also allow the systems on the enterprise network to be access the resources connected to the Orbit's OpenVPN client.

A use case for this section would be as following:

- Having an enterprise level SCADA system manage the RTUs and PLUs connected to multiple Orbit OpenVPN clients. In this case there are two redundant enterprise level OpenVPN servers for load management and resiliency.

The network layout for this is shown in the figure below.

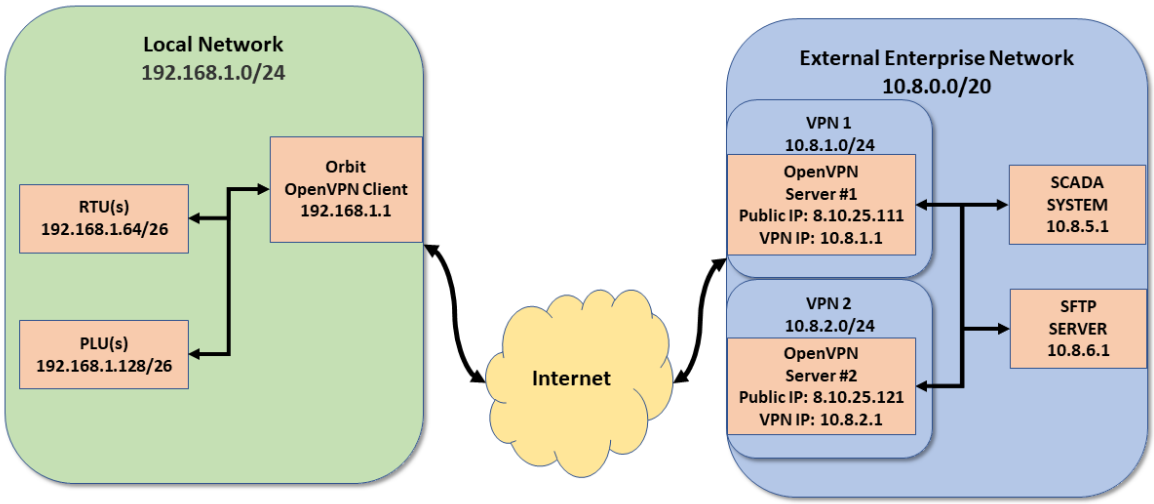

Figure 3 Connecting an Orbit OpenVPN client to an external OpenVPN server

CONFIGURATION

The enterprise OpenVPN servers are configured as such:

- Protocol: UDP
- Port: 1194
- Encryption Algorithm: AES256
- MAC Algorithm: SHA256
- Compression: None

To configure the Orbit on the local network you will need access to the CLI of that Orbit.

The first step would be to create a TUNTAP interface on the Orbit for the OpenVPN client to use.

```
> configure
% set interfaces interface TUN_CLIENT_1 type tuntap
% set interfaces interface TUN_CLIENT_1 tuntap-config mode ip-over-tuntap
% set interfaces interface TUN_CLIENT_1 filter input IN_TRUSTED
% set interfaces interface TUN_CLIENT_1 filter output OUT_TRUSTED
% set interfaces interface TUN_CLIENT_1 enabled true
% commit
% show interfaces interface TUN_CLIENT_1 | details
```

```
type    tuntap;
enabled true;
tuntap-config {
    mode ip-over-tuntap;
}
filter {
    input  IN_TRUSTED;
    output OUT_TRUSTED;
}
```

The above configuration creates a TUNTAP interface named TUN_CLIENT_1 that uses the layer 3 TUN virtual network device that has been configured and enabled for use by the OpenVPN client on the Orbit.

The next step would be to create and configure an OpenVPN client on the Orbit that will connect to the enterprise OpenVPN servers specified in this section.

```
> configure
% set services openvpn client TUN_CLIENT_1
% set services openvpn client TUN_CLIENT_1 server 8.10.25.111 1194 protocol udp
% set services openvpn client TUN_CLIENT_1 server 8.10.25.121 1194 protocol udp
% set services openvpn client TUN_CLIENT_1 server-random-selection true
% set services openvpn client TUN_CLIENT_1 auth-type pub-key
% set services openvpn client TUN_CLIENT_1 pki cert-type rsa
% set services openvpn client TUN_CLIENT_1 pki ca-cert-id ovpn_ca_crt
% set services openvpn client TUN_CLIENT_1 pki cert-id ovpn_client_1_crt
% set services openvpn client TUN_CLIENT_1 pki key-id ovpn_client_1_key
% set services openvpn client TUN_CLIENT_1 ciphersuite encryption-algo aes256-cbc
% set services openvpn client TUN_CLIENT_1 ciphersuite mac-algo sha256-hmac
% set services openvpn client TUN_CLIENT_1 compression disable
% set services openvpn client TUN_CLIENT_1 enabled true
% commit
% show services openvpn client TUN_CLIENT_1 | details
enabled                 true;
server 8.10.25.111 1194 {
    protocol udp;
}
server 8.10.25.121 1194 {
    protocol udp;
}
server-random-selection true;
auth-type               pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_client_1_crt;
    key-id     ovpn_client_1_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key         3600;
ciphersuite {
    encryption-algo aes256-cbc;
    mac-algo        sha256-hmac;
}
compression             disable;
% exit
```

NOTE: The above configuration assumes that the files [ovpn_ca_crt, ovpn_client_1_crt, ovpn_client_1_key] needed by OpenVPN's PKI (public key infrastructure) have already been generated and uploaded onto the Orbit's using the Certificate Management system.

NOTE: The above configuration randomly selects one of the OpenVPN servers.

NOTE: For the above configuration to work as intended the OpenVPN servers will need prior knowledge of the OpenVPN clients that are going to connect to it and the networks that will be behind those clients. The OpenVPN client's certificate common name is used by the OpenVPN server to determine what routes need to be established when a client connects to it.

To verify the status of the OpenVPN client on the Orbit the following commands can be used.

```
> show services openvpn client-state
services openvpn client-state TUN_CLIENT_1
 last-updated            "2021-11-08 12:02:12"
```

```
last-state-message      "CONNECTED SUCCESS 10.8.1.2 8.10.25.111 1194 192.168.1.1 34346"
device-bytes-received   0
device-bytes-sent       0
protocol-bytes-received 7412
protocol-bytes-sent     8919
auth-bytes-received     624
```

The above shows that the local Orbit's OpenVPN client has been connected to the Orbit OpenVPN server using the IP 8.10.25.111 on port 1194. It also shows that the Orbit's OpenVPN client has been assigned a VPN IP address of 10.8.1.2. For this use case the OpenVPN server would have reserved the VPN IP for this client.

CONFIGURING AN ORBIT OPENVPN SERVER FOR ORBIT OPENVPN CLIENTS

DESCRIPTION

This section describes setting up the Orbit's OpenVPN server instance to allow for connections from up to two other Orbit OpenVPN clients.

A use case for this section would be as following:

- Setting up an ad-hoc network between three Orbit devices with individual subnets and resources being accessible from the server device.

The network layout for this is shown in the figure below.

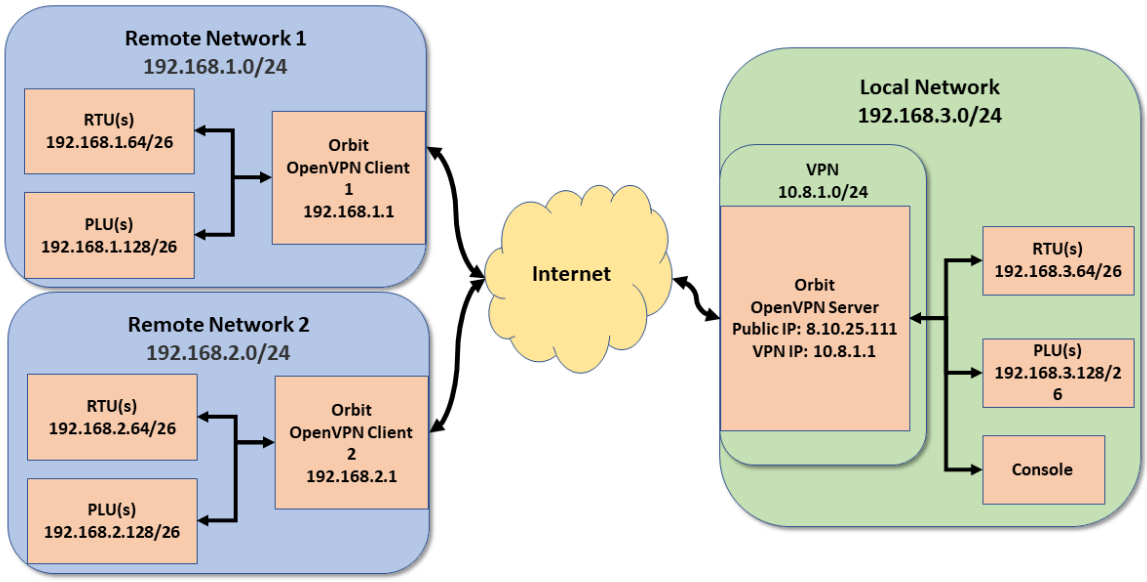

Figure 4 Configuring an Orbit OpenVPN server for Orbit OpenVPN clients

CONFIGURATION

The OpenVPN parameters for the devices are listed below:

- Protocol: TCP
- Encryption Algorithm: AES256
- MAC Algorithm: SHA256
- Compression: LZ4

To configure the Orbit on the local network you will need access to the CLI of that Orbit.

The first step would be to create a TUNTAP interface on the Orbit for the OpenVPN server to use.

```
> configure
% set interfaces interface TUN_SERVER type tuntap
% set interfaces interface TUN_SERVER tuntap-config mode ip-over-tuntap
% set interfaces interface TUN_SERVER filter input IN_TRUSTED
% set interfaces interface TUN_SERVER filter output OUT_TRUSTED
% set interfaces interface TUN_SERVER enabled true
```

```
% commit
% show interfaces interface TUN_SERVER | details
type    tuntap;
enabled true;
tuntap-config {
    mode ip-over-tuntap;
}
filter {
    input  IN_TRUSTED;
    output OUT_TRUSTED;
}
```

The above configuration creates a TUNTAP interface named TUN_SERVER that uses the layer 3 TUN virtual network device that has been configured and enabled for use by the OpenVPN server on the Orbit.

The next step would be to create and configure an OpenVPN server on the Orbit as describe by the network topology in this section.

```
> configure
% set services openvpn server TUN_SERVER_1
% set services openvpn server TUN_SERVER_1 protocol tcp
% set services openvpn server TUN_SERVER_1 port 1194
% set services openvpn server TUN_SERVER_1 tunnel-ip-subnet 10.8.1.0/24
% set services openvpn server TUN_SERVER_1 auth-type pub-key
% set services openvpn server TUN_SERVER_1 pki cert-type rsa
% set services openvpn server TUN_SERVER_1 pki ca-cert-id ovpn_ca_crt
% set services openvpn server TUN_SERVER_1 pki cert-id ovpn_server_1_crt
% set services openvpn server TUN_SERVER_1 pki key-id ovpn_server_1_key
% set services openvpn server TUN_SERVER_1 ciphersuite encryption-algo aes256-cbc
% set services openvpn server TUN_SERVER_1 ciphersuite mac-algo sha256-hmac
% set services openvpn server TUN_SERVER_1 compression lz4
% set services openvpn server TUN_SERVER_1 local-ip-subnets 192.168.3.64/26
% set services openvpn server TUN_SERVER_1 local-ip-subnets 192.168.3.128/26
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_1
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_1 tunnel-ip 10.8.1.11
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_1 ip-subnets 192.168.1.64/26
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_1 ip-subnets
192.168.1.128/26
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_2
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_2 tunnel-ip 10.8.1.22
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_2 ip-subnets 192.168.2.64/26
% set services openvpn server TUN_SERVER_1 client MDS_CLIENT_2 ip-subnets
192.168.2.128/26
% set services openvpn server TUN_SERVER_1 enabled true
% commit
% show services openvpn server TUN_SERVER_1 | details
enabled          true;
protocol         tcp;
port             1194;
tunnel-ip-subnet 10.8.1.0/24;
local-ip-subnets [ 192.168.3.64/26 192.168.3.128/26 ];
client MDS_CLIENT_1 {
    tunnel-ip  10.8.1.11;
    ip-subnets [ 192.168.1.64/26 192.168.1.128/26 ];
}
client MDS_CLIENT_2 {
    tunnel-ip  10.8.1.22;
    ip-subnets [ 192.168.2.64/26 192.168.2.128/26 ];
}
auth-type        pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_server_1_crt;
    key-id     ovpn_server_1_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key  3600;
ciphersuite {
    encryption-algo aes256-cbc;
    mac-algo        sha256-hmac;
}
compression      lz4;
```

```
% exit
```

NOTE: The above configuration assumes that the files [ovpn_ca_crt, ovpn_server_1_crt, ovpn_server_1_key] needed by OpenVPN's PKI (public key infrastructure) have already been generated and uploaded onto the Orbit's using the Certificate Management system.

NOTE: At present client to client direct communications via OpenVPN is not supported on the OpenVPN feature for the Orbits

To configure the Orbit on remote networks 1 and 2 you will need access to the CLI of that Orbit for the next steps.

The next step would be to create and configure the OpenVPN client on the Orbit on the remote network 1. Again, the first step would be to create a TUNTAP interface on the Orbit for the OpenVPN client to use.

```
> configure
% set interfaces interface TUN_CLIENT_1 type tuntap
% set interfaces interface TUN_CLIENT_1 tuntap-config mode ip-over-tuntap
% set interfaces interface TUN_CLIENT_1 filter input IN_TRUSTED
% set interfaces interface TUN_CLIENT_1 filter output OUT_TRUSTED
% set interfaces interface TUN_CLIENT_1 enabled true
% commit
% show interfaces interface TUN_CLIENT_1 | details
type    tuntap;
enabled true;
tuntap-config {
    mode ip-over-tuntap;
}
filter {
    input  IN_TRUSTED;
    output OUT_TRUSTED;
}
```

The above configuration creates a TUNTAP interface named TUN_CLIENT_1 that uses the layer 3 TUN virtual network device that has been configured and enabled for use by the OpenVPN client on the Orbit.

The next step would be to create and configure an OpenVPN client on the Orbit on network 1.

```
> configure
% set services openvpn client TUN_CLIENT_1
% set services openvpn client TUN_CLIENT_1 server 8.10.25.111 1194 protocol tcp
% set services openvpn client TUN_CLIENT_1 auth-type pub-key
% set services openvpn client TUN_CLIENT_1 pki cert-type rsa
% set services openvpn client TUN_CLIENT_1 pki ca-cert-id ovpn_ca_crt
% set services openvpn client TUN_CLIENT_1 pki cert-id ovpn_client_1_crt
% set services openvpn client TUN_CLIENT_1 pki key-id ovpn_client_1_key
% set services openvpn client TUN_CLIENT_1 ciphersuite encryption-algo aes256-cbc
% set services openvpn client TUN_CLIENT_1 ciphersuite mac-algo sha256-hmac
% set services openvpn client TUN_CLIENT_1 compression lz4
% set services openvpn client TUN_CLIENT_1 enabled true
% commit
% show services openvpn client TUN_CLIENT_1 | details
enabled                 true;
server 8.10.25.111 1194 {
    protocol tcp;
}
server-random-selection false;
auth-type               pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_client_1_crt;
    key-id     ovpn_client_1_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key         3600;
ciphersuite {
    encryption-algo aes256-cbc;
    mac-algo        sha256-hmac;
}
compression             lz4;
% exit
```

NOTE: The above configuration assumes that the files [ovpn_ca_crt, ovpn_client_1_crt, ovpn_client_1_key] needed by OpenVPN's PKI (public key infrastructure) have already been generated and uploaded onto the Orbit's using the Certificate Management system.

Like the above process, we now create and configure the OpenVPN client on the Orbit on the remote network 2. Again, the first step would be to create a TUNTAP interface on the Orbit for the OpenVPN client to use.

```
> configure
% set interfaces interface TUN_CLIENT_2 type tuntap
% set interfaces interface TUN_CLIENT_2 tuntap-config mode ip-over-tuntap
% set interfaces interface TUN_CLIENT_2 filter input IN_TRUSTED
% set interfaces interface TUN_CLIENT_2 filter output OUT_TRUSTED
% set interfaces interface TUN_CLIENT_2 enabled true
% commit
% show interfaces interface TUN_CLIENT_2 | details
type    tuntap;
enabled true;
tuntap-config {
    mode ip-over-tuntap;
}
filter {
    input  IN_TRUSTED;
    output OUT_TRUSTED;
}
```

The above configuration creates a TUNTAP interface named TUN_CLIENT_2 that uses the layer 3 TUN virtual network device that has been configured and enabled for use by the OpenVPN client on the Orbit.

The next step would be to create and configure an OpenVPN client on the Orbit on network 2.

```
> configure
% set services openvpn client TUN_CLIENT_2
% set services openvpn client TUN_CLIENT_2 server 8.10.25.111 1194 protocol tcp
% set services openvpn client TUN_CLIENT_2 auth-type pub-key
% set services openvpn client TUN_CLIENT_2 pki cert-type rsa
% set services openvpn client TUN_CLIENT_2 pki ca-cert-id ovpn_ca_crt
% set services openvpn client TUN_CLIENT_2 pki cert-id ovpn_client_2_crt
% set services openvpn client TUN_CLIENT_2 pki key-id ovpn_client_2_key
% set services openvpn client TUN_CLIENT_2 ciphersuite encryption-algo aes256-cbc
% set services openvpn client TUN_CLIENT_2 ciphersuite mac-algo sha256-hmac
% set services openvpn client TUN_CLIENT_2 compression lz4
% set services openvpn client TUN_CLIENT_2 enabled true
% commit
% show services openvpn client TUN_CLIENT_2 | details
enabled                 true;
server 8.10.25.111 1194 {
    protocol tcp;
}
server-random-selection false;
auth-type               pub-key;
pki {
    cert-type  rsa;
    cert-id    ovpn_client_2_crt;
    key-id     ovpn_client_2_key;
    ca-cert-id ovpn_ca_crt;
}
renegotiate-key         3600;
ciphersuite {
    encryption-algo aes256-cbc;
    mac-algo        sha256-hmac;
}
compression             lz4;
% exit
```

NOTE: The above configuration assumes that the files [ovpn_ca_crt, ovpn_client_2_crt, ovpn_client_2_key] needed by OpenVPN's PKI (public key infrastructure) have already been generated and uploaded onto the Orbit's using the Certificate Management system.

To verify the status of the OpenVPN server on the Orbit the following commands can be used.

```
> show services openvpn server-state
services openvpn server-state TUN_SERVER_1
 last-updated       "2021-11-04 18:03:47"
 last-state-message "CONNECTED SUCCESS 10.8.1.1"
 build-info         "OpenVPN 2.5.2 arm-cortex_a9-linux-gnueabi [SSL (OpenSSL)] [LZO]
[LZ4] [EPOLL] [MH/PKTINFO] [AEAD] built on Jun 16 2021"
```

The above shows that Orbit OpenVPN server has been established on the VPN address of 10.8.1.1 as well as the version of the OpenVPN that the server is using.

To verify that the OpenVPN clients have connected to the OpenVPN server the following command can be used on the server Orbit device.

```
> show services openvpn server-clients-state
services openvpn server-clients-state TUN_SERVER_1 MDS_CLIENT_1
 client-id       3
 real-address    192.168.1.1:43778
 virtual-address 10.8.1.11
 bytes-received  4598
 bytes-sent      4467
 connected-since "2021-11-04 16:03:22"
services openvpn server-clients-state TUN_SERVER_1 MDS_CLIENT_2
 client-id       2
 real-address    192.168.2.1:43568
 virtual-address 10.8.1.22
 bytes-received  30460
 bytes-sent      64708
 connected-since "2021-11-04 16:38:29"
```

The above shows that OpenVPN clients with the common names MDS_CLIENT_1 and MDS_CLIENT_2 have connected to the server. The common name is obtained from the certificate that the clients use. The clients have also been assigned their designated VPN IP addresses as configured from the Orbit OpenVPN server.

To verify that the correct routes have been established on the Orbit OpenVPN server the following command can be used.

```
> show routing-state routes outgoing-interface TUN_SERVER_1
                             OUTGOING
DEST PREFIX        NEXT HOP  INTERFACE     SOURCE
-------------------------------------------------
10.8.1.0/24        -         TUN_SERVER_1  kernel
192.168.1.64/26    10.8.1.2  TUN_SERVER_1  kernel
192.168.1.128/26   10.8.1.2  TUN_SERVER_1  kernel
192.168.2.64/26    10.8.1.2  TUN_SERVER_1  kernel
192.168.2.128/26   10.8.1.2  TUN_SERVER_1  kernel
```

The above shows the routing information for the TUN virtual network device (named TUN_SERVER_1) on the Orbit OpenVPN server.

To verify that the route to the external networks as configured on the Orbit OpenVPN server works the following commands can be used.

```
> traceroute 192.168.1.66
 1?: [LOCALHOST]                                      pmtu 1500
 1:  10.8.1.11                                          0.783ms
 2:  192.168.1.66                                       2.745ms reached
> traceroute 192.168.2.128
 1?: [LOCALHOST]                                      pmtu 1500
 1:  10.8.1.22                                          0.692ms
 2:  192.168.2.128                                      1.952ms reached
```

In both the cases the correct hop is used to get to the desired subnet from the server.