EVERTRUST

# Horizon Client v1.5

## *Horizon Client Guide*

EVERTRUST

# Table of Contents


1. Introduction . . . 1
   1.1. Description . . . 1
   1.2. Scope . . . 1
2. General Configuration and Usage . . . 2
3. Discovery Operations . . . 4
   3.1. Local Scan . . . 4
   3.2. Network Scan . . . 4
   3.3. nmap import . . . 5
   3.4. Qualys Certificate View import . . . 5
4. Import Operations . . . 6
   4.1. Local Import . . . 6
   4.2. Network Import . . . 6
5. EST Certificate Lifecycle Operations . . . 8
   5.1. EST Enrollment . . . 8
   5.2. EST Renewal . . . 9
   5.3. Revocation . . . 10
   5.4. Data parameters . . . 10
   5.5. Key and Certificate parameters . . . 10
   5.6. Discovery parameter . . . 11
   5.7. Script parameter . . . 11
   5.8. Examples . . . 12
6. Bulk operations . . . 14
   6.1. Bulk update . . . 14
   6.2. Bulk migrate . . . 14
   6.3. Bulk revoke . . . 15
7. Automation . . . 16
   7.1. Microsoft IIS . . . 16
   7.2. Apache . . . 16
   7.3. Nginx . . . 16

# 1. Introduction

## 1.1. Description

Horizon Client is the client software associated to EverTrust Horizon. This client is developed in Golang, and compiled for the following platforms:

- Linux for x86-64 and arm64 processors
- Windows for x86-64 processors
- Darwin for x86-64 and arm64 processors
- AIX for ppc64 processors

This document is specific to Horizon Client version **1.5**, which may be used with EverTrust Horizon 2.2.0 or later.

## 1.2. Scope

This document is a guide describing how to use the Horizon Client to perform the following tasks:

- Certificate discovery
- Certificate import
- Certificate lifecycle management

# 2. General Configuration and Usage

General parameters of Horizon Client are configured through a file placed in one of the following locations:

- `/opt/horizon/etc/horizon-cli.conf`
- `/usr/local/etc/horizon-cli.conf`
- `[C|D]:\ProgramData\EverTrust\Horizon\horizon-cli.conf`

The configuration file is in JSON format and contains the following:

```
{
  "api_id": "API-ID",
  "api_key": "API-Key",
  "endpoint": "endpoint url. e.g. https://horizon-test.evertrust.fr",
  "debug": true,
  "timeout": 2,
  "proxy": "proxy. e.g. http://myproxy.corp.local:3128",
  "root_ca": "Root CA PEM Certificate(s)."
}
```

These parameters may be instead specified or overridden using environment variables, as detailed in the table below.

*Table 1. General configuration parameters*

| Parameter | Environment variable | Description |
|---|---|---|
| `api_id` | `APIID` | The API ID: the identifier of a local account user defined in Horizon. Used for discovery, import modes and for the revocation in the EST module |
| `api_key` | `APIKEY` | The API Key. Used together with API ID |
| `endpoint` | `ENDPOINT` | The URL of the Horizon instance, starting with `http` or `https` and without trailing "/" |
| `debug` | `DEBUG` | Set to true to enable debug mode of the Horizon Client, defaults to false if unspecified. |
| `timeout` | | Connection timeout in seconds, defaults to 2 seconds if unspecified. |

| Parameter | Environment variable | Description |
|---|---|---|
| `proxy` | `https_proxy` | HTTPS proxy used to reach Horizon (if any), in URL form which can contain login and password if needed. |
| `root_ca` | | PEM chain of CA certificates that issued the TLS certificate exposed by Horizon. This parameter is optional, as preferred way is to put these CA certificates in the machine trust store. |

Finally, Horizon Client is designed to output informative messages on STDOUT and logs on STDERR. Thus, a typical command line to launch Horizon Client will look the following:

```
horizon-cli <command> <parameters> 2>horizon-cli.log
```

**TIP**

You can use the "--help" parameter to get online help on any command or sub-command.

```
horizon-cli <command> <subcommand> --help
```

# 3. Discovery Operations

These operations aim at feeding Horizon with certificates discovered on the network through different means. These certificates will be fed along with appropriate Discovery metadata, such as IP address or Hostname of the machine on which the discovered certificate is held.

## 3.1. Local Scan

In local scan mode, the Horizon Client will scan the machine it is installed on for certificates, and reports them to Horizon. Certificates are discovered if they match following conditions:

- They are saved in PEM or DER format in a file that is pointed in a configuration file
- They are contained in a Machine or User "MY" certificate store (Windows Only)
- They are not CA certificates

In local scan mode, Horizon client should be launched with root or administrator rights, or it will probably fail to discover all certificates.

**TIP**

```
horizon-cli localscan --campaign=test
```

## 3.2. Network Scan

In network scan mode, the Horizon Client will first connect to Horizon to get the campaign's scanning parameters (Hosts and Ports), then perform the network scanning and feed Horizon with the scan results.

The following algorithm is used for network scanning:

1. If `--ping-first` flag is given, perform ICMP ping on the defined hosts and discard hosts that are not reachable
2. Scan the hosts and ports for an open TCP port
3. If TCP port is opened:
    - If port is not '25', try a TLS handshake. If handshake succeeds, retrieve the certificate and report it to Horizon
    - If port is '25', perform SMTP STARTTLS, retrieve the certificate and report it to Horizon

The "timeout" global configuration variable has an impact on both open ports discovery and TLS handshake. In case you get unexpected handshake errors or EOF, try to increase the timeout. However, this will also make the network scan perform slower.

**TIP**

```
horizon-cli netscan --campaign=test
```

## 3.3. nmap import

In nmap import mode, the discovery itself is performed by nmap, using the `ssl-cert` plugin. Horizon Client then has the ability to import the nmap scanning results into Horizon using the nmap import mode.

To be able to do so, nmap needs to be launched with the `-oX` option, in order to export its scan result as XML file. This XML file is then passed on to Horizon Client.

**TIP**

```
horizon-cli importscan nmap --campaign=test --xmlfile=nmapresults.xml
```

## 3.4. Qualys Certificate View import

In Qualys Certificate View (CV) import mode, the discovery itself is performed by Qualys CV. Horizon Client then has the ability to import the Qualys CV scanning results into Horizon using the `qualyscv` import mode.

To be able to do so, a technical account must have been created into Qualys CV for Horizon Client, with appropriate rights to be able to view the scanning results. You need also to identify your Qualys CV API Gateway URL using the following link.

**TIP**

```
horizon-cli importscan qualyscv --campaign=test --endpoint
=https://gateway.qg1.apps.qualys.eu --username=testlogin --password
=testpassword
```

# 4. Import Operations

Import operations are designed to import certificate into Horizon without any metadata. This is useful mainly when installing Horizon, e.g. to import all certificates from an existing PKI database.

## 4.1. Local Import

In order to be able to import certificates, you need to put them as PEM files in a folder, and launch Horizon Client by pointing at that folder. Horizon Client will recurse on the folder, find all PEM files, and import certificates into Horizon. It is advised to use sub-folders to store certificates, so that you avoid to hit any file-per-folder file system limit.

**TIP**

```
horizon-cli localimport --campaign=test --path=/path/to/certs --source
=MyADCS
```

If you wish to import certificates along with their private keys (e.g. when importing from a PKI escrow), you need to put them as PKCS#12 files in a folder, and launch Horizon Client by pointing at that folder. Horizon Client will recurse on the folder, find all PEM files, and import certificates into Horizon. It is advised to use sub-folders to store certificates, so that you avoid to hit any file-per-folder file system limit. All the PKCS#12 files must be encrypted using the same password that will be passed to Horizon Client using the command line.

**TIP**

```
horizon-cli localimport --campaign=test --path=/path/to/certs --source
=MyADCS --pfx-password=<PKCS#12 password>
```

## 4.2. Network Import

### 4.2.1. DigiCert CertCentral

You can import all your valid certificates from DigiCert CertCentral. Please note that only certificates in "issued" state can be imported. Certificates that are revoked will not be imported.

**TIP**

```
horizon-cli netimport digicert --campaign=test --digicert-api-key=<api-
key>
```

### 4.2.2. AWS ACM

You can import all your valid certificates from AWS ACM. Please note that only certificates in "issued" state can be imported. Certificates that are revoked will not be imported.

**TIP**

```
horizon-cli netimport aws-acm --campaign=test --aws-region=<aws-region>
```

```
--access-key-id=<aws-access-key-id> --secret-access-key=<aws-secret-
access-key>
```

**NOTE** AWS Role Assumption is supported. You need to provide the ARN of the role you wish to assume using the `--assume-role-arn` option.

### 4.2.3. Azure Key Vault

You can import all your valid certificates from Azure Key Vault. Please note that only certificates in "issued" state can be imported. Certificates that are in pending state will not be imported.

**TIP**

```
horizon-cli netimport akv --campaign=test --vault-name=<vault short name>
--azure-tenant=<tenant name> --client-id=<client app Id> --client-secret
=<client app secret>
```

### 4.2.4. F5 BIG-IP

You can import all your valid certificates from F5 BIG-IP.

# 5. EST Certificate Lifecycle Operations

## 5.1. EST Enrollment

Horizon Client is able to use the EST module of Horizon to enroll certificates. The following enrollment modes are supported:

- Static and authorized user/password in decentralized mode
- Static and authorized user/password in centralized mode
- Challenge password in decentralized mode
- Challenge password in centralized mode
- Certificate swap in decentralized mode
- Certificate swap in centralized mode

### 5.1.1. Static and authorized user

In this enrollment mode, a local user account is created in Horizon for Horizon Client, and the EST profile on Horizon is configured in "authorized" mode thus a static username and password can be provided to Horizon Client for enrollment. They need to be set in general configuration as `APIID` and `APIKEY`.

**TIP**

```
horizon-cli est --enroll=static --profile=test --cn=TestCN [data
parameters] [key and certificate parameters]
```

### 5.1.2. Challenge password

In this enrollment mode, the EST profile on Horizon is set to "challenge" mode. A request must then be made on Horizon in order to retrieve the one-time password "challenge" to be used to authenticate the EST request. No `APIID` nor `APIKEY` need to be set.

**TIP**

```
horizon-cli est --enroll=<challenge> --profile=test --cn=TestCN [data
parameters] [key and certificate parameters]
```

### 5.1.3. Certificate swap

In this enrollment mode, the EST profile on Horizon is set to "x509" mode. The client is then able to make a request to Horizon by authenticating with an existing certificate. This certificate can be specified either:

- by using the `--key` and `--cert` parameters, respectively pointing at the key and the certificate to be used to authenticate

- by using the `--win-store-auth` parameter (Windows only), that will look into the "MY" certificate store (user by default, unless `--win-machine-store` is specified) for a non-expired certificate whose CN matches the Common Name specified in `--cn` parameter

**TIP**

```
horizon-cli est --enroll=cert --profile=test --key=/path/to/key --cert
=/path/to/cert --cn=TestCN [data parameters] [key and certificate
parameters]

horizon-cli est --enroll=cert --profile=test ---win-store-auth --cn=TestCN
[data parameters] [key and certificate parameters]
```

### 5.1.4. Decentralized mode

In decentralized mode, which is the default mode, Horizon Client generates a private key and a CSR. The CSR is generated according to Data parameters, and the private key and the retrieved certificate are then stored according to the Key and Certificate parameters

### 5.1.5. Centralized mode

In Centralized mode, triggered by adding the “--centralized” parameter to the command line, Horizon Client generates a fake private key and a CSR. The CSR is generated according to Data parameters. The private key generated by Horizon Client is discarded. A random password is generated and inserted into the CSR. If the enrollment is successful, Horizon generates a private key and a certificate and sends them back to Horizon Client as PKCS#12, which Horizon Client decodes using the randomly generated password. The retrieved private key and the retrieved certificate are then stored according to the Key and Certificate parameters

**IMPORTANT** The random password generated has 16 characters, letters and numbers. If a password policy is enforced on Horizon side for the centralized mode in the considered EST profile, ensure that it is compatible with such characteristics.

## 5.2. EST Renewal

The certificate renewal is performed by using the “--renewal” parameter. It works the same way as the Certificate swap mode, except that:

- it is designed to renew a certificate already issued by Horizon on the same profile
- it can be scheduled as a periodic task (cron or Scheduled Task), that will perform the renewal only when the certificate is N days before its expiration. N can be specified using the “--renewal-interval” parameter, and defaults to 30.

**TIP**

```
horizon-cli est --renew --profile=test --key=/path/to/key --cert
=/path/to/cert [key and certificate parameters]

horizon-cli est --renew --profile=test ---win-store-auth --cn=TestCN [key
```

```
and certificate parameters]
```

## 5.3. Revocation

The certificate revocation is performed using the "--revoke" parameter, and used the `APIID` and `APIKEY` provided in the general configuration to authenticate to Horizon. This means the user identified by the `APIID` parameter must have revocation rights on the certificate to revoke.

**TIP**

```
horizon-cli est --revoke --cert=/path/to/cert
```

## 5.4. Data parameters

These parameters are used to be inserted in the Certificate Signing Request sent to Horizon.

*Table 2. CSR data parameters*

| Parameter | Description |
|---|---|
| `--cn` | Requested subject Common Name, single value, and mandatory. |
| `--ou` | Requested subject `OU`. Can contain multiple values. Optional. |
| `--dnsnames` | Requested subject alternative name DNS entries. Can contain multiple values. Optional. |
| `--emails` | Requested subject alternative name RFC822Name entries. Can contain multiple values. Optional. |

## 5.5. Key and Certificate parameters

These parameters define how to store the retrieved certificate and its associated private key. The following alternatives are available:

- Key and certificate stored separately in two files, in PEM format. This is typically used to be used by Apache or NGINX web servers;
- Key and certificate stored together in a PKCS#12 file. This is typically used by Tomcat application server;
- Key and certificate stored together in Windows certificate store. This is typically used by IIS web server.

*Table 3. Platform-independent key and certificate parameters*

| Parameter | Description |
|---|---|
| `--key` | Path to the private key to store |

| Parameter | Description |
|---|---|
| `--cert` | Path to the certificate to store |
| `--pfx` | Path to the PKCS#12 file storing the certificate and its key |
| `--pfxpwd` | Password used to encrypt the PKCS#12 file specified in the `--pfx` parameter |
| `--export-jks` | Path to export the certificate and private key as JKS. Optional, must be used along pfx/pfxpwd options as pfxpwd option is used to protect the JKS |
| `--jks-alias` | Alias of the private key entry within the JKS. Optional, only used with `--export-jks` |
| `--jks-key-password` | Password of the private key entry within the JKS. Optional, defaults to pfxpdw, only used with `--export-jks` |

*Table 4. Windows-only key and certificate parameters*

| Parameter | Description |
|---|---|
| `--win-store-save` | Triggers the ability to store the certificate and private key into the "MY" certificate store |
| `--win-machine-store` | Triggers the ability to store in the Machine store. If not specified, the User store is used. This parameter requires Horizon Client to be executed with appropriate elevated rights. |
| `--win-use-tpm` | Triggers the ability to store the certificate in the `Microsoft Platform Crypto Provider` KSP. If not specified, the `Microsoft Software Key Storage Provider` KSP will be used. |

## 5.6. Discovery parameter

You can chain a certificate enrollment operation with the discovery operation by using the `--discovery` parameter. It will use the `APIID` and `APIKey` defined in the general configuration to authenticate to Horizon and feed it with the enrolled certificate and its discovery metadata.

The `--discovery` parameter takes a value, which is the name of the Discovery campaign to use to report the certificate to Horizon.

## 5.7. Script parameter

You can tell Horizon Client to launch a script upon successful certificate enrollment or renewal by using the `--script` parameter, which takes the path to the script as an argument.

The script will receive arguments passed by Horizon Client in the following order:

1. Issued certificate serial number
2. Issued certificate fingerprint (SHA-1 hash of the certificate in DER format)
3. Issued certificate Subject DN
4. Issued certificate Issuer DN

Below is an example of a very simple bash script:

```
#!/bin/sh

echo $1
echo $2
echo $3
echo $4
```

Below is an example of a very simple PowerShell script:

```
param($serial, $fingerprint, $subject, $issuer)

Write-Output $serial
Write-Output $fingerprint
Write-Output $subject
Write-Output $issuer
```

## 5.8. Examples

You will find below a few examples detailing how to use the client for EST enrollment in various context

### 5.8.1. Decentralized enrollment with challenge, output as key and certificate

```
horizon-cli est --enroll=<challenge> --profile=<profile> --key=/path/to/key --cert
=/path/to/cert --cn=test.example.com --dnsnames=test.example.com,www.test.example.com
```

### 5.8.2. Decentralized enrollment with challenge, output as PKCS#12

```
horizon-cli est --enroll=<challenge> --profile=<profile> --key=/path/to/key --cert
=/path/to/cert --cn=test.example.com --dnsnames=test.example.com,www.test.example.com
--pfx=/path/to/p12 --pfxpwd=<pkcs12_password>
```

NOTE

You need to specify `--key` and `--cert` options, because you will need key and cert separately later on to trigger renewal

### 5.8.3. Centralized enrollment with challenge, output as key and certificate

```
horizon-cli est --enroll=<challenge> --profile=<profile> --key=/path/to/key --cert
=/path/to/cert --cn=test.example.com --dnsnames=test.example.com,www.test.example.com
--centralized
```

### 5.8.4. Centralized enrollment with challenge, output as PKCS#12

```
horizon-cli est --enroll=<challenge> --profile=<profile> --key=/path/to/key --cert
=/path/to/cert --cn=test.example.com --dnsnames=test.example.com,www.test.example.com
--pfx=/path/to/p12 --pfxpwd=<pkcs12_password> --centralized
```

NOTE

You need to specify `--key` and `--cert` options, because you will need key and cert separately later on to trigger renewal

### 5.8.5. Decentralized enrollment with challenge, output in machine windows store

```
horizon-cli est --enroll=<challenge> --profile=<profile> --cn=test.example.com
--dnsnames=test.example.com,www.test.example.com --win-store-save --win-machine-store
```

### 5.8.6. Decentralized renewal, output as key and certificate

```
horizon-cli est --renew --profile=<profile> --key=/path/to/key --cert=/path/to/cert
```

### 5.8.7. Decentralized renewal using machine windows store

```
horizon-cli est --renew --profile=<profile> --cn=test.example.com --win-store-auth
--win-store-save --win-machine-store
```

# 6. Bulk operations

The horizon client allow to perform bulk operations on certificates using the *Horizon Certificate Query Language* (HCQL).

## 6.1. Bulk update

the `bulk update` command allow update of certificate metadata en masse. The command takes a HCQL query as parameter and updates the matching certificates with the provided metadata. You can update the **owner**, the **team** and the **labels** of the certificates.

*Table 5. Bulk update command parameters*

| Parameter | Description |
|---|---|
| `--query` | The HCQL query string. Update will be performed on results. |
| `--confirm` | Skip confirm. |
| `--owner` | The owner to set on certificates matching the query. (Optional) |
| `--team` | The team to set on certificates matching the query. (Optional) |
| `--label` | The label, in the forme key:value, to set on certificates matching the query. (Optional) |

**TIP**

```
horizon-cli bulk update --query 'module equals "est" and status is valid'
--owner "myuser" --team "myteam" --label "mylabel:myvalue"
```

## 6.2. Bulk migrate

The `bulk migrate` command allow certificate migration from one profile to another. The command takes an HCQL query as parameter and migrate the matching certificates to the provided profile. The command can also update the certificates metadata.

*Table 6. Bulk migrate command parameters*

| Parameter | Description |
|---|---|
| `--query` | The HCQL query string. Update will be performed on results. |
| `--confirm` | Skip confirm. |
| `--profile` | The target profile for the migration. |
| `--owner` | The owner to set on certificates matching the query. (Optional) |

| Parameter | Description |
| --- | --- |
| `--team` | The team to set on certificates matching the query. (Optional) |
| `--label` | The label, in the forme key:value, to set on certificates matching the query. (Optional) |

**TIP**

```
horizon-cli bulk migrate --query 'module equals "est" and status is valid'
--profile new-est-profile --team myteam --label mylabel:myvalue
```

## 6.3. Bulk revoke

The `bulk revoke` command allow certificate revocation en masse. The command takes an HCQL query as parameter and revoke the matching certificates.

*Table 7. Bulk revoke command parameters*

| Parameter | Description |
| --- | --- |
| `--query` | The HCQL query string. Update will be performed on results. |
| `--confirm` | Skip confirm. |

**TIP**

```
horizon-cli bulk revoke --query 'team equals "myterminatedteam" and status
is valid' --confirm
```

# 7. Automation

The Horizon Client can help you automate the installation of your TLS certificates. Helper commands are available for **Microsoft IIS**, **Apache** and **Nginx**. Note that the two latter commands only work on Linux.

## 7.1. Microsoft IIS

The automate command for Microsoft IIS will enroll or renew a certificate, store it in the *Windows Machine Store* using the *software backend*, and bind it to the specified host and port.

**TIP**

```
horizon-cli automate iis --enroll=<challenge> --profile=myestprofile
--dnsnames=www.example.com --bind-host www.example.com:443
```

## 7.2. Apache

The automate command for Apache works exactly the same as the est client, except that it will restart your Apache server after the certificate has been installed.

**NOTE** To restart your server, the client will try to use the `service` command to restart the service named `httpd` if the client is run on a RHEL machine or `apache2` if any other distibution is used.

**TIP**

```
horizon-cli automate apache --enroll=<challenge> --profile=myestprofile
--dnsnames=www.example.com --cert /etc/ssl/certs/apache.pem --key
/etc/ssl/private/apache.key
```

## 7.3. Nginx

The automate command for Nginx works exactly the same as the est client, except that it will restart your Nginx server after the certificate has been installed.

**NOTE** To restart your server, the client will try to use the `service` command to restart the service named `nginx`.

**TIP**

```
horizon-cli automate nginx --enroll=<challenge> --profile=myestprofile
--dnsnames=www.example.com --cert /etc/ssl/certs/nginx.pem --key
/etc/ssl/private/nginx.key
```