EVERTRUST

Horizon Client v1.7

Horizon Client Guide

EVERTRUST

Table of Contents

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

1.2. System requirements

To run Horizon Client the underlying system must comply with the following minimum requirements :

For certificate lifecycle purposes

- 1 gigahertz (GHz) or faster with 1 or more cores
- 1 gigabyte (GB) of RAM for Linux environments
- 2 gigabyte (GB) of RAM for Microsoft environments
- 10 gigabyte (GB) or larger storage device

For discovery purposes

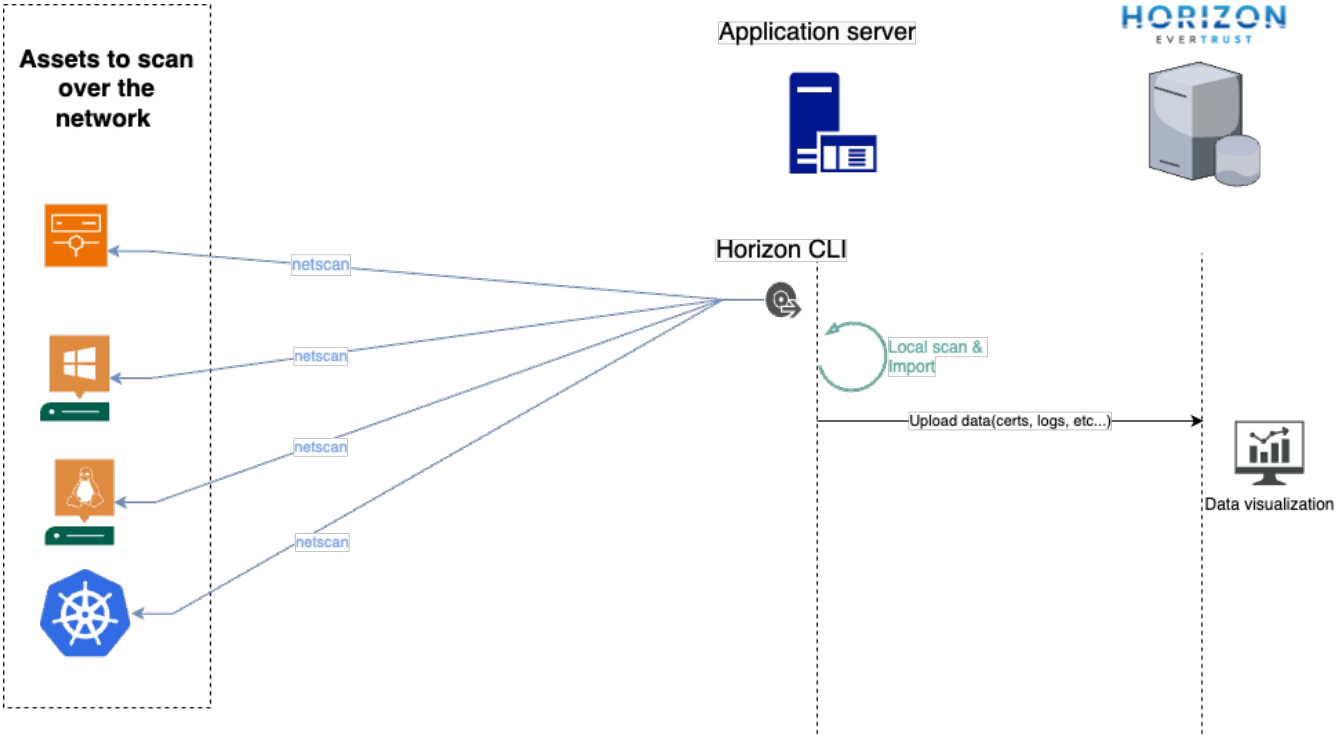
- 2 gigahertz (GHz) or faster with 2 or more cores
- 2 gigabyte (GB) of RAM for Linux environments
- 4 gigabyte (GB) of RAM for Microsoft environments
- 20 gigabyte (GB) or larger storage device

This document is specific to Horizon Client version **1.7**, which may be used with EverTrust Horizon 2.4.0 or later.

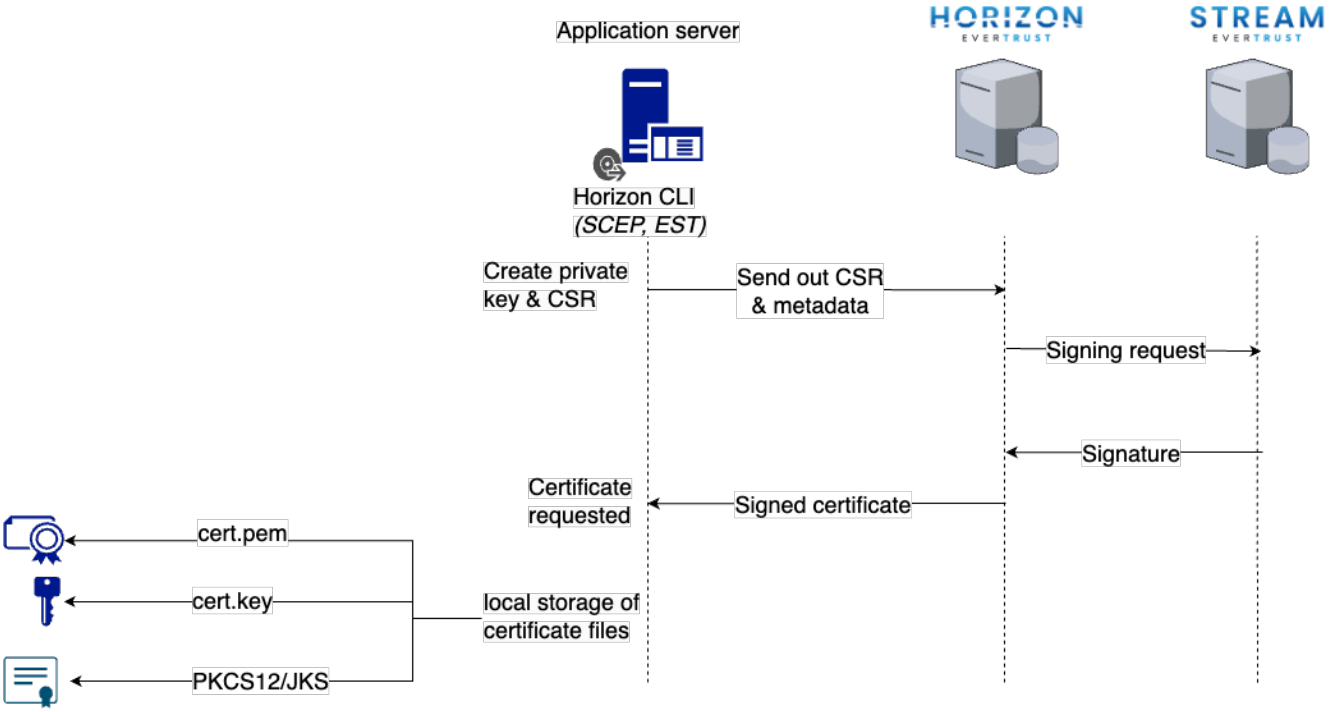
1.3. Scope

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

- Certificate discovery & import



- Certificate lifecycle management



2. General Configuration and Usage

2.1. Installations

2.1.1. Package install/uninstall

Using RPMs file

- Installing the package

```
# yum install horizon-cli-<version>-1.x86_64.rpm
```

- Uninstalling the package

```
# yum remove horizon-cli
```

Using MSI file:

To install the package, double click on the MSI file and follow the instructions. To uninstall the package, simply browse to the Applications & program menu and uninstall the program.

Using binary file:

The linux binary file is usable on any linux distribution, to install it follow the steps below :

- Add the binary file to the "PATH", in order to easily launch it on your shell.
- Apply the executable permission on the binary file

```
# chmod +x horizon-cli.bin
```

2.1.2. Command line installation & initialization

Use the command below to install the client and generate interactively your configuration file:

```
# horizon-cli install
```

The configuration file can also be created using command line parameters:

```
# horizon-cli install --endpoint https://horizon-test.com
```

Use the help to get the full list of available parameters.

NOTE	If you did not use an installer, this command should always be run first to ensure everything is set up correctly.

2.2. Configuration content

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": false,
  "timeout": 2,
  "proxy": "proxy. e.g. http://myproxy.corp.local:3128",
  "root_ca": "Root CA PEM Certificate(s).",
  "log_file": "The log file of Horizon."
}
```

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.

Parameter	Environment variable	Description
`timeout`		Connection timeout in seconds, defaults to 2 seconds if unspecified.
`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.
`log_file`	`LOGFILE`	Log file of horizon. This parameter is optional, but a default value is set as the Horizon Client displays useful messages on STDOUT and logs should always be kept.

TIP

You can use the "--help" parameter to get command line 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
```

In order to perform network scans on a recurring schedule, the Horizon Client offers the possibility

to create periodic tasks to run a scan.

The three supported options for the `period` parameter are:

- daily - runs the task everyday between 0-4 AM UTC
- weekly - runs the task every Sunday between 0-4 AM UTC
- monthly - runs the task on the first day of the month between 0-4 AM UTC

TIP

```
# horizon-cli netscan --campaign=test --create-periodic-task --period
=monthly
```

The created task can then be removed using:

TIP

```
# horizon-cli netscan --campaign=test --remove-periodic-task
```

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=<pkcs12_password>
```

You can also import certificates from a csv file. Certificates must be in a column named "certificate". As of now, three formats are supported:

1. DERBase64: Certificate in DER (binary) Base 64 encoded (default);
2. DERHex: Certificate in DER (binary) Hex String encoded;
3. PEM: Certificate in PEM (with or without the certificate header and footer).

TIP

```
# horizon-cli localimport --campaign=test --csv /path/to/csv/file.csv
--csv-separator ";"
```

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.

TIP

```
# horizon-cli netimport bigip --campaign=test --hostname=<F5 BigIp hostname> --login=<F5 BigIp login> --password=<F5 BigIp password>
```

It is also possible to import the certificates as managed certificates in Horizon. This will allow renewal and removal of the certificate upon revocation using Horizon's triggers mechanism.

In order to activate this behavior, the `connector` property must reference a valid F5 Connector in Horizon.

TIP

```
# horizon-cli netimport bigip --campaign=test --connector=<Horizon F5 Connector name> --hostname=<F5 BigIp hostname> --login=<F5 BigIp login> --password=<F5 BigIp password>
```

CAUTION	In order for the trigger mechanism to work correctly, an Horizon WebRA profile must use the F5 Connector trigger and a schedule task should reference the connector and the WebRA profile.

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 --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=<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 --auth-cert --profile=test --key=/path/to/key
--cert=/path/to/cert --cn=TestCN [data parameters] [key and certificate
parameters]

# horizon-cli est enroll --auth-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 “renew” command. 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 --in-cert=/path/to/cert/to/renew
[key and certificate parameters]

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

```
and certificate parameters]
```

5.3. Data parameters

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

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.

5.4. 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 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
`--cert`	Path to the certificate to store
`--pfx`	Path to the PKCS#12 file storing the certificate and its key
`--pfx-pwd`	Password used to encrypt the PKCS#12 file specified in the `--pfx` parameter

Parameter	Description
`--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-pwd`	Password for the JKS output. Mandatory if `--jks` is set
`--jks-alias`	Alias of the private key entry within the JKS. Optional, only used with `--jks`
`--jks-alias-pwd`	Password of the private key entry within the JKS. Optional, defaults to pfxpwd, only used with `--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.

5.5. 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.

5.6. 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
```

5.7. Examples

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

5.7.1. Decentralized enrollment with challenge, output as key and certificate

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

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

```
# horizon-cli est enroll --challenge=<challenge> --profile=<profile> --cn
=test.example.com --dnsnames=test.example.com,www.test.example.com --pfx
=/path/to/pkcs12 --pfx-pwd=<pkcs12_password>
```

5.7.3. Centralized enrollment with challenge, output as key and certificate

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

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

```
# horizon-cli est enroll --centralized --challenge=<challenge> --profile=<profile>
--cn=test.example.com --dnsnames=test.example.com,www.test.example.com --pfx
=/path/to/pkcs12 --pfx-pwd=<pkcs12_password>
```

5.7.5. Decentralized enrollment with challenge, output in machine windows store

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

5.7.6. Decentralized renewal from certificate and key, output as key and certificate

```
# horizon-cli est renew --profile=<profile> --in-cert=/path/to/old/cert --in-key
=/path/to/old/key --cert=/path/to/new/cert --key=/path/to/new/key
```

5.7.7. Decentralized renewal from PKCS#12, output as key and certificate

```
# horizon-cli est renew --profile=<profile> --in-cert=/path/to/old/pkcs12 --cert
=/path/to/cert --key=/path/to/key
```

5.7.8. 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. ACME Certificate Lifecycle Operations

6.1. ACME Enrollment

The Horizon Client uses the LEGO ACME client under the hood to enroll certificates with Horizon. As per the ACME protocol, the client will need to prove its ownership of the domain name(s) it wants to enroll a certificate for. This is done either by HTTP, TLS-ALPN or DNS challenge. An email address is also needed to create an ACME account with Horizon in order to enroll certificates.

Table 5. ACME Validation parameters

Parameter	Description
`--account`	Email of the account to use.
`--dns-names`	DNS names to enroll a certificate for.
`--http-01-port`	Port to use for http01 challenge. (Optional)
`--tls-alpn-01-port`	Port to use for tlsalpn01 challenge. (Optional)
`--dns-01-provider`	DNS provider name to use for dns01 challenge, from the list at https://go-acme.github.io/lego/dns#dns-providers"

6.1.1. ACME Account

The ACME account is used to store the ACME account key, which is used to sign the ACME requests. All of this is taken care of by the ACME modules of Horizon and Horizon Client.

Given the email from the `--account` CLI parameter, the client will create an ACME account with Horizon, and store it's private key in the `/var/db/acme/accounts/<email>.pem` file relative to the Horizon Client data folder (`/opt/horizon` on unix and `C:\ProgramData\Horizon` on Windows). If such a file already exists, the client will use it instead of creating a new account.

6.1.2. ACME validation

The ACME protocol requires the client to prove its ownership of the domain name(s) it wants to enroll a certificate for. This is done either by HTTP, TLS-ALPN or DNS challenge.The order in which the client will try the validation methods is the following:

1. TLS-ALPN challenge
2. HTTP challenge
3. DNS challenge

The allowed challenge validation modes for a profile need to be configured in Horizon. If the profile does not allow a challenge validation mode, the client will not be able to enroll a certificate for it.

DNS challenge

Note that the DNS challenge requires the client to be able to update the DNS records of the domain name(s) it wants to enroll a certificate for. This is done by providing the proper credentials for the DNS provider specified in the `--dns-01-provider` CLI parameter. Each provider will have different needs, but the credentials will need to be provided as environment variables in every case. Please refer to the LEGO documentation for more details.

6.2. ACME Renewal

The `acme renew` command is designed to work similarly to the `acme enroll` command, but with a few differences:

The following input parameters can be used to specify the certificate to renew:

Table 6. ACME Renewal input certificate parameters

Parameter	Description
`--in-cert`	Path to the Certificate to renew (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries
`--in-pfx-pwd`	Password for the PKCS#12 file to renew
`--in-jks-pwd`	Password for the JKS file to renew
`--in-jks-alias`	Alias for the certificate to renew in the JKS file
`--in-jks-alias-pwd`	Alias password for the JKS file to renew

6.3. ACME Revocation

You can revoke an ACME enrolled certificate using the account that enrolled it. The `acme revoke` command will revoke the certificate specified by the `--cert` parameter with the reason specified by the `--reason` parameter.

The revocation reason must be one of the following:

- `unspecified`
- `keycompromise`
- `affiliationchanged`
- `superseded`
- `cessationofoperation`
- `cacompromise`

6.4. 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 7. Platform-independent key and certificate parameters

Parameter	Description
`--key`	Path to the private key to store
`--cert`	Path to the certificate to store
`--pfx`	Path to the PKCS#12 file storing the certificate and its key
`--pfx-pwd`	Password used to encrypt the PKCS#12 file specified in the `--pfx` parameter
`--jks`	Path to export the certificate and private key as JKS
`--jks-pwd`	Password used to encrypt the JKS file specified in the `--jks` parameter
`--jks-alias`	Alias of the private key entry within the JKS

Table 8. 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

6.5. 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.

6.6. 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
```

6.7. Examples

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

6.7.1. Enrollment with HTTP-01 validation, output as key and certificate

```
# horizon-cli acme enroll --profile=<profile> --account=myemail@example.com --key
=/path/to/key --cert=/path/to/cert --dnsnames=test.example.com,www.test.example.com
--http-01-port=5002
```

6.7.2. Enrollment with Cloudflare DNS-01 validation, output as PKCS#12

```
CLOUDFLARE_EMAIL=myemail@example.com \
CLOUDFLARE_API_KEY=<apikey> \
horizon-cli acme enroll \
  --account=myemail@example.com \
  --profile=<profile> \
  --cn=test.example.com \
  --dnsnames=test.example.com,www.test.example.com \
  --pfx=/path/to/pkcs12 \
  --pfx-pwd=<pkcs12_password> \
  --dns-01-provider=cloudflare
```

6.7.3. Enrollment with TLS-ALPN-01 validation, output in machine windows store

```
# horizon-cli acme enroll --profile=<profile> -account=myemail@example.com --dnsnames
=test.example.com,www.test.example.com --win-store-save --win-machine-store --tls-alpn
-01-port=5001
```

6.7.4. Renewal with HTTP-01 or TLS-ALPN-01 validation, output as key and certificate

```
# horizon-cli acme renew --profile=<profile> --account myemail@example.com --in-cert
/path/to/cert --out-key=/path/to/key --out-cert=/path/to/cert --tls-alpn-01-port=5001
--http-01-port=5002
```

NOTE The Client will use either HTTP-01 or TLS-ALPN-01 validation, depending on the authorized validation modes in the profile configuration. If both are authorized, the Client will try them in this order.

6.7.5. Revoking a certificate using the account that enrolled it

```
# horizon-cli acme revoke --profile=<profile> --account myemail@example.com --in-cert
/path/to/cert
```

7. SCEP Certificate Lifecycle Operations

The Horizon Client includes a SCEP client to perform challenge based pre-validated enrollments and renewals. Its usage is similar to that of the EST client in challenge mode.

Usage:

```
# horizon-cli scep [command] [flags]
```

7.1. SCEP Enrollment

The `enroll` command allows you to perform a SCEP enrollment operation. It will generate a new key pair and a CSR based on the content parameters, and send it to the SCEP server to obtain a certificate.

7.1.1. Validation

Get a SCEP challenge password from the Horizon web app or REST API. You can then provide it with the `--challenge` parameter to the `enroll` command.

7.1.2. Content Parameters

You can customize the contents of the CSR using the following parameters:

Table 9. SCEP enrollment content parameters

Parameter	Description
`--profile`	SCEP Profile to use
`--challenge`	SCEP pre-validated request challenge
`--cert`	Path to the certificate file to be written
`--key`	Path to the key file to be written
`--cn`	Common Name of the certificate to request
`--ou`	Organizational Units of the certificate to request
`--dnsnames`	SAN DNS Names of the certificate to request
`--ip`	SAN IP Addresses of the certificate to request
`--emails`	SAN RFC822 Names of the certificate to request
`--contact-email`	Contact email
`--owner`	Certificate owner
`--team`	Certificate owning team
`--labels`	Labels, in the form key:value

Parameter	Description
`--ca-chain`	Path to write the CA Chain to be written, as a PEM Bundle
`--pfx`	Path to write the PKCS#12 output to be written
`--pfx-pwd`	Password for the PKCS#12 output
`--key-type`	Type and size of the key to generate (defaults to `rsa-2048`)
`--script`	Bash or powershell script to execute upon enrollment completion
`--discovery`	Discovery profile to use in order to report the certificate to Horizon after enrollment
`--jks`	Path to write the JKS output to be written
`--jks-pwd`	Password for the JKS output
`--jks-alias`	Alias for the JKS output
`--jks-alias-pwd`	Password for the alias in the JKS output
`--overwrite`	Overwrite existing files

7.2. SCEP Renewal

The `renew` command is designed to work similarly to the `enroll` command, but with a few differences:

- It will enroll a certificate based on the `--in-cert` parameter (or similar, see below) instead of the content parameters. Only the `--key-type` parameter is used to generate a new key pair.
- No challenge is needed for a SCEP renewal operation

The following input parameters can be used to specify the certificate to renew:

Table 10. SCEP input certificate parameters

Parameter	Description
`--profile`	SCEP Profile to use
`--key-type`	Key type
`--contact-email`	Contact email
`--owner`	Certificate owner
`--team`	Certificate owning team
`--labels`	Labels, in the form key:value
`--discovery`	Discovery profile to use in order to report the certificate to Horizon after renewal

Parameter	Description
`--in-cert`	Path to the Certificate to renew (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries
`--in-pfx-pwd`	Password for the PKCS#12 file to renew
`--in-jks-pwd`	Password for the JKS file to renew
`--in-jks-alias`	Alias for the certificate to renew in the JKS file
`--in-jks-alias-pwd`	Alias password for the JKS file to renew
`--cert`	Path to the Certificate to save as a PEM file
`--key`	Path to the Key to save as a PEM file
`--ca-chain`	Path to the Chain to save as a PEM file
`--pfx`	Path to write the PKCS#12 output
`--pfx-pwd`	Password for the PKCS#12 output. Mandatory if `--pfx` is set
`--jks`	Path to write the JKS output
`--jks-pwd`	Password for the JKS output. Mandatory if `--jks` is set
`--jks-alias`	Alias for the JKS output. Mandatory if `--jks` is set
`--jks-alias-pwd`	Password for the alias in the JKS output. Mandatory if `--jks` is set
`--script`	Execute bash or powershell script upon enrollment or renewal completion
`--renewal-interval`	Number of days before expiration to trigger the renewal. Optional, renewal modeonly

7.3. Output Parameters

Choose how the created certificate and its associated private key are stored. 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 or JKS 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 11. Platform-independent output 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
`--pfx-pwd`	Password used to encrypt the PKCS#12 file specified in the `--pfx` parameter
`--jks`	Path to export the certificate and private key as JKS. Optional
`--jks-pwd`	Password used to encrypt the JKS file specified in the `--jks` parameter
`--jks-alias`	Alias of the private key entry within the JKS
`--jks-key-pwd`	Password of the private key entry within the JKS

Table 12. Windows-only output 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.

7.4. Metadata parameters

Each certificate enrolled via horizon **must** have a profile, specified with the `--profile` flag. You can add extra metadata according to your needs using the following parameters:

Table 13. SCEP metadata parameters

Parameter	Description
`--owner`	Owner of the certificate
`--contact-email`	Contact email of the certificate owner
`--team`	Team owning the certificate
`--labels`	Labels to attach to the certificate, in the form `key:value`

7.5. 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.

7.6. 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
```

7.7. Examples

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

7.7.1. Enrollment with output as key and certificate

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

7.7.2. Enrollment with lots of metadata and output as PKCS#12

```
# horizon-cli scep enroll \
  --profile=<profile> \
  --challenge=<challenge> \
  --key-type=rsa-2048 \
  --cn=test.example.com \
  --dnsnames=test.example.com,www.test.example.com \
  --owner="John Doe" \
  --ou="IT" \
  --team="IT" \
  --labels="env:prod" \
  --pfx=/path/to/pkcs12 \
  --pfx-pwd=<pkcs12_password>
```

7.7.3. Renewal with output as key and certificate

```
# horizon-cli scep renew --profile=<profile> --in-cert /path/to/cert --cert
=/path/to/cert --key=/path/to/key
```

8. WebRA Certificate Lifecycle Operations

The Horizon Client can perform post-validated lifecycle operations using the WebRA protocol. This includes certificate enrollment, renewal and revocation.

8.1. WebRA operations validation

Like SCEP and EST, WebRA operations requires the intervention of a third party to validate the request. Unlike SCEP and EST though, it is a post-validation protocol, meaning that no challenge is produced before the operation, instead a request is created and sent to the WebRA server, which will need to be validated or cancelled by an operator with the appropriate rights on the web app.

This means the Horizon Client performs enrollment and renewal operations in two steps:

1. Create the request
2. Once the request is validated, retrieve the certificate

Depending on the time it takes for the request to be validated, the Horizon Client can be configured to either enter a blocking loop and wait for the request to be validated, or merely create the request and exit.

If the latter is chosen, the Horizon Client will keep in its internal database the pending request, and will check for its validation each time the `horizon-cli automate routine` command is executed. If the request is validated, the certificate will be retrieved and stored in the appropriate location. If it is denied, the request will be removed from the database. In some cases you would want to configure a crontab or scheduled task to perform this check periodically. You can use the command `horizon-cli automate create-periodic-task <period>` to help you in the process, or create it manually.

By default, the behavior is to create the request and exit. If you wish for the client to enter a blocking loop until the request is validated, specify the `--now` flag.

8.2. WebRA Enrollment

8.2.1. Content Parameters

You can customize the contents of the CSR using the following parameters:

Table 14. WebRA enrollment content parameters

Parameter	Description
`--profile`	WebRA Profile to use
`--discovery`	Discovery profile to use in order to report the certificate to Horizon after enrollment
`--contact-email`	Contact email
`--cn`	Common Name of the certificate to request
`--ou`	Organizational Units of the certificate to request

Parameter	Description
`--dnsnames`	SAN DNS Names of the certificate to request
`--ip`	SAN IP Addresses of the certificate to request
`--emails`	SAN RFC822 Names of the certificate to request
`--ms-guid`	SAN MS GUID of the certificate to request
`--ms-sid`	SAN MS SID of the certificate to request
`--key-type`	Select a key type for your enrollment other than the default one of the profile
`--owner`	Certificate owner
`--team`	Certificate owning team
`--labels`	Labels, in the form key:value
`--cert`	path to the cert file to be created
`--key`	path to the key file to be created
`--ca-chain`	Path to write the CA Chain to, as a PEM Bundle
`--pfx`	Path to write the PKCS#12 output to be created
`--pfx-pwd`	Password for the PKCS#12 output, mandatory if pfx is set
`--jks`	Path to write the JKS output to be created
`--jks-pwd`	Password for the JKS output, mandatory if `--jks` is set
`--jks-alias`	Alias for the JKS output, mandatory if `--jks` is set
`--jks-alias-pwd`	Password for the alias in the JKS output, mandatory if `--jks` is set
`--overwrite`	Overwrite existing files
`--win-store-save`	[Windows Only] Triggers the use of Windows certificate store to save the certificate after enrollment
`--win-machine-store`	[Windows Only] Triggers the use of local machine store on Windows
`--win-use-tpm`	[Windows Only] Triggers the use of the Microsoft Platform Crypto Provider for certificate store save. Used only with win-store-save.
`--script`	Execute bash or powershell script upon enrollment or renewal completion
`--now`	If user has only "request enroll" permission, start a blocking loop until review, instead of waiting for the next routine execution

8.3. WebRA Renewal

The `webra renew` command is designed to work similarly to the `webra enroll` command, except that it will enroll a certificate based on the `--in-cert` parameter (or similar, see below) instead of the content parameters.

The following input parameters can be used to specify the certificate to renew:

Table 15. WebRA input certificate parameters

Parameter	Description
`--discovery`	Discovery profile to use in order to report the certificate to Horizon after enrollment
`--in-cert`	Path to the Certificate to renew (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries
`--in-key`	Path to the private key of the certificate to renew if it is not included in the certificate file
`--in-pfx-pwd`	Password for the PKCS#12 file to renew
`--in-jks-pwd`	Password for the JKS file to renew
`--in-jks-alias`	Alias for the certificate to renew in the JKS file
`--in-jks-alias-pwd`	Alias password for the JKS file to renew
`--key-type`	The key type to use for the renewed certificate
`--cert`	Path to the cert file to be created
`--key`	Path to the key file to be created
`--ca-chain`	Path to write the CA Chain to, as a PEM Bundle
`--overwrite`	Overwrite existing files
`--pfx`	Path to write the PKCS#12 output to be created
`--pfx-pwd`	Password for the PKCS#12 output. Mandatory if pfx is set
`--jks`	Path to write the JKS output to be created
`--jks-pwd`	Password for the JKS output, mandatory if `--jks` is set
`--jks-alias`	Alias for the JKS output, mandatory if `--jks` is set
`--jks-alias-pwd`	Password for the alias in the JKS output, mandatory if `--jks` is set
`--overwrite`	Overwrite existing files
`--win-store-save`	[Windows Only] Triggers the use of Windows certificate store to save the certificate after enrollment

Parameter	Description
`--win-machine-store`	[Windows Only] Triggers the use of local machine store on Windows
`--win-use-tpm`	[Windows Only] Triggers the use of the Microsoft Platform Crypto Provider for certificate store save. Used only with win-store-save.
`--script`	Execute bash or powershell script upon enrollment or renewal completion
`--now`	If user has only "request enroll" permission, start a blocking loop until review, instead of waiting for the next routine execution
`--renewal-interval`	Number of days before expiration to trigger the renewal. Optional, renewal mode only

8.4. WebRA Revocation

The `webra revoke` command takes the following parameters:

Table 16. WebRA revocation parameters

Parameter	Description
`--discovery`	Discovery profile to use in order to report the certificate to Horizon after enrollment
`--cert`	Path to the Certificate to revoke (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries
`--key`	Path to the private key of the certificate to revoke if it is not included in the certificate file
`--pfx-pwd`	Password for the PKCS#12 file to revoke
`--jks-pwd`	Password for the JKS file to revoke
`--jks-alias`	Alias for the certificate to revoke in the JKS file
`--jks-alias-pwd`	Alias password for the JKS file to revoke
`--reason`	Reason for revocation (unspecified, keycompromise, cacompromise, affiliationchanged, superseded, cessationofoperation)

For a revocation operation, the `--now` flag is unavailable, and the `automate routine` command will not track the revocation status, as no actions are to be performed after the revocation is complete. This command thus merely creates the revocation request and exits.

8.5. Output Parameters

Choose how the created certificate and its associated private key are stored. 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 or JKS 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 17. Platform-independent output parameters

Parameter	Description
`--cert`	Path to the certificate to store
`--key`	Path to the private key to store
`--pfx`	Path to the PKCS#12 file storing the certificate and its key
`--pfx-pwd`	Password used to encrypt the PKCS#12 file specified in the `--pfx` parameter
`--jks`	Path to export the certificate and private key as JKS. Optional
`--jks-pwd`	Password used to encrypt the JKS file specified in the `--jks` parameter
`--jks-alias`	Alias of the private key entry within the JKS
`--jks-key-pwd`	Password of the private key entry within the JKS

Table 18. Windows-only output 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.

8.6. Metadata parameters

Each certificate enrolled via horizon **must** have a profile, specified with the `--profile` flag. You can add extra metadata according to your needs using the following parameters:

Table 19. SCEP metadata parameters

Parameter	Description
`--owner`	Owner of the certificate
`--contact-email`	Contact email of the certificate owner
`--team`	Team owning the certificate
`--labels`	Labels to attach to the certificate, in the form `key:value`

8.7. 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.

8.8. 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
```

8.9. Examples

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

8.9.1. Enrollment with output as key and certificate, waiting for the certificate to be issued

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

8.9.2. Enrollment with lots of metadata, output as PKCS#12 and no blocking loop

```
# horizon-cli webra enroll \
  --profile=<profile> \
  --cn=test.example.com \
  --dnsnames=test.example.com,www.test.example.com \
  --owner="John Doe" \
  --ou="IT" \
  --team="IT" \
  --labels="env:prod" \
  --pfx=/path/to/pkcs12 \
  --pfx-pwd=<pkcs12_password>
```

TIP

after this command, run periodically:

```
# horizon-cli automate routine > /path/to/my/logfile
```

8.9.3. Renewal with output as key and certificate, waiting for the certificate to be issued

```
# horizon-cli webra renew --in-cert /path/to/old/cert --in-key /path/to/old/key --cert
```

```
/path/to/cert --key /path/to/key --now
```

8.9.4. Revocation of a certificate

```
# horizon-cli webra revoke --in-cert /path/to/cert --in-key /path/to/key
```

9. Update operations

The horizon client can perform update operations on certificates using the `update-cert` command. This will modify the information associated with the certificate on Horizon.

CAUTION	The `update-cert` command need the 'Update (pop)' common configuration permission enabled on the profile the certificate is linked to.

9.1. General Parameters

`--confirm`	The command asks for confirmation after the changes are computed. Use this flag to disable this behavior and proceed directly. (Optional)
`--prompt`	Use this flag to be prompted for edition of all the certificate fields. In this mode, using enter on an existing value means the value is not changed. (Optional)

9.2. Input parameters

The update is only possible on local certificates for which you possess the key:

`--cert`	Path to the Certificate to update (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries.
`--key`	Path to the private key of the certificate to update if it is not included in the certificate file. (Optional)
`--pfx-pwd`	Password for the PKCS#12 file to update. (Optional)
`--jks-pwd`	Password for the JKS file to update. (Optional)
`--jks-alias`	Alias for the JKS file to update. (Optional)
`--jks-alias-pwd`	Alias password for the JKS file to update. (Optional)

9.3. Update Parameters

An update concerns only metadata fields, that is fields added by Horizon.

`--owner`	Set the owner of the certificate. An empty string means deletion of this information. (Optional)
`--team`	Set the team of the certificate. An empty string means deletion of this information. (Optional)

`--contact-email`	Set the contact email of the certificate. An empty string means deletion of this information. (Optional)
`--labels`	Set the labels of the certificate. An empty string means deletion of this information. (Optional)
`--metadata`	Set the technical metadata of the certificate. To use with caution. An empty string means deletion of this information. (Optional)

9.4. Examples

You will find below a few examples detailing how to use the client to update certificates in various context

9.4.1. Updating the owner of a certificate

```
# horizon-cli update-cert --cert=/path/to/cert --key=/path/to/key --owner=newowner
```

9.4.2. Removing the team from a certificate stored in JKS file

```
# horizon-cli update-cert --cert=/path/to/cert.jks --jks-pwd=<jks_password> --team=""
```

9.4.3. Updating labels and metadata of a certificate stored in windows certificate store

```
# horizon-cli update-cert --cert=<certificate_thumbprint> --labels
="label1:value1,label2:value2" --metadata="metadata1:value1,metadata2:value2"
```

10. Bulk operations

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

10.1. Bulk update

The `bulk update` command allows 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**, the **labels** and the **contact email** of the certificates. To unset an existing value use the value *"unset"*.

Table 20. 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)
`--labels`	The labels, in the comma separated key:value form, to set on certificates matching the query. (Optional)
`--contact-email`	The contact email to set on certificates matching the query. (Optional)
`--metadata`	The metadata 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" --labels "mylabel:myvalue"
--contact-email "unset"
```

10.2. Bulk migrate

The `bulk migrate` command allows 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. You can update the **owner**, the **team**, the **labels** and the **contact email** of the certificates. To unset an existing value use the value *"unset"*.

Table 21. 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)
`--team`	The team to set on certificates matching the query. (Optional)
`--labels`	The labels, in the comma separated key:value form, to set on certificates matching the query. (Optional)
`--contact-email`	The contact email to set on certificates matching the query. (Optional)
`--metadata`	The metadata 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 --labels mylabel:myvalue
```

10.3. Bulk revoke

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

Table 22. Bulk revoke command parameters

Parameter	Description
`--query`	The HCQL query string. Update will be performed on results.
`--confirm`	Skip confirm.
`--reason`	Reason for revocation. Default is "unspecified"

TIP

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

11. Automation

The Horizon Client can help you automate the installation of your TLS certificates. This page documents both the **legacy automation module**, as it was in the 1.5.x series, and the **new automation module**, which is available in the 1.6.x series.

11.1. Legacy Automation

NOTE The legacy automation is now deprecated. Please use the New Automation instead

11.1.1. 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
```

11.1.2. 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
```

11.1.3. 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
```

11.2. New Automation

The `automate enroll` command is the entry point for the new automation module. The difference between the legacy and new automation module resides in the discovery of certificates on your machine, and on the automatic renewal.

11.2.1. Automation policies

The new automation module relies on *automation policies* to know **how** the certificates should be enrolled and renewed. Automation policies are configurable in the Horizon web app, contain an EST, SCEP or ACME profile and other options, like preferred enrollment CA and hash algorithm. For each automation operation, an automation policy needs to be specified using the `--automation-policy` parameter.

NOTE EverTrust recommends using EST for most use cases of server automation.

CAUTION SCEP automation is not yet supported on windows servers.

11.2.2. Discovery

The new automation module will discover certificates on your machine by parsing your webserver's or TLS service's configuration files. Supported services are:

- *apache*
- *nginx*
- *tomcat*
- *jboss wildfly*
- *lighttpd*
- *microsoft iis*
- *winhorizon*
- *horizon adcsconnector*

By default, the `automate enroll` command will try to search for all supported services. If you want to limit the search to a specific list of services, you can use the `--target` option.

TIP

```
# horizon-cli automate enroll --target=apache,nginx [...]
```

NOTE The discovery of config files relies on the "usual" locations for the services configuration folder. If you have an "unusual" configuration folder for your service, you can use the `--config-folder` option to specify it. Note that it only works if you

restrict the discovery to a single service.

You can use the `--analyze-only` flag to only perform the discovery phase, print results and exit.

11.2.3. Enrollment

Once the discovery phase is done, you will be presented with a list of certificates that can be enrolled. You can then select the certificates you want to enroll, and enrollments will be performed according to the automation policy and it's corresponding profile.

For example, if your automation policy specifies an EST profile with challenge password validation, the client will enroll the certificate using the EST protocol, and you will be prompted for the challenge password.

The enrollment is meant to be somewhat intelligent, in that it does not enroll certificates that are already known by Horizon and compliant to your automation policy. For example, if you have initially enrolled your web server's certificate using the EST, SCEP or ACME client on the same profile, and you run the `automate enroll` command, the client will not try to enroll the certificate again, but will simply update its internal database with the new information. If you want to re-enroll the certificates anyway, you can use the `--force-enroll` option.

Additional enrollment parameters

You can chain a certificate enrollment operation with the discovery operation by using the `--discovery-campaign` parameter to specify your discovery campaign. 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. Upon enrollment, you can add additional metadata to the certificate using the following parameters:

Table 23. Metadata parameters

Parameter	Description
`--owner`	Owner of the certificate
`--contact-email`	Contact email of the certificate owner
`--team`	Team owning the certificate
`--labels`	Labels to attach to the certificate, in the form `key:value`

Table 24. Protocol specific parameters

Parameter	Description
`--acme-account`	Email of the acme account to use for enrollment

11.2.4. Installation

After a successful enrollment or renewal, the certificate will be installed on your machine. The installation process will depend on the services that use the certificate, and the storage backend

that was used to store the certificate. For example, if the certificate was used by *Tomcat* and was stored as a PKCS#12 file, then this file will be overwritten, with the same password. The old PKCS#12 file will be backed up by the client.

The impacted services will be restarted automatically after each certificate enrollment or renewal.

NOTE	If you wish for the client to **not install** your new certificate, that is, not replace the old certificate and not restart the impacted services, you can use the `--no-install` option. Each new file (cert, key, CA chain, keystore…) will then be placed in the same folder as it's predecessor, with the `.new` extension.

11.2.5. Renewal

Each time a certificate is discovered and enrolled by the automation module of the Horizon Client, its details are stored in the internal database for future reference. Each time the `automate routine` command is run, the client will check if any of the locally known certificates need to be renewed. Reasons for renewal can be:

- The certificate is about to expire
- The certificate has been revoked
- Preferences such as key type or enrollment CA were changed in the profile or automation policy

If a certificate needs to be renewed, the client will perform the renewal according to the automation policy, and its corresponding profile.

TIP	We recommend that you run the `automate routine` command periodically as a cron job or scheduled task. You can use the command `horizon-cli automate create-periodic-task <period>` or the flag `--auto-renew` on the `automate enroll` command to help you in the process, or create it manually.

This mechanism allows for more resilient web servers, as the certificates will be renewed automatically, before any interruption of service can happen because of an expired or revoked certificate. It also helps your organisation migrate your TLS certificates to a new CA quickly, by simply changing the preferred enrollment CA in the automation policy and waiting a few hours for all your instances of the Horizon Client to execute their routine tasks.

11.2.6. Backup

Each time a file is replaced by the Horizon Client, the old file is backed up. The backup files are stored in the `cert/backup/<HASH(BACKUPED FILE PATH)>` directory relative to the Horizon Client data folder (`/opt/horizon` on unix and `C:/ProgramData/EverTrust/Horizon` on Windows), as `filename_n.ext` where `n` is the number of the backup. Thus, the `filename_0.ext` is the original version, before any intervention of the Horizon Client.

11.2.7. Internal Database operations

The internal database is used to store the details of the certificates that are discovered and enrolled

by the automation module. You can list them using the `automate list` command, and delete them using the `automate delete` command. Certificates are indexed by their bindings, which are the combination of all the services along with the hostnames and ports that use the certificate. For example, if you have a certificate that is used by *Apache* for all hosts on the port 443, it's "id" in the local database will be `apache-*:443`. We found this to be a human-readable way of uniquely identifying a local certificate.

TIP

You can choose the output format of the `automate list` command. By default it outputs a string, but you can use the `--json` option to output a JSON object. example:

```
# horizon-cli automate list --json | jq
```

The `automate remove <id1> ⋯ <idn>` command erases certificates from the local database. This command will not remove the certificate files from your machine, only remove it from the "managed certificates" local database. This way, the client will not check its status at each routine execution anymore.

TIP

You can use the `automate remove all` command to remove all certificates from the local database.

The `--restore` option of the `automate remove` command can be used to restore a certificate from a backup file. The backup file to be restored will always be the older one, in most cases the `filename_0.ext`, that is, the original file before any tampering by the Horizon Client. For certificates stored in the windows store, the store thumbprints will be stored in a file corresponding to the server type, like `iisbackups`.

11.2.8. Interactivity

Two options are available to control the interactivity of the `automate enroll` command:

- The `--no-interactive` flag will prevent any prompt from being displayed, and will use the default values or those provided in the command line arguments. It will:
  - select all discovered certificates for enrollment
  - if a challenge is required, use the provided `--challenge` argument. If no challenge is provided, or the given challenge has already been used, the enrollment will fail.
  - not add any additional SANs
- the `--prompt` flag will force the client to prompt the user for any missing information. If specified, any other command line arguments are optional. It will:
  - prompt the user to select which services to search for on the machine (equivalent to the `--target` option)
  - prompt the user for the automation policy (equivalent to the `--automation-policy` option)

11.2.9. Examples

Enroll certificates used by nginx and apache

```
# horizon-cli automate enroll \
  --target=nginx,apache \
  --automation-policy=MyPolicy
```

Use the interactive mode

```
# horizon-cli automate enroll --prompt
```

Check if the previously enrolled certificates need to be renewed

```
# horizon-cli automate routine
```

Get the DN of all the certificates enrolled by the automation module

```
# horizon-cli automate list --json | jq -r '.[] | .certificate | .subject'
```

Remove the certificate used by nginx on the port 443 from the automatically renewed certificates

```
# horizon-cli automate remove nginx-*:443
```

Remove the certificate used by tomcat on the port 8443 from the automatically renewed certificates and restore the original certificate

```
# horizon-cli automate remove --restore tomcat-*:8443
```