



Horizon Client v1.8

Horizon Client Guide

EVERTRUST

Table of Contents

1. Introduction	1
1.1. Description	1
1.2. System requirements	1
1.3. Scope	1
2. General Configuration and Usage	3
2.1. Installations	3
2.2. Configuration content	4
3. Discovery Operations	6
3.1. Local Scan	6
3.2. Network Scan	6
3.3. nmap import	7
3.4. Qualys Certificate View import	7
3.5. Nessus Scan Import	8
4. Import Operations	9
4.1. Local Import	9
4.2. Network Import	10
5. EST Certificate Lifecycle Operations	12
5.1. EST Enrollment	12
5.2. EST Renewal	13
5.3. Data parameters	14
5.4. Key and Certificate parameters	14
5.5. Discovery parameter	15
6. Script parameter	15
6.1. Examples	16
7. ACME Certificate Lifecycle Operations	18
7.1. ACME Enrollment	18
7.2. ACME Renewal	19
7.3. ACME Revocation	19
7.4. Key and Certificate parameters	19
7.5. Discovery parameter	20
8. Script parameter	21
8.1. Examples	21
9. SCEP Certificate Lifecycle Operations	23
9.1. SCEP Enrollment	23
9.2. SCEP Renewal	24
9.3. Output Parameters	25
9.4. Metadata parameters	26
9.5. Discovery parameter	26
10. Script parameter	26

10.1. Examples	27
11. WebRA Certificate Lifecycle Operations	29
11.1. WebRA operations validation	29
11.2. WebRA Enrollment	29
11.3. WebRA Renewal	31
11.4. WebRA Revocation	32
11.5. Output Parameters	33
11.6. Metadata parameters	34
11.7. Discovery parameter	34
12. Script parameter	34
12.1. Examples	35
13. Update operations	37
13.1. General Parameters	37
13.2. Update Parameters	37
13.3. Certificate selection parameters	37
13.4. Examples	38
14. Bulk operations	40
14.1. Bulk update	40
14.2. Bulk migrate	40
14.3. Bulk revoke	41
15. Automation	42
15.1. Subcommands	42
15.2. Supported Services	43
15.3. Generic Service Automation	44
15.4. Automation policies	45
15.5. Existing WebServer certificates	46
15.6. Additional enrollment parameters	46
15.7. Installation	48
15.8. Renewal	48
15.9. Interactivity	49
15.10. Certificate commands	49
15.11. Backup	60
15.12. Internal Database operations	60
15.13. Routine	61
15.14. Management commands	61
15.15. Examples	63

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

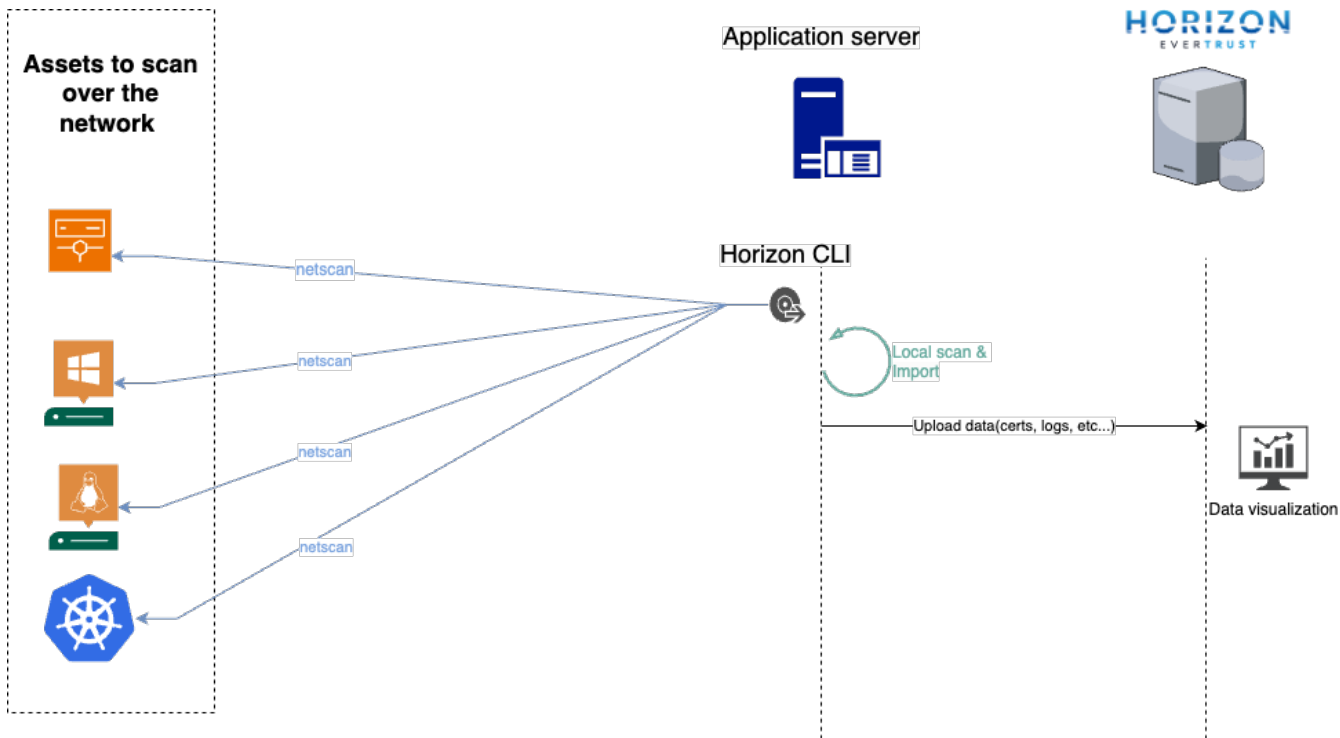
- 2 gigahertz (GHz) or faster with 2 or more cores
- 2 gigabyte (GB) of RAM for Linux environments
- 4 gigabyte (GB) of RAM for Microsoft environments
- 20 gigabyte (GB) or larger storage device

This document is specific to Horizon Client version **1.8**, which may be used with EverTrust Horizon 2.4.0 or later.

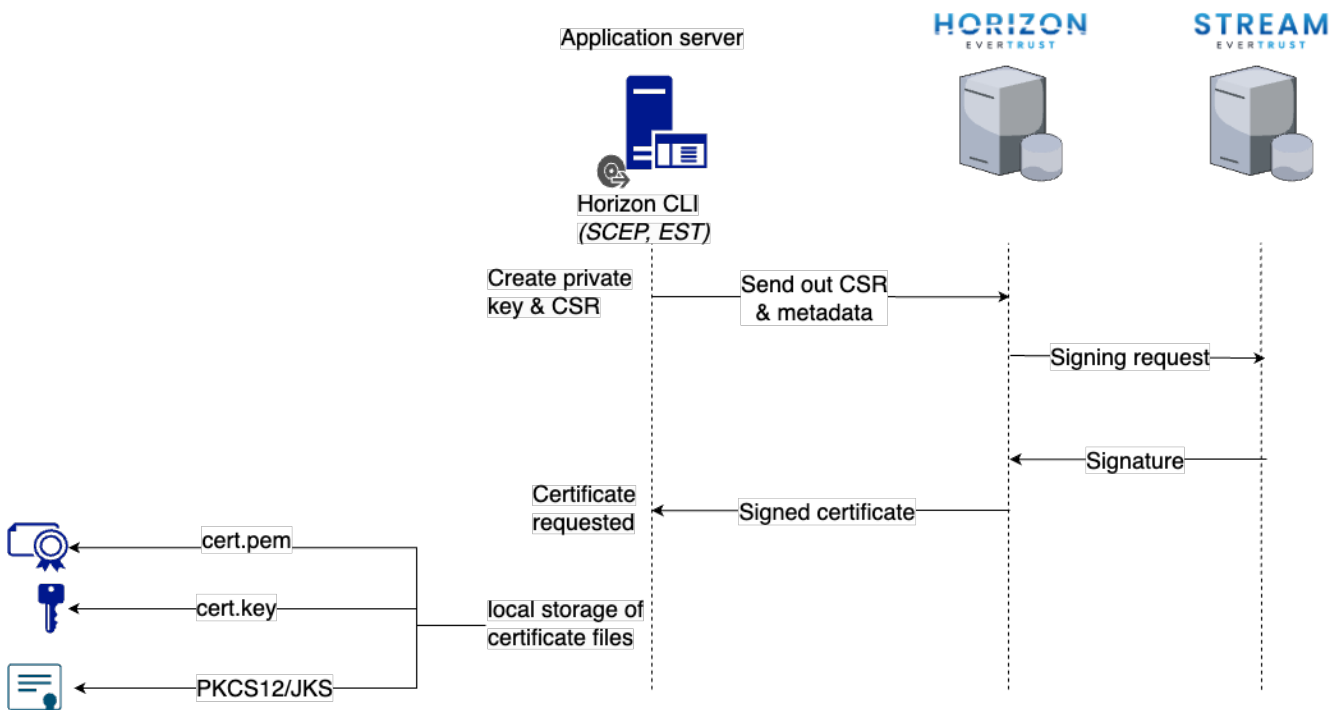
1.3. Scope

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
<code>api_id</code>	<code>APIID</code>	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
<code>api_key</code>	<code>APIKEY</code>	The API Key. Used together with API ID
<code>endpoint</code>	<code>ENDPOINT</code>	The URL of the Horizon instance, starting with <code>http</code> or <code>https</code> and without trailing <code>/</code>
<code>debug</code>	<code>DEBUG</code>	Set to true to enable debug mode of the Horizon Client, defaults to false if unspecified.

Parameter	Environment variable	Description
<code>timeout</code>		Connection timeout in seconds, defaults to 2 seconds if unspecified.
<code>proxy</code>	<code>https_proxy</code>	HTTPS proxy used to reach Horizon (if any), in URL form which can contain login and password if needed.
<code>root_ca</code>		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.
<code>log_file</code>	<code>LOGFILE</code>	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.

You can use the “--help” parameter to get command line help on any command or sub-command.

TIP

```
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
```

This periodic task can be run with a specific user identity on Linux using the `user` parameter.

TIP

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

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
```

3.5. Nessus Scan Import

In Nessus scan import mode, Horizon Client enables the importation of scanning results from Nessus into Horizon. This mode allows for a seamless integration of Nessus vulnerability scans into the Horizon environment.

To utilize this feature, you need to ensure that you have valid credentials for Nessus with the necessary permissions to access and export scan data and the scan id on which you want to perform the import. Additionally, you must know your Nessus URL through which Horizon Client will communicate with the Nessus API and use the "SSL Certificate Information" plugin output to get the certificates into horizon.

TIP

```
horizon-cli importscan nessus --campaign=test --endpoint  
=https://cloud.tenable.com --username=testlogin --password=testpassword  
--scan-id=5
```

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
```

By default, this command does not import CA certificates. To import CA certificates, use the `--enable-ca-import` flag.

TIP

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

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=<pkes12_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 ";"
```

In order to add technical metadata to the imported certificate, the `--csv-metadata` flag can be used to import metadata from a column with the same name. For example, to configure a `pki_connector` on each certificate with a file containing the `pki_connector` column:

TIP

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

Supported metadata are:

- `pki_connector`
- `certeuropa_id`
- `digicert_id`
- `digicert_order_id`
- `entrust_id`
- `fcms_id`
- `gsatlas_id`
- `metapki_id`

TIP

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
```


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
<code>--cn</code>	Requested subject Common Name, single value, and mandatory.
<code>--ou</code>	Requested subject OU . Can contain multiple values. Optional.
<code>--dnsnames</code>	Requested subject alternative name DNS entries. Can contain multiple values. Optional.
<code>--emails</code>	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
<code>--key</code>	Path to the private key to store
<code>--cert</code>	Path to the certificate to store
<code>--pfx</code>	Path to the PKCS#12 file storing the certificate and its key
<code>--pfx-pwd</code>	Password used to encrypt the PKCS#12 file specified in the <code>--pfx</code> parameter

Parameter	Description
<code>--jks</code>	Path to export the certificate and private key as JKS. Optional, must be used along <code>pfx/pfxpwd</code> options as <code>pfxpwd</code> option is used to protect the JKS
<code>--jks-pwd</code>	Password for the JKS output. Mandatory if <code>--jks</code> is set
<code>--jks-alias</code>	Alias of the private key entry within the JKS. Optional, only used with <code>--jks</code>
<code>--jks-alias-pwd</code>	Password of the private key entry within the JKS. Optional, defaults to <code>pfxpwd</code> , only used with <code>--jks</code>

Table 4. Windows-only key and certificate parameters

Parameter	Description
<code>--win-store-save</code>	Triggers the ability to store the certificate and private key into the "MY" certificate store
<code>--win-machine-store</code>	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.
<code>--win-use-tpm</code>	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.

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 - windows store)

thumbprint)

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
```

6.1. Examples

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

6.1.1. 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
```

6.1.2. 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>
```

6.1.3. 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
```

6.1.4. 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>
```

6.1.5. 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
```

6.1.6. 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
```

6.1.7. 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
```

6.1.8. 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
```

7. ACME Certificate Lifecycle Operations

7.1. 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
<code>--account</code>	Email of the account to use.
<code>--dns-names</code>	DNS names to enroll a certificate for.
<code>--http-01-port</code>	Port to use for http01 challenge. (Optional)
<code>--tls-alpn-01-port</code>	Port to use for tlsalpn01 challenge. (Optional)
<code>--dns-01-provider</code>	DNS provider name to use for dns01 challenge, from the LEGO supported list

7.1.1. 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 its 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.

7.1.2. 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.

7.2. 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
<code>--in-cert</code>	Path to the Certificate to renew (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries
<code>--in-pfx-pwd</code>	Password for the PKCS#12 file to renew
<code>--in-jks-pwd</code>	Password for the JKS file to renew
<code>--in-jks-alias</code>	Alias for the certificate to renew in the JKS file
<code>--in-jks-alias-pwd</code>	Alias password for the JKS file to renew

7.3. 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`

7.4. 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
<code>--key</code>	Path to the private key to store
<code>--cert</code>	Path to the certificate to store
<code>--pfx</code>	Path to the PKCS#12 file storing the certificate and its key
<code>--pfx-pwd</code>	Password used to encrypt the PKCS#12 file specified in the <code>--pfx</code> parameter
<code>--jks</code>	Path to export the certificate and private key as JKS
<code>--jks-pwd</code>	Password used to encrypt the JKS file specified in the <code>--jks</code> parameter
<code>--jks-alias</code>	Alias of the private key entry within the JKS
<code>--jks-key-pwd</code>	Password of the private key entry within the JKS

Table 8. Windows-only key and certificate parameters

Parameter	Description
<code>--win-store-save</code>	Triggers the ability to store the certificate and private key into the "MY" certificate store
<code>--win-machine-store</code>	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
<code>--win-use-tpm</code>	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

7.5. 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.

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 - windows store thumbprint)
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
```

8.1. Examples

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

8.1.1. 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
```

8.1.2. 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
```

8.1.3. 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
```

8.1.4. 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.

8.1.5. Revoking a certificate using the account that enrolled it

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

9. 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]
```

9.1. 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.

9.1.1. 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.

9.1.2. Content Parameters

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

Table 9. SCEP enrollment content parameters

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

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

9.2. 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
<code>--profile</code>	SCEP Profile to use
<code>--key-type</code>	Key type
<code>--contact-email</code>	Contact email
<code>--owner</code>	Certificate owner
<code>--team</code>	Certificate owning team
<code>--labels</code>	Labels, in the form <code>key:value</code>
<code>--discovery</code>	Discovery profile to use in order to report the certificate to Horizon after renewal

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

9.3. 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
<code>--key</code>	Path to the private key to store
<code>--cert</code>	Path to the certificate to store
<code>--pfx</code>	Path to the PKCS#12 file storing the certificate and its key
<code>--pfx-pwd</code>	Password used to encrypt the PKCS#12 file specified in the <code>--pfx</code> parameter
<code>--jks</code>	Path to export the certificate and private key as JKS. Optional
<code>--jks-pwd</code>	Password used to encrypt the JKS file specified in the <code>--jks</code> parameter
<code>--jks-alias</code>	Alias of the private key entry within the JKS
<code>--jks-key-pwd</code>	Password of the private key entry within the JKS

9.4. 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 12. SCEP metadata parameters

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

9.5. 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.

10. 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 - windows store thumbprint)
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
```

10.1. Examples

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

10.1.1. 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
```

10.1.2. 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 \  
  --key=/path/to/key
```

```
--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>
```

10.1.3. 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
```

11. WebRA Certificate Lifecycle Operations

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

11.1. 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.

11.2. WebRA Enrollment

11.2.1. Content Parameters

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

Table 13. WebRA enrollment content parameters

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

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

Parameter	Description
<code>--now</code>	If user has only "request enroll" permission, start a blocking loop until review, instead of waiting for the next routine execution

11.3. 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 14. WebRA input certificate parameters

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

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

11.4. WebRA Revocation

The `webra revoke` command takes the following parameters:

Table 15. WebRA revocation parameters

Parameter	Description
<code>--discovery</code>	Discovery profile to use in order to report the certificate to Horizon after enrollment
<code>--cert</code>	Path to the Certificate to revoke (PEM file, PKCS#12 file, JKS file) or cert thumbprint for Windows certificate store entries
<code>--key</code>	Path to the private key of the certificate to revoke if it is not included in the certificate file
<code>--pfx-pwd</code>	Password for the PKCS#12 file to revoke
<code>--jks-pwd</code>	Password for the JKS file to revoke
<code>--jks-alias</code>	Alias for the certificate to revoke in the JKS file
<code>--jks-alias-pwd</code>	Alias password for the JKS file to revoke
<code>--reason</code>	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.

11.5. 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 16. Platform-independent output parameters

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

Table 17. Windows-only output parameters

Parameter	Description
<code>--win-store-save</code>	Triggers the ability to store the certificate and private key into the "MY" certificate store
<code>--win-machine-store</code>	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.

Parameter	Description
<code>--win-use-tpm</code>	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.

11.6. 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 18. SCEP metadata parameters

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

11.7. 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.

12. 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 - windows store thumbprint)
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
```

12.1. Examples

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

12.1.1. 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
```

12.1.2. 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>
```

after this command, run periodically:

TIP

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

12.1.3. 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
```

12.1.4. Revocation of a certificate

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

13. 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.

This command can either be used to update a certificate present on your machine or to update certificates on Horizon using an account with sufficient permission.

CAUTION

To update a local certificate, the 'Update (pop)' common configuration permission must be enabled on the profile the certificate is linked to.

13.1. General Parameters

<code>--confirm</code>	The command asks for confirmation after the changes are computed. Use this flag to disable this behavior and proceed directly. (Optional)
<code>--prompt</code>	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)

13.2. Update Parameters

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

<code>--owner</code>	Set the owner of the certificate. An empty string means deletion of this information. (Optional)
<code>--team</code>	Set the team of the certificate. An empty string means deletion of this information. (Optional)
<code>--contact-email</code>	Set the contact email of the certificate. An empty string means deletion of this information. (Optional)
<code>--labels</code>	Set the labels of the certificate. An empty string means deletion of this information. (Optional)
<code>--metadata</code>	Set the technical metadata of the certificate. To use with caution. An empty string means deletion of this information. (Optional)

13.3. Certificate selection parameters

13.3.1. Local certificate

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

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

13.3.2. Certificate on Horizon server

An account must be configured on the client using `horizon-cli install` and it must have update permissions on the certificate

<code>--id</code>	Id of the certificate to update (Optional)
-------------------	--

13.4. Examples

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

13.4.1. Updating the owner of a certificate

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

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

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

13.4.3. 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"
```

13.4.4. Updating contact email of a certificate referenced on Horizon

```
horizon-cli update-cert --id=<certificate_id> --contact-email="test@evertrust.fr"
```

14. Bulk operations

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

14.1. 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 19. Bulk update command parameters

Parameter	Description
<code>--query</code>	The HCQL query string. Update will be performed on results.
<code>--confirm</code>	Skip confirm.
<code>--owner</code>	The owner to set on certificates matching the query. (Optional)
<code>--team</code>	The team to set on certificates matching the query. (Optional)
<code>--labels</code>	The labels, in the comma separated key:value form, to set on certificates matching the query. (Optional)
<code>--contact-email</code>	The contact email 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"
```

14.2. 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 20. Bulk migrate command parameters

Parameter	Description
<code>--query</code>	The HCQL query string. Update will be performed on results.

Parameter	Description
<code>--confirm</code>	Skip confirm.
<code>--profile</code>	The target profile for the migration.
<code>--owner</code>	The owner to set on certificates matching the query. (Optional)
<code>--team</code>	The team to set on certificates matching the query. (Optional)
<code>--labels</code>	The labels, in the comma separated key:value form, to set on certificates matching the query. (Optional)
<code>--contact-email</code>	The contact email 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
```

14.3. 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 21. Bulk revoke command parameters

Parameter	Description
<code>--query</code>	The HCQL query string. Update will be performed on results.
<code>--confirm</code>	Skip confirm.

TIP

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

15. Automation

The horizon-cli automate command helps you automate the installation of your TLS certificates. It is designed to streamline the process of certificate enrollment, renewal, and installation. This functionality is particularly useful for managing Transport Layer Security (TLS) on web servers, ensuring secure and encrypted connections. It simplifies complex operations through its suite of subcommands, each tailored for specific aspects of certificate management.

15.1. Subcommands

To begin with Horizon CLI automation, use the `--help` flag with any subcommand for detailed usage information:

```
./horizon-cli automate <subcommand> --help
```

Table 22. Automation module subcommands

Subcommand	Description
enroll	This command is versatile, functioning similarly to the init command on servers without certificates, while also capable of re-enrolling or taking control of servers with existing certificates.
init	This command is designed to automatically configure SSL on any newly installed web server. When executed, it seamlessly sets up a secure connection by arranging SSL certificates and enabling SSL on the default HTTPS port (usually port 443). The process includes configuring the necessary SSL settings and restarting the server with SSL enabled.
control	This command is designed to locate certificates within a web server's configuration and offers to bring them under automation control. Taking a certificate under control involves adding necessary metadata and incorporating it into the local memory for routine management and oversight. This process ensures that the certificates are systematically monitored and managed as part of the automated workflow. Be aware that enrolling a new certificate will replace any certificate currently bound to an application running on your machine.
modify	This command allows the user to select one of the managed certs and modify its sans.

Subcommand	Description
create-periodic-task	Sets up a routine task for automated certificate renewal. The task frequency can be specified, with a default period of 6 hours. This helps in automating the renewal process to ensure certificates remain valid without manual intervention.
remove-periodic-task	Removes the previously set periodic task for certificate renewal. This is useful when automated renewal is no longer needed or needs to be reconfigured.
routine	Performs a routine check on all managed certificates to assess if any need renewal. This is a part of proactive certificate management to avoid service disruptions.
list	Lists all managed certificates. This provides an overview of all certificates under management, including details like expiration dates, domains covered, etc.
remove	Removes a managed certificate or a group of certificates. The <id> parameter specifies which certificates to remove, with the option to remove all services using a keyword like 'all'.

TIP

If you're looking for a more guided experience while using horizon-cli automate, the `--prompt` flag is an excellent tool. It enables interactive prompts that guide you step-by-step through the command's options, making it easier to configure and execute your commands accurately.

```
./horizon-cli automate <subcommand> --prompt
```

15.2. Supported Services

The horizon-cli automate feature supports a variety of web server and application server services. This ensures a wide range of compatibility and flexibility for users working with different server environments. The `--target` flag is used within the tool to specify the specific service on which to perform automation tasks, allowing for precise and targeted configuration and management.

CAUTION

In order for horizon-cli to function properly, the targeted service should be started on the machine.

List of the supported services:

- *nginx* (linux)
- *tomcat* (linux & windows)
- *jboss wildfly* (linux)
- *lighttpd* (linux)
- *microsoft iis* (windows)
- *evertrust winhorizon* (windows)
- *evertrust adcsconnector* (windows)
- *generic* (linux & windows)

Example of a standard setup on the default HTTPS port (443) for NGINX:

```
./horizon-cli automate init --target=nginx --automation-policy=<POLICY_NAME>
```

To set up SSL on a custom port (e.g., 9000), use the `--PORT` flag. This option allows SSL configuration on a specific port of your Nginx server.

TIP

```
./horizon-cli automate init --target=nginx --automation-policy  
=<POLICY_NAME> --PORT=9000
```

15.3. Generic Service Automation

The Generic Service within the Horizon Client’s automation module provides a versatile and user-friendly method for obtaining TLS certificates, eliminating the need to specify a pre-existing service or certificate file. This feature is especially beneficial in scenarios where certificates are required to be generated dynamically or for users who seek a more automated certificate management process.

Through the interactive mode:

```
./horizon-cli automate init --target=generic --prompt
```

Non-Interactive Mode: For an even more streamlined experience, certificates can be automatically generated with default parameters in non-interactive mode.

```
./horizon-cli automate init --target=generic --automation-policy=<POLICY_NAME> --cert  
=my_cert.pem --key=my_key.key --chain-file=my_chain.pem --no-interactive
```

Default Storage Location: Newly generated certificates are automatically stored in a default location, which is `/opt/horizon/var/generic` on Unix systems, and `C:\ProgramData\EverTrust\Horizon\Var\Generic` on Windows systems.

Configuration Folder Override: Users have the flexibility to override the default storage location

using the `--config-folder` option. This allows for customization of the storage path as per individual requirements or organizational standards.

```
./horizon-cli automate init --config-folder=/path/to/folder --target=generic
--automation-policy=<POLICY_NAME> --pfx=my_cert.p12 --pfx-pwd=pass123 --chain-file
=my_chain.pem --no-interactive
```

The Generic Service's emphasis on flexibility and user-friendliness makes it a valuable tool for a wide range of users, from those requiring on-the-fly certificate generation to those preferring a hands-off, automated approach.

Windows-Specific Features: On Windows systems, the Horizon CLI offers additional flags to specify the certificate store location:

--user-store: Save the certificate in the user store. This option is beneficial when certificates need to be accessible on a per-user basis.

--machine-store: Save the certificate in the machine store. Ideal for certificates that must be available system-wide. These options provide flexibility in managing certificate storage, catering to different security and accessibility requirements on Windows systems.

15.4. Automation policies

Automation policies are central to the operation of the automation module in the Horizon system. These policies dictate how certificates should be enrolled and renewed, providing a customizable framework to suit various client requirements.

Before using automation policies, they must be pre-configured in the Horizon web app. Each policy is given a unique name for easy identification.

Profile Selection: Policies can be based on EST, SCEP, or ACME profiles, depending on the specific requirements of the enrollment and renewal process.

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

Execution Policy: Includes settings that define how and when the automation should be executed.

Compliance Settings: Specify which CAs are authorized for use within the policy.

Authorized Hash Algorithms: Determine which hash algorithms are acceptable.

Trust Chains: Configure the trust chains that are essential for establishing the trustworthiness of the certificates.

Parameter Specification: When performing automation operations, the relevant automation policy is specified using the `--automation-policy` parameter.

15.5. Existing WebServer certificates

The Horizon CLI's automation module is equipped with a discovery feature that scans and identifies certificates on your machine. It does this by parsing configuration files of your web server or TLS service.

By default, it searches for certificates across all supported services.

```
./horizon-cli automate enroll --automation-policy=<POLICY_NAME>
```

Limit the search to specific services. Separate multiple services with commas.

```
./horizon-cli automate enroll --automation-policy=<POLICY_NAME> --target  
=apache,nginx
```

If you use a different config folder than the default one you can specify your custom folder using `--config-folder`

TIP

```
./horizon-cli automate enroll --automation-policy=<POLICY_NAME> --target  
=apache,nginx --config-folder=/path/to/folder
```

Perform only the discovery phase and print the results without enrolling certificates using `--analyze-only`:

```
./horizon-cli automate enroll --automation-policy=<POLICY_NAME> --target  
=apache,nginx --analyze-only
```

The `--discovery` parameter allows you to integrate certificate enrollment with a discovery campaign pre configured in the Horizon web app before running the CLI command. It includes additional information on the certificate's usage and location on the host machine.

```
./horizon-cli automate enroll --automation-policy=<POLICY_NAME> --discovery-campaign  
=<DISCOVERY_CAMPAIGN_NAME>
```

15.6. Additional enrollment parameters

15.6.1. Challenge Passwords for Enrollment

The `--challenge` parameter is used to provide challenge passwords for EST or SCEP during the enrollment process. Challenge passwords are crucial for the authentication phase in these protocols, ensuring secure communication and identity verification.

TIP

When enrolling certificates using EST or SCEP protocols, the `--challenge` parameter allows you to specify one or more challenge passwords required by the enrollment server. Multiple challenge passwords can be provided, separated by commas, to support various scenarios or multiple servers enrollment.

15.6.2. Post-Enrollment Script Execution

The `--script` parameter allows the execution of a custom script upon the successful completion of a certificate enrollment process. It supports both Bash scripts in Linux environments and PowerShell scripts in Windows environments.

```
./horizon-cli automate enroll --automation-policy=<POLICY_NAME> --script  
=/home/user/post_enroll.sh
```

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 - windows store thumbprint)
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
```

15.6.3. ACME Account Specification for Enrollment

The `--acme-account` parameter is mandatory when enrolling certificates using the ACME protocol. It specifies the ACME account to be used for the enrollment process.

```
./horizon-cli automate enroll --acme-account=myAcmeAccount
```

Add metadata to certificates during the enrollment process: These certificate information parameters enhance the management and traceability of TLS certificates. By using these optional fields, organizations can maintain better oversight and control over their certificate infrastructure.

Table 23. Metadata parameters

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

15.7. Installation

After a successful enrollment or renewal, the certificate will be installed on your machine. The impacted services will be restarted automatically after each certificate enrollment or renewal.

In certain scenarios, you might not want the Horizon CLI to automatically install the new certificate or restart the related services.

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 its predecessor, with the `.new` extension.

15.8. 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.

Using `--auto-renew` flag will check and check the certificate every 6 hours:

TIP

```
./horizon-cli automate enroll --target=nginx --automation-policy  
=<POLICY_NAME> --auto-renew
```

15.9. 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)
 - prompt the user for the configuration folder (equivalent to the `--config-folder` option)

15.10. Certificate commands

15.10.1. Enroll

The `enroll` command can either setup a certificate and empty https configuration from scratch, or take control of certificates, reenrolling them to be compliant when necessary.

Parameters

Table 24. Enroll general parameters

Parameter	Mandatory	Type	Description
--automation-policy	<input checked="" type="checkbox"/>	string	The automation policy to link the certificate to.
--challenge		string array (comma separated)	See challenge section.
--script		string (path to file)	See script section.
--auto-renew		boolean	Configures a status check every 6 hours.
--discovery-campaign		string	Also add discovery info on the enrolled certificate on the campaign passed.
--force-enroll		boolean	Reenroll all selected certificates regardless of their compliance.
--analyze-only		boolean	Instead of executing the enrollments, only displays a summary of possible actions.
--no-interactive		boolean	Disables all interactive inputs. All parameters must then be given using cli flags.
--prompt		boolean	Enable all interactive inputs. All parameters will now be asked for, except ones given using cli flags.

Table 25. Enroll webServer configuration

Parameter	Mandatory	Type	Description
--target		string array (comma separated)	List of services to target. If not given, all available services are targeted.
--no-install		boolean	Disables installation.
--config-folder		string (path to folder)	Explicitly point your webserver configuration folder.
--port		integer	When initializing an empty https configuration on a blank server, choose on which port to listen for https (defaults to the standard https port for the webserver 443 or 8443)
--keystore-password		string	Password of the webserver's keystore if it cannot be deduced from configuration

Table 26. Enroll certificate configuration

Parameter	Mandatory	Type	Description
--dnsnames		string array (comma separated)	List of DNS SANs. If not given, the machine hostname is used.
--ip		string array (comma separated)	List of IP SANs. If not given, no ips are set.
--labels		string array (comma separated, in label:value form)	Horizon labels to add to the certificate on enrollment.
--team		string	Horizon team to add to the certificate on enrollment
--owner		string	Horizon owner to add to the certificate on enrollment
--contact-email		string	Horizon contact email to add to the certificate on enrollment

Table 27. Enroll ACME options

Parameter	Mandatory	Type	Description
--acme-account	<input checked="" type="checkbox"/> (if using ACME)	string	The identifier (email) of the ACME account to use.

Table 28. Enroll generic options

Parameter	Mandatory	Type	Description
--destination-folder		string (path to folder)	Folder to write the new certificates to. Default to <code>/opt/horizon/var/generic</code> on linux and <code>C:\ProgramData\EverTrust\Horizon\Generic</code> on windows.
--pfx		string	Name of the PKCS#12 file to write the enrolled certificate to.
--pfx-pwd		string	Password of the PKCS#12 file to write the enrolled certificate to.
--jks		string	Name of the JKS file to write the enrolled certificate to.
--jks-pwd		string	Password of the JKS file to write the enrolled certificate to.

Parameter	Mandatory	Type	Description
--jks-alias		string	Alias of the JKS file to write the enrolled certificate to.
--cert		string	Name of the file to write the enrolled certificate to in PEM format.
--key		string	Name of the file to write the enrolled key to.
--der		boolean	Save the certificate and key in DER format.
--chain-file		string	Name of the file to write the enrolled certificate chain to.
--user-store		boolean	Save the certificate in the windows user store.
--machine-store		boolean	Save the certificate in the windows machine store.

NOTE

Storage output for generic must choose between pfx, certificate, jks or windows store output.

Examples

Use the interactive mode

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

Enroll a new certificate in the default generic folder

```
horizon-cli automate enroll --automation-policy=<automation policy> --target=generic --cert cert.pem --key key.pem --chain-file chain.pem
```

Add https or control a nginx service

```
horizon-cli automate enroll --automation-policy=<automation policy> --target=nginx
```

Add https to a nginx service without interaction and with custom san values

```
horizon-cli automate enroll --automation-policy=<automation policy> --target=nginx --no-interactive --dnsnames="nginx.test,*.test" --ip "1.1.1.1"
```

15.10.2. Init

The `init` command can setup a certificate and empty https configuration from scratch.

CAUTION

The `init` command can only be used on webservers where no certificate is configured in order to avoid conflicts. See the other commands to match your use case.

Parameters

Table 29. Init general parameters

Parameter	Mandatory	Type	Description
<code>--automation-policy</code>	<input checked="" type="checkbox"/>	string	The automation policy to link the certificate to.
<code>--challenge</code>		string array (comma separated)	See challenge section.
<code>--script</code>		string (path to file)	See script section.
<code>--auto-renew</code>		boolean	Configures a status check every 6 hours.
<code>--discovery-campaign</code>		string	Also add discovery info on the enrolled certificate on the campaign passed.
<code>--analyze-only</code>		boolean	Instead of executing the enrollments, only displays a summary of possible actions.
<code>--no-interactive</code>		boolean	Disables all interactive inputs. All parameters must then be given using cli flags.
<code>--prompt</code>		boolean	Enable all interactive inputs. All parameters will now be asked for, except ones given using cli flags.

Table 30. Init webServer configuration

Parameter	Mandatory	Type	Description
<code>--target</code>		string array (comma separated)	List of services to target. If not given, all available services are targeted.
<code>--no-install</code>		boolean	Disables installation.
<code>--config-folder</code>		string (path to folder)	Explicitly point your webserver configuration folder.

Parameter	Mandatory	Type	Description
--port		integer	When initializing an empty https configuration on a blank server, choose on which port to listen for https (defaults to the standard https port for the webserver 443 or 8443)
--keystore-password		string	Password of the webserver's keystore if it cannot be deduced from configuration

Table 31. Init certificate configuration

Parameter	Mandatory	Type	Description
--dnsnames		string array (comma separated)	List of DNS SANs. If not given, the machine hostname is used.
--ip		string array (comma separated)	List of IP SANs. If not given, no ips are set.
--labels		string array (comma separated, in label:value form)	Horizon labels to add to the certificate on enrollment.
--team		string	Horizon team to add to the certificate on enrollment
--owner		string	Horizon owner to add to the certificate on enrollment
--contact-email		string	Horizon contact email to add to the certificate on enrollment

Table 32. Init ACME options

Parameter	Mandatory	Type	Description
--acme-account	<input checked="" type="checkbox"/> (if using ACME)	string	The identifier (email) of the ACME account to use.

Table 33. Init generic options

Parameter	Mandatory	Type	Description
--destination-folder		string (path to folder)	Folder to write the new certificates to. Default to <code>/opt/horizon/var/generic</code> on linux and <code>C:\ProgramData\EverTrust\Horizon\Generic</code> on windows.
--pfx		string	Name of the PKCS#12 file to write the enrolled certificate to.
--pfx-pwd		string	Password of the PKCS#12 file to write the enrolled certificate to.
--jks		string	Name of the JKS file to write the enrolled certificate to.
--jks-pwd		string	Password of the JKS file to write the enrolled certificate to.
--jks-alias		string	Alias of the JKS file to write the enrolled certificate to.
--cert		string	Name of the file to write the enrolled certificate to in PEM format.
--key		string	Name of the file to write the enrolled key to.
--der		boolean	Save the certificate and key in DER format.
--chain-file		string	Name of the file to write the enrolled certificate chain to.
--user-store		boolean	Save the certificate in the windows user store.
--machine-store		boolean	Save the certificate in the windows machine store.

NOTE Storage output for generic must choose between pfx, certificate, jks or windows store output.

Examples

Use the interactive mode

```
horizon-cli automate init --prompt
```

Enroll a new certificate in the default generic folder

```
horizon-cli automate init --automation-policy=<automation policy> --target=generic  
--cert cert.pem --key key.pem --chain-file chain.pem
```

Add https to a nginx service

```
horizon-cli automate init --automation-policy=<automation policy> --target=nginx
```

Add https to a nginx service without interaction and with custom san values

```
horizon-cli automate init --automation-policy=<automation policy> --target=nginx --no  
-interactive --dnsnames="nginx.test,*.test" --ip "1.1.1.1"
```

15.10.3. Control

The `control` command can take control of an existing certificate on your machine.

CAUTION

The `control` command can only be used on known and compliant with Horizon certificates. If the certificate needs to be enrolled, see the other commands to match your use case.

Parameters

Table 34. Control general parameters

Parameter	Mandatory	Type	Description
<code>--automation-policy</code>	<input checked="" type="checkbox"/>	string	The automation policy to link the certificate to.
<code>--challenge</code>		string array (comma separated)	See challenge section.
<code>--script</code>		string (path to file)	See script section.
<code>--auto-renew</code>		boolean	Configures a status check every 6 hours.
<code>--discovery-campaign</code>		string	Also add discovery info on the enrolled certificate on the campaign passed.
<code>--analyze-only</code>		boolean	Instead of executing the control, only displays a summary of possible actions.

Parameter	Mandatory	Type	Description
--no-interactive		boolean	Disables all interactive inputs. All parameters must then be given using cli flags.
--prompt		boolean	Enable all interactive inputs. All parameters will now be asked for, except ones given using cli flags.

Table 35. Control webServer configuration

Parameter	Mandatory	Type	Description
--target		string array (comma separated)	List of services to target. If not given, all available services are targeted.
--no-install		boolean	Disables installation.
--config-folder		string (path to folder)	Explicitly point your webserver configuration folder.
--keystore-password		string	Password of the webserver's keystore if it cannot be deduced from configuration

Table 36. Control certificate configuration

Parameter	Mandatory	Type	Description
--labels		string array (comma separated, in label:value form)	Horizon labels to add to the certificate on enrollment.
--team		string	Horizon team to add to the certificate on enrollment
--owner		string	Horizon owner to add to the certificate on enrollment
--contact-email		string	Horizon contact email to add to the certificate on enrollment

Table 37. Control ACME options

Parameter	Mandatory	Type	Description
--acme-account	<input checked="" type="checkbox"/> (if using ACME)	string	The identifier (email) of the ACME account to use when renewing.

CAUTION

For generic control, it is as of now disabled outside the default folder

(/opt/horizon/var/genric or C:\ProgramData\EverTrust\Horizon\Generic) to ensure no service interruption. Use post enrollment scripts to copy the certificate.

Table 38. Control generic options

Parameter	Mandatory	Type	Description
--pfx		string	Name of the PKCS#12 file to control.
--pfx-pwd		string	Password of the PKCS#12 file to control.
--jks		string	Name of the JKS file to control.
--jks-pwd		string	Password of the JKS file to control.
--jks-alias		string	Alias of the JKS file to control.
--cert		string	Name of the certificate file to control.
--key		string	Name of the key file to control.
--der		boolean	If true, the certificate and key to control are in DER format.
--chain-file		string	Name of the file to write the enrolled certificate chain to.
--user-store		boolean	Control the certificate in the windows user store.
--machine-store		boolean	Control the certificate in the windows machine store.
--thumbprint		boolean	Thumbprint of the certificate to control. To use with --machine-store or --user-store

NOTE Input for generic must choose between pfx, certificate, jks or windows store.

Examples

Use the interactive mode

```
horizon-cli automate control --prompt
```

Control a certificate in the default generic folder

```
horizon-cli automate control --automation-policy=<automation policy> --target=generic
--cert cert.pem --key key.pem --chain-file chain.pem
```

Control a certificate on an already configured nginx

```
horizon-cli automate control --automation-policy=<automation policy> --target=nginx
```

15.10.4. Modify

The `modify` command allows to select a managed certificate and reenroll it, modifying its sans.

NOTE Automation policy cannot be modified

Parameters

Table 39. Modify general parameters

Parameter	Mandatory	Type	Description
<code>--challenge</code>		string array (comma separated)	See challenge section.
<code>--script</code>		string (path to file)	See script section.
<code>--discovery-campaign</code>		string	Also add discovery info on the enrolled certificate on the campaign passed.
<code>--prompt</code>		boolean	Enable all interactive inputs. All parameters will now be asked for, except ones given using cli flags.

Table 40. Modify webServer configuration

Parameter	Mandatory	Type	Description
<code>--target</code>		string array (comma separated)	List of services to target. If not given, all available services are targeted.
<code>--no-reload</code>		boolean	Do not reload webservers after certificate enrollment.

Table 41. Modify certificate configuration

Parameter	Mandatory	Type	Description
<code>--dnsnames</code>		string array (comma separated)	List of DNS SANs. If not given, the machine hostname is used.
<code>--ip</code>		string array (comma separated)	List of IP SANs. If not given, no ips are set.

Parameter	Mandatory	Type	Description
--labels		string array (comma separated, in label:value form)	Horizon labels to add to the certificate on enrollment.
--team		string	Horizon team to add to the certificate on enrollment
--owner		string	Horizon owner to add to the certificate on enrollment
--contact-email		string	Horizon contact email to add to the certificate on enrollment

Table 42. Modify ACME options

Parameter	Mandatory	Type	Description
--acme-account	<input checked="" type="checkbox"/> (if using ACME)	string	The identifier (email) of the ACME account to use.

Examples

Use the interactive mode

```
horizon-cli automate modify --prompt
```

Set sans for the generic certificates

```
horizon-cli automate modify --target=generic --dnsnames="nginx.test,*.test" --ip "1.1.1.1"
```

15.11. 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.

15.12. 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, its "id" in the local database will be `apache-*:443`.

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`.

15.13. Routine

This command does not take any parameter. Its usage is described in the renewal section.

15.14. Management commands

15.14.1. Periodic task

To run the routine at specified intervals, the periodic task command can create scheduled execution on windows (scheduled task) and linux (cron).

Two commands are available, to remove and add the task.

Parameters

Table 43. Create periodic task arguments

Argument	Mandatory	Type	Description
Period		duration	Run the routine command every period from midnight UTC. Must be superior to 1h and inferior to 24h. Defaults to 6h.

Table 44. Create periodic task parameters

Parameter	Mandatory	Type	Description
--user		string	Linux only: The user to impersonate while running the routine. Defaults to root user.

Examples

Create a periodic task to run routine every 7h

```
horizon-cli automate create-periodic-task 7h
```

Remove the periodic task

```
horizon-cli automate remove-periodic-task
```

15.14.2. List

The **list** command lists the currently managed certificates and various information about them, notably the associated automation policy.

Parameters

Table 45. List parameters

Parameter	Mandatory	Type	Description
--json		string	Output the current state as json instead of human readable format.

Examples

Show the currently managed certificates

```
horizon-cli automate list
```

Show the currently managed certificates as json

```
horizon-cli automate list --json
```

15.14.3. Remove

The `remove` command erases a certificate binding from the horizon-cli managed certificates.

Parameters

Table 46. Remove arguments

Argument	Mandatory	Type	Description
id		string	Id of the services to remove. See <code>automate list</code> to get the id. Use <code>all</code> to erase every managed certificate.

Table 47. Remove parameters

Parameter	Mandatory	Type	Description
--restore		boolean	Enables restoration of the oldest backup for the removed certificate.

Examples

Remove all currently managed certificates

```
horizon-cli automate remove all
```

Remove the certificate linked to nginx https and restore the original one

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

15.15. Examples

15.15.1. Enroll certificates used by nginx and apache

```
horizon-cli automate enroll \  
  --target=nginx,apache \  
  --automation-policy=<POLICY_NAME>
```

15.15.2. Enroll certificates using the generic target

```
horizon-cli automate enroll \  
  --target=generic \  
  --automation-policy=<POLICY_NAME>
```

15.15.3. Use the interactive mode

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

15.15.4. Check if the previously enrolled certificates need to be renewed

```
horizon-cli automate routine
```

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

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

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

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

15.15.7. 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
```