Krestfield EzSign

Installation and Configuration Guide

version 4.2.1

Introduction

The Krestfield EzSign suite enables applications to quickly generate and verify digital signatures or encrypt and decrypt data without the need for complex programming

It provides the following key features:

Compliant Signature Generation and Verification

The server produces PKCS#7 compliant signatures (RSA or Elliptic Curve), which include signed attributes and the certificate chain. The SHA-1, SHA-2 and SHA-3 suite of digest algorithms are supported

The server performs full signature validation including path building and revocation checking, supporting both CRL and OCSP revocation checking

OCSP validation also supports the signing of OCSP requests and the inclusion of the correct Service Locator extension for use with the IdenTrust OCSP four corner model

Support for proxies to access CRL and OCSP servers is also supported, including proxies requiring authentication

AES Encryption and Decryption

AES keys of 128, 196 or 256 bits can be generated for encryption/decryption purposes. Data is encrypted using CBC (Cipher Block Chaining) and a random IV (Initialisation Vector) is generated for each and every encryption operation, ensuring the data is secured to the maximum level

Multi Token Support

The server supports several mechanisms for secure key storage, including:

- Cloud Based HSMs (including AWS Cloud HSM, Google KMS and the Thales DPoD Cloud HSM)
- PKCS#11 based HSMs (such as the nCipher and Thales/Gemalto Luna range)
- Thales PayShield HSMs

- Software

  For testing or applications that do not require hardware key protection, a software key store may be used. Keys and certificates are AES encrypted

Java based

The server is completely java based, and supports Java versions 8 onwards

Simple Client API

A thin java or .NET client is available with a simple interface to the server enabling rapid integration. You can start to generate signatures by writing only two lines of code!

A REST API is also available via the PKCloud product

Multi-Channel

The server provides key separation and the ability to support different configuration options per channel e.g. one channel can use a software key store whilst another makes use of an HSM, all from the same server

The number of channels is not limited (technically or by license)

Installation

The software is delivered as either a zip or gzip (tar.gz) file. Unzip and unpack the installation file to a location of your choice

The installation files are organised as follows

```
[Installation Folder]/EzSignVX.Y.Z/EzSignClient
[Installation Folder]/EzSignVX.Y.Z/EzSignServer
```

(Where X.Y.Z is the version number e.g. `EzSignV4.2.1`)
The EzSignClient folder (or just the `kezsign-client-X.Y.Z.jar` file) should be copied to all clients that will be accessing the server

Within the EzSignClient folder there are the following directories

- `/doc` – contains EzSignClient documentation
- `/lib` – contains the EzSignClient jar files
- `/samples` – contains client samples

Within the EzSignServer folder there are the following directories

- `/bin` – scripts to start, stop and manage the server
- `/doc` – contains this document
- `/lib` – the EzSignServer libraries
- `/logconf` – contains the logging (log4j2.xml) configuration file
- `/logs` – the default location for log files
- `/samples` – contains sample configurations

The product is bundled with a Java runtime but if you prefer to use your own, edit the `envs.sh` file in the `EzSignServer/bin` directory and edit the `JAVA_HOME` parameter so that it points to your JDK/JRE installation folder e.g.

```
JAVA_HOME=/fs01/app/jdk1.8.0_251
```

The account used to run the server must have permissions to access the `EzSignServer` folder and be able to execute the scripts (located in the `/bin` directory) and jar files (located in the `/lib` directory). The account must also have read/write access to the keystore location (see the Key Store Files section later)

Client applications must be able to access the jar files located in the `EzSignClient/lib` folder

For nCipher HSMs the account running the server must have access to the nCipher `kmdata/local` folder. This is usually achieved by adding the user to the `nfast` group

Components

The EzSign product consists of the following individual components:

- The EzSign Server
  - The processing engine which manages the keys, HSMs and performs the signature generation and verification

- The EzSign Client
  - The client component which is integrated with applications and makes the calls to the server

- The EzSign Management Utility
  - The utility which allows for certificate management, the configuration of passwords and the generation of CSRs (Certificate Signing Requests)

- The EzSign Control Utility
  - A utility which permits the remote pausing/resuming of the server and the ability to alter the logging level on the fly

These are discussed in more detail below

EzSign Server

The EzSign Server is a multi-channel, digital signature and encryption processing application capable of interfacing with Hardware Security Modules for secure signature generation and validation or encryption/decryption operations

The Server supports several different key stores including a software key store (useful for non-production or systems that do not require a high level of security) and hardware key stores

Hardware key stores support any HSM (Hardware Security Module) which exposes the industry standard PKCS#11 interface (including the nCipher range of HSMs). It also supports several cloud based HSMs and the Thales PayShield HSMs

The Server makes use of a properties file for its configuration. Certain operations such as the setting of passwords, generation of CSRs (Certificate Signing Requests) and importing of issued certificates are carried out via the Management utility

Channels

A single Server can be configured with multiple channels. A channel contains only the keys and certificates which were generated or imported for that particular channel. Therefore a channel provides key separation such that different applications can access only the keys and certificates of interest to them A channel also supports its own token

Each channel can utilise a different key store and different configuration options. For example, one channel can make use of a PayShield and be configured to perform OCSP revocation checking whilst another channel can be configured for an nCipher HSM and not perform any revocation checking

There are two types of channel

- PKI: Performs digital signature generation and verification
- Symmetric: Provides encryption an decryption operations

Key Store Files

Whichever key store is used the server stores information locally about each key and certificate held by, or protected by the key store

The server will store these files in a folder which has the same name as the channel. This folder will be located beneath the key store directory (specified by the `keyStoreDir` property)

For example, if `keyStoreDir` is set to `/opt/ezsign/keystore` and a channel is configured called `ChannelOne`, then the keys and certificates associated with ChannelOne will be stored here:

```
/opt/ezsign/keystore/ChannelOne
```

These files are encrypted by the server and may contain encrypted key information (in the case of a software key store). If an HSM is used, either references to keys will be stored (identifiers or labels) or key data that is encrypted by the HSM's itself (as in the case of the PayShield HSMs)

Access to these files should be controlled such that only the server is able to access them as although they are encrypted, accidental deletion or corruption may prevent the server from operating correctly

The account used to run the server must have read/write permissions to the keystore location

Logging

Logging is performed via Apache Log4J2. See http://logging.apache.org/log4j/2.x/

By default, logs are written to:

```
EzSignServer/logs/ezsign.log
```

Logs are rolled over once they reach 100mb. At which point they are zipped and stored in a folder (a separate folder is created per month) e.g.

```
EzSignServer/logs/2019-05/ezsign-2019-05-21-1.log.gz
```

Thus saving log files automatically whilst preserving space

Log4J is highly configurable and these options can easily be altered by editing the configuration file located here:

```
EzSignServer/logconf/log4j2.xml
```

If an alternative location for the configuration file is required, update the `LOGCONFIG` property in the `ezsign-daemon-start` script to point to the new location e.g.

```
LOGCONFIG=/opt/ezsign/server1/log4j2.xml
```

Refer to the Apache documentation for more information on the configuration options

The Thread Pool

When the server starts it creates a pool of channels. Each channel loads its keys and certificates and creates a connection to the HSM (if required)

The size of the pool is configured via the `threadPoolSize` property and defaults to 1 if omitted. Larger sizes will allow for greater parallelism but may result in slower start up times. Some degree of experimentation with a particular setup may be required to find the optimum value. Usually, a good starting point is to set this value to the number of clients (or client threads) expected to connect to the server at any one time

As each request arrives the next available free channel instance processes the request. If no free instance is available the server will wait a short time and then check again whether any instances have become free. It will do this until the `waitTimeoutIfAllBusy` value has been reached at which point an error will be returned to the client. This prevents hanging if, for example external OCSP servers were running particularly slowly. The client may choose to try again if this does occur

Properties Configuration

The server's configuration is contained within a properties file. The properties file must be passed to the server (as a parameter) at start up. The server will then load the properties and then wait for client requests

Essentially the properties file contains information about the IP Addresses and Ports to listen on, the key store location, logging settings and channel configurations

For detailed information about the available properties, refer to the following:

https://krestfield.github.io/docs/ezsign/ezsign_properties.html

A sample properties file can be found here:

https://krestfield.github.io/docs/ezsign/ezsign_sample_properties.html

Certificate Path Checking

During signature verification, EzSign will perform the following operations:

1. Verify the signature data against the signer certificates' public key
2. Build a certificate path
3. Perform path checks
4. Check certificate revocation

Step 1 performs the mathematical calculations over the signature data I.e. digesting the data, decryption and digest comparisons

Step 2 builds a path, using the certificates from the signature and certificates that may have been uploaded into the channel. A trusted root must have been imported into the channel for this step to succeed as the path must terminate on a trusted root

Each certificate is checked for time validity (the Valid From date is before the current time and the Valid To date after) and its signature is verified against the issuing certificates public key

Step 3 then performs the following steps:

1. If the setting `channel.N.verify.denyWeakCertificateHash` is true, if any of the certificates in the path have a weak hash (anything weaker than SHA-2) they will be rejected
2. If the setting `channel.N.verify.relaxAllCertExtensionChecks` is true, no further checks will be performed on the path, if this setting is false (or not set at all), then the additional checks will be performed:
    1) If the settings for key size (`channel.N.verify.minKeySize` and `channel.N.verify.maxKeySize`) are set, each certificate's key size must be within these limits
    2) The certificates must have the keyUsage extension and this must be marked as critical
    3) Signer certificates must have the Digital Signature key usage set
    4) If the setting `channel.N.verify.nonRepudiationRequired` is true, signer certificates must also have the Non Repudiation key usage set
    5) For CA and Root CA certificates they must have the Key Cert Sign key usage set and if the setting `channel.N.verify.caBasicConstraintsRequired` is true, they must also have the Basic Constraints extension.
    When Basic Constraints are checked the path length permitted will also be checked
    If the setting `channel.N.verify.relaxRootCertExtensionChecks` is true, these additional checks will not be carried out on root certificates. This may be required if legacy root certificates are being used

All the checks performed in Step 3 may be overridden by developing a custom path check class

Custom Certificate Path Checking

Specific checks may be performed on certificate paths by developing a custom java class. You may develop the custom class yourself following the details below, or Krestfield can develop one to your specific requirements. Custom path checking may be required, if for example you wish to check a certificate has been registered, check custom extensions or any other specific certificate checks your system may require

To create a custom path checker perform the following operations:

1. Create a Java project and add a reference to the `KEzSign.jar` (located in the `EzSignServer/lib` directory of the installation)
2. Create a new class (e.g. `MyCustomPathChecker`) which implements the `KPathCheckBase` interface e.g.

```
package com.myorg.ezsign.pathcheck;

import com.krestfield.ezsign.KEzSignException;
import com.krestfield.ezsign.KPathException;
import com.krestfield.ezsign.log.KSigLog;
import com.krestfield.ezsign.path.KPathCheckBase;

public class MyCustomPathChecker implements KPathCheckBase
{
    public void loadProperties(int channelNum, Properties props) throws KEzSignException
    {
        KSigLog.LogEvent("Loading properties from CustomPathCheck.loadProperties");

        // Load any specific properties required
    }

    public void check(ArrayList<X509Certificate> certPath) throws KPathException
    {
        // Perform custom checks on the path
        throw new KPathException("MyCustomPathChecker - Not yet implemented");
    }
}
```

3. Implement the `loadProperties` and `check` methods (see below), add the compiled class to the server classpath and reference this class in the server properties as follows:

   ```
   channel.N.verify.pathCheckClass=com.myorg.ezsign.pathcheck.MyCustomPathChecker
   ```

Note: If a custom path checker is implemented, the default checks will not be performed. If you require any of the custom checks to be implemented, Krestfield can supply source code snippets to assist with this

Implement the loadProperties Method

This method is passed the channel number and the properties object – which is the loaded server properties

Therefore, you may add any specific properties into an existing server properties file. These can then be read in this method

E.g. You could add in the following specific properties for channel 1:

```
channel.1.mypathchecker.allowedExtensions=2.5.29.19,2.5.29.31,1.2.840.114021.1.4.2
```

```
channel.1.mypathchecker.validityTimeDays=30
```

Or you could have server wide settings such as:

```
mypathchecker.allowedExtensions=2.5.29.19,2.5.29.31,1.2.840.114021.1.4.2
mypathchecker.validityTimeDays=30
```

If there is a failure to load any properties, throw an `KEzSignException`

Implement the check Method

All certificates in the path are included in the ArrayList, index 0 is always the end-entity (signer) certificate, other certificates are CA certificates and the last certificate in the list will be the root

Perform the required checks on these certificates and throw a `KPathException` if there are any failures or rejections

Note, that you may use the EzSign logging functions which will result in the messages being sent to the standard EzSign log files. Use the `KSigLog` class to perform this logging

Start Scripts

The server is started by calling the `ezsign-daemon-start` script located in the `EzSignServer/bin` directory. This starts the server managing process (the daemon) and also starts the server listening on the interface and port configured

The script must be passed the properties file and the master password. The master password may be passed as an additional parameter e.g.

```
ezsign-daemon-start.sh server.properties masterpassword
```

Or if the master password is held within a file, the filename may be passed after the -f switch e.g.

```
ezsign-daemon-start.sh server.properties -f masterpassfile.txt
```

Finally, the master password may be set within the environment variable `EZMASTERPASS`, in which case no password parameter is required e.g.

```
ezsign-daemon-start.sh server.properties
```

Once the daemon has started the status can be monitored via the EzSign Control utility (see below). The listening server can be stopped and started by running the following scripts

```
ezsign-server-stop
ezsign-server-start
```

These scripts do not halt the daemon process but stop and start the server listening process which will either prevent or permit further requests from the client being processed

To stop the main daemon process run the following script

```
ezsign-daemon-stop
```

Overriding Properties

The following properties can be overridden by system properties

keyStoreDir
server.port
server.bindIpAddress
server.threadPoolSize
server.allowedSourceIpAddresses
server.waitTimeoutIfAllBusy
log.level
server.ctrl.port
server.ctrl.bindIpAddress
server.ctrl.allowedSourceIpAddresses

If included as `-D` parameters at server start up, they will override any settings configured within the properties file

For example. The `ezsign-daemon-start.sh` script could be updated as follows:

```
java -Dserver.port=1234 -Dserver.bindIpAddress=127.0.0.1
     -cp $CLASSPATH com.krestfield.ezsign.server.EzSignServer $1 $2
```

(where the highlighted items were added)

This would result in the `server.port` being set to `1234` and the `bindIpAddress` to `127.0.0.1` whatever values the properties file held

These fields could be configured as parameters which would allow for the dynamic setting of these values. For example:

```
java -Dserver.port=$PORT -cp $CLASSPATH
                com.krestfield.ezsign.server.EzSignServer $1 $2
```

Or if you intended to pass these as parameters to the start script, such as:

```
ezsign-daemon-start.sh [port] [properties file] [password]
```

e.g.

```
ezsign-daemon-start.sh 5006 server.properties mypassword
```

You could update the start script as follows:

```
java -Dserver.port=$1 -cp $CLASSPATH
                com.krestfield.ezsign.server.EzSignServer $2 $3
```

This functionality allows for the dynamic control of the server which may be of use to automate multi-instance deployments

Passwords

The EzSign server requires two passwords: A Token Password and a Master Password. The Token Password is used to either encrypt software keys or to authenticate to HSMs and is stored in the properties file, encrypted. The Master Password is not stored but is used to derive a key under which the Token Passwords are encrypted

Token Passwords are per channel and each channel can be configured with a different Token Password

The Master password is per server. The same Master Password must be provided each time a Token Password is configured or the server started

When the server is started the Master Password must be provided. This password is then used to decrypt the Token Passwords stored in the properties file. If the Master Password provided is incorrect, the server cannot start

If using the Start Script (e.g. `ezsign-daemon-start.sh`) the master password can be provided in one of three ways:

As clear text e.g. the server could be started as follows:

```
ezsign-daemon-start.sh server.properties masterpassword
```

Where `masterpassword` is the master password

Or the master password can be stored within a file, and the filename them passed to the start script. If this option is chosen then the -f switch must be specified before the filename e.g.

```
ezsign-daemon-start.sh server.properties -f masterpassfile.txt
```

Where `masterpassfile.txt` is the file containing the master password

Finally, the master password may be set within the environment variable `EZMASTERPASS`, in which case no password parameter is required when calling the start script e.g.

```
ezsign-daemon-start.sh server.properties
```

On UNIX/Linux the master password could be set as follows:

```
export EZMASTERPASS=masterpassword
```

Note: If no password is passed in this way and the `EZMASTERPASS` variable is not available you will be prompted to enter a password before the server will start

If automating start-up you may wish to add further protection around the master password. For example, on a Microsoft OS, the server could be started by a PowerShell script or Scheduled Task that run under a specific windows account. This account could then store the master password in the Windows Credential Manager and either set as an environment variable or pass to the script when starting

On Linux/UNIX systems the master password could be stored in a file. The account starting the server could then be configured to be the only account with read access to this file

EzSign Client

The EzSign client is a lightweight java (or .NET) component which exposes a simple interface enabling applications to quickly utilise the PKI and encryption functions offered by the EzSign Server

The communication between the client and server is socket based. The server name (or IP Address) and the port the server is listening on must be specified at the client

The communications between the client and server can be protected by using an Authentication Code. This is a passphrase which is provided at the client and also configured at the server (see Set Authentication Code below)

Refer to the *EzSign Client Integration Guide* for details on how to integrate the client with your applications

EzSign Management

The Management utility enables the following operations:

- Setting of Token Passwords
- Generation of CSRs (Certificate Signing Requests)
- Specifying Signing Keys
- Importing Certificates
- Importing existing HSM keys
- Deleting Certificates
- Displaying Certificates
- Generating AES Keys
- Displaying AES Keys
- Deleting AES Keys
- Translating Keystore Objects from an old password to new
- Setting the Authentication Code

The server will not start until the token passwords have been set and you will be unable to sign or verify signatures until the required keys and certificates have been generated and imported. Encrypting and decrypting requires an AES key to be generated. Therefore, the Management tool must usually be run before the server can be started (if you are duplicating configurations there is no need to re-run the tool)

Starting

To start the management utility, run the following script which requires that the server properties file is passed as a parameter e.g.

```
ezsign-manage.sh server.properties
```

(On Windows run `ezsign-manage.bat`)

Once the script starts you will be asked to enter the Master Password before being able to proceed

```
Krestfield EzSign Management
----------------------------

The master password is required to manage the server

Enter password:
```

Type the Master Password and press enter. This will then display the menu

```
Krestfield EzSign Management
----------------------------

1.  Set Passwords

2.  Generate CSR
3.  Import Certificate
```

```
  4.  Import Existing HSM Keys

  5.  Set Signing Key

  6.  Display Certificates
  7.  Display Certificates Details
  8.  Delete Certificate

  9.  Generate AES Key
  10. Display AES Keys
  11. Delete AES Key

  12. Translate Keystore Objects
  13. Display all Channel Objects
  14. Set Authentication Code

  15. Exit

Command:
```

Setting Master and Token Passwords

To set the Master and Token passwords for a channel choose option 1. You will then be prompted to choose the channel, enter the Master Password and Token Passwords and then whether you want to write the new data to the properties file. This option must be run before the server is started in order to encrypt the Token Passwords

```
  Channels
  --------

    1. TEST
    2. TechTest1
    3. TechTest2

Select Channel Number: 1

  Master Password
  ---------------

  This is the password which is used to start the server
  It is used to encrypt other passwords (Token Passwords etc). It is not stored

 Enter Password:
Retype Password:

  Token Password
  --------------

  For software tokens, this is the password used to encrypt keys
  For PKCS#11 tokens, this is the token PIN/Password used to authenticate,
      also referred to as the operator password
  For HSM9000 tokens, this is password is used to re-encrypt the local keys
     (although they are already encrypted under the HSM Master Key)
  This password is stored in the configuration, encrypted under the master password

 Enter Password:
Retype Password:
```

```
  A backup of the original has been saved to ..\test.properties.2016.11.12.09.03
  Ready to update properties file ..\test.properties with the new passwords

Proceed? (y/n): y

  The properties file ..\test.properties has been updated successfully.
```

Note: For sensitive operations including:

- Generating CSRs
- Importing certificates
- Deleting certificates
- Setting the signing key
- Generating AES keys
- Deleting AES keys

The channels token password will be required. When required you will receive a prompt as follows:

```
The token password is required for this operation.  Please enter the token password below

Token Password:
```

The token password must then be entered to proceed

Generating a CSR

To generate a CSR (Certificate Signing Request) choose option 2. You will be prompted to enter the selected channel again, then requested DN (Distinguished Name), key size and the filename to store the CSR

```
Generating CSR for channel TEST

Distinguished Name: CN=Test,O=Krestfield Ltd,C=GB

CSR Filename: krestfield.p10

  Generated CSR Data:
-----BEGIN CERTIFICATE REQUEST-----
MIICYTCCAUkCAQAwHjEPMA0GA1UEAwwGVGVzdC5PMQswCQYDVQQGEwJHQjCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAIMvY7RaJtHTz2jB7Nfqb2OLANmjOqACybd5FSlwFxvCSjTzoRoG
Y97aFWldDHueyVLmKKJtYMScTIs1gZSvm9guUpOhVsvQ0KaX58ZFMmRvlmeBlP2rbQIe0F1Fp724
XggI/5dXr9OKVbdIWrJkZTsFYn8bXU7nY1MAuRl5NK6CSkl6XZTvODRezL9ioFmkE09EWP4wIKyQ
zQW0Z/mn7a51eiJA+utBf3MgtUkmEzTc8Z73xrGflwt0fCnghs1gXnHNlZMkiR3l1iOPV/ppQ9l2
FsVv85JFk4eMJxP1cp+niEtoTFIc49JF/nB4u+B1aslnBBimn2oMzD/gItOdCc0CAwEAATANBgkq
hkiG9w0BAQsFAAOCAQEAYh4erwAkHL+ZuHMRcOmufK9ZJCxLXgbF2DjCgCC53xMuNwiQ7wIL6at6
N1jK8v4WlIaRE1WRSOO7k7OG3MwXBtDy3oVmr5NWEKu0WSVbwM/7mVjkczuBxLXdr/pEeyH9Nm02
h3kH0sO25xt5BjoCzcoHIYmmHxa5tyzjchqTE5Fw68S/7rusodvAbEwNKYQQhHTCwAzuusGDxb0D
+JHNTk63zenDHw56pcURsvmA16BSXF30MatVcoTd7elqmAho+yUm0c0CVOfGoH2zFv7jcyl8jDtm
w3Kg45sND7L2A9hII0QGy3L+seE/GpMaoxhhDBPxbIfAkqecxyapA+aX3w==
-----END CERTIFICATE REQUEST-----

  CSR written to krestfield.p10
```

The CSR should then be processed by the CA who will issue the certificate. Once the certificate has been issued the Import Certificate option can be chosen to import the certificates

Importing Certificates

To import root certificates or certificates issued from a CSR choose option 3. You will then be prompted for the channel and a path to the certificate.

```
  Channels
  --------

    1. TEST

Select Channel Number: 1

Path to certificate: /opt/ezsign/certs/signing.cer

  Certificate imported successfully
```

If the certificate is associated with a private key on the token, you will also be prompted if you wish to set this as the new signing certificate:

```
Do you wish to set this as the default signing certificate? (y/n): y
  A backup of the original has been saved to ..\samples\test.properties.2016.10.12.11.00
```

```
  Ready to update properties file ..\samples\test.properties with the new passwords
Proceed? (y/n): y

  The properties file ..\samples\test.properties has been updated successfully.
```

Choosing **y** will result in the properties file being updated. If you do not wish to set this as the signing certificate now, you can do this later by running option 5. Set Signing Key

Note that for signing certificates, all certificates in the path must be imported

Importing Existing Keys and Certificates

If you are using a PKCS#11 HSM which has been used by another system you may import these objects into EzSign. To do this choose option `4.Import Existing HSM Keys`

Choose the channel to import the objects to. Note: the channel selected must be configured to use the HSM from which the objects are to be imported

If not previously entered, you will be prompted for the token password:

```
  The token password is required for this operation.  Please enter the token password
  below

Token Password:

  Password verified OK

  Importing objects from the HSM...

  Objects imported from the HSM OK
```

The objects imported can be examined by running the `Display Certificates` or `Display all Channel Objects` options. The signing key can be selected by running the `Set Signing Key` option

Setting the Signing Key

To set the default signing key (that is the key that will be used to sign data), choose option 5. You will then be presented with a list of available certificates which have associated private keys and can therefore be used to sign data. Note: a certificate will not be available for signing if there is not a complete path for that certificate

```
  Current Signing Certificate:
  ----------------------------

       Subject: CN=Test Cert 1, O=Krestfield Ltd, C=GB
       Issuer : CN=Krestfield Test CA, OU=Engineering, O=Krestfield Ltd, C=GB
       Valid From: Wed Oct 12 08:12:48 BST 2016
       Valid To  : Thu Oct 12 08:12:48 BST 2017
       Serial Number: 5b000000348a17f73059f07217000000000034


  Available Signing Certificates:
```

```
------------------------------
  1. Subject: CN=Test Cert 2, O=Krestfield Ltd, C=GB
     Issuer : CN=Krestfield Test CA, OU=Engineering, O=Krestfield Ltd, C=GB
     Valid From: Wed Nov 12 18:12:48 BST 2016
     Valid To  : Thu Nov 12 18:12:48 BST 2017
     Serial Number: 5b000000348a17f73059f07217000002010F13


Select Certificate Number: 1

  A backup of the original has been saved to ..\test.properties.2016.10.12.09.28
  Ready to update properties file ..\test.properties with the new passwords

Proceed? (y/n): y

  The properties file ..\test.properties has been updated successfully.
```

Displaying and Deleting Certificates

Options 6, 7 and 8 are used to delete and display the available certificates. When certificates are displayed they are shown in the following format

```
  Current Signing Certificates:
  -----------------------------
    1. Subject: CN=Test 1, O=Krestfield Ltd, C=GB
       Issuer : CN=Krestfield Test CA2, OU=Engineering, O=Krestfield Ltd, C=GB
       Valid From: Wed Oct 12 08:12:48 BST 2016
       Valid To  : Thu Oct 12 08:12:48 BST 2017
       Serial Number: 5b000000348a17f73059f07217000000000034

    2. Subject: CN=Krestfield Test CA2, OU=Engineering, O=Krestfield Ltd, C=GB
       Issuer : CN=Krestfield Test Root CA2, OU=Engineering, O=Krestfield Ltd, C=GB
       Valid From: Fri Dec 04 08:22:54 GMT 2015
       Valid To  : Thu Nov 29 08:22:54 GMT 2035
       Serial Number: 3e000000022542122l0f37d14f000000000002

    3. Subject: CN=Krestfield Test Root CA2, OU=Engineering, O= Krestfield Ltd, C=GB
       Issuer : CN=Krestfield Test Root CA2, OU=Engineering, O= Krestfield Ltd, C=GB
       Valid From: Thu Dec 03 09:13:47 GMT 2015
       Valid To  : Wed Dec 03 09:23:47 GMT 2036
       Serial Number: 5de549fbaf4b14b141d63d3c631b27c0


  Other Certificates:
  -------------------

    4. Subject: CN=Krestfield Test Root CA1, OU=Engineering, O=Krestfield Ltd, C=GB
       Issuer : CN=Krestfield Test Root CA1, OU=Engineering, O=Krestfield Ltd, C=GB
       Valid From: Thu Nov 03 09:13:47 GMT 2015
       Valid To  : Wed Nov 03 09:23:47 GMT 2036
       Serial Number: 5de549abaf4b14b141d63d3c631bde54
```

The first section displays the Current Signing Certificates. This is based on the selected signing key and displays the complete path. The second sections displays all other certificates which are stored but not included in the current signing path

Generating AES Keys

Choose option 9 to generate an AES key. Enter the key size and the key label as follows:

```
Enter AES Key Size (128, 192 or 256): 256

Enter a unique label for this key: key10

AES key generated OK
```

Once the key has been created it can be used to encrypt and decrypt data via the client, where the label set above must be specified to select this key

Displaying and Deleting AES Keys

AES keys can be displayed and deleted by choosing options 10 and 11

```
Current AES Keys:
-----------------------------
     #    Key Size   Date Created         Label
     ---  --------   -------------------  --------
       1  128bits    11-3-2017 17:16:45   key2
       2  256bits    13-3-2017 21:33:00   testkey5
       3  192bits    11-3-2017 17:14:52   key1
       4  256bits    19-3-2017 09:20:26   key10
       5  256bits    11-3-2017 17:17:05   key3
```

Key details will be shown including the key size, date created and associated label

Translate KeyStore Objects

If you wish to translate keystore objects from one token password to another. For example, when refreshing passwords for software tokens or if an HSM's objects have been translated to another operator cardset, choose option 12

You will be prompted to choose the channel and then enter the current token password

```
Please enter the current token password

Enter Password:
Retype Password:
```

Then enter the new token password. This is the new password or new operator cardset passphrase:

```
Please enter the NEW token password

Enter Password:
Retype Password:
```

```
Objects have been translated to the new password successfully
A backup was made of the original objects and stored in the SIGN keystore folder
```

The objects will be translated and re-encrypted under the new password. The pre-translated objects will be backed up to a timestamped folder within the keystore directory e.g. `/20181101_1015_BACKUP`

Display KeyStore Objects

To display all the objects stored within a channel, select option 13, then enter the channel

All objects details will be displayed indicating what type of object they are (i.e. private key, certificate etc), the ID and filename e.g.

```
Current Objects:
----------------
Object ID: 15fbc93ade39910
Created  : Tue Nov 14 22:08:35 GMT 2017
Type     : CERTIFICATE
  Subject: CN=Krestfield CA, OU=PKI Services, O=Krestfield Ltd, C=GB
  Issuer : CN=Krestfield Root, OU=PKI Services, O=Krestfield Ltd, C=GB
  Serial Number: 380000000e3308b4434ca3142100000000000e
Filename : 15fbc93ade39910.cer

Object ID: 15fbc93956f0472
Created  : Tue Nov 14 22:08:29 GMT 2017
Type     : CERTIFICATE
  Subject: CN=Krestfield Root, OU=PKI Services, O=Krestfield Ltd, C=GB
  Issuer : CN=Krestfield Root, OU=PKI Services, O=Krestfield Ltd, C=GB
  Serial Number: 5f0609d62d60709e45c1051774a13021
Filename : 15fbc93956f0472.cer

Object ID: PRVK:CC035985F170B51460A3B659523A8D757AD0CBCD
Created  : Thu Apr 05 22:03:12 BST 2018
Type     : PRIVATE KEY
Filename : PRVK_CC035985F170B51460A3B659523A8D757AD0CBCD.priv

Object ID: PUBK:13996A889E52A844660D083D631EE0F30405C576
Created  : Tue Nov 14 22:08:35 GMT 2017
Type     : CERTIFICATE
  Subject: CN=SSAS Cert, O=Krestfield, C=GB
  Issuer : CN=Krestfield CA, OU=PKI Services, O=Krestfield Ltd, C=GB
  Serial Number: 450000000716edae60376f2200000000000007
Filename : PUBK_13996A889E52A844660D083D631EE0F30405C576.cer
```

Set Authentication Code

To set the authentication code which is used to encrypt traffic between the client and server, choose option 14

Enter the master password, followed by the authentication code:

```
You will now be asked to enter the Master Password
Followed by the Authentication Code Password
```

```
  Master Password
  ---------------

  This is the password which is used to start the server
  It is used to encrypt other passwords (Token Passwords etc). It is not stored

 Enter Password:
Retype Password:

  Enter the Authentication Code
  -----------------------------

  This is a password used to secure traffic from the client to the server
  Once this has been set, the client must provide this same password to the EzSign client

 Enter Password:
Retype Password:
```

You will now be prompted whether to set this as the server code only (i.e. securing comms between the client and the server when sending messages), as the server control code only (i.e. securing comms between the client utils scripts and the control server) or both:

```
  Do you want to set this password as the:

  1. Server Authentication Code
  2. Server Control Authentication Code
  3. Both
  4. Cancel

Enter Choice: 1
  A backup of the original has been saved to ..\config\config.properties.2018.11.05.12.23
```

The server configuration will be updated and a backup made

EzSign Control

The Control utility connects to the running server on the specified control port and enables the live monitoring of the server status (running or stopped), the pausing and resuming of the server and the ability to alter the logging level whilst the server is running

Run the `ezsign-server-ctrl` script to start the EzSign Control utility

The script requires two parameters: the IP Address and Control Port of the server e.g:

```
ezsign-server-ctrl 127.0.0.1 5657
```

These are the values set by the following properties in the server properties file:

```
server.ctrl.bindIpAddress

server.ctrl.port
```

If an Authentication Code is being used on the server's control interface, this should also be provided at startup via the authCode parameter e.g.

```
ezsign-server-ctrl 127.0.0.1 5657 authCode=1f84-66c2-29f5-a60b
```

In this case the Authentication Code is `1f84-66c2-29f5-a60b`

Starting the utility in this way will display the menu, enabling the user to select the option required

This script may also be called with an action supplied to carry out the operation with no further interaction required e.g.

To obtain the running status of the server:

```
ezsign-server-ctrl 127.0.0.1 5657 status
```

To pause the server:

```
ezsign-server-ctrl 127.0.0.1 5657 pause
```

To resume the server

```
ezsign-server-ctrl 127.0.0.1 5657 resume
```

To shut down the server process

```
ezsign-server-ctrl 127.0.0.1 5657 stopdaemon
```

To change the log level

```
ezsign-server-ctrl 127.0.0.1 5657 loglevel=3
```

Support

All questions, queries around the API described within this document should be directed to Krestfield Support at the following email address:

```
support@krestfield.com
```

Or visit our site here:

```
https://www.krestfield.com
```