Krestfield PKCloud

Servers

version 2.0

Overview

A Server is an instance processing messages from clients to perform operations such as sign, verify, encrypt etc.

It listens on the ports configured in the server settings and makes use of the Channel, KeyStore, Proxy and Logging settings configured

Multiple Servers are able to run simultaneously listening on different ports which allows for load balanced and redundant setups

Each server configured will host all channels configured i.e. all channels are available via all servers

Configuration

The items available for configuration are:

Client API

This is the client API that the server will listen on. The Server supports the following interfaces:

- **KrestfieldClient**
  - This interface requires client applications to use the Krestfield Client component to make calls to the server. The Client Component is a simple, thin client which exposes a simple interface enabling applications to make calls to the Server. It is available for Java and .NET applications. The client can be downloaded from the Downloads section

  - The communications between the client and the server can be secured with an Auth Code. This code is used to authenticate the client and secure messages between the client and server (traffic is encrypted with an AES-256 key derived from the Auth Code). To configure the Auth Code, select the **Secure Messages with Auth Code** check box and enter the Auth Code value (passphrase). The more complex this passphrase the more secure the communications will be. Note: The client must specify this same Auth Code when making calls to the server

- **RESTful**
  - This exposes a RESTful web service which may be called by constructing and posting JSON messages over http. Any client that can construct these messages may use this interface

- The communications between the client and the server can be secured with TLS by selecting the **Use TLS** check box. When this is selected, the *pkrest.jks* java key store must be configured as per the instructions in the Installation and Configuration guide (REST API Setup section)

Server API Interface

This is the IP address and port the server will listen on for client messages

- **IP Address**
    - The IP Address (interface) the server will bind to. If 'All' is selected, the server binds to all interfaces
- **Port**
    - The port the Server will listen on for messages from the clients
- **Client IP List**
    - This is a white-list of permitted client IP Addresses, indicating what clients may connect to the server. The list must be comma separated e.g. 127.0.0.1,196.254.16.24. If left empty, any client may connect

Server Control Interface

The control interface allows the PKCloud server to monitor the EzSign instance, detecting whether it is running or stopped etc. As well as sending stop and change log level messages

This interface may also be utilised via the command line tools contained in the ./pkcloud/ezsign/bin directory. But unless there is a requirement to monitor the individual running servers, start/stop manually or update log settings via scripts, these settings should remain at their defaults

- **IP Address**
    - The IP Address (interface) the control server will bind to. If 'All' is selected, the control server binds to all interfaces. Defaults to 127.0.0.1
- **Control Port**
    - The port the control server will listen on
- **Client IP List**
    - This is a list of permitted client IP Addresses indicating what clients may call the control interface. The list must be comma separated e.g. 127.0.0.1,196.254.16.24. If left empty, the default of 127.0.0.1 will be set

Thread Pool

When an EzSign Server instance starts it creates a number of threads for parallel processing of requests. By default this is set to 5 but can be reduced to 1 for low volumes (tens of transactions per second) or higher for larger volumes. More threads will consume more resources

Comms Settings

Timeout and security settings between the client and server can be set here

- **Timeout if Busy**
  - When an EzSign Server processes messages it passes the message to the next free thread. If all threads are busy this timeout determines how long to wait until a thread is free before returning a failure
- **Message Timeout**
  - When the client sends a message to the EzSign Server, if the server is busy there will be a delay until the message is processed. This setting determines how long to wait before returning a failure to the client