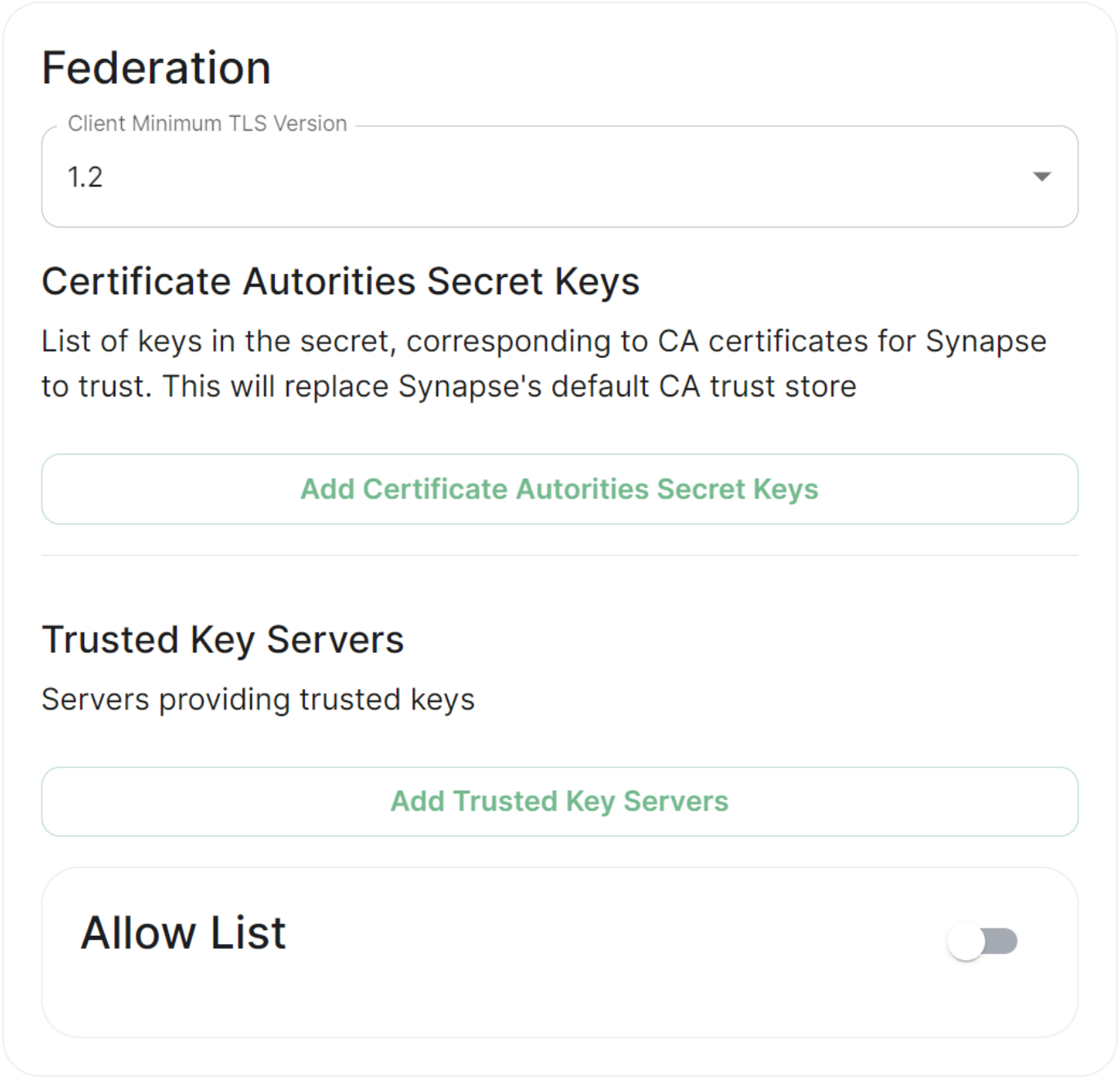
Synapse Section: Federation

Detailed information on configuring homeserver Federation including Trusted Key Servers.

Federation is the process by which users on different servers can participate in the same room. For this to work, all servers participating in a room must be able to talk to each other.

When Federation is `Open`, you will not need to configure anything further, however to privately federate you will need to make use of the `Federation` section found under `Advanced`.



How do I turn Federation On / Off?

How Federation is enabled is automatic based on how you configure it within this Federation section.

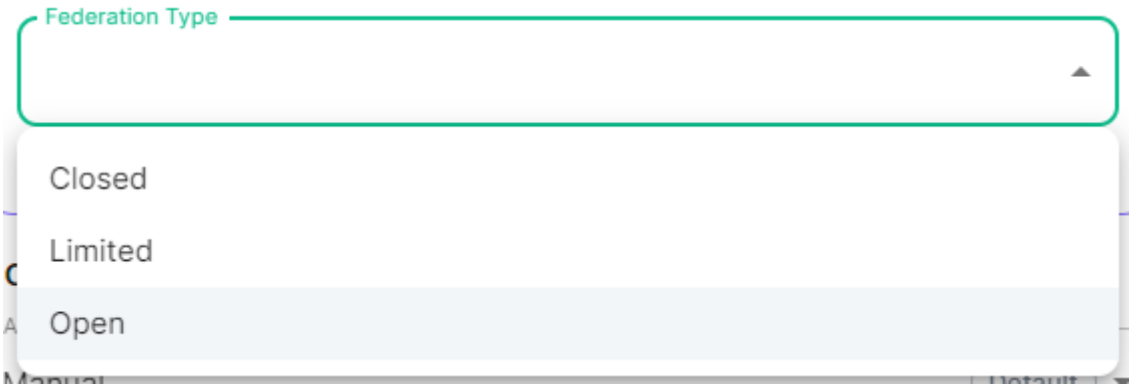
By default Federation is enabled, to close Federation simply enable the Allow List without adding any allowed servers.

Federation Profile

At the top of the Synapse Section you can configure a Federation Type. This Profile section specifically configures the performance profile of your deployed homeserver.

As such, setting this to `Open` will automatically configure Synapse Workers for Federation Endpoints to better support an openly federating server.

This should not be confused with the Federation section detailed in this document.



> Previous setups may have used the Synapse Additional config. Configuration of Federation settings via Additional Config, that are in conflict with any set via the UI, will not override the UI set values. As such, we do not advise including them or any related settings within the Additional Config as they are of increased risk to causing issues with your deployment and are not supported.

Client Minimum TLS Version

`federation_client_minimum_tls_version`

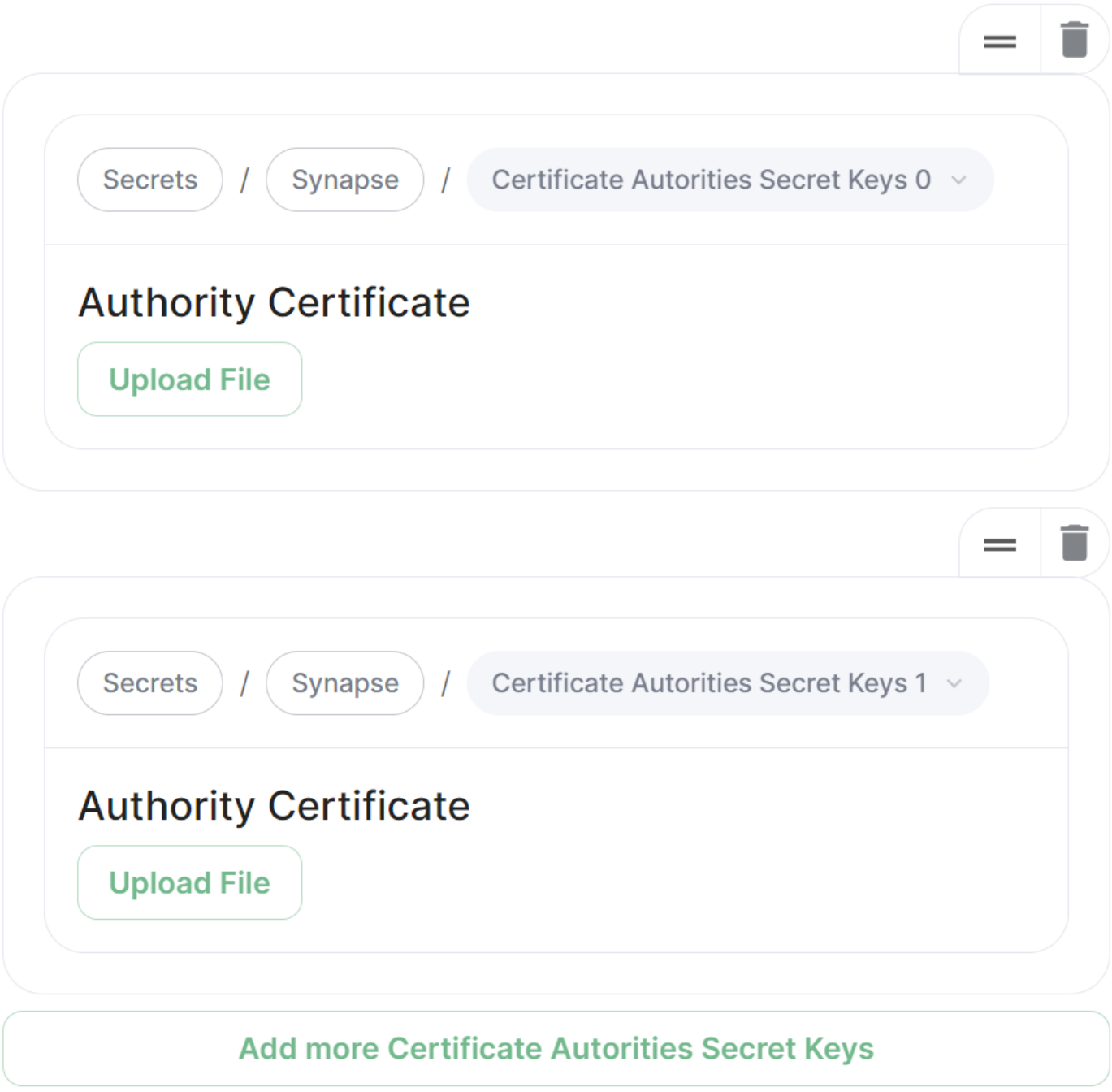
Allows you to choose the minimum TLS version that will be used for outbound federation requests. Defaults to "1.2". Configurable to "1.2" or "1.3".

> Setting this value higher than "1.2" will prevent federation to most of the public Matrix network: only configure it to "1.3" if you have an entirely private federation setup and you can ensure TLS 1.3 support.

Certificate Autorities Secret Keys

Certificate Autorities Secret Keys

List of keys in the secret, corresponding to CA certificates for Synapse to trust. This will replace Synapse's default CA trust store



Configure when you are federating between homeservers' whose certificates are signed by different Certificate Authorities, click the `Add Certificate Authorities Secret Keys` / `Add More Certificate Authorities Secret Keys` button to reveal the option to upload your CA certificate.

> Uploaded certificates should be PEM encoded and include the full chain of intermediate CAs and the root CA. You can simply concatenate these files prior to uploading.

Trusted Key Servers

`trusted_key_servers`

Trusted Key Servers

Servers providing trusted keys

Server Name

A federated server name. This is not necessarily the domain name the server is available at, it is the server name in Matrix IDs and where either SRV records are created or where the WellKnownDelegation is hosted.

Verify Keys

Key ID

A key id to verify

Public Key

A public key to verify

Add more Verify Keys

Add more Trusted Key Servers

Used to specify the trusted servers to download signing keys from. When synapse needs to fetch a signing key, each server is tried in parallel. Normally, the connection to the key server is validated via TLS certificates. Verify keys provide additional security by making synapse check that the response is signed by that key.

Click `Add Trusted Key Servers` / `Add More Trusted Key Servers` to add a new key server, then provide the homeservers' federated server name, i.e. the base domain of the homeserver you with to federate with. Under `Verify Keys` for the server, you will need to provide it's `Key ID` and `Public Key`.

Getting a Homeservers' `Key ID` and `Public Key` from your browser

Simply access the Synapse endpoint `GET /_matrix/key/v2/server`. You must use the domain where your Synapse is exposed, this might be different than the domain you have in your Matrix IDs. For example `https://matrix.yourcomapany.com/_matrix/key/v2/server`.

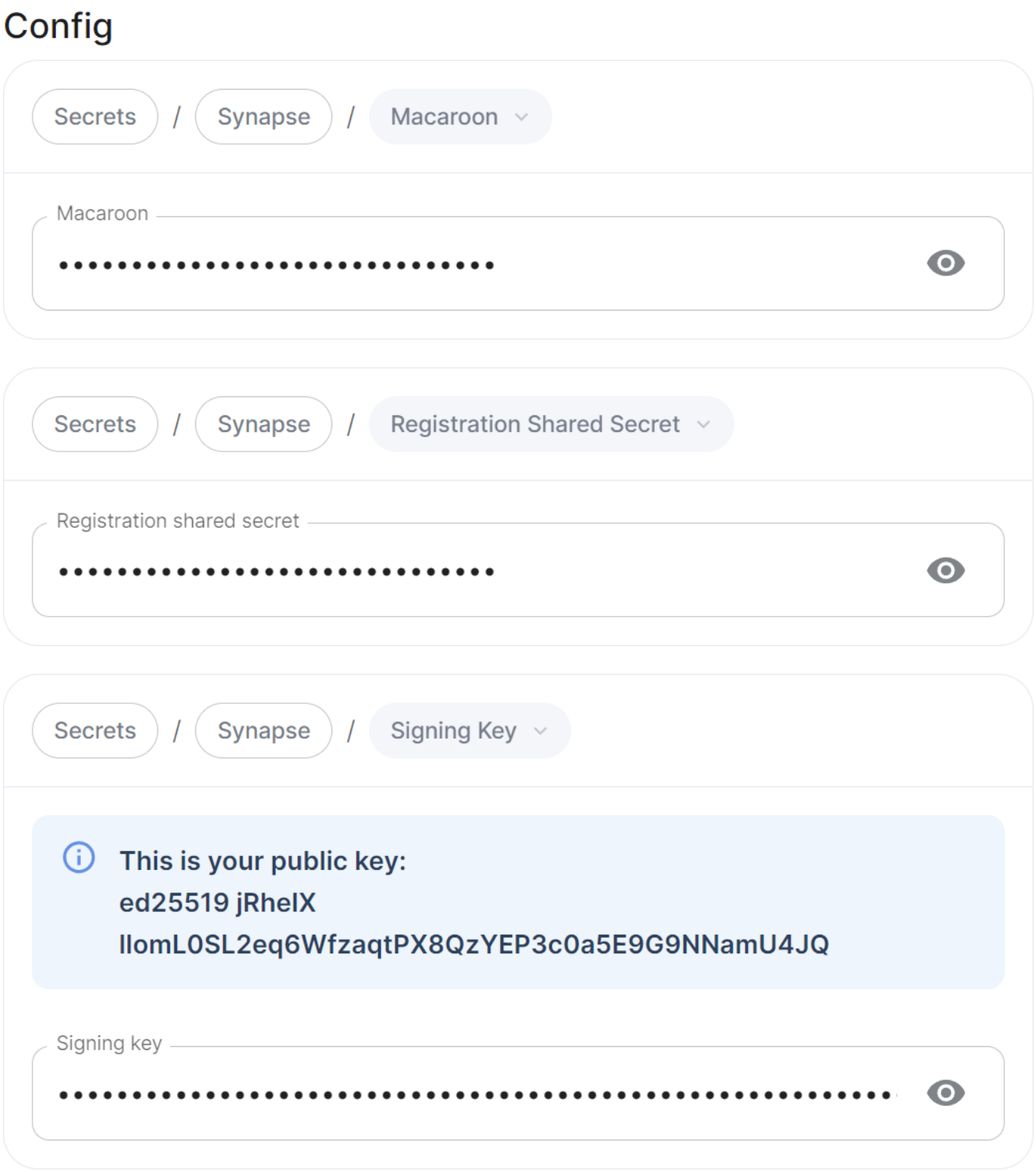
For the element.io homeserver, https://element.ems.host/_matrix/key/v2/server returns


```
{
  "old_verify_keys": {},
  "server_name": "element.io",
  "signatures": {
    "element.io": {
      "ed25519:DnK8xk":
"oOgEpir32XvnuMXQs+GvB6nOuIWgYathJ8kbzDhh9TT/BVSEH116Kk9NYUVPeXHJO0HhzBeTjmAiuUTVFS8nCg"
    }
  },
  "valid_until_ts": 1715307962481,
  "verify_keys": {
    "ed25519:DnK8xk": {
      "key": "EgdGx+0oy/9IX5k7tCobr0JoiwMvmmQ8sDOVlZODh/o"
    }
  }
}
```


Under `verify_keys`, `ed25519:DnK8xk` is the Key ID and `EgdGx+0oy/9IX5k7tCobr0JoiwMvmmQ8sDOVlZODh/o` is the Public Key.

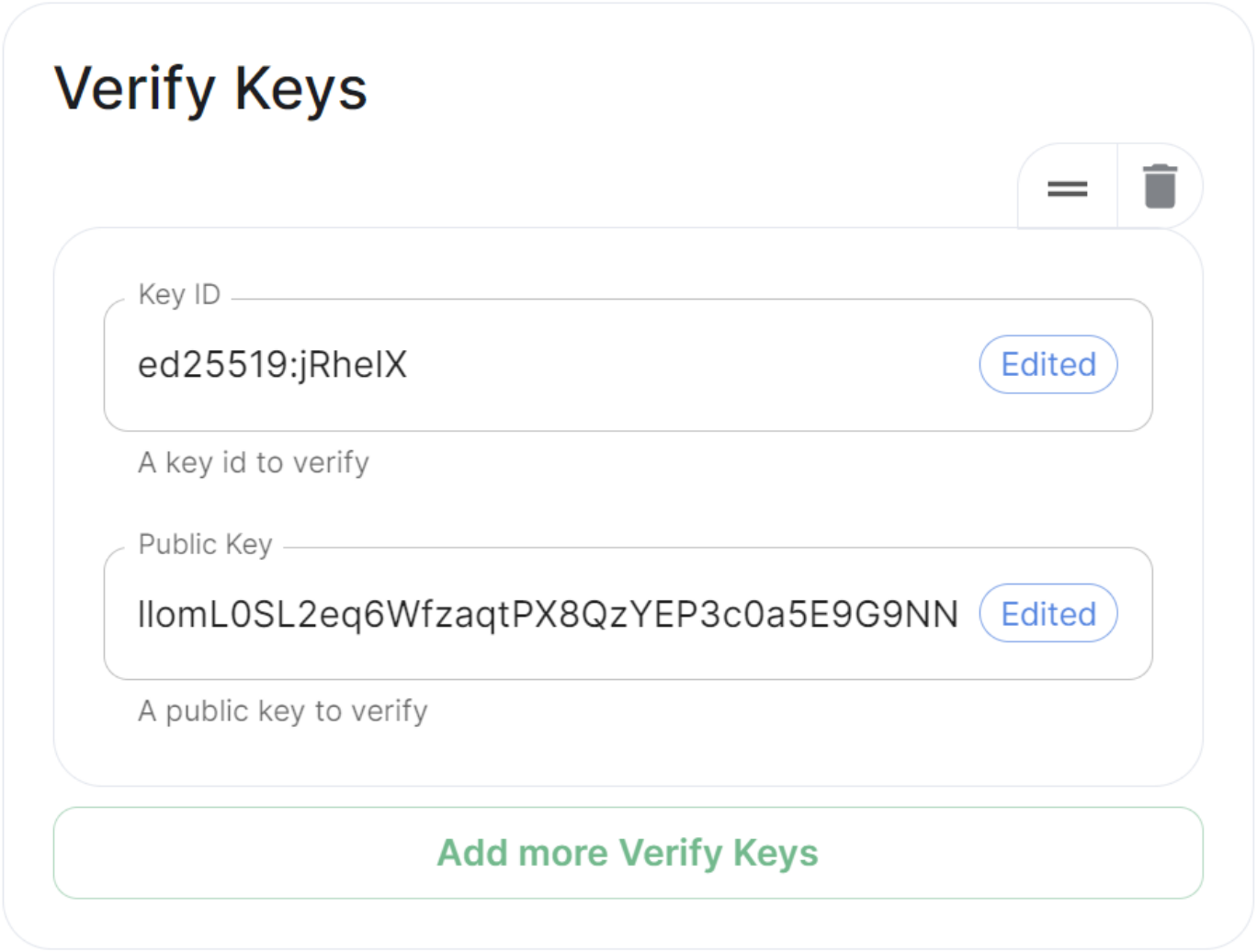
Getting an On-Premise Homeservers' `Key ID` and `Public Key` via the Installer

You can retrieve the `Public Key` of an On-Premise Homeserver by re-running the installer on the host, then navigating to the `Synapse` section. Under `Advanced`, `Config` you will be presented with the homeservers' Public Key in a blue box.



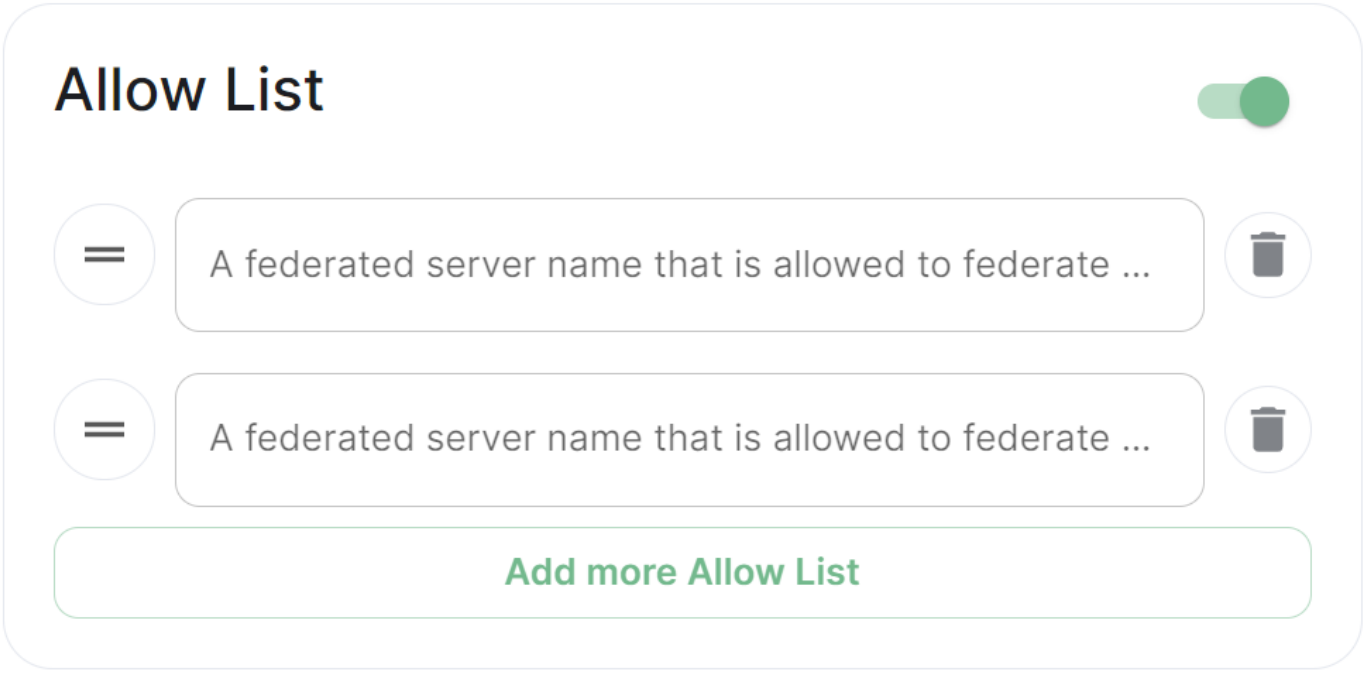
Copy the entire string, taking the example above, it would be `ed25519 jRheIX llomL0SL2eq6WfzaqtPX8QzYEP3c0a5E9G9NNamU4JQ`. From this string, you can derive the `Key ID` and `Public Key` required when you wish to add this homeserver to another homeservers' Federation Trusted Key Servers.

1. The `Key ID` is the first two sections joined with a `:`, so `ed25519:jRheIX`
2. The `Public Key` is the remainder of the string, so `llomL0SL2eq6WfzaqtPX8QzYEP3c0a5E9G9NNamU4JQ`



Allow List

`federation_domain_whitelist`



Use the Allow List to restrict federation to the given whitelist of domains, if not specified, the default is to whitelist everything. Simply provide the homeservers' federated server name, i.e. the base domain of the homeservers' you with to federate with.

> We recommend also firewalling your federation listener to limit inbound federation traffic as early as possible, rather than relying purely on this application-layer restriction.

> This does not stop a server from joining rooms that servers not on the whitelist are in. As such, this option is really only useful to establish a "private federation", where a group of servers all whitelist each other and have the same whitelist.

Please also note that by default an `ip_range_blacklist` is configured to block all private IP address ranges. If your servers require communicating on any of the below ranges, you will need to configure `ip_range_whitelist`. See Allowing Private Federation via `ip_range_whitelist` for information on configuring this.

Revision #8
Created 2024-11-06 10:22:14 UTC by Kieran Mitchell Lane
Updated 2025-05-28 11:25:49 UTC by Kieran Mitchell Lane