Technical Reference Document

AXIS One Click Cloud Connection Server 2.33.0

AXIS O3C Server Reference.

Windows and Linux Versions

Last updated: January 20, 2023
Rev: 1

About this Document

This document lists the functionality included in AXIS One Click Cloud Connection (O3C) and explains how the O3C Server can be integrated and used in containing applications. Read the License agreement carefully before proceeding. The license file is located in the release archive.

Intellectual Property Rights

Axis Communications AB has intellectual property rights relating to technology embodied in the product described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the patents listed at http://www.axis.com/global/en/patent and one or more additional patents or pending patent applications in the US and other countries.

Legal Considerations

Video and audio surveillance can be regulated by laws that vary from country to country. Check the laws in your local region before using this product for surveillance purposes.

Liability

Every care has been taken in the preparation of this document. Please inform your local Axis office of any inaccuracies or omissions. Axis Communications AB cannot be held responsible for any technical or typographical errors and reserves the right to make changes to the product and manuals without prior notice. Axis Communications AB makes no warranty of any kind with regard to the material contained within this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Axis Communications AB shall not be liable nor responsible for incidental or consequential damages in connection with the furnishing, performance or use of this material. This product is only to be used for its intended purpose.

Trademark Acknowledgments

AXIS COMMUNICATIONS, AXIS, ETRAX, ARTPEC and VAPIX are registered trademarks or trademark applications of Axis Communications AB in various jurisdictions. All other company names and products are trademarks or registered trademarks of their respective companies. Acrobat, Adobe, Apache, Debian, Ethernet, Internet Explorer, EvoStream, Iomega, LaCie, Linux, Macintosh, Microsoft, Mozilla, MySQL, Red Hat, UNIX, Windows, WWW, Wi-Fi are registered trademarks of the respective holders. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. UPnP™ is a certification mark of the UPnP™ Implementers Corporation. WPA is a mark of the Wi-Fi Alliance.

Support

Should you require any technical assistance, please contact Axis support at http://www.axis.com.

Table of Contents

1 Introduction

This documentation lists the functionality included in AXIS One Click Cloud Connection (O3C) Server and explains how the server can be integrated and used in containing applications.

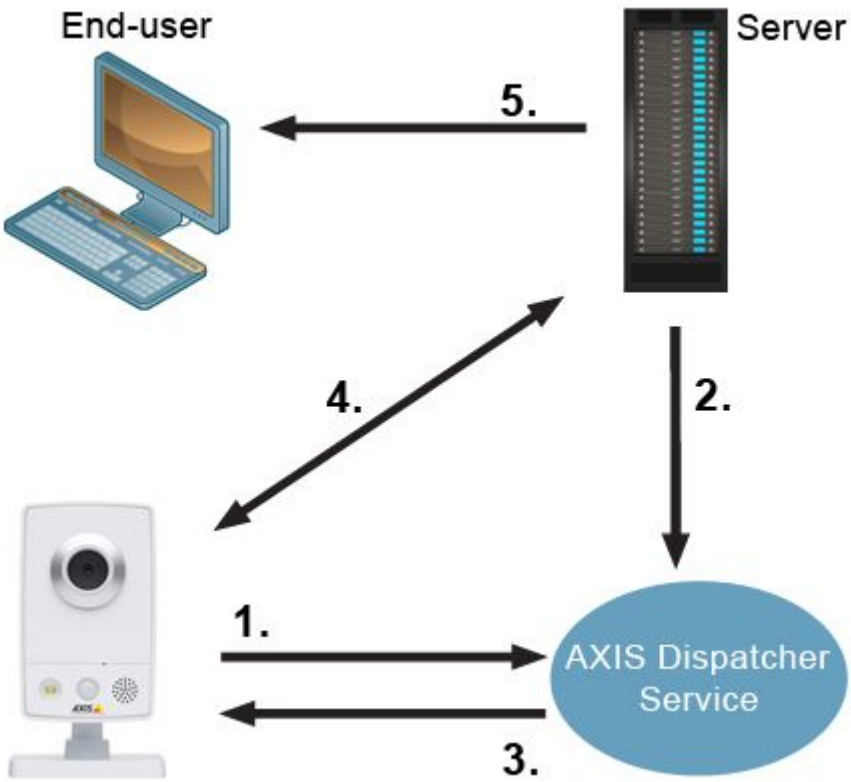
AXIS One Click Cloud Connection allows easy and secure access to Axis devices. The devices contain a special client software (the O3C Client) which connects to the O3C Server on demand. Once connected, API requests can be sent to the device through the server.

With AXIS One Click Cloud Connection, systems can be built that do not require configuration of the remote device connection. Integrating AXIS One Click Cloud Connection in your system makes it easy for end users to add devices to a monitoring system regardless of the Internet service provider, routers or firewall settings. The solution can be used both for web based surveillance systems offering viewing of live and stored/recorded video from any computer on the Internet or for a traditional computer based software with easy handling of remote devices.

The end user registers the devices in your software using a special Owner Authentication Key (OAK) delivered with the device and then presses the control button on the device. The device automatically connects to the O3C Dispatcher service. From there, it is possible for your software to ask the O3C Dispatcher service to redirect the device to connect to your O3C Server.

Typical Setup

Below is an illustration of a setup where the O3C Server is put into use. Note that in the illustration below, "Server" represents a partner's server software which includes AXIS O3C Server.



Steps:

1. Make sure that the device is up and running, then press and hold the device's control button until the device's status LED flashes green. The device connects to the O3C Dispatcher service.
2. Register the device at an O3C Server instance via `admin/dispatch.cgi?action=register`. The O3C Server sends the O3C Dispatcher service a request to let the device redirect and connect to the O3C Server.
3. The dispatcher service reconfigures the device to be connected to the O3C Server.
4. The device connects to the server through an encrypted tunnel.
5. The end user can retrieve images and other device data through the O3C Server.

2 Installation and Configuration

AXIS O3C Server 2.33.0 is released for the 64-bit versions of the following operating systems:

- Microsoft Windows Server 2019
- Debian GNU/Linux 10
- Debian GNU/Linux 11

Sections 2.1 and 2.2 describe examples on how to do new installations of O3C Server on Linux and Windows respectively. Section 2.3 describes an example on how to do an upgrade from the previous O3C Server version.

Warning

During the installation, a certificate authority will be created, including cryptographic keys. It is extremely important not to lose your certificate authority files, nor the password to its encrypted key file.

Although all configuration parameters are briefly described in the configuration example file, parameters in need for extra explanation are covered in Section 2.4.

2.1 Installation on Debian GNU/Linux 10 and 11

The release archives are named:

- `o3c-server-2.33.0-debian10-x86_64.tar.gz`
- `o3c-server-2.33.0-debian11-x86_64.tar.gz`

2.1.1 Archive contents

The archive contains the following files:

- `o3c-server`: The O3C Server application.
- `o3c-server.conf.example`: O3C Server example configuration.
- `o3c-server.service`: Example service definition for systemd
- `dispatcher_ca_bundle.pem`: CA certificate chain for communication with the dispatcher (for registration requests).
- `dispatcher_stclient_ca.pem`: Legacy CA certificate that signs the device certificates. This makes the server trust devices when they have been dispatched.
- `dispatcher_client_ca_2.pem`: New CA certificate that signs the device certificates. This makes the server trust devices when they have been dispatched.
- `dispatcher_client_ca_bundle.pem`: Combination of dispatcher_stclient_ca.pem and dispatcher_client_ca_2.pem.
- `pkitool`: Basic tool for certificate generation. This tool can generate a limited PKI with a root CA for signing server certificates.
- `LICENSE`: Axis Communications AB's proprietary license and third-party open-source library licenses.
- `RELEASENOTES`

2.1.2 Installation example

A typical installation of O3C Server involves setting up a certificate authority structure, issuing a server certificate, creating the server configuration and loading the service definition into systemd.

Note

The number of file descriptors that the O3C server is allowed to utilize can be configured by editing the value `LimitNOFILE` in `o3c-server.service`. This value should be tuned to the use case of the O3C server; the number of connected devices, the number of continuous ongoing requests and any limitations imposed by the operating system affects this limit.

1. Install the runtime dependency packages.

   Debian GNU/Linux 10:

   - `libglib2.0-0`
   - `libssl1.1`
   - `libevent-2.1-6`
   - `libevent-openssl-2.1-6`

   Debian GNU/Linux 11:

   - `libglib2.0-0`
   - `libssl1.1`
   - `libevent-2.1-7`
   - `libevent-openssl-2.1-7`

2. Extract the release archive to `/opt/o3c-server/`. This directory should now contain what is listed in Section 2.1.1.

3. Setup a provider certificate authority. The `pkitool` with command `setup-provider-ca` can be used for this purpose. See also 8.2.

4. Issue a server certificate. The `pkitool` with command `issue-server-cert` can be used for this purpose. See also 8.2.

5. Copy the CA certificate and the issued server certificate to `/opt/o3c-server/`, and **move** the CA directory to a secure location outside your production environment.

6. Modify the O3C Server example configuration and save it to `/opt/o3c-server.conf`. The following parameters must be set:

   - `stserverid`
   - `cert_file`
   - `provider_ca`
   - `provider_name`

7. Load the systemd definition by:

   (a) Copy `o3c-server.service` to /etc/systemd/system.

   (b) Calling `systemctl enable o3c-server`

8. Start the service by calling `systemctl start o3c-server`

2.1.3 Multiple instances

Multiple instances of the O3C Server can be configured to run on the same host. Follow the instructions in Section 2.1.2, and do the following:

1. Create copies of `o3c-server.conf`, and namespace them as `o3c-server-<number>.conf` or according to a similar convention by your own choice.
2. Change the `stserverid`, `listen` and `logfile` parameters in the new configuration files. The server certificate can be shared between the instances, as long as the instances are accessed using the same FQDNs/IPs.
3. Create copies of /etc/systemd/system/o3c-server.service, and namespace them according to the same convention used for the configuration files. Change the `ExecStart` variable of the new service definitions to refer to their respective configuration file.
4. Load the new services by calling `systemctl enable o3c-server-<number>`.

2.2 Installation in Windows Server 2019

2.2.1 Archive contents

The release archive (`o3c-server-2.33.0-win32-x86_64.zip`) contains the following files:

- `o3c-server.exe`: The O3C Server application.
- `o3c-server.conf.example`: O3C Server example configuration.
- `setup_service.ps1`: Utility for Windows SCM registration.
- `dispatcher_ca_bundle.pem`: CA certificate chain for communication with the dispatcher (for registration requests).
- `dispatcher_stclient_ca.pem`: Legacy CA certificate that signs the device certificates. This makes the server trust devices when they have been dispatched.
- `dispatcher_client_ca_2.pem`: New CA certificate that signs the device certificates. This makes the server trust devices when they have been dispatched.
- `dispatcher_client_ca_bundle.pem`: Combination of dispatcher_stclient_ca.pem and dispatcher_client_ca_2.pem.
- `pkitool.exe`: Basic tool for certificate generation. This tool can generate a limited PKI with a root CA for signing server certificates.
- `LICENSE`: Axis Communications AB's proprietary license and third-party open-source library licenses.
- `RELEASENOTES`
- A set of runtime DLL's.

2.2.2 Installation example

A typical installation of O3C Server involves setting up a certificate authority structure, issuing a server certificate, creating the server configuration and registering the service in Windows Service Control Manager.

1. Extract the release archive to `C:\o3c-server\`. This directory should now contain what is listed in Section 2.2.1.
2. Setup a provider certificate authority. The `pkitool` with command `setup-provider-ca` can be used for this purpose. See also 8.2.
3. Issue a server certificate. The `pkitool` with command `issue-server-cert` can be used for this purpose. See also 8.2.
4. Copy the CA certificate and the issued server certificate to `C:\o3c-server\`, and **move** the CA directory to a secure location outside your production environment.
5. Modify the O3C Server example configuration and save it to `C:\o3c-server\o3c-server.conf`. The following parameters must be set:
   - `stserverid`
   - `cert_file`
   - `provider_ca`
   - `provider_name`
6. Register O3C Server as a service in Windows SCM by calling `setup_service.ps1 add -c C:\o3c-server\o3c-server.conf`.
7. Start the service.

2.2.3 Multiple instances

Multiple instances of the O3C Server can be configured to run on the same host. Follow the instructions in Section 2.2.2, and do the following:

1. Create copies of `o3c-server.conf`, and namespace them as `o3c-server-<number>.conf` or according to a similar convention by your own choice.
2. Change the `stserverid`, `listen` and `logfile` parameters in the new configuration files. The server certificate can be shared between the instances, as long as the instances are accessed using the same FQDNs/IPs.
3. Register additional services in Windows SCM by calling `setup_service.ps1 add -i <number> -c C:\o3c-server\o3c-server-<number>.conf`

2.3 Upgrading from previous version

To upgrade from the previous version of O3C Server, it is necessary to set up a new O3C Server instance, with its own IP address and DNS, and thereafter move the devices to the new server. This is briefly described in the steps below.

1. Setup a new O3C Server instance on new system as described in Sections 2.1.2 and 2.2.2, except setup the CA directory structure using the already existing CA certificates. If using `pkitool`, see Section 2.3.1 for a detailed instruction.
2. Move devices from the old O3C Server to the new O3C Server by:
   (a) Append the new server's address to the server lists of the devices, using the param.cgi. The server list is located at root.RemoteService.ServerList, and comma (,) is used as separator. For further reference on the param.cgi, see the VAPIX documentation.
   (b) Move each device using the redirect.cgi of O3C Server, described in Section 3.3. This is to make sure that devices are moved in a controlled fashion.
3. When all devices have successfully been moved to the new server, shut down the old server.
4. Remove the old server's address from the server list, by using the param.cgi as described in Step 2a.
5. Update the provider account's server list on the AXIS Dispatcher by registering a device as usual with the new server list, see Section 4.2

Note

When performing the upgrading procedure described above, always start with a non-production device to verify the new installation.

2.3.1 Using the PKI Tool with existing CA certificates

When extending an existing O3C Server setup, you need to reuse your existing CA certificate in order to allow previously registered devices to connect to the O3C Servers using the new server certificates.
This can be achieved by performing the following steps:

1. Create a new CA directory structure using the PKI Tool: `pkitool setup-provider-ca`.
2. Overwrite the created `stserver_ca.crt` file with your existing certificate.
3. Overwrite the created `stserver_ca.key` file with your existing key.
4. Now issue server certificates as usual with `pkitool issue-server-cert`.

2.4 Configuration

2.4.1 Logging

O3C Server supports the following log methods:

- File, with optional rotation described in Section 2.4.1.1
- Syslog (Linux only)
- Standard error stream (interactive mode)
- HTTP stream (referred to as the debug.cgi), described in Section 3.7

Each log message is prepended with the local time, formatted as YYYY-MM-DD HH:MM:SS.SSSSSS.

2.4.1.1 Log rotation

O3C Server supports two log file rotation disciplines: filesize based and nightly. Rotation is performed when passing midnight local time, or when the maximum allowed file size threshold has been reached. Both metrics are checked every minute. With that said, rotation may occur at most one minute past midnight, as well as some bytes past the threshold.

To enable filesize based rotation, set the `logrotation_filesize` configuration value to the maximum allowed file size measured in bytes. To enable nightly rotation, set the `logrotation_nightly` configuration value to `1`.

3 Administration Interface

The administration interface is used for server administration. The different commands are handled by different paths. The administration interface is accessed either through HTTP (configured with the `listen_admin` setting) or through HTTPS (configured with the `listen_admin_https` setting). Both fields for `listen_admin` or `listen_admin_https` in the o3c-server.conf can be set, or only one of them, but both fields cannot be set empty.

3.1 Status

`http://<listen_admin>/admin/status.cgi`
`https://<listen_admin_https>/admin/status.cgi`

The `status.cgi` returns a status report listing all connected devices and information on all open request connections to each of them (initiated through the O3C Server user interface).

This CGI takes a format argument to present the status report in different formats, as shown in Table 1. If no argument is specified it will produce a plain-text output that is easy to parse and process with stream manipulation utilities. See Section 3.1.1 for details on the plain-text output. Output can also be rendered in JSON format. Attributes and their key names varies slightly between the plain-text- and the JSON formatted output. See Section 3.1.2 for details on the JSON output.

Argument	Description
`format`	Desired format. Set to `json` for JSON output. Anything else will produce plain-text output.

*Table 1: Arguments to */admin/status.cgi*

3.1.1 Plaintext

Device Info	Description
`id`	The id of the client connection on the form `connection_id.mac_address`. `connection_id` is a device internal counter for successful O3C Client/Server connection attempts since restart `mac_address` is the MAC address of the device
`srcaddr`	The IP address and port of the client connection
`accepted`	Indicates if the client has been accepted and is available for use
`v`	Protocol version number
`connected`	The time the client was connected to the server (RFC3339)
`rx`	Estimated number of bytes received by O3C Server from the O3C Client
`tx`	Estimated number of bytes transmitted by O3C Server to the O3C Client

Table 2: Device information

Each open data connection against a device will show up as a separate row in the output below the corresponding device row. These rows will be indented by four white spaces.

Connection Info	Description
`id`	The id of the request connection
`srcaddr`	The IP address and port of the request origin
`request`	The request made to the client
`connected`	The time the request was initiated (RFC3339)
`rx`	Estimated number of bytes received by O3C Server for the request
`tx`	Estimated number of bytes transmitted by O3C Server for the request

Table 3: Connection information

3.1.1.1 Example output with one connected device

```
id=1.00408cfd49b3 srcaddr=127.0.0.1:2 accepted=1 v=2 rx=392654 tx=0 \
   connected=2022-12-16T10:01:02.000000Z

Total number of clients: 1
```

3.1.1.2 Example output with one connected device and two ongoing requests

```
id=1.00408cfd49b3 srcaddr=127.0.0.1:2 accepted=1 v=2 rx=392947 tx=0 \
   connected=2022-12-16T10:01:02.000000Z
    id=85b11b8467015304dfe075411cdfdf54 srcaddr=127.0.0.1:3 \
       request=http://00408cfd49b3/axis-cgi/serverreport.cgi \
       rx=0 tx=207 connected=2022-12-16T10:03:04.000000Z
    id=d509d6679ad843031b5cf7e46d6bf583 srcaddr=127.0.0.1:4 \
       request=http://00408cfd49b3/axis-cgi/serverreport.cgi \
       rx=0 tx=208 connected=2022-12-16T10:03:06.000000Z

Total number of clients: 1
```

3.1.2 JSON

In addition to the plaintext version, the JSON version also provides the uptime and version of the server.

Attribute key name	Attribute type	Description
`clients`	`array`	An array of client objects, described in Table 5
`client_count`	`integer`	The number of currently connected clients
`uptime`	`integer`	Server uptime in seconds
`server_version`	`string`	The server application version

Table 4: JSON top-level object schema

Attribute key name	Attribute type	Description
`serial`	`string`	Client device serial number (MAC)
`count`	`integer`	Device internal counter for successful O3C Client/Server connection attempts since restart
`source`	`string`	The IP address and port of the client connection
`accepted`	`boolean`	Indicates if the client has been accepted and is available for use
`protocol_version`	`integer`	Protocol version number
`connected`	`string`	The time the client was connected to the server (RFC3339)
`rx`	`integer`	Estimated number of bytes received by O3C Server from the O3C Client
`tx`	`integer`	Estimated number of bytes transmitted by O3C Server to the O3C Client
`requests`	`array`	An array of requests forwarded by O3C Server to the O3C Client, described in Table 6

Table 5: JSON client object schema

Attribute key name	Attribute type	Description
`id`	`string`	The id of the request connection
`source`	`string`	The IP address and port of the request origin
`url`	`string`	The request URL as provided to O3C Server
`connected`	`string`	The time the request was initiated (RFC3339)
`rx`	`integer`	Estimated number of bytes received by O3C Server for the request
`tx`	`integer`	Estimated number of bytes transmitted by O3C Server for the request

Table 6: JSON request object schema

3.1.2.1 Example output with one connected device

```
{
  "clients": [
    {
      "serial": "00408C7AC667",
      "count": 1,
      "source": "127.0.0.1:2",
      "accepted": true,
      "protocol_version": 2,
      "connected": "2022-12-16T10:01:02.000000Z",
      "rx": 0,
      "tx": 0,
      "requests": []
    }
  ],
  "client_count": 1,
  "uptime": 2051,
  "server_version": "2.33.0"
}
```

3.1.2.2 Example output with one connected device and one ongoing request


```
{
  "clients": [
    {
      "serial": "ACCC8E7BF64D",
      "count": 1,
      "source": "127.0.0.1:2",
      "accepted": true,
      "protocol_version": 2,
      "connected": "2022-12-16T10:01:02.000000Z",
      "rx": 3111645,
      "tx": 0,
      "requests": [
        {
          "id": "79426bfdf213963e50ceba4c192f9f1b",
          "source": "127.0.0.1:3",
          "url": "http://ACCC8E7BF64D/axis-cgi/mjpg/video.cgi",
          "connected": "2022-12-16T10:03:04.000000Z",
          "rx": 0,
          "tx": 757018
        }
      ]
    }
  ],
  "client_count": 1,
  "uptime": 2051,
  "server_version": "2.33.0"
}
```


3.2 Metrics

`http://<listen_admin>/admin/metrics.cgi`
`https://<listen_admin_https>/admin/metrics.cgi`

The `metrics.cgi` returns a collection of values in JSON format representing the state of the server.

This CGI takes no arguments.

3.2.1 JSON

Attribute key name	Attribute type	Description
`current_clients`	`integer`	The number of currently connected clients
`total_clients`	`integer`	Total aggregated client connections made
`total_client_timeouts`	`integer`	Total aggregated client time-outs
`current_proxy_requests`	`integer`	Currently active or pending proxy requests
`total_proxy_requests`	`integer`	Total aggregated proxy requests made
`total_proxy_responses`	`integer`	Total aggregated proxy responses made
`total_proxy_rx`	`integer`	Total aggregated bytes received to proxy
`total_proxy_tx`	`integer`	Total aggregated bytes transmitted from proxy
`crl_load_failures`	`integer`	Total number of failures to load a valid revocation list
`crl_valid_until`	`string`	The date and time the loaded revocation list expires
`server_uptime`	`integer`	Server uptime in seconds
`configuration_reloads`	`integer`	The number of times the server configuration has been reloaded
`configuration_uptime`	`integer`	The time in seconds since last server configuration reload
`server_version`	`string`	The server application version

Table 7: JSON top-level object schema

3.2.1.1 Example output with one connected device but no activity

```
{
  "current_clients": 1,
  "total_clients": 1,
  "total_client_timeouts": 0,
  "current_proxy_requests": 0,
  "total_proxy_requests": 0,
  "total_proxy_responses": 0,
  "total_proxy_rx": 0,
  "total_proxy_tx": 0,
  "crl_load_failures": 0,
  "crl_valid_until": "Wed Nov 23 07:21:25 2022 GMT",
  "server_uptime": 123,
  "configuration_reloads": 0,
  "configuration_uptime": 123,
  "server_version": "2.33.0"
}
```

3.2.1.2 Example output with one connected device and one ongoing request

```
{
  "current_clients": 1,
  "total_clients": 1,
  "total_client_timeouts": 0,
  "current_proxy_requests": 1,
  "total_proxy_requests": 1,
  "total_proxy_responses": 1,
```

```
    "total_proxy_rx": 1337,
    "total_proxy_tx": 1337,
    "crl_load_failures": 0,
    "crl_valid_until": "Wed Nov 23 07:21:25 2022 GMT",
    "server_uptime": 321,
    "configuration_reloads": 0,
    "configuration_uptime": 321,
    "server_version": "2.33.0"
  }
```


3.3 Redirect

`http://<listen_admin>/admin/redirect.cgi`
`https://<listen_admin_https>/admin/redirect.cgi`

The `redirect.cgi` can be used to redirect clients to another server instance. This can be used for balancing clients between multiple server installations.

Note

To be able to redirect a client to another server, the server needs to be listed in the clients server list, given by the `root.RemoteService.ServerList` device parameter. If the server is not listed in the client server list, an "OK" message will be shown indicating that the request was received by the client, but the client will not be redirected.

The CGI takes two arguments as shown in table 8.

Argument	Description
`client`	The client to be redirected.
`server`	The server `listen_client` interface to redirect to in `<address:port>` format.

*Table 8: Arguments to */admin/redirect.cgi*

Example:

`http://127.0.0.1:3128/admin/redirect.cgi?client=00408cdc8194&server=127.0.0.1:8081`

3.4 Reconnect

`http://<listen_admin>/admin/reconnect.cgi`
`https://<listen_admin_https>/admin/reconnect.cgi`

The `reconnect.cgi` provides a simple way to reconnect a client. This is done by forcing the server to close the open TCP connection to the client. The client will then try to reconnect to one of the servers given in its server list.

The CGI takes one argument as shown in table 9.

Argument	Description
`client`	The client to be reconnected

*Table 9: Argument to */admin/reconnect.cgi*

Example:

```
http://127.0.0.1:3128/admin/reconnect.cgi?client=00408cdc8194
```

3.5 Restart

```
http://<listen_admin>/admin/restart.cgi
https://<listen_admin_https>/admin/restart.cgi
```

The `restart.cgi` is used to restart a client without going through VAPIX.

The CGI takes one argument as shown in table 10.

Argument	Description
`client`	The client to be restarted

*Table 10: Argument to */admin/restart.cgi*

Example:

```
http://127.0.0.1:3128/admin/restart.cgi?client=00408cdc8194
```

3.6 Shortcut

```
http://<listen_admin>/admin/shortcut.cgi
https://<listen_admin_https>/admin/shortcut.cgi
```

The O3C Server can be used as a proxy to enable public access to a client-side cgi. This is obtained in two steps:

1. As an administrator, register the client-side cgi URL with the O3C Server and acquire a time-limited access token via `shortcut.cgi`
2. As a user, use the access token through the O3C `listen_user_ext` interface to communicate with the client

The `shortcut.cgi` takes three arguments as shown in table 11.

The cgi URL is specified by the argument `url` and should specify client, path and optionally query parameters. The `url` argument must be `base64` encoded. If a query string is included in the `url` argument, it is locked and cannot be altered by the user, but if left unspecified the user is free to alter it. If the path in the `url` argument is left unspecified the user of the shortcut is allowed to specify it. It is also possible to specify which user credentials to use; e.g. `http://user:pass@host/`.

The argument `timeout` limits the lifetime of the shortcut's accessibility in seconds. When accessing the shortcut the timeout will be reset.

It is possible to limit the HTTP methods allowed when accessing the shortcut by specifying the argument `allowmethods` which takes a comma separated list of HTTP methods.

Arguments `url` and `timeout` are required. `allowmethods` will default to a GET.

By setting `X-No-Internal-Authentication` to `1` as a header or a cookie all authentication headers will be removed. This is useful when accessing pages that require no authentication.

Argument	Description
`url`	Full URI to client, base64 encoded
`timeout`	Time in seconds the shortcut should be available. Default: 10
`allowmethods`	Comma separated list of allowed HTTP methods for accessing the shortcut.

*Table 11: Arguments to */admin/shortcut.cgi*

If successful, `shortcut.cgi` will return a reference token to the time-limited shortcut (depicted `id=` in the result).

When accessing the shortcut using cookies, make a call to the root of the `listen_user_ext` interface and set a HTTP cookie named "`o3c-shortcut-id`" to the value of the shortcut token.

When accessing the shortcut using path, use this syntax:

```
http://<listen_user_ext>/o3c-shortcut/<serverid>/<shortcut-token>
```

The `serverid` part is a mandatory identifier, that can be used for redirecting purposes in a load balanced system setup to assure that consequent calls to the same device will go through the same O3C instance every time. Yet mandatory, the actual value of the `serverid` part can be set to any non-empty string.

Example: Create a shortcut

```
curl "http://127.0.0.1:3128/admin/shortcut.cgi? \
  timeout=10& \
  allowmethods=POST,GET& \
  url=$(printf "http://00408C123456/axis-cgi/param.cgi?action=list" | base64 -w0)"

id=ec5bb9447f41dd6bb3a6d01cf2abbe6f
```

Example: Access a shortcut using cookie accessor

```
curl \
  -H "Cookie: o3c-shortcut-id=ec5bb9447f41dd6bb3a6d01cf2abbe6f" \
  "http://example.com:8081/"
```

Example: Access a shortcut using path accessor

```
curl "http://example.com:8081/o3c-shortcut/serverid/ec5bb9447f41dd6bb3a6d01cf2abbe6f"
```

3.7 Debug

```
http://<listen_admin>/admin/debug.cgi
https://<listen_admin_https>/admin/debug.cgi
```

The `debug.cgi` returns a continuous log from the server and the log level presented is defined either in the configuration file of O3C Server or as parameter in the request. For a description of possible arguments see table 12.

Argument	Description
`level`	The debug level *0* = nothing, *1* = error, *2* = warning, *3* = notice, *4* = debug.
`client`	The device which stream should be listened to, on the form `connection_id.mac_address` given by a `status.cgi` call. If left empty all connected devices will be listened to.

*Table 12: Arguments to */admin/debug.cgi*

Example:

```
Method: GET
http://127.0.0.1:3128/admin/debug.cgi

[2022-12-16 13:37:01.000000] [debug] [client.c 174] [user 3a1@127.0.0.1:2] \
   [client 1.accc8e7bf64d@127.0.0.1:2] Sending request command to client for user request
[2022-12-16 13:37:02.000000] [debug] [client.c 176] [user 3a1@127.0.0.1:2] \
   [client 1.accc8e7bf64d@127.0.0.1:2] Request URL: http://accc8e7bf64d/
```

3.8 Dispatch

```
http://<listen_admin>/admin/dispatch.cgi
https://<listen_admin_https>/admin/dispatch.cgi
```

The dispatch.cgi is used to manage devices with the Axis dispatcher service. See section 4 for more information.

3.9 Reload server configuration

`http://<listen_admin>/admin/reload_config.cgi`
`https://<listen_admin_https>/admin/reload_config.cgi`

The `reload_config.cgi` instructs the server to reload its configuration.

This CGI takes no arguments.

Some configuration options should not be changed and applied through a configuration reload as this may lead to unwanted behavior. These include `listen_client`, `keepalive` and `providername`.

Any connected client will remain connected and will be using their established SSH context. They can be reconnected individually, see 3.4.

3.9.1 Output

The `reload_config.cgi` will on success respond with HTTP/1.1 200 OK without response body.

If the configuration change could not be applied an HTTP/1.1 500 error is returned.

Example:

```
HTTP/1.1 500 Server Error
Content-Type: text/plain
Connection: close
Content-Length: 49

Request failed. (Reason: Internal Server Error.)
```

Note

The reload of the configuration on a Debian GNU/Linux system can be accomplished by signal `SIGUSR1`, for example `killall -SIGUSR1 o3c-server`, or by calling systemd, for example `systemctl reload o3c-server`, with appropriate changes in `o3c-server.service`.

3.10 Print server configuration

`http://<listen_admin>/admin/print_config.cgi`
`https://<listen_admin_https>/admin/print_config.cgi`

The `print_config.cgi` will output the current server configuration in a simple key-value listing.

This CGI takes no arguments.

The value of the configuration option `credentials`, if defined in the configuration, is obfuscated and replaced by a single asterisk.

4 The Axis O3C Dispatcher Service

Before a device can be accessed through O3C Server, it has to be dispatched to the server. For this to work properly the device needs to be connected to the AXIS O3C Dispatcher service.

4.1 Client Setup

The device need to be factory reset.

To connect a device to the AXIS O3C Dispatcher service the O3C client need to be enabled. This is accomplished by pressing and holding the control button until the status LED flashes green.

An alternative way to enable the O3C client is by using the device web pages. The configuration of your device using the device web pages depends on the device model and firmware. The user manual for your device can be found on http://www.axis.com.

4.2 Register a Device with the Axis Dispatcher Service

Device is dispatched by sending a HTTP request to `admin/dispatch.cgi` in the O3C Server. Use the following syntax:

```
http://<listen_admin>/admin/dispatch.cgi?action=[&=...]&server=[&=...]
https://<listen_admin_https>/admin/dispatch.cgi?action=[&=...]&server=[&=...]
```

with the following arguments:

Argument	Description
`action`	Use the literal string `register` for this operation.
`user`	Your dispatcher user name provided by Axis.
`pass`	Your dispatcher password provided by Axis.
`mac`	The device's MAC address, e.g. 00408CA03AF0.
`oak`	The device's OAK (owner authentication key), e.g. 5A8A-26D0-52AC, supplied on a document with the device.
`server`	IP address and port of your running service. This value should contain the value of the configuration parameter `listen_client`. If the service is behind a NAT, the value should contain a fully qualified domain name and port of the public address that is reachable from the clients, the corresponding IP address and port or both. To include more than one value, use commas to separate the values.

*Table 13: Register arguments to */admin/dispatch.cgi*

Note

The `dispatch.cgi` argument `server` is mandatory but it can be empty. If `server` is specified, the provider account is updated with the specified server list. If `server` is empty, no update will occur, the previously registered server list is retained for the provider account.

Note

The server list for a provider account, affects all devices registered to that provider account. The action `/dispatch.cgi?action=register` updates the provider accounts server list, not the individual device.

A typical call to the register function looks like this:

```
http://127.0.0.1:3128/admin/dispatch.cgi?action=register&user=example&pass=example
&mac=00408CABCDEF&oak=1234-5678-90AB&server=example.com:8080,example.com:8081
```

You can use any programming language that is able to send HTTP requests. If everything goes well this call will return `HTTP 200`, and the HTTP body does not contain any `error`. If there is a problem, `HTTP 400` will be returned and the response body will contain an error message.

The device should now be configured to connect to your O3C Server and you should be able to communicate with the device through the server.

4.3 Unregister a Device from the Axis Dispatcher Service

The `admin/dispatch.cgi` can be used to unregister devices from the dispatcher. The following parameters need to be specified:

Argument	Description
`action`	Use the literal string `unregister` for this operation.
`user`	Your dispatcher user name provided by Axis.
`pass`	Your dispatcher password provided by Axis.
`mac`	The device's MAC address, e.g. 00408CA03AF0.

*Table 14: Unregister arguments to */admin/dispatch.cgi*

A typical call to the unregister function looks like this:

```
http://127.0.0.1:3128/admin/dispatch.cgi?action=unregister&user=example
&pass=example&mac=00408CABCDEF
```

You can use any programming language that is able to send HTTP requests. If everything goes well this call will return `HTTP 200`, and the HTTP body does not contain any `error` . If there is a problem, `HTTP 400` will be returned and the response body will contain an error message.

Note

After a device has been unregistered it is recommended to factory reset the device. Otherwise it is possible for the device to connect to the O3C Server.

5 User Interface

AXIS O3C Server can be used as a proxy server and transfer requests and data from a connected device's internal web server even if the accessing client is on a different network. This type of traffic is handled by the O3C Server user interface.

The user interface is accessed either through HTTP (configured with the `listen_user` setting) or through HTTPS (configured with the `listen_user_https` setting). Both fields for `listen_user` or `listen_user_https` in the o3c-server.conf can be set, or only one of them, but both fields cannot be set empty.

5.1 Accessing VAPIX® through the O3C Server User Interface

The proxy connection through O3C Server is typically used to access VAPIX® on connected devices. VAPIX® is Axis own open API (Application Programming Interface) using standard protocols enabling integration into a wide range of solutions on different platforms.

Extensive documentation of the VAPIX® API can be found on the Axis web site:

http://www.axis.com

All common HTTP request methods can be used through the user interface (`HEAD`, `GET`, `POST`, `OPTIONS`, `PUT`, `DELETE`, `TRACE` and `PATCH`).

The typical syntax to access a connected device is:

```
<http_scheme>://<mac_address>/<path>
```

where `mac_address` is the MAC address of the device and `path` is the resource requested from the device's internal web server.

Note

In the latest available firmware, the root password will be marked as set (if not already set) after the first internally authenticated request through O3C. See the `'force_authentication_mode'` configuration setting. This effectively disables all VAPIX® requests outside of the internally authorized O3C-server and direct access to the device will require updating the root password. For examples on how to update the root password please refer to pwdgrp.cgi in the VAPIX® documentation.

Note

If requests to a device are made more frequently than they can be handled by the O3C server they are queued on the O3C server. This is shown as ongoing requests for a client in `/admin/status.cgi`. The max number of queued requests is limited by the number of `file descriptors` available to the O3C server. The configuration option `client_dictates_max_conn` will prevent this queue to increase beyond the number of supported concurrent requests as reported by a device.

5.2 Command-Line Access using cURL

cURL is an open source command line tool and library for transferring data with URL syntax. cURL is used in command lines or scripts to transfer data. cURL is built with libcurl, which is a free and easy-to-use client-side URL transfer library.

Use the cURL -x option to set your O3C Server as proxy using your `listen_user` setting, and then you have an easy way via the command-line to access the internal web server of your connected devices.

GET request example:

```
curl -x 127.0.0.1:3128 "http://00408cfd49b3/axis-cgi/param.cgi?action=list
  &group=root.RemoteService"
```

POST request example:

```
curl -x 127.0.0.1:3128 --data "action=list&group=root.RemoteService"
  "http://00408cfd49b3/axis-cgi/param.cgi"
```

Tip

In the examples above, the VAPIX® `axis-cgi/param.cgi` is used to list the device's parameters. In the device parameter list, the O3C related parameters (e.g. the server list) are listed in the `root.RemoteService` group.

Note

cURL and libcurl does only support connections to an HTTP proxy using HTTP. Even though the O3C Server accepts HTTPS connections through the `listen_user_https` interface, this interface cannot be used with cURL due to this limitation.

5.3 Web Client Access using a Web Browser

It is possible to access the web interface of a O3C Server connected Axis device, even if you are not on the same local network as the device.

Specify your `listen_user` server setting as the proxy server address to use.

To access a device use the following format:

```
<http_scheme>://<mac_address>/<path>
```

where `mac_address` is the MAC address of the device and `path` is the resource requested from the device's internal web server.

6 Notification

Using the `notify_urls` configuration setting, the O3C Server can be set up to send notifications. HTTP keep-alive is recommended on these connections.

For more information see `force_accept_clients`, `notify_urls`, and `notification_ca` in the example configuration.

6.1 Server Up

This notification indicates that the server has started. The identity of the server (the O3C Server configuration setting `stserverid`) is passed in a '`X-Stserver-Id`' header.

Example HTTP request:

```
PUT notification.example.com?action=server_up HTTP/1.1
Host: 127.0.0.1
X-Stserver-Id: my_o3c_server
```

The receiver should reply with an `HTTP 200` response if the notification has been successfully received.

6.2 Server Down

This notification indicates that the server stopped. The identity of the server is passed in a '`X-Stserver-Id`' header.

Example HTTP request:

```
DELETE notifications.example.com?action=server_down HTTP/1.1
Host: 127.0.0.1
X-Stserver-Id: my_o3c_server
```

The receiver should reply with an `HTTP 200` response if the notification has been successfully received.

6.3 Client Hello

This notification indicates that a client is attempting to connect to the server. The identity of the server is passed in a '`X-Stserver-Id`' header.

Example HTTP request:

```
HEAD notifications.example.com?action=client_hello&client_id=1.00408C123456 HTTP/1.1
Host: 127.0.0.1
X-Stserver-Id: my_o3c_server
Connection: Keep-Alive
```

The notified end can control how the connecting client should be handled. If the client connection is accepted, the notified end should reply with an `HTTP 204` response.

The receiver can redirect the client with an `HTTP 302` response if it determines that another server should handle the connection.

Example HTTP redirect:

```
HTTP/1.1 302 Found
Location: 10.21.31.41:8080
```

All other responses are considered as errors and will result in the client re-connecting.

Note

To be able to redirect a client to another server, the server needs to be listed in the clients server list, given by the `root.RemoteService.ServerList` device parameter.

6.4 Client Connect

This notification indicates that the client is connected and accessible. The identity of the server is passed in a '`X-Stserver-Id`' header.

Example HTTP request:

```
PUT notifications.example.com?action=client_connect&client_id=1.00408C123456&
client_srcaddr=10.22.33.44 HTTP/1.1
Host: 127.0.0.1
X-Stserver-Id: my_o3c_server
Connection: Keep-Alive
```

The receiver should reply with an `HTTP 200` response to acknowledge the notification.

The receiver can redirect the client with an `HTTP 302` response if it determines that another server should handle the connection.

Example HTTP redirect:

```
HTTP/1.1 302 Found
Location: 10.21.31.41:8080
```

Note

To be able to redirect a client to another server, the server needs to be listed in the clients server list, given by the `root.RemoteService.ServerList` device parameter.

6.5 Client Disconnect

This notification indicates that the client is no longer connected and accessible. The identity of the server is passed in a 'X-Stserver-Id' header.

Example HTTP request:

```
DELETE notifications.example.com?action=client_disconnect&client_id=1.00408C123456
HTTP/1.1
Host: 127.0.0.1
X-Stserver-Id: my_o3c_server
```

The receiver should reply with an HTTP 200 response if the notification has been successfully received.

7 Troubleshooting

This section covers some of the basic errors that might occur when using the AXIS One Click Cloud Connection (O3C) and the O3C Server. If the camera cannot connect to the server, that is the connection is being refused, in most cases this is caused by an error in the configuration file, or that the server is not running.

1. Make sure that the service is up and running.
2. Make sure that there is no firewall blocking the server.
3. Make sure that the IP address specified in the configuration file matches the IP address of the computer it is running on.

If the camera is successfully dispatched to the server, but not seen in the connection list, this may be caused by a firewall/NAT.

1. Make sure the port is open and not blocked by any firewall on the running computer
2. If there is still no connection, get a server report of the camera if possible and use it when communicating with our support

If you experience any server related issues, simply access the `debug.cgi` to get continuous log information from the server. Check the log file (see the `logfile` setting in the configuration file where it is located) to get more information that may help you solve the problem.

7.1 Status LED Error Indications

When a One Click enabled device fails to connect to O3C Server or the AXIS O3C Dispatcher, the devices status LED indicator flashes to indicate the problem (see table 15).

Indicator	Description
1 amber/off flash	Host or net unreachable. Most likely reasons: no connection to the router, no DHCP server, user not logged in to the Internet, the ISP only has a limited number of IP addresses available.
2 amber/off flashes	DNS error. A probable reason is that DNS has not been set by DHCP or manual IP configuration. Consider using an IP address and a DNS name for the AXIS O3C Server in the list of servers in the camera configuration.
3 amber/off flashes	Connection timed out. The camera cannot reach the O3C Server. Is there a proxy blocking the traffic? Is O3C Server offline? Can the NAT router handle all connections?
1 amber/green flash	Connection refused. Is O3C Server running?
2 amber/green flashes	Connection reset.
4 amber/green flashes	Camera connected, but the system could not register the camera.
5 amber/off flashes	Other error.

Table 15: Status LED error indications

8 Security

The connection between O3C clients (devices) and the O3C Server is secured by Transport Layer Security (TLS). The TLS implementation ensures that data transferred across a non-trusted network (e.g. The Internet) is authenticated, maintains integrity and confidentiality.

Within the O3C stack, both the client and server are authenticated using bi-directional Public Key Infrastructure (PKI), which prevents man-in-the-middle attacks. The PKI is described in Section 8.1

Integrity and confidentiality is dictated by the cipher that is negotiated by the client and the server. The allowed set of ciphers can be configured to comply with adopted security standards. As devices may not support compliant ciphers, either technically or practically, notes on choosing appropriate ciphers are found in Section 8.3.

Note

If the ciphers selection is left with its default value, the O3C server will log a warning to indicate that additional hardening can be done.

8.1 Public Key Infrastructure

The main parts in the PKI are:

1. AXIS Dispatcher; the Client Certificate Authority.
2. AXIS Device; the host running an O3C client.
3. Provider server installation; the hosts running O3C servers.

Figure 1 illustrate the resources and how they relate to each other.

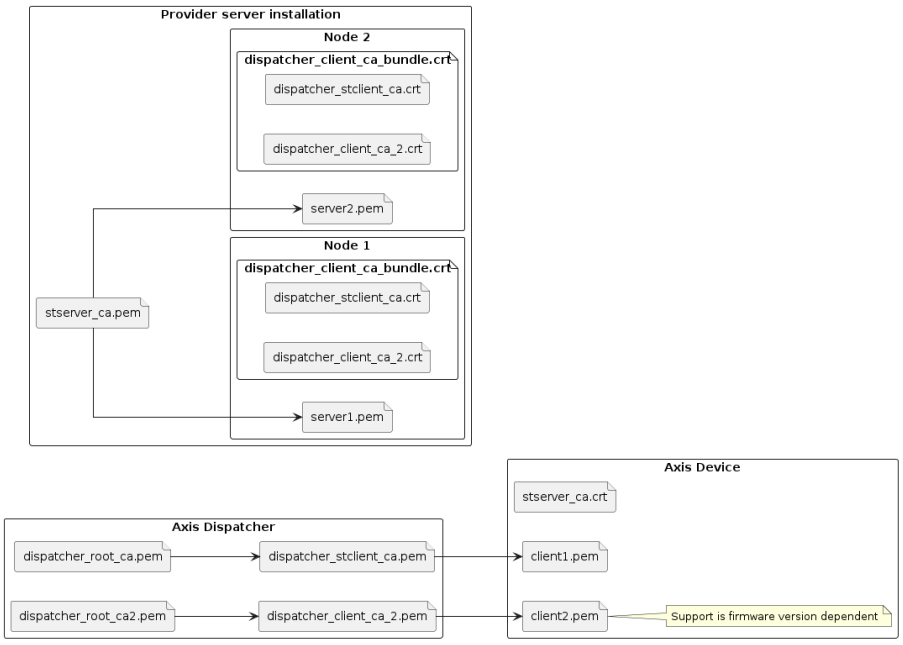


Figure 1: Certificate setup diagram.

The pointed arrows denotes **signs**, as in "`stserver_ca.pem` **signs** `server1.pem`".

In the diagram the `Provider server installation` consists of two nodes, each running an instance of the O3C Server. Each node has a server certificate signed by the certificate authority generated for this particular `Provider server installation`, `stserver_ca.pem`. The server certificates are bundled together with their respective private keys, and stored in the files `server1.pem` and `server2.pem` respectively.

The private key of `dispatcher_stclient_ca.pem` signs `client1.pem`. Depending on the device firmware version the private key of `dispatcher_client_ca_2.pem` also signs `client2.pem`. This is performed by the `AXIS Dispatcher` when an `AXIS Device` is dispatched.

The `AXIS Dispatcher` will also install `stserver_ca.crt` on the `AXIS Device` when it is dispatched. This file is uploaded to the dispatcher, by the `Provider O3C Server Installation`, when the `AXIS Device` registration request is made.

This means the `AXIS Device` will now trust any O3C servers as long as their certificates are signed by `stserver_ca.pem`.

The nodes in the `Provider server installation` also contain files `dispatcher_stclient_ca.pem`, `dispatcher_client_ca_2.pem` and a bundle of these two files called `dispatcher_client_ca_bundle.pem`. These files contains the public parts of the `AXIS Dispatcher`'s client CA certificates. This means the O3C servers in the `Provider O3C Server Installation` will trust the `AXIS Device` as long as it has been dispatched by `AXIS Dispatcher`.

8.2 Server certificate notes

An `AXIS Device` requires the subject OU field in server certificates to have the value `ST Server`.

The valid signature algorithms to use in the certificates is determined by the devices to be used in the system. Stronger signature algorithms may induce performance issues on certain device models and certain firmware versions may not support certain algorithms. For reference, `pkitool` creates certificates with sha256WithRSAEncryption as signature algorithm.

The valid key lengths to use in the certificates is determined by the devices to be used in the system. Longer key lengths may induce performance issues on certain device models. For reference, `pkitool` creates certificates with 4096 bit key length.

8.3 Configuration notes

The ciphers that are presented and accepted by the server can be controlled by the `ciphers` parameter in the configuration file, whereas TLS versions can be controlled by the `client_disable_tls_x_y` parameters.

For TLS1.3, cipher suites can be configured using the `ciphersuites` parameter. If not configured it will use the TLS1.3 standard ciphers. See https://wiki.openssl.org/index.php/TLS1.3#Ciphersuites for reference.

A restrictive configuration may inhibit devices from connecting to the server as the chosen ciphers may not be supported by the device and version, or not be feasible for use as the device hardware performance may be a limiting factor.

A device may be equipped with a chip that provides hardware acceleration of cryptographic algorithms. The ARTPEC-3, ARTPEC-4 and ARTPEC-5 chips have hardware acceleration for AES128. If AES256 induces high load on devices based on these chips, AES128 may be considered as a way to decrease the load.

With the introduction of a second PKI (depicted by `dispatcher_client_ca_2.pem` in the 1) the support for revoking the `client2.pem` certificate is introduced. This client certificate is revoked when the device is dispatched and when a device is un-registered from a dispatcher account. The CRL is available at

`http://crl.o3c.axis.com/crl/66af0131-d6e8-40c1-93ba-edb6f2cc92d2.crl`

For more information see `crl_url`, `crl_path` and `crl_mode` in the example configuration.

The `Provider server installation` can be configured to trust either of, or both, `client1.pem` and `client2.pem` by setting the appropriate value for the `ca_cert_file`. The file `dispatcher_stclient_ca.pem` is the CA certificate chain supplied in O3C Server versions before 2.33.0. The file `dispatcher_client_ca_2.pem` is the CA certificate chain added with O3C Server version 2.33.0. A bundle of these two files is also supplied, called `dispatcher_client_ca_bundle.pem`.