

Chipkin BACnet SC Hub - User Manual

Version 1.1.1 · Chipkin Automation Systems Inc. · <https://store.chipkin.com/>

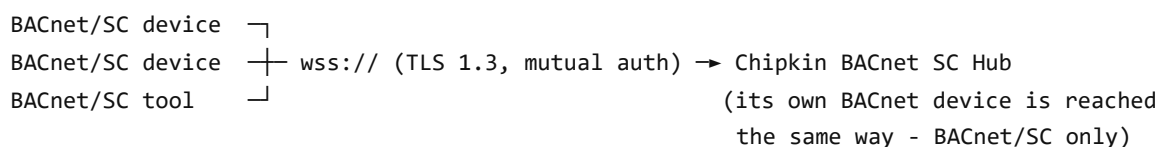
This manual is for people who install, configure and run the Chipkin BACnet SC Hub. `Chipkin-BACnet-SC-Hub --version` prints the version you are running, with the CAS BACnet Stack version it is built on.

Contents

1. [What the hub does](#)
2. [Installation](#)
3. [Quick start](#)
4. [Certificates](#)
5. [Configuration](#)
6. [Connecting devices](#)
7. [Managing certificates over BACnet](#)
8. [Status page and HTTP endpoints](#)
9. [Security](#)
10. [Troubleshooting](#)
11. [The BACnet device](#)
12. [Version, licensing and support](#)

1. What the hub does

BACnet Secure Connect (BACnet/SC, ANSI/ASHRAE 135 Annex AB) carries BACnet over TLS-secured WebSockets instead of plain UDP. Devices on a BACnet/SC network don't talk to each other directly: each one connects to a **hub**, which relays their traffic - the role a BACnet/IP broadcast domain plays for UDP devices. The Chipkin BACnet SC Hub is that hub, built on the [CAS BACnet Stack](#).



It implements the **B-SCHUB** device profile (ANSI/ASHRAE 135 Annex L):

- **BACnet/SC hub function** (BIBB NM-SCH-B) on Network Port 2, the hub's only data link: listens on `wss://0.0.0.0:4443/` (TLS 1.3, mutual certificate authentication, WebSocket subprotocol `hub.bsc.bacnet.org`) and relays traffic between the connected devices, including the hub's own BACnet device.
- **Optional hub connector**: with `--sc-hub-uri`, the hub also connects out to another BACnet/SC hub (with an optional failover hub).
- **BACnet/SC only**. The hub has no BACnet/IP port: its own BACnet device (discovery, `ReadProperty`, `ReadPropertyMultiple`, `DeviceCommunicationControl`, certificate management) is reached over BACnet/SC, through the hub, like every other device on the network. See [BACnet/SC only](#).
- **Certificates from your Certificate Authority**, in the CARI file format (ANSI/ASHRAE 135-2024 Annex AA.2). The hub makes its own private key and a certificate request; your CA signs it. The hub never signs

certificates.

- **Certificate management over BACnet:** a client can add an issuer and replace the hub's operational certificate (clause 19.8.3).
- **Status page with a health checklist, health and metrics** over HTTP or HTTPS, for people and for monitoring tools, and a **set-up guide** in the browser. Started from a console, the hub opens the status page in your web browser.

Demo mode. Until you have a licence, the hub runs as a demo: you type demo to start it, it accepts at most **4 BACnet/SC devices at a time**, it **stops after 24 hours** (start it again to continue), and it runs from a console, not as a service. See [Demo mode](#). For a licence, contact sales@chipkin.com.

Supported BIBBs

BIBB	Name
DS-RP-B	Data Sharing - ReadProperty - B
DS-RPM-B	Data Sharing - ReadPropertyMultiple - B
DM-DDB-B	Device Management - Dynamic Device Binding - B
DM-DOB-B	Device Management - Dynamic Object Binding - B
DM-DCC-B	Device Management - DeviceCommunicationControl - B
NM-SCH-B	Network Management - BACnet/SC Hub Function - B

Services executed

Service	Use
ReadProperty, ReadPropertyMultiple	Read any property of any object.
Who-Is / I-Am, Who-Has / I-Have	Discovery, over BACnet/SC.
DeviceCommunicationControl	Silence or resume the device, optionally with a password.
AtomicReadFile	Read the certificate File objects.
AtomicWriteFile, WriteProperty	Certificate management only: write a certificate into File 1, 3 or 4, and set their File_Size. No other property is writable.
ReinitializeDevice	ACTIVATE_CHANGES or WARMSTART applies written certificates. Other states are refused.

The PICS lists every object and property and who answers it.

2. Installation

Each release comes as:

Download	What it is
Chipkin-BACnet-SC-Hub-<version>-setup.exe	Windows installer. Installs the program in C:\Program Files\Chipkin\BACnet SC Hub; settings, certificates and logs go in C:\ProgramData\Chipkin\BACnetSCHub. Adds Start-menu shortcuts that run the hub in a console window and open this manual, and optionally a firewall rule for TCP 4443.
chipkin-bacnet-sc-hub_<version>_amd64.deb	Debian/Ubuntu package: sudo apt install ./chipkin-bacnet-sc-hub_<version>_amd64.deb. The program goes in /opt/chipkin-bacnet-sc-hub, settings in /etc/chipkin-bacnet-sc-hub/hub.conf, certificates in /etc/chipkin-bacnet-sc-hub/certs, logs in /var/log/chipkin-bacnet-sc-hub, and creates the bacnethub user the hub runs as.
Chipkin-BACnet-SC-Hub-<version>-linux-x64.tar.gz	The same for other Linux distributions: unpack it and run sudo ./install.sh. sudo ./uninstall.sh removes it (--purge also removes settings and certificates).
Chipkin-BACnet-SC-Hub.exe, Chipkin-BACnet-SC-Hub	The bare program, to run from a folder of your choice.
manual.pdf, fact-sheet.pdf, PICS.pdf	This manual, the two-page fact sheet, and the PICS.
sbom-windows.cdx.json, sbom-linux.cdx.json	Software bill of materials (CycloneDX): the exact CAS BACnet Stack, OpenSSL, libwebsockets and other library versions.
SHA256SUMS.txt	SHA-256 checksums of every download: sha256sum -c SHA256SUMS.txt --ignore-missing.

After installing, the hub needs its certificate. When the certificate folder is empty, both installers run --generate-csr: the hub makes its private key and a certificate request, hub-cari-request.zip, in the certificate folder. Until your Certificate Authority's answer is installed, BACnet/SC can't start, so the hub is unreachable over BACnet; its status page says what to do and /health reports degraded. See [Certificates](#). Settings and certificates are kept when you upgrade or uninstall.

Signed downloads. The Windows program and the Windows installer are code-signed by Chipkin (Azure Artifact Signing, SHA-256 with a timestamp); check them with Get-AuthenticodeSignature <file>. SHA256SUMS.txt lists every file's checksum. See code-signing.md.

Supported platforms

Windows 10/11 and Windows Server 2016 or later (x64). Linux x64: the release build is made on the current Ubuntu LTS, so it needs a distribution with that glibc version or newer. For another platform, contact Chipkin.

The hub runs fine on Windows 10, but BACnet/SC tools on a Windows 10 computer that use Windows' own TLS (such as YABE) can't connect to it or to any other BACnet/SC hub: BACnet/SC requires TLS 1.3, and Windows 10 can't make TLS 1.3 client connections. The hub logs a warning about this when it starts on Windows 10.

Firewall

Allow in:

Port	Protocol	For
4443	TCP	BACnet/SC devices connecting to the hub (--sc-port).
8080	TCP	The status page and HTTP endpoints (--http-port) - only if you bind it beyond 127.0.0.1.

The Windows installer can add the first rule for you. The hub has no BACnet/IP port (UDP 47808); an older version's "BACnet SC Hub (BACnet/IP)" firewall rule is removed when you install this version.

Running the hub

The demo runs from a console window. It first asks you to type `demo` and press Enter (see [Demo mode](#)), then opens the status page in your web browser. Leave the window open while the hub runs; press `q` (or `Ctrl+C`) to stop it.

Windows. Start **Chipkin BACnet SC Hub (demo)** from the Start menu. It runs the hub with the installed settings, the same as:

```
cd /d C:\ProgramData\Chipkin\BACnetSCHub
"C:\Program Files\Chipkin\BACnet SC Hub\Chipkin-BACnet-SC-Hub.exe" --config hub.conf
```

Relative paths in `hub.conf` (`sc-cert-dir = certs`, `log-file = logs\hub.log`) are relative to that folder. Only SYSTEM, Administrators and the user who installed the hub can open the folder, so another user needs an Administrator command prompt.

Linux. After installing the `.deb` or running `install.sh`, start the hub as the `bacnethub` user, which owns its settings, certificates and log:

```
sudo -u bacnethub /opt/chipkin-bacnet-sc-hub/Chipkin-BACnet-SC-Hub --config /etc/chipkin-bacnet-sc-hub/hub.conf
```

Its output also goes to `/var/log/chipkin-bacnet-sc-hub/hub.log`. To start it from a script, answer the demo question on its input: `echo demo | sudo -u bacnethub /opt/chipkin-bacnet-sc-hub/Chipkin-BACnet-SC-Hub --config /etc/chipkin-bacnet-sc-hub/hub.conf`.

In demo mode the hub stops after 24 hours; start it again to continue. See [Demo mode](#).

As a service. Running the hub as a Windows service or a systemd service needs a licence; for one, contact sales@chipkin.com. Without a licence, `--install-service` and `--service` are refused with a message saying so (exit code 1), and on Linux the hub refuses to start under systemd. A service has no console, so a licensed service with a password-protected key needs `sc-key-password` in its config file (see [Password-protected private keys](#)), and `log-file` to keep a log.

An older version's service. Installing this version with the `.deb` or `install.sh` stops and removes the `chipkin-bacnet-sc-hub` systemd service an older version set up. On Windows, uninstalling removes the `BACnetSCHub` service; to remove it without uninstalling, run `Chipkin-BACnet-SC-Hub --uninstall-service` from an Administrator command prompt.

3. Quick start

From a folder where you unpacked the program (the installers do step 1 for you; to start an installed hub, see [Running the hub](#)):

1. Make the hub's private key and its certificate request.

```
Chipkin-BACnet-SC-Hub --generate-csr
```

2. Send certs/hub-cari-request.zip to your Certificate Authority.

Install the CARI response it sends back:

```
Chipkin-BACnet-SC-Hub --import-cari hub-cari-response.zip
```

3. Start the hub. In demo mode, type demo and press Enter when it asks.

```
Chipkin-BACnet-SC-Hub
```

Then:

- The hub opens <http://127.0.0.1:8080/> in your web browser (it always prints the address too). The status page's health checklist says what works and what to do about what doesn't, beside the hub's health figures, the demo-mode limits and the connected devices.
- Open <http://127.0.0.1:8080/setup>, the [set-up guide](#). It can do step 2 for you (download the request, upload the CA's response), tells each device what it needs, and shows live why a connection attempt failed.

Each device gets its own certificate from the same CA. See [Connecting devices](#).

On Windows the program is Chipkin-BACnet-SC-Hub.exe. A typical start-up:

```
Chipkin BACnet SC Hub v1.1.1
```

```
CAS BACnet Stack version: 6.0.23.0
```

```
Common helper (common/) version: 3.1.0
```

```
=====
```

```
2026-10-07 18:10:19 [WARNING] DEMO MODE: this copy of the Chipkin BACnet SC Hub is an unlicensed demo.
It accepts at most 4 BACnet/SC devices at a time and stops after 24 hours (start it again to continue).
For a licence, contact Chipkin sales: sales@chipkin.com
```

```
=====
```

```
Type demo and press Enter to start the demo (Ctrl+C to quit): demo
```

```
2026-10-07 18:10:21 [INFO] starting the demo
```

```
2026-10-07 18:10:21 [OK] Device 389022 ("Chipkin BACnet SC Hub") ready. Vendor ID 389. Press 'h' for
help, 'm' for a health/metrics snapshot.
```

```
=====
```

```
2026-10-07 18:10:21 [WARNING] DEMO MODE: at most 4 BACnet/SC devices at a time, and the hub stops after
24 hours (restart it to continue). For a licence, contact Chipkin sales: sales@chipkin.com
```

```
=====
```

```
2026-10-07 18:10:21 [OK] status page: http://127.0.0.1:8080/ - health checklist, connected devices and
traffic
```

```
2026-10-07 18:10:21 [INFO] opened the status page in your web browser
```

```
2026-10-07 18:10:21 [OK] BACnet/SC: listening for WebSocket/TLS connections on wss://0.0.0.0:4443/
(subprotocol "hub.bsc.bacnet.org", TLS 1.3, mutual auth)
```

On a terminal the lines are coloured by level: errors red, warnings yellow, [OK] lines green, information in the terminal's own colour. The colour is left out when the output goes to a file or a pipe, when the NO_COLOR environment variable is set (to anything), and always in the --log-file copy.

The browser. Started from a console (its input and output both a terminal), the hub opens the status page in your default web browser once its HTTP server is up: on Windows with the default browser, on Linux with xdg-open when

a desktop session is running (DISPLAY or WAYLAND_DISPLAY is set). If that can't be done it says so and carries on; the address is printed either way. A script, a pipe or a service never opens a browser.

At start-up the hub also checks its certificates and logs each one's subject, days until expiry (a warning under 30 days), whether the private key matches, and the subjectAltName entries.

Console keys

Key	Action
h	Help and version.
m	Health and metrics snapshot.
up / down	Change Analog Input 1 by 1.1.
q	Quit.

4. Certificates

BACnet/SC support

BACnet/SC runs over TLS with **mutual authentication**: the hub and every device present a certificate, and each side accepts the other only if that certificate was signed by an issuer (a Certificate Authority, CA) it trusts. Without its own certificate, the hub can't listen for BACnet/SC, so nothing can reach it over BACnet; its status page and set-up guide still work, and /health reports degraded.

The hub never signs certificates. A Certificate Authority does. You can use:

- the **Chipkin BACnet SC Certificate Authority**, Chipkin's CA application for BACnet/SC sites;
- **BACCARI**, from BACnet International;
- **any CARI-compatible CA**, including your own PKI (see production-certificates.md).

Getting the hub its certificate takes four steps:

1. **Make the request.** `Chipkin-BACnet-SC-Hub --generate-csr` makes the hub's private key and a certificate signing request (CSR), and writes `hub-cari-request.zip` in the certificate folder. The installers do this for you. The zip holds the CSR only; the private key never leaves the hub.
2. **Send it to your CA.** Download `hub-cari-request.zip` from the [set-up guide](#) (step 2) or take it from the certificate folder, and send it to the CA.
3. **Install the CA's response.** The CA sends back a CARI response zip. Install it with `--import-cari <response.zip>` and restart the hub, or upload it on the set-up guide (step 3), which installs it and starts using it straight away.
4. **Check.** The status page says the hub is listening, and /health reports ok.

Both ways of installing check the response first: the certificate in it must be for **this hub's request** (its public key matches the hub's CSR), it must not have expired, and one of the issuer certificates in the zip must have signed it. Anything else is refused and nothing is written.

The hub's certificate can also be installed [over BACnet](#) or with `POST /certs/<slot>`.

The CARl file layout

Certificate files use the **CARl** format (Certificate Authority Requirements Interchange, ANSI/ASHRAE 135-2024 Annex AA.2, added to 135-2020 by Addendum cs). CARl is the folder layout BACnet uses to take certificate signing requests to a CA and bring the certificates back. Its root folder is cert1/:

- **Folders:** one device-<instance>/ folder per device, and one port-<id>/ folder per BACnet/SC port in it. A hub/ folder in a port folder marks a hub function, and a router/ folder in a device folder marks a device that routes between BACnet/SC networks. Both are empty.
- **Files in a port folder:** csr-<name>.pem (the request), opr-<name>.pem (the operational certificate the CA returns) and, optionally, key-<name>.pem (the private key).
- **The issuer:** cert1/issuer/iss-1.pem (and iss-2.pem).

A **CARl request** is a zip of such a tree with the CSRs in it; the CA's **CARl response** is the same tree with the opr-certificates and the issuer certificate(s) added.

The hub's certificate folder (--sc-cert-dir, default ./certs) is a CARl tree for the hub itself:

File	What it is
cert1/device-<deviceID>/port-2/hub/	Empty: Network Port 2 is a hub function.
cert1/device-<deviceID>/port-2/key-hub.pem	The hub's private key, made by --generate-csr. Private.
cert1/device-<deviceID>/port-2/csr-hub.pem	The hub's certificate signing request (File 2, Certificate_Signing_Request_File).
cert1/device-<deviceID>/port-2/opr-hub.pem	The hub's certificate, from the CA's response (File 1, Operational_Certificate_File).
cert1/issuer/iss-1.pem	The CA that signed it (File 3, Issuer_Certificate_Files). Devices need a copy to trust the hub.
cert1/issuer/iss-2.pem	File 4: the CA response's second issuer. With one issuer it is a copy of iss-1.pem, so no old issuer stays trusted. It can also be changed over BACnet or with POST /certs/issuer2.
hub-cari-request.zip	The hub's CARl request, to send to the CA. Holds the CSR, never the key.
key-hub-pending.pem	Only after a GENERATE_CSR_FILE over BACnet: the new key behind csr-hub.pem, until a certificate for it is installed. Private.
trusted-issuers.pem	Written by the hub: every issuer from both slots, which is what TLS trusts.
issuer-crl.pem	Optional, from your CA: revoked device certificates (see Security).
readme.txt	A walkthrough of the folder, including which files are private.

<deviceID> is the hub's device instance (--deviceID, default 389022). If you change the device instance after making the request, the hub looks in a different folder: run --generate-csr again with the new device-id (use --config with the hub's config file, so it sees the same settings).

Making the request

```
Chipkin-BACnet-SC-Hub --generate-csr
Chipkin-BACnet-SC-Hub --generate-csr --cert-hub-uri wss://bacnet-hub.example.site:4443/
```

The request is for an ECDSA P-256 key, signed with SHA-256. Its subject Common Name is the device name (--device-name), and it asks for the TLS purposes a hub needs (serverAuth and clientAuth) and a subjectAltName with the host of the hub URI devices dial. That host comes from --cert-hub-uri, or by default this computer's IPv4 address. The CA decides what the certificate finally contains.

Running --generate-csr again keeps the key and writes the zip again from the same request; it never replaces a key. To make a new request (for another host name, say), move key-hub.pem and csr-hub.pem out of the folder first. To renew with a new key while the hub keeps running, use GENERATE_CSR_FILE over BACnet (see [Managing certificates over BACnet](#)).

For an installed hub, run the commands with its settings, as the account that owns the certificate folder:

```
# Linux
sudo -u bacnethub /opt/chipkin-bacnet-sc-hub/Chipkin-BACnet-SC-Hub \
    --config /etc/chipkin-bacnet-sc-hub/hub.conf --generate-csr

REM Windows, as the user who installed the hub (or from an Administrator command prompt)
cd /d C:\ProgramData\Chipkin\BACnetSCHub
"C:\Program Files\Chipkin\BACnet SC Hub\Chipkin-BACnet-SC-Hub.exe" --config hub.conf --generate-csr
```

Installing the CA's response

On the command line:

```
Chipkin-BACnet-SC-Hub --import-cari hub-cari-response.zip
```

It prints the certificate's subject, issuer, expiry and SHA-256 fingerprint, and writes opr-hub.pem, cert1/issuer/iss-1.pem and iss-2.pem (the response's second issuer, or a copy of the first). Restart the hub to use it. For an installed hub, run it as in [Making the request](#), with --import-cari <file> in place of --generate-csr; on Linux, put the zip somewhere the bacnethub user can read, such as /tmp.

On the set-up guide: step 3, *Install the CA's response*, takes the zip and installs it while the hub runs. BACnet/SC reloads with the new certificate; connected devices reconnect. See [The set-up guide](#) for who may do this.

The issuers in the response replace the hub's issuer slots. Devices whose certificates come from another CA are then no longer accepted. To keep trusting a second CA (while a site moves to a new one, say), ask the CA to include both issuers in the response, or add the second one afterwards over BACnet or with POST /certs/issuer2.

Older folders

A certificate folder from an earlier release keeps working unchanged. That covers the flat names (operational-certificate.pem, private-key.pem, issuer-certificate.pem, ...) and the older ones (hub.crt, hub.key, ca.crt). The start-up log names the layout it found. --migrate-certs copies such a folder into the CARI layout and leaves the old files as they are, so going back only means deleting cert1/. --generate-csr and --import-cari need the CARI layout.

Production certificates

production-certificates.md covers running the hub with your own CA in more depth: choosing a CA, root and intermediate layouts, signing the hub's request with your own PKI, rotating certificates and changing the CA without an outage, revocation, what the certificates should contain, and protecting private keys.

Password-protected private keys

A private key may be stored encrypted (the file starts -----BEGIN ENCRYPTED PRIVATE KEY-----), for example when you replace the hub's key with one protected by your own tools. The hub asks for its password **once**, when it starts, with the typing hidden, and checks it by decrypting the key; a wrong password is asked for again, up to three times. It keeps the password in memory and uses it everywhere the key is needed - the listener, the connector's every connect attempt, the HTTPS status page - so it is never asked for again during the run.

A service (licensed copies only) has no console to ask on: set the password in the config file as sc-key-password (config file only, like dcc-password; restrict the file's permissions as described in [Configuration file](#)). When it is set, the hub doesn't ask.

The same password unlocks the [set-up guide](#)'s certificate installation from any computer that reaches it over HTTPS. Without one, it works only on the hub's own computer.

5. Configuration

Chipkin-BACnet-SC-Hub [options]

Options

Option	Default	Meaning
--ip-address <address or name>	the first IPv4 adapter	The network adapter the hub advertises, on a computer with several networks: an IPv4 address of this computer (192.168.1.20) or an adapter name (Ethernet, eth0). Sets the hub URI --generate-csr asks for in the certificate's subjectAltName, and the one the set-up guide gives devices. The listeners still accept on every adapter. A name that matches nothing stops the hub and lists the adapters.
--deviceID <n>	389022	BACnet device instance.
--device-name <name>	Chipkin BACnet SC Hub	The Device's Object_Name. Must be unique on the BACnet internetwork, so name each hub. 1 to 128 bytes of UTF-8; for a non-ASCII name use device-name in a config file saved as UTF-8.
--sc-network-number <n>	not set	Network_Number of Network Port 2 (1..65534), reported with quality configured. Unset, it reports 0, quality unknown.
--log-file <path>	none	Also write everything the hub prints to this file (see Logging).
--log-max-size-mb <n>, --log-max-files <n>	10, 5	Rotate the log file at this size, keeping this many old files (<path>.1, .2, ...).
--sc-port <n>	4443	BACnet/SC hub (wss://) port.
--sc-cert-dir <dir>	./certs	Certificate folder.

Option	Default	Meaning
--sc-hub-uri <wss://host:port/>	off	Also connect out to this BACnet/SC hub.
--sc-failover-uri <wss://host:port/>	off	Failover hub for --sc-hub-uri.
--sc-max-hub-connections <n>	4	Maximum BACnet/SC devices connected at once, 1 to 4 in demo mode (see Demo mode).
--sc-rate-limit <n>	10	Maximum new BACnet/SC connection attempts per second from any one source address (0 = no limit). Excess attempts are refused before the TLS handshake.
--sc-rate-limit-total <n>	50	Maximum new BACnet/SC connection attempts per second for the whole listener (0 = no limit).
--sc-accept-hub-without-hello	off	Compatibility: let --sc-hub-uri connect to a hub that omits the Hello option. See BACnet/SC compatibility .
--sc-accept-device-without-hello	off	Compatibility: accept a device whose Connect-Request omits the Hello option, such as YABE. See BACnet/SC compatibility .
--http-port <n>	8080	Status page and HTTP endpoints.
--http-bind <addr>	127.0.0.1	Interface for the HTTP listener. See Security before changing it.
--http-tls	off	Serve the HTTP endpoints over HTTPS.
--http-tls-cert <file>, --http-tls-key <file>	the hub's operational certificate and key	Certificate and key for --http-tls.
--sc-keylog-file <file>	off	Debugging only. Append each BACnet/SC connection's TLS session secrets to <file>, so Wireshark can decrypt a capture. Command line only. See Decoding BACnet/SC traffic in Wireshark .
--no-update-check	checks	Don't ask Chipkin's update service for a newer version (see Update check). Config file: check-for-updates = false.
--config <path>	none	Read settings from a config file (below).
--generate-csr	-	Make the hub's private key and certificate request, and hub-cari-request.zip for the CA, then exit. With a key already there, writes the zip from its request. See Making the request .
--import-cari <zip>	-	Install the CA's CARi response (the hub's certificate and the issuer certificates) after checking it, then exit. Restart the hub afterwards.
--cert-hub-uri <wss://host:port/>	this computer's IPv4 and --sc-port	The hub URI devices dial. --generate-csr asks for its host in the certificate's subjectAltName.
--migrate-certs	-	Copy an older release's certificate folder into the CARi layout, then exit.

Option	Default	Meaning
<code>--inspect <file></code>	-	Say what a certificate, CSR, key, .pfx or CARI zip is, list every field and check it against this hub (the set-up guide's file inspector), then exit; 1 if a check fails.
<code>--install-service, --uninstall-service</code>	-	Windows: set up or remove the BACnetSCHub service. Setting one up needs a licence; removing one works without (see Running the hub).
<code>--help, --version</code>	-	Usage, or version information.

An unknown option, a stray argument or an option missing its value is an error. The BACnet/IP options removed in 1.1.0 - `--port <n>`, `--bacnet-ip <on/off>` and `--ip-network-number <n>` - are the exception: the hub warns that it ignores them and starts, so an old shortcut or script keeps working. The on/off switches (`--http-tls`, `--sc-accept-hub-without-hello`, `--sc-accept-device-without-hello`) mean "on" alone and also take on or off, so the command line can turn off what the config file turned on.

Configuration file

`--config <path>` reads `key = value` lines. A line starting with # is a comment; a # later in a line is part of the value, so passwords and tokens may contain one. `example.conf` (shipped with the program) lists every key with its default: `device-id`, `device-name`, `sc-network-number`, `log-file`, `log-max-size-mb`, `log-max-files`, `ip-address`, `sc-port`, `sc-cert-dir`, `sc-hub-uri`, `sc-failover-uri`, `dcc-password`, `sc-key-password`, `http-upload-token`, `http-port`, `http-bind`, `http-tls`, `http-tls-cert`, `http-tls-key`, `sc-max-hub-connections`, `sc-rate-limit`, `sc-rate-limit-total`, `sc-accept-hub-without-hello`, `sc-accept-device-without-hello` and `check-for-updates`. On/off keys take true/false (or on/off, yes/no, 1/0). A command-line option always wins over the config file. An unknown key or bad value is logged and skipped. A config file from an older version that still has `port`, `bacnet-ip` or `ip-network-number` (BACnet/IP, removed in 1.1.0) gets a warning for each of those lines, and the hub starts; delete them.

`dcc-password`, `sc-key-password` and `http-upload-token` can only be set in the config file, so they never show up in process listings or shell history. `sc-key-password` is the password of an encrypted private key (see [Password-protected private keys](#)). `dcc-password` protects DeviceCommunicationControl and ReinitializeDevice; `http-upload-token` is the separate secret for the HTTP certificate upload (use a different value - the hub warns if they match). Both are compared in constant time. Restrict the file's permissions (`chmod 600 hub.conf`, or on Windows `icacls hub.conf /inheritance:r /grant:r *S-1-5-18:F *S-1-5-32-544:F "%USERNAME%:F"`, which keeps access for SYSTEM, Administrators and you); the hub warns at start-up if the file looks readable by other users. The Windows installer already restricts `C:\ProgramData\Chipkin\BACnetSCHub` to SYSTEM, Administrators and the user who installed the hub.

Logging

Everything the hub prints goes to the console. With `--log-file <path>` (config: `log-file`) it also goes to that file, which is appended to across restarts. When the file reaches `--log-max-size-mb` (default 10 MB) it is renamed `<path>.1` (the previous `.1` becomes `.2`, and so on) and a new file is started; `--log-max-files` (default 5) old files are kept. Every line written through the hub's logger starts with a UTC timestamp and a level ([INFO], [OK], [WARNING], [ERROR]). Connection audit lines start with `SC audit:.` The log file never contains colour codes, even when the console copy is coloured.

BACnet/SC only

The hub has no BACnet/IP port (it was removed in version 1.1.0): it opens no UDP port, its only Network Port is Network Port 2 "BACnet SC", and its own BACnet device is reachable only over BACnet/SC - discovery (Who-Is/I-Am) and every other service go through the hub, like every device on the network. A BACnet tool reaches it by connecting to the hub as a BACnet/SC device (it needs a certificate from your CA, like any device). The hub doesn't send an I-Am at start-up, since no device is connected yet; devices find it with Who-Is once they connect.

The BACnet/SC Network Port keeps **instance 2**: the hub's certificate folder is named after it (cert1/device-`<n>/port-2/`, ANSI/ASHRAE 135-2024 Annex AA.2), so every installed hub and every certificate request a CA has signed stays valid.

The hub warns a few seconds after start-up if BACnet/SC isn't listening (no certificate yet, say), because the device can't then be reached over BACnet at all. BACnet/SC keeps retrying every 5 seconds, /health reports degraded until it is listening, and the status page's health checklist says what to fix.

BACnet/SC compatibility

ANSI/ASHRAE 135-2024 (clause AB.2.2) requires the BACnet/SC Connect-Request and Connect-Accept messages to carry a *Hello* option. By default the hub follows the standard strictly and refuses a connection that leaves it out.

`--sc-accept-hub-without-hello` (config: `sc-accept-hub-without-hello = true`) relaxes this for the **hub connector** only: when the hub connects out to another hub (`--sc-hub-uri`), it accepts that hub's Connect-Accept without Hello and treats the other hub as having no optional capabilities. Turn it on only for a hub that needs it; the hub logs a warning at start-up while it is on.

`--sc-accept-device-without-hello` (config: `sc-accept-device-without-hello = true`) does the same for **devices** connecting to this hub: it accepts a Connect-Request without Hello, which YABE sends, and records the device as having no optional capabilities. Without it, such a device is refused and the hub logs `Connect-Request refused by the hub: "Connect messages require the Hello destination option"`. Turn it on only while a device like that needs to connect; the hub logs a warning at start-up while it is on.

The two settings are independent; either, both or neither can be on.

Demo mode

Until licensing is available for your copy, the hub runs in **demo mode**:

- **You type demo to start it.** The hub shows the demo terms between two horizontal rules and asks you to type demo and press Enter. Anything else asks again; after three other answers, or if its input ends with no answer, the hub doesn't start: it says why and exits with code 1. There is no option that skips the question. A script answers it on the hub's input: `echo demo | Chipkin-BACnet-SC-Hub [options]`. The one-shot tools (`--generate-csr`, `--import-cari`, `--migrate-certs`, `--inspect`) and `--help` don't ask.
- **At most 4 BACnet/SC devices at a time.** A fifth device is refused at its BACnet/SC Connect-Request until one disconnects, and the hub logs `DEMO MODE: a BACnet/SC device was refused (at most once a minute)`. You can lower the limit with `--sc-max-hub-connections` or `sc-max-hub-connections` in the config file, but not raise it: asking for more than 4 prints an error and the hub shuts down.
- **It stops after 24 hours.** The hub logs `DEMO MODE: the 24 hours demo run time is over - stopping and exits`. Start it again to continue; each start gives another 24 hours.
- **It runs from a console only.** Running it as a Windows service or a systemd service needs a licence (see [Running the hub](#)).

Every DEMO MODE: message in the console and the log stands between two horizontal rules of =, so it can't be missed. The status page has a demo banner with the time left and a **Licence** warning in its health checklist, and /health reports demo_mode and demo_stops_in_seconds.

For a licence without these limits, contact Chipkin sales at sales@chipkin.com.

6. Connecting devices

Each BACnet/SC device needs three things:

1. **Its own certificate and private key**, from the **same CA** as the hub's certificate (or from a CA in the hub's other issuer slot). The device makes its own key and CSR, or a CARI request; the CA signs it and sends back the certificate (opr-<name>.pem in a CARI response). Give each device its own certificate: the hub tells devices apart by them.
2. **The issuer certificate**, so the device trusts the hub: iss-1.pem (and iss-2.pem, if the hub has two issuers). The CA's response for the device usually includes it; the hub also serves it on the set-up guide.
3. **The hub URI**, wss://<hub address>:4443/.

The hub trusts every certificate its issuers signed, straight away; there is nothing to add on the hub for a new device. Revoked certificates are refused when you use a revocation list (see [Security](#)).

The set-up guide

On the hub's computer, open <http://127.0.0.1:8080/setup> (the status page links to it, and every page's app bar calls it **Set-up**). It takes you through four steps:

1. **This hub's certificate.** How the trust works, and the hub's and its issuer's certificates with every field and every check.
2. **Get it signed.** Download hub-cari-request.zip (the hub's CSR, never its key) and send it to your CA. If the hub has no request yet, the page says to run --generate-csr.
3. **Install the CA's response.** Upload the CARI response zip. The hub checks it answers this hub's request and that the CA signed it, then installs it and starts using it. Connected devices reconnect.
4. **Connect devices.** The URI(s) devices dial, the issuer certificate to download (iss-1.pem, iss-1.cer for Windows tools, iss-2.pem), and which file goes into which Network Port property of the device:
5. Operational_Certificate_File: the device's opr-<name>.pem from the CA;
6. the private key the device made with its CSR (it stays on the device);
7. Issuer_Certificate_Files: iss-1.pem (in both slots if there are two);
8. the primary hub URI.

Three tools are always available:

- **Why did a connection fail?** Start the device's BACnet/SC connection. The guide shows, live, what the hub sees, and turns each failure into a sentence and a fix:
- a TCP connection with nothing after it: the device sent no certificate, only speaks TLS 1.2, or doesn't trust the hub's certificate;
- a refused certificate: OpenSSL's reason, such as signed by another CA, expired, not yet valid, no clientAuth, or revoked;
- the wrong WebSocket subprotocol;
- a refused Connect-Request (duplicate VMAC, missing Hello, the demo limit);
- and finally, "Connected!" with the device's VMAC and UUID.

- **File inspector.** Drop in any certificate, CSR, key, .pfx (empty password) or CARL zip. It shows every field and checks it against this hub; nothing is stored. The same inspector runs on the command line as Chipkin-BACnet-SC-Hub --inspect <file>, which exits with 1 if a check fails. Don't upload a production private key to any tool.
- **Support report.** A text summary to send to support: versions, the hub URI, the certificates with their checks, connected devices and recent connection attempts. It contains no keys or passwords.

Who may install a certificate. Installing the CA's response (step 3) needs the **password of the hub's private key**: the config file's sc-key-password, or the password typed when the hub started (see [Password-protected private keys](#)). With a key that has no password (what --generate-csr makes), it works only on the hub's own computer (from 127.0.0.1). Everything else in the guide is open, like the status page. See [Security](#).

Setting up a device

Most BACnet/SC devices and tools, such as the [CAS BACnet Explorer](#), take the files directly:

1. **A tool that imports CARL files** takes the CA's response for the device as it is.
2. **Any other tool or device:** unzip the response and install opr-<name>.pem (in cert1/device-<n>/port-<id>/) and the device's own private key as its operational certificate and key, and cert1/issuer/iss-1.pem as its issuer certificate, in both slots if it has two. Set its primary hub URI to wss://<hub address>:4443/.

The Chipkin BACnet Explorer imports the device's CARL zip as it is (Connect, BACnet/SC, **Import...**). It connects through the network adapter you select, so give it the hub's **LAN address** (wss://192.168.1.20:4443/), not wss://127.0.0.1:4443/ - even when it runs on the hub's own computer. Its "Test connection" checks each step and says which one failed. 3. **YABE (Yet Another BACnet Explorer)** reads its certificate and key from a .pfx file. Make one from the device's files with an empty password, for example openssl pkcs12 -export -in opr-yabe.pem -inkey key-yabe.pem -certfile iss-1.pem -passout pass: -out yabe.pfx, keep it private, and give YABE iss-1.cer (from the set-up guide) as its trusted issuer. YABE leaves out the Hello option, so start the hub with --sc-accept-device-without-hello on. YABE uses Windows' own TLS, which can't make the TLS 1.3 connections BACnet/SC requires on Windows 10: use Windows 11 / Server 2022 or later.

The hub logs refused TLS handshakes with the reason, and keeps an audit trail of every connection (lines start with SC audit:): the device's address and the subject of the certificate it presented, then its BACnet/SC VMAC and device UUID once the hub accepts it, and the same identity again when it disconnects:

```
2026-10-07 18:10:21 [INFO] SC audit: peer "wss://0.0.0.0:4443/client=3" connected from
10.0.0.31:64150, certificate "O = Example Site, CN = AHU-3 controller"
2026-10-07 18:10:21 [INFO] SC audit: peer "wss://0.0.0.0:4443/client=3" (10.0.0.31:64150) is BACnet/SC
device VMAC 0a:0b:0c:0d:0e:0f, UUID 50515253-5455-5657-5859-5a5b5c5d5e5f, certificate "O = Example
Site, CN = AHU-3 controller" - connected
```

The status page lists the devices connected right now the same way.

7. Managing certificates over BACnet

A BACnet client can change the hub's certificates with the BACnet/SC certificate procedures (ANSI/ASHRAE 135 clause 19.8.3), for example from the [CAS BACnet Explorer](#)'s certificate page:

1. **Write** - WriteProperty File_Size = 0, then AtomicWriteFile the new PEM certificate: File 1 for the hub's own certificate, File 3 or 4 for an issuer.

2. **Staged** - reading the File back shows the new certificate and Network Port 2's `Changes_Pending` is `TRUE`, but nothing is saved yet and the hub keeps using its current certificates.
3. **Activate** - `ReinitializeDevice ACTIVATE_CHANGES` (or `WARMSTART`). The hub checks the staged certificates first: each must be a PEM certificate, at least one issuer must remain, and the hub certificate must match its private key and chain to an issuer. If they pass, it saves them and restarts BACnet/SC; devices reconnect under the new certificates. If not, it answers `INVALID_CONFIGURATION_DATA` and nothing changes, so a mistake can't lock the hub out. The staged writes stay, so the client can correct them and activate again.

Supported procedures:

- **Add an issuer** - write the new CA into slot 2 (File 4). Devices signed by either CA are then accepted, which lets a site move to a new CA gradually.
- **Replace the hub certificate** - read the CSR (File 2), have your CA sign it, write the result to File 1, activate.
- **Replace the hub certificate with a new key pair** - write `GENERATE_CSR_FILE` (9) to Network Port 2's Command. The hub makes a new key pair and a new CSR (File 2) with the same subject, saving the new key as `key-hub-pending.pem`; it keeps using its current key and certificate meanwhile. Have the new CSR signed, write the result to File 1 and activate: the new key then replaces `key-hub.pem`. (The set-up guide's `hub-cari-request.zip` also carries the new CSR, and installing the CA's response there or with `--import-cari` finishes the swap the same way.) Refused with `INVALID_VALUE_IN_THIS_STATE` while certificate writes are staged - and on a freshly started hub, which reads `Changes_Pending TRUE` with nothing staged (see the known issue below): send `ReinitializeDevice ACTIVATE_CHANGES` once first. If the hub stops between saving the new certificate and swapping the key, it finishes the swap at the next start.
- **Discard staged writes** - write `DISCARD_CHANGES` (1) to Network Port 2's Command. The staged certificate writes are dropped, the File objects read the files on disk again, and `Changes_Pending` goes back to `FALSE`. A restart also discards staged writes. Don't send it to a freshly started hub: it also reverts the stack's start-up state, after which `Changes_Pending` no longer follows certificate writes (restart the hub to recover).

Known issue: right after the hub starts, Network Port 2 reads `Changes_Pending = TRUE` (and `Current_Health` may report `invalid-configuration-data`) although nothing has been written. The CAS BACnet Stack leaves the hub's own start-up certificate settings pending. The certificates in use are correct. The first `ReinitializeDevice ACTIVATE_CHANGES` clears it.

If `dcc-password` is set, `ReinitializeDevice` requires it.

8. Status page and HTTP endpoints

The hub serves HTTP on `--http-port` (default 8080), on `127.0.0.1` only by default. With `--http-tls` it serves HTTPS instead (TLS 1.2 or 1.3), using the hub's own certificate and key (`opr-hub.pem`, `key-hub.pem`) unless `--http-tls-cert/--http-tls-key` name another certificate - for example one your browsers already trust. With the hub's own certificate, point curl at its issuer: `curl --cacert certs/cert1/issuer/iss-1.pem https://localhost:8080/health` (with Windows' own curl, add `--ssl-no-revoke` if your CA publishes no revocation list Windows can check). The HTTPS certificate is loaded at start-up.

Path	What it returns
GET /	Status page for a browser. First the connected devices , one row each: status, address, VMAC, device UUID, how long it has been connected (and since when, UTC) and its certificate's subject. Then the health checklist (hidden until you click its heading; its summary - passed, warnings, failed - always shows), the system health figures (BACnet/SC state, connections, certificates), traffic counters, and the endpoints (hidden until clicked too). Your browser remembers which of the two you opened. A banner says when a newer version is available. The versions are on the About page . Its figures update in place every 5 seconds without reloading the page (so it doesn't flicker or lose your place); if the hub stops answering, the page says The hub is not responding until it answers again. Without JavaScript it reloads every 5 seconds instead.
GET /api/status	Everything the status page shows, as JSON (what keeps the page current).
GET /setup	The set-up guide , and the /api/... endpoints it uses (below).
GET /about	The About page : the versions, Check for updates , and the feedback form. (/feedback takes you there.)
GET /api/about	The About page's figures as JSON: version, stack, common, commit, platform, os, and update (state - checking, up-to-date, available, failed or off - with checking, latest, url, checkedAt and error).
GET /health	<i>Is the hub working?</i> JSON with status ok (HTTP 200) or degraded (HTTP 503 - the BACnet/SC hub isn't listening, usually because the hub has no certificate yet or a certificate is bad). Also demo_mode and demo_stops_in_seconds. Point uptime monitors and load balancers here.
GET /metrics	<i>How much is it doing?</i> JSON counters: uptime, connected devices, connects, disconnects, rate-limit refusals, messages and bytes in and out, and connections closed because the device stopped reading (sc_tx_queue_overflows).
POST /certs/<slot>	Upload a certificate file (operational, csr, issuer1, issuer2). Needs Authorization: Bearer <http-upload-token>; disabled when no http-upload-token is set. At most 5 attempts a minute from one address and 30 in total; more get HTTP 429 and a log line.

```
$ curl -s http://127.0.0.1:8080/health
{"status":"ok","version":"1.1.1","uptime_seconds":154,"sc_hub_function_listening":true,"sc_hub_connections_current":1,"staged_certificate_changes":false,"tls_keylog":false,"demo_mode":true,"demo_stops_in_seconds":86246}
```

```
$ curl -s http://127.0.0.1:8080/metrics
{"uptime_seconds":154,"uptime":"2m34s","sc_hub_connections_current":1,"sc_hub_connections_max":4,"sc_total_connects":3,"sc_total_disconnects":2,"sc_rate_limit_rejections":0,"sc_rx_messages":12,"sc_rx_bytes":456,"sc_tx_messages":12,"sc_tx_bytes":456,"sc_tx_queue_overflows":0}
```

The health checklist

Beside the system health figures, the status page lists the checks the hub makes, each with an icon and the word **Pass**, **Warning** or **Fail**, and for anything but a pass, what to do (with a link to the set-up guide or to sales where that helps):

Check	Passes when
Certificates installed	The hub's certificate, its private key and an issuer certificate are in the certificate folder.
Certificate chain valid	The hub's certificate matches its private key, isn't a CA certificate, and is signed by an installed issuer for both TLS roles (the same check as an activation over BACnet). A staged change made over BACnet is a warning until it is activated.
Certificates not expiring soon	The hub's and the issuers' certificates are valid now and for more than 30 days (a warning within 30 days; a failure once expired, or not yet valid).
BACnet/SC hub listening	The hub function is listening for devices.
Connections under the limit	Fewer devices are connected than the limit (a warning when full: the next device is refused).
Licence	Licensed. In demo mode, a warning with the time left.
Clock looks right	The computer's clock is not earlier than the day this version was built (certificates are checked against it).
TLS key log off	Shown only while --sc-keylog-file is on, as a warning.
Software up to date	Shown once the update check has an answer: a pass when this is the latest version, a warning with the download link when a newer one is available. Not shown while the check is off or couldn't reach the update service.

A certificate set from the B-SCHUB example's public demo CA (whose private key is public) is never trusted: the hub refuses to import it, and BACnet/SC does not start while it is installed - the log says why, and the checklist shows **BACnet/SC hub listening** failing.

Update check

At start-up and every 24 hours the hub asks Chipkin's update service whether a newer version exists. It runs in the background, so the hub never waits for it, and gives up after about 10 seconds.

- **A newer version:** a yellow UPDATE AVAILABLE line on the console (and in the log file) with the download link, a banner on the status page, and a **Software up to date** warning in the health checklist. The hub **never downloads or installs anything** - install the new version when it suits you (on Windows the setup program, on Debian/Ubuntu the .deb; for other Linux distributions take the .tar.gz from the same release).
- **Up to date:** one log line, and the checklist item passes.
- **No answer** (an offline or air-gapped site, a firewall, a proxy): one quiet INFO line saying it couldn't check. Nothing else changes.

What is sent: one HTTPS request, GET `https://app-updates.chipkin.com/chipkin-bacnet-sc-hub/<platform>/x64/version.json` (<platform> is win32, or linux/deb), with a random number on the end so no cache answers it, and a User-Agent naming the hub's version and platform (Chipkin-BACnet-SC-Hub/1.1.1 (win32/x64)). Nothing about your site, devices, certificates or settings is sent, and no cookie. Like any web request, it shows Chipkin's service the public IP address of your site. The service's certificate is checked (Windows' trusted roots on Windows, the system CA bundle on Linux). On Windows the system proxy settings are used.

Checking now: the [About page's Check for updates](#) button asks the update service at once instead of waiting for the next check, and shows the answer: up to date, a newer version with its download link, couldn't reach the service,

or turned off. It works from the hub's own computer only (a page opened as `http://127.0.0.1` or `localhost`), at most once every 30 seconds. Behind it is `POST /api/updates/check`, which needs an `Authorization: HubKey` - header and a JSON body like the set-up guide's actions, a loopback connection and a loopback Host header (so another web site can't make the hub contact the service), and answers 202 (409 when checking is turned off, 429 when it was asked moments ago); `GET /api/about` then shows the result.

Turning it off: `--no-update-check` on the command line, or `check-for-updates = false` in the config file. The About page then says the check is off.

About and feedback

The **About** page (`/about`, in the app bar of every page) shows the hub's version, the CAS BACnet Stack and common helper versions, the build (the commit it was built from and the platform), the operating system and the device, and has the **Check for updates** button ([Update check](#)). Its figures stay current like the status page's.

Its **Send feedback** form sends a message to the people who make the hub: a problem, a question or an idea. Your email address is optional - give it if you would like an answer. Tick **Include diagnostics** to add the hub's version, operating system, CAS BACnet Stack version, demo mode, uptime, connection count and the health checklist's results; **Show exactly what is added** shows the text before you send it. No certificates, keys, passwords, addresses, hub URIs, folder names or device names are ever included.

The page sends the message to the hub, and the hub forwards it over HTTPS to Chipkin's feedback service (`https://app-updates.chipkin.com/feedback`), which emails it to Chipkin and keeps a copy. The hub's version is always sent with it; the service sees your site's public IP address. Nothing is sent until you press **Send feedback**. A message is plain text up to 8000 characters; the hub accepts at most 3 messages every 10 minutes from one computer and 10 an hour in total. If the hub can't reach the service, the page says so - email support@chipkin.com instead. A copy of the hub built without the feedback key (a build of your own, for instance) says feedback isn't available and offers email.

`POST /api/feedback` (JSON `{"message", "email", "diagnostics"}`) needs an `Authorization: HubKey` - header like the set-up guide's actions, so another web site's page can't send feedback through your hub; it answers 202 and `GET /api/feedback/status?id=<n>` says whether the message was sent.

Uploading a certificate

An upload is checked exactly like a certificate written [over BACnet](#) before anything is saved: it must be a PEM certificate, and the certificates as they would be afterwards must still let the hub run (the hub certificate matches its private key and chains to an issuer, and at least one issuer remains). If they pass, the file is written atomically and BACnet/SC reloads with it straight away; devices reconnect. If not, the hub answers HTTP 400 with the reason and nothing changes. An upload to `csr` must be a certificate signing request for the hub's own private key. While a certificate change made over BACnet is waiting for `ACTIVATE_CHANGES`, uploads are refused with HTTP 409.

```
curl -X POST --data-binary @new-hub-cert.pem \  
-H "Authorization: Bearer $UPLOAD_TOKEN" http://127.0.0.1:8080/certs/operational
```

Set-up guide API

The [set-up guide](#) is a page in front of these endpoints, which also work with curl. Request bodies are the raw file (PEM, DER or zip), not a form. Answers are JSON except downloads and the report. **Privileged** endpoints need the hub key's password (see [Security](#)), sent as `Authorization: HubKey <base64 of the password>`. When the key has

no password, send Authorization: HubKey -: the header itself is required, because it is what keeps other web sites from using the hub.

Path	What it does
GET /api/setup/info	The hub URI(s) a device dials, the hub's and its issuers' certificates with every field and check, whether the hub has a request, and whether this caller may install a certificate.
GET /api/setup/issuer/iss-1.pem (iss-1.cer, iss-2.pem)	The issuer certificate(s) devices need to trust the hub. Public.
GET /api/hub/request	hub-cari-request.zip, the hub's CARL request for the CA (its CSR, never the key). 404 if there is no request yet (run --generate-csr).
POST /api/hub/import	Privileged. Install the CA's CARL response zip: checked as for --import-cari, then saved, and BACnet/SC reloads. 400 if it isn't for this hub's request (nothing changed); 409 while a certificate change made over BACnet is waiting for ACTIVATE_CHANGES.
POST /api/inspect?name=<file>	What a file is (certificate, CSR, key, .pfx with an empty password, CRL, CARL zip), every field, and the checks against this hub, each with a fix. Nothing is stored.
POST /api/check-password	Privileged. 200 if the password is right. Counts as an attempt.
GET /api/diagnostics?after=<n>	The last 100 BACnet/SC connection events (each with a sequence number; after returns only newer ones) and the devices connected now.
GET /api/report	A plain-text diagnostic report for support: versions, hub URI, certificates and their checks, connected devices, recent events. No keys or passwords.

```
# Get the hub's request, and install the CA's response, from the hub's own computer
# (key without a password):
curl -o hub-cari-request.zip http://127.0.0.1:8080/api/hub/request
curl --data-binary @hub-cari-response.zip -H "Authorization: HubKey -"
http://127.0.0.1:8080/api/hub/import
# With a password-protected hub key:
curl --data-binary @hub-cari-response.zip -H "Authorization: HubKey $(printf %s "$PW" | base64)" \
http://127.0.0.1:8080/api/hub/import
```

9. Security

- **BACnet/SC** uses TLS 1.3 only, with mutual authentication. A device is accepted if its certificate chains to an issuer in File 3 or 4, and isn't revoked (below).
- **The hub never signs certificates.** It holds no CA key, so taking over the hub doesn't let anyone issue certificates for your BACnet/SC network. Your CA does that.
- **Certificate revocation.** Put your CA's certificate revocation list(s) in issuer-crl.pem in --sc-cert-dir (PEM; one CRL per issuer, concatenated). The hub then refuses any device whose certificate is on a list. It checks the file every 5 seconds: a new or changed file restarts BACnet/SC, every device reconnects, and revoked ones are refused, so no restart is needed. Revocation checking fails closed: while the file exists, a device whose issuer has no CRL in it, or whose CRL is past its next-update date, is refused too, and a file with no readable

CRL stops BACnet/SC (/health reports degraded). Keep the CRLs current. Without the file, revocation isn't checked. The start-up log lists each CRL, how many certificates it revokes, and when it expires.

- **Host names aren't checked** when the hub connects out to another hub. BACnet/SC certificates identify devices, not DNS names; the certificate chain is still verified.
- **The HTTP listener** binds to 127.0.0.1 by default. If you need it from another machine, turn on `--http-tls` before `--http-bind`, so the upload token and status data are encrypted (or use an SSH tunnel or a TLS reverse proxy). The hub logs a warning on every start bound off loopback over plain HTTP. `/`, `/health` and `/metrics` have no authentication, even over HTTPS.
- **Private keys** (`key-hub.pem`, `key-hub-pending.pem`, and any device key or `.pfx`) must stay private. The hub's request zip never contains its key.
- **Installing a certificate from the set-up guide** (`POST /api/hub/import`) changes what the hub presents and which CA it trusts, so it is gated by the hub key's password (`sc-key-password`, or the one typed at start-up):
- **Comparison:** the password is compared in constant time.
- **Rate limit:** at most 5 wrong passwords a minute from one address, and 30 in total; then HTTP 429, and every refusal is logged.
- **No clear text:** it is never accepted over plain HTTP from another computer, where it would cross the network unencrypted. Turn on `--http-tls` to install from elsewhere.
- **No password on the key:** installing works only from the hub's own computer, to a page loaded as `127.0.0.1` or `localhost`.
- **Other web sites can't use it:** every privileged request must carry an `Authorization: HubKey ...` header, which another web site's page can't add without a CORS preflight that the hub never answers. A page loaded under any name other than a loopback one (DNS rebinding) is refused too while the key has no password.
- **Audit log:** every installation, and every refusal, is logged with the address, the certificate's subject, issuer, expiry and fingerprint.
- **Separate token:** `http-upload-token` for `POST /certs/<slot>` is a different secret, and unaffected.
- **TLS key log.** `--sc-keylog-file` writes the BACnet/SC session secrets to a file, so that anyone with the file and a capture can read the traffic. It's for debugging only, and is off unless given on the command line (see [Decoding BACnet/SC traffic in Wireshark](#)). `/health's` `tls_keylog` tells a monitor when it is on.
- **Rate limiting:** new BACnet/SC connection attempts are limited per source address (`--sc-rate-limit`, default 10 a second) and for the whole listener (`--sc-rate-limit-total`, default 50 a second), before the TLS handshake. One flooding host uses up only its own allowance, so devices on other addresses can still reconnect. The hub logs each refusal with the address. A device that stops reading is disconnected after 64 messages are waiting for it.
- **No unencrypted BACnet.** The hub has no BACnet/IP port; every BACnet message to or through it is BACnet/SC (TLS 1.3, mutual authentication).
- **Connections to Chipkin.** The hub connects out to Chipkin's update service (<https://app-updates.chipkin.com>) for two things only: the [update check](#) (start-up and every 24 hours; turn it off with `--no-update-check` / `check-for-updates = false`) and a message someone sends from the [About page](#). Both use HTTPS with the service's certificate checked; neither sends certificates, keys or settings, and the hub never downloads or runs anything from the service.
- **The web pages load nothing from elsewhere.** The status page, the set-up guide and the About page carry their styles, logo and icons inside the page and are served with a strict Content-Security-Policy, so they work offline and can't pull in outside content. The status and About pages' scripts, `/status.js` and `/about.js`, come from the hub, and their policy allows no inline script.

To report a vulnerability, see [SECURITY.md](#).

10. Troubleshooting

Symptom	Cause and fix
cannot start listening ... certificate file(s) missing/unreadable	The hub has no certificate yet. Run <code>--generate-csr</code> , have your CA sign <code>hub-cari-request.zip</code> , and install its response with <code>--import-cari</code> or on <code>/setup</code> . Until then the hub is unreachable over BACnet; <code>/health</code> reports degraded and the status page's checklist says what to do.
<code>--import-cari</code> or <code>/setup</code> says the response is "for another request"	The CA signed a different CSR - an older request, or another hub's. Send it this hub's current <code>hub-cari-request.zip</code> .
<code>--import-cari</code> or <code>/setup</code> says "no issuer certificate in the zip signed the hub's certificate"	The response is missing the CA that signed it (or has only a root, and an intermediate signed the certificate). Ask the CA for a response with the issuing CA in <code>cert1/issuer/</code> .
<code>--generate-csr</code> : "uses an older certificate layout"	The folder has certificates from an earlier release. Run <code>--migrate-certs</code> first.
<code>lws_create_context</code> failed ... retrying every 5 s	The BACnet/SC port is in use, or a certificate file doesn't load. The hub keeps retrying; fix the cause and it starts listening without a restart.
SC TLS handshake REJECTED - client certificate failed verification	The device's certificate isn't signed by an issuer the hub trusts, has expired, or is revoked (certificate revoked, unable to get certificate CRL when <code>issuer-crl.pem</code> has no CRL for its issuer, CRL has expired). Check it with <code>openssl verify -CAfile cert1/issuer/iss-1.pem <the device's opr-*.pem></code> (add <code>-crl_check -CRLfile issuer-crl.pem</code> with a CRL).
private key ... DOES NOT MATCH at start-up	The hub's key and certificate (<code>key-hub.pem</code> and <code>opr-hub.pem</code> , or <code>private-key.pem</code> and <code>operational-certificate.pem</code> in an older folder) don't belong together. Install the CA's response for this hub's current request.
A device won't connect, and you don't know why	Open <code>/setup</code> , <i>Why did a connection fail?</i> , and start the device: it shows what the hub saw and what to change. The file inspector (or <code>--inspect <file></code>) checks the device's certificate against the hub.
The set-up guide won't install the response from another computer	The hub key has no password: open the guide on the hub's own computer (127.0.0.1), or give the key a password and set <code>sc-key-password</code> . With a password, plain HTTP from another computer is refused too: start the hub with <code>--http-tls</code> .
A device can't reach the hub	Check the hub URI it uses, TCP 4443 in the firewall, and whether 4 devices are already connected (see Demo mode). The <code>SC audit</code> : lines show what the hub saw.
DEMO MODE: a BACnet/SC device was refused	4 devices are already connected, the most demo mode allows. Disconnect one, or contact sales@chipkin.com for a licence.
The hub stops with DEMO MODE: ... demo run time is over	Demo mode runs for 24 hours at a time. Start the hub again, or contact sales@chipkin.com for a licence.
refusing a WebSocket upgrade ... (HTTP 400)	The device asked for a WebSocket subprotocol other than <code>hub.bsc.bacnet.org</code> . Check its BACnet/SC settings.
Connect-Request refused by the hub	Another connected device has the same VMAC, the device left out the Hello option, or the hub is full.

Symptom	Cause and fix
SC rate limit: refusing a new connection	A device (or something else) is reconnecting faster than <code>--sc-rate-limit</code> allows.
ERROR: TOO MANY BACnet/SC CONNECTIONS REQUESTED and the hub exits	<code>sc-max-hub-connections</code> is above 4. Set it to 4 or less. See Demo mode .
/health is degraded	BACnet/SC isn't listening, so the device is unreachable over BACnet. Open the status page: its health checklist says which check fails and what to do.
ignoring <code>--port ...</code> BACnet/IP was removed or ignoring "port" in the config file	The hub has no BACnet/IP port since 1.1.0. Remove the option or the port, <code>bacnet-ip</code> and <code>ip-network-number</code> lines.
The hub asks Type demo and press Enter and, run from a script, exits with The hub did not start	In demo mode the hub starts only after <code>demo</code> is typed. Send it on the input: <code>echo demo \n Chipkin-BACnet-SC-Hub ...</code>
The status page doesn't open in a browser	It opens only when the hub is started from a console; on Linux only in a desktop session. Open the address the hub prints.
Escape codes like <code>[33m</code> in a console	Your console doesn't show colour. Set <code>NO_COLOR=1</code> .
The Chipkin BACnet Explorer's test stops at "Reach the hub's port": "The local address could not be used"	It binds to the network adapter selected on its Connect page and can't reach <code>127.0.0.1</code> from there. Use the hub's LAN address in the hub URI.
A BACnet/SC tool on Windows 10 (e.g. YABE) fails with <code>A call to SSPI failed / a TLS handshake error</code> , and the hub logs nothing	Windows 10's own TLS can't do TLS 1.3, which BACnet/SC requires. The hub warns about this at start-up on Windows 10. Run the tool on Windows 11 / Server 2022 or later, or use a client with its own TLS 1.3 (CAS BACnet Explorer).
The hub says running as a service needs a licence	<code>--install-service</code> , <code>--service</code> and starting the hub under systemd need a licence. Run the demo from a console (see Running the hub), or contact sales@chipkin.com for a licence.

Decoding BACnet/SC traffic in Wireshark

BACnet/SC is TLS 1.3, so a capture shows only encrypted records. To see the BACnet/SC messages inside, have the hub write each connection's TLS session secrets to a key log file, and give that file to Wireshark:

```
Chipkin-BACnet-SC-Hub --sc-keylog-file sc-keys.log
```

1. Start the capture (for example `tcp port 4443`; on Windows, capture traffic to the same computer with Npcap's loopback adapter), then start the hub with `--sc-keylog-file`. Only connections made while the option is on can be decoded, because Wireshark needs each TLS handshake.
2. In Wireshark: **Edit > Preferences > Protocols > TLS > (Pre)-Master-Secret log filename**, and select `sc-keys.log`. With tshark: `tshark -r capture.pcapng -o tls.keylog_file:sc-keys.log`.
3. The records then decode as WebSocket, and the BACnet/SC frames as **BSCVLC** (Connect-Request, Connect-Accept, Encapsulated-NPDU, ...). Filter with `bscv1c`. If you run BACnet/SC on a port Wireshark doesn't recognise, use **Decode As > TCP port > TLS**.

The file covers both BACnet/SC roles: devices connecting to the hub, and the hub's own connection to another hub (`--sc-hub-uri`). It doesn't cover the HTTPS status page. The hub appends to the file and flushes every line straight

away, so you can decode a capture while the hub is still running.

Anyone with this file and a capture can read that traffic. Use the option for debugging only, then delete the file. It never holds a private key. It can be set on the command line only (there is no config-file key), so a service can't be left logging by a forgotten line in its config. While it is on, the hub logs a warning at start-up and whenever BACnet/SC starts listening, /health reports "tls_keylog": true, and the status page shows a banner.

11. The BACnet device

Device 389022	"Chipkin BACnet SC Hub"	Vendor 389 (Chipkin Automation Systems)
└─ Analog Input 1	"Bronze"	sample value, degrees C
└─ Network Port 2	"BACnet SC"	BACnet/SC hub function (and optional connector)
└─ File 1	"Operational Certificate"	the hub's certificate (writable)
└─ File 2	"Certificate Signing Request"	CSR for the hub's key (read-only)
└─ File 3	"Issuer Certificate Slot 1"	trusted CA certificate (writable)
└─ File 4	"Issuer Certificate Slot 2"	second trusted CA certificate (writable)

Every object has a **Description** saying what it is for.

The device instance defaults to 389022 (--deviceID), and the Device's name to "Chipkin BACnet SC Hub" (--device-name). Both must be unique on the BACnet internetwork. Analog Input 1 is a sample value: the up/down arrow keys change it while the hub runs.

Network Port 2 (BACnet/SC) is the device's only Network Port; it keeps instance 2 (see [BACnet/SC only](#)). It reports Network_Number 0 with Network_Number_Quality **unknown** unless --sc-network-number sets it; this device isn't a router. The private key is never served by any object. The PICS lists every property.

12. Version, licensing and support

- **Version:** this manual describes version 1.1.1. What changed in each release is in CHANGELOG.md; what a version number promises, and how long a version is supported, is in SUPPORT.md: each MAJOR version gets security fixes for 5 years from its first release.
- **Licensing:** the Chipkin BACnet SC Hub is proprietary software of Chipkin Automation Systems Inc. (see LICENSE). Without a licence it runs in [demo mode](#). For a licence, contact sales@chipkin.com. It is built on the **CAS BACnet Stack**, a separate Chipkin product, and on OpenSSL (Apache-2.0), libwebsockets (MIT), libuv (MIT), zlib (Zlib) and, on Windows, pthreads4w (Apache-2.0); its web pages use Lucide icons (ISC) - see THIRD-PARTY-NOTICES.md. Each release's sbom-*.cdx.json lists the exact versions.
- **Support:** questions and problem reports go to support@chipkin.com. Include Chipkin-BACnet-SC-Hub --version and the set-up guide's support report. Security problems: SECURITY.md.
- **Related documents:** the fact sheet, the PICS, the production certificate guide, the support policy and the BTL test plan.