Velocity HTTPS and certificates

Source file: doc/eng-us/features/6.0/velocity-https-certificates.md Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/features/6.0/velocity-https-certificates.md

This page is for administrators who serve a site over HTTPS with Velocity. Velocity's HTTPS looks after itself. With no certificate configured, the server makes its own and serves HTTPS anyway. With a source configured (your own files, an archive, a PKCS#12 bundle, Let's Encrypt or any other ACME CA), it checks the certificate, serves it, watches it, renews it and swaps the new one into the running server, without a restart and without dropping a connection. Applies to Exponential Velocity 0.0.4.27 and later.

What you can rely on

- No moment when the server answers plain HTTP but not HTTPS (HTTPS comes up first).
- A broken or expired certificate never reaches a visitor: the pair is verified (parses, not expired, key belongs to it) before use, and a self-signed certificate stands in until the real one is usable again.
- Renewal needs nothing from you: Let's Encrypt certificates renew in a background job with growing pauses (5 minutes doubling up to a day) after a failure, so a CA's rate limits are never spent.
- HTTP/2 comes with it (HTTP/2 and security).

Quick start

All settings live under `Q.web.https` in the site file (for example `/etc/qbix/sites-enabled/example.com.conf`, or `/etc/vc/...` for Velocity) or in the `--config` JSON.

Let's Encrypt (port 80 of this server reachable from the internet, DNS pointing here):

JSON

```
{ "Q": { "web": { "https": {
  "port": 443,
  "mode": "letsencrypt",
  "acme": { "email": "your-contact-address", "domains": ["example.com", "www.example.com"] }
} } } }
```

Your own files:

JSON

```
{ "Q": { "web": { "https": {
  "port": 443,
  "mode": "files",
  "cert": "/etc/ssl/example.com/fullchain.pem",
  "key":  "/etc/ssl/example.com/privkey.pem"
} } } }
```

Development, no certificate at all:

JSON

```
{ "Q": { "web": { "https": { "port": 8443, "mode": "self-signed" } } } }
```

Start the server. The console prints one line such as `tls: certificate ready (EC, openssl-ecdsa, 0.1s, 397 days, localhost ...)`. Check at any time:

Shell

```
qbixconsole ssl:show
```

Always try the Let's Encrypt **staging** directory first (`"directory": "letsencrypt-staging"`): its certificates are untrusted but its rate limits are generous.

Settings (under `Q.web.https`)

Setting	Default	Meaning
`port`	`443`	HTTPS port; `--https-port` overrides
`mode`	`"manual"`	`files` (same as `manual`), `archive`, `pkcs12`, `letsencrypt` (same as `acme`), `certbot`, `remote`, `self-signed`
`fallback`	`"self-signed"`	When the source has nothing usable: `"self-signed"` or `"none"` (HTTPS off)
`watchInterval`	`60`	Seconds between checks of the certificate files
`domain`	none	Main host name; default for Let's Encrypt and certbot domains
`selfSigned.dir`	`<conf dir>/ssl`	Where the self-signed pair, imported pairs and ACME files live
`selfSigned.hosts`	automatic	Names and addresses of the self-signed certificate: `domain`, virtual host names and aliases, this machine's name, `localhost`, `127.0.0.1`, `::1`, the bound address
`sources.<mode>`	none	A class implementing `Q_WebServer_Certificate_Source`, to add a mode of your own

`files`: `cert` (default `config/certs/fullchain.pem`; leaf alone or with chain; PEM, DER or PKCS#7), `key` (default `config/certs/privkey.pem`; may be the same file), `chain` (optional), `keyPassword` / `keyPasswordFile` / `keyPasswordEnv`.

`archive`: `archive` (`.zip`, `.tar`, `.tar.gz`/`.tgz`, `.tar.bz2`, `.tar.xz`, `.rar`, `.7z`), `password*` for an encrypted archive, `keyPassword*` for an encrypted key.

`pkcs12`: `bundle` (`.p12` / `.pfx`) and its `password`, `passwordFile`, `passwordEnv`.

`letsencrypt` / **ACME (`Q.web.https.acme`)**:

Setting	Default	Meaning
`email`	none	Contact address for expiry notices (recommended)
`domains`	`[domain]`	Names to certify; the first names the directory; `*.example.com` needs `dns-01`
`directory`	`"letsencrypt"`	`letsencrypt`, `letsencrypt-staging`, `zerossl`, `google`, `google-staging`, or any ACME directory URL
`challenge`	`"http-01"`	`http-01` or `dns-01`

Setting	Default	Meaning
`webroot`	none	Also write HTTP-01 answers to `<webroot>/.well-known/acme-challenge/` when another server owns port 80
`dnsHook`	none	`dns-01`: a program run as `HOOK add NAME VALUE` and `HOOK remove NAME VALUE`
`dnsWait`	`30`	Seconds to wait after the hook added the record
`keyType`	`"ec256"`	`ec256`, `ec384`, `rsa2048`, `rsa3072`, `rsa4096`; a new key for every certificate
`renewAt`	`0.33`	Renew when this share of the lifetime is left
`eab.kid`, `eab.hmacKey*`	none	External account binding (ZeroSSL, Google)
`timeout`	`300`	Seconds to wait for validation and issuance
`dir`, `challengeDir`	`<ssl dir>/acme`, `<acme dir>/challenges`	Where ACME files live

`certbot` and `remote` (a URL downloaded every `checkInterval`, default `86400`) are also available; see the engine's `docs/https.md`.

How it behaves

1. A source provides the certificate; the pair is checked; HTTPS reads only a **private copy** (`<ssl dir>/active-<port>-<fingerprint>-<pid>.pem`) that belongs to this server process, so a file being rewritten elsewhere cannot break a handshake. A second server on the same directory and port writes its own copy (fixed in 0.0.4.28: before it, a second server could take HTTPS away from the first).
2. Every `watchInterval` seconds the source's files are checked; a changed, valid pair is presented to the next connection. A certificate written before its key gets one interval to settle.
3. **Self-signed** is tried in order: PHP openssl with ECDSA P-256; PHP openssl with RSA 2048 signed with SHA-256; the system `openssl` program; the system snakeoil pair. It lasts 397 days, renews 30 days before expiry and whenever the names change, and lives in `ssl/self-signed.*`. (Release 0.0.4.27 signs with SHA-256 because systems that refuse SHA-1 refused the earlier one.)
4. **Per-domain selection (SNI)**: the certificate that covers each domain is chosen per connection and shown in the panel's Domains tab (control panel).

What's in `ssl/`

```
ssl/
  self-signed.pem, self-signed.key      the self-signed pair
  active-443-<fp>-<pid>.pem/.key        the copy one server reads (do not edit)
  imported/                             pairs imported from DER, archives, bundles, remote
  acme/account-<ca>.pem                 one account key per CA: keep it
  acme/<first domain>/fullchain.pem, privkey.pem, state.json
  acme/challenges/<token>               pending HTTP-01 answers (short-lived)
```

Back up `ssl/`. Everything in it can be made again except the ACME account keys.

Operate it

- `qbixconsole ssl:show` (and `--json`): served certificate, every known certificate by expiry. Watch `daysLeft`: below 14 for a CA certificate means renewal has been failing for two weeks; `acme.lastError` says why.
- The control panel's **SSL** tab shows the same, edits the safe settings (mode, ACME email and directory, `renewAt`, host names), renews and reloads without a restart.
- Keep NTP running: certificates are checked against the system clock.

Troubleshooting

Symptom	Cause	Fix
ACME error `Invalid response from http://.../.well-known/acme-challenge/...`	Port 80 does not reach this server	Open port 80, proxy `/.well-known/acme-challenge/` to it, or set `acme.webroot` for the server that owns port 80
`the DNS hook failed`	The hook exited non-zero	Run it by hand with `add NAME VALUE`
`no private key found (is it encrypted? ...)`	Encrypted key, no password given	Set `keyPassword`, `keyPasswordFile` or `keyPasswordEnv`
`none of the certificates belongs to the private key`	Key and certificate from different requests	Use the key the certificate was issued for
`no tool here reads ...rar`	No `bsdtar`/`unrar`	Install `libarchive-tools` or repack as `.zip`

Limits and security notes

- No external program is required for `files`, `pkcs12`, `.zip`/`.tar*`, self-signed or Let's Encrypt; PHP's openssl extension is enough.
- Private keys are written `0600`, atomically, and never logged. External programs run with an argument list, never through a shell. The ACME client verifies the CA's certificate (turn `verify` off only for a test CA).
- Wildcards need `dns-01` with a hook program.

Related pages

- Control panel (SSL and Domains tabs), scheduler (renewal jobs), Velocity web server
- Specifications: HTTP/2 and security, engine settings (TLS session resumption, `--https-port`)
- Upgrade: Velocity engine upgrade notes, Velocity engines (`[HTTPSSettings]`)
- Changelog: Exponential Velocity engine
- History: 24 September (certificate subsystem), 25 to 30 September (SSL tab, per-domain

certificates)

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.