

V2T vendor guide

Day-to-day work for the vendor: publishing updates, issuing and managing licences, and handling customer contact details. The deployment runbook (first deploy, nginx, upgrades, backups) is [activation_server/README.md](#); this guide is the short version of what you do after the service is running.

Where things are

What	Where
Admin page	<code>https://d2g.dev/v2t/admin</code> (one password; seven tabs: Dashboard, Customers, Codes, Activations, Trials, Updates, Releases)
Admin CLI	on the host, in <code>activation_server/</code> : <code>A="docker compose run --rm activation python -m activation_server.admin"</code> , then <code>\$A <subcommand></code>
Licence signing key	on the host only (<code>V2T_KEY_FILE</code>) and in your backup
Release signing key	on the dev PC only (<code>..\v2t-release-key.pem</code>), never on the host
Hosted installers	<code>https://d2g.dev/v2t-download/</code> (nginx serves <code>/home/sschang/v2t_download/</code>)
Signed release files and patch zips	<code>V2T_DATA_DIR</code> on the host, seen by the container as <code>/data</code>

The admin page and the CLI do the same things. Use the page for daily work; the CLI always works, also when the page is locked by the IP allowlist or the password is lost.

Give `$A` exactly one subcommand per line, and define it again in each new shell.

Publish an update

When a client release needs a new server route, deploy the service first and publish the client only after the route answers through the public URL.

There are two kinds. Pick by what changed:

Change	Release type
Python code, templates, static files only	patch (small zip, applied from the panel)
requirements.txt , a new top-level module, launch.cmd or recover.py	full installer

Every release needs a version above the newest published one. Versions compare as numbers (3.1.10 is above 3.1.9). Release notes are a UTF-8 text file, one short line per change, each line "Traditional Chinese / English".

Patch

1. If activation_server/ changed since the last deploy, redeploy the service first: keep an own copy of activation.db outside the data dir, then follow "Upgrading" in the [runbook](#) and check curl -s https://d2g.dev/v2t/healthz .
2. On the dev PC, in the checkout the installed version was built from, change only what the patch fixes and bump version.py . Run the tests.
3. Build and sign:

```
.venv\Scripts\python.exe -m scripts.build_patch --key ..\v2t-release-key.pem --notes-file notes.1
```

It prints the zip, the signed release-X.Y.Z.json and the deps digest. The build refuses when the digest is not the sha256 of the installed requirements.txt (the 3.1.0 digest, which starts 6a66a1f7a5ab): installed PCs would be offered an installer instead and never see the patch.

4. Copy both files into V2T_DATA_DIR on the host, then:

```
$A publish-release /data/release-X.Y.Z.json --zip /data/v2t-patch-X.Y.Z.zip # "published X.Y.Z
$A list-releases
```

On the admin page: Releases tab, upload the JSON and the zip together.

5. Check from any PC:

```
curl -s -o /dev/null -w "%{http_code} %{content_type}" https://d2g.dev/v2t/update/download/X.Y.Z
answers 200 application/zip .
```

Teachers see **Apply update** on the panel's Status tab at the next panel start. A patch that fails to start is rolled back on that PC and hidden there; the panel shows a red line with both versions.

Full installer

1. If `activation_server/` changed since the last deploy, redeploy the service first: keep an own copy of `activation.db` outside the data dir, then follow "Upgrading" in the [runbook](#) and check `curl -s https://d2g.dev/v2t/healthz`.
2. Bump `version.py`, run the tests, tag the commit, and build:

```
python scripts\build_installer.py --full --output build\vX.Y.Z-full
```

The result is `V2T-Setup-full.exe` plus its `.bin` slices. Zip them together, stored (no compression needed), as `V2T-X.Y.Z-full.zip`. See [BUILD_INSTALLER.md](#).

3. Test it before anything is published: install over the previous version (licence, models and term files must survive; V2T must be stopped first) and into an empty folder (first start must need no download).
4. Upload the zip to `/home/sschang/v2t_download/` on the host and check it from the dev PC:

```
curl -sI https://d2g.dev/v2t-download/V2T-X.Y.Z-full.zip      # 200 and Content-Length equal to the  
sha256sum /home/sschang/v2t_download/V2T-X.Y.Z-full.zip    # on the host; equal to the local st
```

5. Sign with exactly that URL, on the dev PC:

```
.venv\Scripts\python.exe -m scripts.sign_release --key ../v2t-release-key.pem --type installer --
```

The notes should say how to install: download the zip, unzip it, run `V2T-Setup-full.exe`.
`sign_release` takes the version from `version.py`, so sign before `version.py` is bumped for the next release. The notes files of a release are kept in `docs/release-notes/`.

6. Copy `release-X.Y.Z.json` into `V2T_DATA_DIR`, then:

```
$A publish-release /data/release-X.Y.Z.json      # "published X.Y.Z (installer)"  
$A list-releases
```

7. Check from the dev PC:

```
curl -s -o /dev/null -w "%{http_code} %{redirect_url}\n" https://d2g.dev/v2t/update/download/X.Y.Z  
answers 302 and the signed URL.
```

Teachers see **Download installer** on the Status tab. Nothing installs by itself: they download the zip, press **Stop & Close**, unzip and run Setup. Publishing an installer is one-way for every PC that installs it, because `recover.py` and `launch.cmd` change only through an installer.

Order matters: redeploy the service if it changed, host the file, check the URL, sign, publish.

A signed URL that does not answer breaks the Download button on every PC. Never move or rename a hosted installer while its release is offered.

Withdraw, re-offer, correct

```
$A withdraw-release X.Y.Z           # PCs stop being offered it at their next check
$A reoffer-release X.Y.Z           # offer a withdrawn release again
$A publish-release /data/release-X.Y.Z.json --replace    # replace the document of the same version
```

A withdraw stops new offers only; it does not undo installs already made. When an installer replaces older patches, withdraw those patches at the publish.

Who has updated

- Updates tab, or `$A list-checks --limit 20` : which PCs asked, their version, what they were offered.
- `$A list-downloads --limit 20` : who downloaded which release.

Clear old releases and logs (one-off)

```
$A purge-history           # lists what would go; changes nothing
$A purge-history --yes     # backs up, then purges
```

- It keeps only the newest live release and deletes every other release with its patch zip, and empties the update-check, download and audit logs.
- Codes, PCs and customer contacts are not touched; every licence keeps working.
- Always read the listing first: the `keep:` line must name the release PCs should be offered.
- The backup in `backups/` holds what was removed and every contact. Delete it once you have confirmed the result.

Licences

A customer gets an **activation code**. Each PC that activates with it uses one **seat** and receives a licence bound to that PC, valid until the code's expiry date. Reinstalling on the same PC does not use another seat.

Issue a code

Codes tab, **New code**, or:

```
$A new-code --licensee "Taipei XX HS" --seats 30 --expires 2027-12-31 --name "Wang Xiaoming" --email
```

- `--licensee` is the name shown in the app ("Licensed to ...") and is part of every licence.
- `--name` is the contact person; `--email` and `--phone` are optional.
- The code (v2T-xxxx-xxxx-xxxx-xxxx) is shown **once**. Copy it then; only its hash is stored.
Afterwards refer to the code by the hash prefix that `list-codes` shows.

Send the customer the code and [LICENSE_GUIDE.md](#).

See codes and PCs

```
$A list-codes                                # code hash, licensee, seats, expiry, disabled, customer_id, bu
$A list-activations --code <prefix>         # the PCs of one code: machine id, licence id, computer name, r
```

The Codes and Activations tabs show the same with search and filters. The Codes tab adds the seats used and a status; the Activations tab has a CSV export.

Extend the expiry date

Codes tab: set the date in the code's row and press **Set expiry**. Or:

```
$A extend --code <prefix> --expires 2028-12-31
```

Every activated PC receives the new date at its next weekly re-check, with no action by the teacher. A PC that is offline keeps its old date until it goes online once.

A customer needs more PCs

Seats cannot be raised on an existing code. Issue a second code for the extra PCs with the same licensee and expiry.

A PC is replaced or retired

```
$A revoke --machine-id <id>                # or --license-id <id>; Activations tab: Revoke
```

Revoke frees the seat. The old PC stops at its next re-check, and the new PC activates with the same code.

Stop or restore a whole code

```
$A disable --code <prefix> --yes      # every PC of the code stops at its next check
$A enable --code <prefix>              # PCs can activate again with the same code, using no new seat
```

Disable is for a customer that left or did not pay. Check `list-activations` first: disabling stops working classrooms.

How fast a change reaches a PC

An activated PC re-checks about every 7 days when online, at the moment V2T is started. Extend, revoke and disable take effect at that check. A PC that cannot reach the server keeps working for 30 days and then needs one internet connection. A PC on a trial asks at every start, so a changed trial end date reaches it at its next start.

A PC with no internet (manual licence)

1. The teacher clicks **Copy** in the License window and sends you the Machine ID.
2. On a PC that holds the licence key:

```
.venv\Scripts\python.exe scripts\issue_license.py --key ..\v2t-signing-key.pem --machine-id <id>
```

3. Send back the `license.json` it writes; the teacher imports it.

A manual licence never contacts the server, so it is not in the admin page, cannot be extended or revoked, and a new file must be issued when it expires. Keep your own list of them.

Trials

A PC without a licence registers itself at its first start online (Machine ID and computer name) and gets a signed trial.

- One trial per Machine ID. It ends 2 calendar months after the first registration, by the date in the container. That date is UTC only because the image and the compose file set no time zone. Every later start gets the same trial back, so a reinstall does not start a second one.
- A Machine ID with an activation record, live or revoked, gets no trial. The PC shows 此電腦不提供試用 / No trial for this computer and must re-activate or import a licence file.
- With no contact a PC runs 7 days from its first start, retrying at every start (the first day is counted too, so its panel shows 8 days left on that day). After that it does not start until one contact, an activation or an import.

- The PC checks at start only. The call is cut off after 4 seconds (the app announces "up to about 5 seconds"). A running class is never stopped, and the end date itself is still a usable day.
- A paid or manual licence makes no trial call.
- Statuses: `running`, `ending soon` (the end date is 14 days away or nearer; the PC's panel counts the end date as a day and turns amber one day later), `ended`, `converted` (the PC has an activation record).
- The service limits one address to 300 new PCs in 24 hours and 600 trial calls an hour. Both counts are kept in memory, so a container restart clears them.

The Trials tab

Search and the chips (Running, Ending soon, Ended, Converted) narrow the list. **Export CSV** downloads the rows shown as `trials.csv`. **Edit** changes one trial's end date or note. **End now** and **Clear name and note** each ask on a confirmation page first.

List trials

```
$A list-trials                # every trial with its status
$A list-trials --status ending # running, ending, ended or converted
$A list-trials --search TEXT   # Machine ID, computer name or note
```

Extend or shorten one trial

```
$A set-trial-end --id N --ends YYYY-MM-DD      # or --machine-id ID instead of --id
```

The PC gets the new date at its next start. A teacher who has the License window open presses **重新檢查 / Check again** and V2T goes on without a restart.

End one trial now

```
$A set-trial-end --id N --now
```

The end date becomes yesterday, because the end date itself is a usable day. A trial that has already ended is refused.

Change many end dates at once

```
$A set-trial-ends --ends YYYY-MM-DD --started-before YYYY-MM-DD      # lists, changes nothing
$A set-trial-ends --ends YYYY-MM-DD --started-before YYYY-MM-DD --yes
```

It needs at least one selector, `--started-before` or `--ends-before` (both mean before, not on). Always run it without `--yes` first and read the listing. Its use: PCs that came from 3.1.0 by the 3.2.1 patch moved from the date built into 3.1.0 (2027-06-30) to a 2-month trial from their first contact; this command gives them one later date. A trial row holds no app version, so the selection is by date only and also takes fresh 3.2.0 installs registered before that date.

Keep a note on a trial

```
$A set-trial-note --id N --note "..."
```

One line of at most 200 characters; an empty text clears it. You type the note; the PC never sends it.

Clear a name and note on request

```
$A blank-trial --id N          # prints the trial, changes nothing
$A blank-trial --id N --yes
```

The computer name and the note are blanked. The row, the Machine ID and the dates stay, and the name stays empty afterwards.

A trial row is never deleted

A deleted row would give that PC a new trial. Extend or shorten the trial instead.

The 3.2.1 patch on a 3.1.0 PC: two corners

- **No licence file, but the Machine ID has an activation record.** The new version is refused a trial at its first start (`No trial for this computer`), the update is undone and hidden on that PC, and it keeps running 3.1.0. The fix is a licence first: activate a code or import a licence file. The 3.2.0 installer is not a fix by itself, because 3.2.0 does not start on that PC without a licence either, while 3.1.0 still runs. The 3.2.1 patch stays hidden there, so once the PC is licensed it reaches 3.2 by the 3.2.0 installer or by a later patch.
- **The trial was stored during the patch and the new version then failed to start.** The PC is put back on 3.1.0, which runs on that stored trial until its end date and then says the licence expired. The fix: a code or a licence file, the 3.2.0 installer, or a later patch. This corner is accepted as it is and was not tried on a real PC.

Customer contact details

The activation dialog collects the activator's name, e-mail and optional phone under the privacy notice. They are used for licence management, support and update notices only.

Task	Admin page	CLI
Find a customer	Customers tab, Search	<code>\$A list-codes</code> (the <code>customer_id</code> column)
Correct a name, e-mail or phone	Customers tab, Edit	<code>\$A edit-contact --code <prefix> --name "..."</code> [--email E] [--phone P]
Export	Customers, Activations or Trials tab, Export CSV	not available
Erase one customer on request	Customers tab, Edit, Erase contact details	<code>\$A erase-customer --id N</code> (prints the listing), then with <code>--yes</code>
Clear a trial's computer name and note on request	Trials tab, Edit, Clear name and note	<code>\$A blank-trial --id N</code> (prints the trial), then with <code>--yes</code>
Routine clean-up	not available	<code>\$A prune --older-than 365 --dry-run</code> , then without <code>--dry-run</code>

- **Erase** blanks the name, e-mail and phone of one customer and removes them from the audit log. The licences keep working. The licensee name and the computer names stay, because they are part of the licence records.
- **Prune** erases the contacts of customers whose codes all ended more than 365 days ago and deletes update-check and download rows older than that. It never runs by itself: run it about once a month, always with `--dry-run` first. Use 365, the period the privacy notice promises.

The same monthly run also blanks the computer name and note of trials that ended more than 365 days ago. That is what keeps the registration note's promise that the computer name is cleared one year after the trial ends.

- After an erase or a prune, delete older database backups, downloaded CSV files and off-host copies that still hold the old text (see "After an erase or a prune" in the runbook).
- Delete an exported CSV (`customers.csv` , `activations.csv` , `trials.csv`) when you are done with it.

Routine and safety

- Back up `V2T_DATA_DIR` (database, `releases/` , the two admin files) and both private keys. Losing the licence key means no licence can be issued; losing the release key means no update can be signed for installed PCs.
- Never paste `.env` , a key file or the admin password into a chat, a ticket or an e-mail.
- Change the admin password with `$A set-password` ; it logs every session out.
- `$A list-events --limit 20` shows the audit log: logins, code and licence changes, erases.
- After any change on the host, `curl -s https://d2g.dev/v2t/healthz` must answer `{"ok":true}` .