

ZEUS AUTOMATION

Zeus PLC Master Image

Controller Master Image Guide

Building a master (golden) SD / eMMC image of a commissioned Zeus PLC controller, and producing new controllers from it — each with its own identity

Software version 0.1.0.13

Edition 1.7 · 2 October 2026

Web optizeus.org

About this guide

This guide explains how to make a **master image** of a Zeus PLC controller — a copy of a fully prepared controller's SD card or eMMC — and how to produce new controllers from it. Every controller made this way starts with the same software and settings, generates its own identity (SSH keys, machine ID) the first time it starts, and gets its own access when it is commissioned.

It covers the IPC-CM400200 (Raspberry Pi Compute Module 4) and the IPC-CM500200 (Compute Module 5). The procedure is the same for both, but each board family needs its **own** master image.

Who should read it

Reader	Read
Commissioning and production engineers who prepare controllers in quantity	The whole guide
Service engineers who replace a controller in the field	Chapters 5 and 6, Appendix A
Quality staff	Chapter 1, Chapter 6.6 (acceptance), Appendix A

You need to be comfortable with the Zeus PLC IDE (connecting to a controller, **Download hardware, Deploy runtime...**) and with typing a few Linux commands over SSH. The *Zeus PLC IDE Operation Manual* describes the IDE.

Conventions

Convention	Meaning
Bold	A menu item, button or field exactly as the IDE or a Windows tool labels it
Monospace	Commands, file names, addresses and values you type
Target... → Deploy runtime...	A navigation path
\$ / >	A command typed on the controller (Linux) / on the Windows PC

Note: Information that helps you understand a step.

Tip: A faster or safer way to do something.

Important: Information you need to avoid a problem — a wasted image or a controller that cannot be reached.

WARNING: A risk to a working machine or to data. Read it before acting.

Safety

WARNING: Preparing a controller for imaging (Chapter 3) permanently deletes its SSH keys, machine ID, access token, control program, retained values and logs. Do it only on a **spare reference controller**, never on a controller that runs a machine.

WARNING: Flashing an image erases everything on the target SD card or eMMC. Check the drive letter and size twice before you write.

WARNING: A controller that has just been flashed has no control program and its outputs are off. Commission it on the bench, not connected to a machine.

Version

This edition describes Zeus PLC **version 0.1.0.13**.

Edition	Date	Change
1.0	30 September 2026	First issue (Zeus PLC 0.1.0.6).
1.1	30 September 2026	Updated for Zeus PLC 0.1.0.7.
1.2	30 September 2026	Updated for Zeus PLC 0.1.0.8.
1.4	2 October 2026	Zeus PLC 0.1.0.10: licensing the controller during commissioning (6.4, acceptance 6.6, checklist 7a).
1.5	2 October 2026	Updated for Zeus PLC 0.1.0.11.
1.6	2 October 2026	Updated for Zeus PLC 0.1.0.12.
1.7	3 October 2026	Updated for Zeus PLC 0.1.0.13.
1.3	2 October 2026	Updated for Zeus PLC 0.1.0.9.

Contents

1. Overview	6
1.1 What a master image is	6
1.2 What is shared and what is unique	6
1.3 The workflow	7
1.4 What you need	7
1.5 Time needed	8
2. Preparing the reference controller	9
2.1 Start from a known state	9
2.2 Install the Zeus runtime	9
2.3 Add what every controller should have	9
2.4 Install the first-boot service	10
2.5 Final check before scrubbing	11
3. Removing the unit-specific data	12
3.1 Run the preparation script	12
3.2 What the script did	12
3.3 Power off — and do not start it again	13
4. Reading the master image	14
4.1 Choose a name	14
4.2 SD-card controllers	14
4.3 eMMC controllers	14
4.4 Shrinking the image (optional)	15
4.5 Checksum and compression	15
4.6 File the image	15
5. Writing the image to a new controller	17
5.1 Before you write	17
5.2 SD-card controllers	17
5.3 eMMC controllers	17
5.4 Label the controller	18
6. Commissioning a controller made from the image	
6.1 One at a time, on the bench	19
6.2 First start	19
6.3 Give it its own address and name	19
6.4 Give it its own access	20
6.5 Download the machine	21
6.6 Acceptance	21
6.7 Where required: hardware root of trust	21
7. Keeping the image current	22
7.1 When to make a new master image	22
7.2 Making the next image	22
7.3 Re-using the reference unit as a normal controller	22
8. Troubleshooting	23
A. Checklists	25
A.1 Making a master image	25
A.2 Each new controller	25

B. The files in controller-image **27**

 B.1 Files on the controller the guide refers to 27

1. Overview

1.1 What a master image is

A master image (also called a *golden image*) is a file — for example `zeus-cm4-master-0.1.0.13-2026-09-30.img` — that holds the complete contents of a prepared controller's storage: the operating system, its hardening, the Zeus runtime, its service and its default configuration. Writing that file to the storage of a new controller gives you a controller that is ready to commission in minutes instead of hours, and identical to every other one made from the same image.

A plain copy of a working controller is **not** a master image. It would give every copy the same SSH host keys, the same machine ID and the same monitor access token:

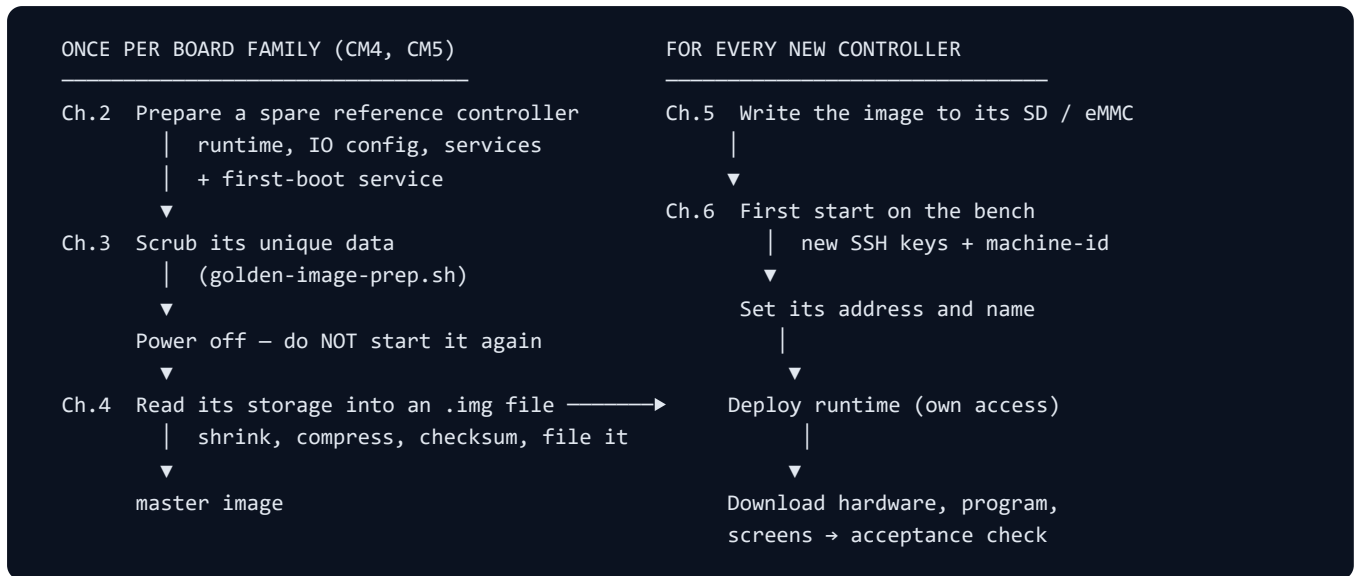
If copies share...	The consequence
SSH host keys	A fake controller cannot be told apart from a real one; one stolen key works for every unit
Monitor token	One leaked token opens every controller
machine-id	Logging, licensing and anything else keyed on it confuses the units

So the master image is made from a controller whose unique data has been removed, and each new controller regenerates its own on first start.

1.2 What is shared and what is unique

Item	In the master image (shared)	Created per controller
Operating system, packages, hardening	✓	
Zeus runtime, its service, folders and permissions	✓	
Carrier-board services (power-fail monitor, local display) that you installed	✓	
Site program-signing public key	✓ (it is public)	
The engineering PC's SSH key in <code>authorized_keys</code>	✓	
Network addresses and host name	✓ (as on the reference unit)	Changed per unit (6.3)
SSH host keys		✓ automatically, first boot
machine-id		✓ automatically, first boot
Monitor access (engineer accounts, or a token)		✓ by Deploy runtime... and Download hardware (6.4, 6.5)
Control program, retained values, operator screens		✓ downloaded from the IDE
Logs and audit trail		✓ start empty
Secure-boot fuses, TPM keys	Cannot be imaged	✓ on the bench, where required

1.3 The workflow



1.4 What you need

Hardware

Item	Notes
A spare controller of the right family	The <i>reference unit</i> . It is used up by Chapter 3 and must be re-commissioned afterwards (Chapter 7).
For an SD-card controller (CM4 Lite / CM5 Lite): a USB SD card reader	The bench IPC-CM400200 boots from a 16 GB SD card.
For an eMMC controller: a USB cable to the carrier's USB-boot port	Plus the carrier's nRPIBOOT jumper or button (see the carrier documentation).
SD cards or eMMC modules at least as large as the reference unit's	A smaller card will not take the image unless it was shrunk (4.4).

Software on the Windows PC

Tool	Used for	Where from
Zeus PLC IDE with an ARM64 runtime build (the Zeus Automation engineering station)	Deploy runtime, hardware and program download	Zeus Automation
scp / ssh (built into Windows 10/11)	Copying the image scripts to the reference unit	Windows
Win32 Disk Imager	Reading a card into an <code>.img</code> , and writing it	sourceforge.net/projects/win32diskimager
Raspberry Pi Imager (alternative for writing)	Writing an <code>.img</code> to SD or eMMC	raspberrypi.com/software
rpiboot (<code>usbboot</code>) — eMMC only	Makes a controller's eMMC appear on the PC as a USB disk	github.com/raspberrypi/usbboot (Windows installer)

Tool	Used for	Where from
7-Zip (optional)	Compressing the image for storage	7-zip.org
WSL with PiShrink (optional)	Shrinking the image so it fits smaller cards	github.com/Drewsif/PiShrink

Files from the Zeus PLC release package (folder `controller-image\`, Appendix B)

File	Purpose
<code>golden-image-prep.sh</code>	Removes the unique data from the reference unit (Chapter 3)
<code>zeus-firstboot.sh</code>	Regenerates identity on a new controller's first start
<code>zeus-firstboot.service</code>	Runs <code>zeus-firstboot.sh</code> exactly once, before SSH and the runtime start

1.5 Time needed

Step	Typical time
Preparing and scrubbing the reference unit	30–60 minutes
Reading a 16 GB card	15–25 minutes
Shrinking and compressing (optional)	10–20 minutes
Writing one new controller	10–20 minutes
Commissioning one new controller	15–30 minutes

2. Preparing the reference controller

The reference controller is the one whose storage becomes the master image. Everything you want on **every** new controller must be on it before Chapter 3; everything that belongs to **one** machine must not be.

WARNING: The reference controller is used up by Chapter 3. Use a spare controller of the right family (CM4 for an IPC-CM400200 image, CM5 for an IPC-CM500200 image) — never a controller that runs a machine.

2.1 Start from a known state

1. Fit the spare controller on the bench with power, a network cable to your PC's network, and nothing connected to its IO ports.
2. Make sure you can reach it from the PC:

```
> ping 192.168.1.50
> ssh bio@192.168.1.50
```

Replace `192.168.1.50` and `bio` with the unit's address and SSH user. SSH must log in **with the engineering PC's key**, without asking for a password — that key is copied into every controller made from the image, which is what lets the IDE reach them.

3. Bring the operating system up to date, so every controller starts from the same patched base:

```
$ sudo apt update && sudo apt full-upgrade -y
$ sudo reboot
```

2.2 Install the Zeus runtime

1. In the IDE, **Target...** → add the unit (**Type** IPC-CM400200 or IPC-CM500200, its **Host** and **SSH user**) and **Connect**.
2. **Target...** → **Deploy runtime...** with **Install service** and **Start after install** ticked. Every step must show **OK**.
3. Check that the target badge shows the right board, and that the run state reads **RUNNING** or **STOPPED** — not an error.

Note: Deploy runtime refuses a runtime build that is older than its source code. If it says *Rebuild it first*, rebuild the ARM64 runtime on the engineering station and deploy again — an image made from an old runtime spreads that old runtime to every controller.

2.3 Add what every controller should have

Install now anything that should come with every controller of this family. Typical items:

Item	How
Power-fail monitor (carrier with UPS/supercap)	Zeus Automation carrier procedure
Local HDMI display (kiosk)	IDE: Controller Settings → Local display → Install on controller (IDE manual, <i>Operator screens</i>)
Real-time tuning	<code>sudo bash prepare-rt-controller.sh --apply</code> (engineering station scripts)
Time synchronisation (NTP)	Point the unit at the site's time server (IDE manual, <i>Retain and maintenance</i>)
TLS on the control link, log split	<code>rollout-cm4.sh <user>@<host> --harden</code> from the engineering station

Do **not** leave plant-specific things on the reference unit: no plant control program, no operator screens, no site recipes. They are deleted in Chapter 3 anyway, and a master image should not carry one customer's logic to another.

Tip: Write down, now, exactly what you installed and in which version — it goes into the image record (4.6). A year from now "what is on this image?" is the first question.

2.4 Install the first-boot service

The first-boot service is what gives each new controller its own SSH keys and machine ID. It must be on the reference unit **before** it is scrubbed.

1. From the Zeus PLC release package, copy the three files in `controller-image\` to the unit:

```
> cd C:\Zeus\ZeusPLC-IDE-0.1.0.13-win-x64\controller-image
> scp golden-image-prep.sh zeus-firstboot.sh zeus-firstboot.service bio@192.168.1.50:/tmp/
```

2. On the unit, install the service:

```
$ sudo install -m 755 /tmp/zeus-firstboot.sh /opt/zeus-plc/bin/zeus-firstboot.sh
$ sudo install -m 644 /tmp/zeus-firstboot.service /etc/systemd/system/zeus-firstboot.service
$ sudo systemctl daemon-reload
$ sudo systemctl enable zeus-firstboot.service
```

3. Check it:

```
$ systemctl is-enabled zeus-firstboot.service
enabled
```

Note: On this unit the service does nothing yet: it runs only when the SSH host keys are missing, and they are still there. It will run once on each controller made from the image, and never again.

Important: The scripts must keep Linux line endings. If you edited them on Windows and the controller reports `$'\r': command not found`, convert them with `sed -i 's/\r$//' /tmp/*.sh /tmp/*.service` and install again.

2.5 Final check before scrubbing

Check	How	Expected
Runtime healthy	IDE connected; or <code>systemctl is-active zeus-plc</code>	<code>active</code> , run state not <code>FAULT</code>
Board correct	IDE target badge	CM4 or CM5, as intended
First-boot service enabled	<code>systemctl is-enabled zeus-firstboot.service</code>	<code>enabled</code>
IDE key login works	<code>ssh bio@<address></code> from the PC	No password prompt
Free space	<code>df -h /</code>	Note the <i>Used</i> figure — it decides how far the image can shrink (4.4)

3. Removing the unit-specific data

This step turns the reference controller into a neutral master: it removes everything that must be unique to one controller, and leaves everything that should be shared.

WARNING: This step cannot be undone. Check once more that you are logged in to the **spare reference controller** — `hostname` and `ip -4 addr` — before you continue.

3.1 Run the preparation script

1. On the reference unit:

```
$ sudo bash /tmp/golden-image-prep.sh
```

2. The script asks for confirmation. Type `YES` (capitals) and press Enter. Anything else aborts without changing anything.
3. It then reports each step:

```
== stopping the runtime ==
== removing SSH host keys (regenerated on first boot) ==
== clearing machine-id (regenerated on first boot) ==
== removing device state (retained data + boot program) ==
== stripping the device token from the config (keeping the site public key) ==
    token removed; bind set to loopback
== clearing shell history and tmp ==
== enabling the first-boot identity-regeneration service ==

DONE. Power off now and image the SD. Do NOT boot this unit again before imaging.
```

If the line `!! zeus-firstboot.service not installed` appears, stop: go back to 2.4. An image made without the first-boot service gives every controller made from it **no** SSH host keys at all, so none of them can be reached over SSH.

3.2 What the script did

Removed or changed	Why
Runtime stopped	So nothing writes to the storage while it is imaged
SSH host keys deleted	Each new controller generates its own on first boot
<code>/etc/machine-id</code> emptied	Regenerated on first boot
<code>retain.bin</code> , the boot program and its signature files deleted	Retained values and programs belong to one machine
<code>license.json</code> deleted	A license names one controller's CPU serial; each unit is licensed when it is commissioned (6.4)

Removed or changed	Why
Runtime logs and audit log deleted	A new controller starts with an empty history
Monitor token removed; monitor bound to 127.0.0.1, token not required	Each unit gets its own access when commissioned (6.4); until then the monitor is reachable only from the unit itself
Shell history and /tmp cleared	Removes commands and files typed during preparation
First-boot service enabled	Makes sure it runs on each clone

What it keeps: the operating system, the runtime and its service, the configuration (apart from the token), the site's program-signing public key, the SSH `authorized_keys` of the user accounts, and everything you installed in 2.3.

Note: Operator screen files in `/var/lib/zeus-plc/hmi` are not removed by the script. If you downloaded screens to the reference unit, delete them before powering off: `sudo rm -rf /var/lib/zeus-plc/hmi/*`.

3.3 Power off — and do not start it again

```
$ sudo poweroff
```

Wait until the green activity LED has stopped flashing (about 10 seconds), then remove power.

Important: Do **not** power the reference unit on again before its storage has been read (Chapter 4). If it starts, the first-boot service gives it a new identity, the neutral state is lost, and you must repeat Chapter 3.

4. Reading the master image

This chapter copies the scrubbed reference unit's storage into an image file on the PC. Follow 4.2 for a controller that boots from an SD card, 4.3 for one with eMMC.

Tip: Not sure which one you have? On a working controller of the same type, `cat /sys/block/mmcblk0/device/type` prints `SD` for an SD card and `MMC` for eMMC. The bench IPC-CM400200 prints `SD`.

4.1 Choose a name

Name the file so that anyone can tell what it is without opening it:

```
zeus-<board>-master-<runtime version>-<date>.img  
zeus-cm4-master-0.1.0.13-2026-09-30.img
```

4.2 SD-card controllers

1. With the reference unit powered off, take out its SD card and put it in the USB card reader on the PC.
2. Windows may offer to format the card, or say it cannot read a partition. **Click Cancel every time.** Formatting destroys the master.
3. Start **Win32 Disk Imager** as administrator.
4. In **Device**, choose the drive letter of the card's small `bootfs` partition. Check the size in Explorer if in doubt — it must be the card, not a USB stick or disk.
5. In **Image File**, enter the full path and name from 4.1, for example `D:\Zeus-Images\zeus-cm4-master-0.1.0.13-2026-09-30.img`.
6. Tick **Read Only Allocated Partitions** if your version offers it (a smaller file); otherwise the whole card is copied.
7. Click **Read** — not *Write*. Wait until it reports *Read Successful*.
8. Eject the card in Windows before you remove it.

WARNING: **Read** copies card → file. **Write** copies file → card and would overwrite the master with whatever file is named. Check the button before you click.

4.3 eMMC controllers

An eMMC controller has no card to take out. The carrier makes the eMMC visible to the PC as a USB disk:

1. With the controller powered off, set the carrier to USB-boot mode: fit the **nRPIBOOT** jumper (or hold the **BOOT** button) as the carrier documentation describes.
2. Connect the carrier's USB-boot (USB device / OTG) port to the PC.
3. Power the controller on, then run **rpiboot** on the PC (from the Start menu after installing usbboot). It reports progress and finishes within a minute; the eMMC then appears in Windows as a new drive.
4. Continue with 4.2 steps 2–8, using the drive that appeared.

5. Power off, remove the jumper (or release the button) and the USB cable.

Note: While in USB-boot mode the controller does not run its own software — the first boot service cannot trigger. Reading the eMMC this way is safe.

4.4 Shrinking the image (optional)

A raw image is as large as the card (about 15 GB for a 16 GB card), even though only about 6–7 GB is used, and it can only be written to cards at least as large as the original. **PiShrink** reduces it to the used size and makes it expand itself to the full card on the first boot. It runs in WSL:

```
> wsl
$ wget https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh
$ chmod +x pishrink.sh
$ sudo ./pishrink.sh /mnt/d/Zeus-Images/zeus-cm4-master-0.1.0.13-2026-09-30.img
```

Important: PiShrink adds a one-time resize to the image. The first boot of each new controller is then about a minute longer, and includes one extra reboot. Wait for it; do not power the controller off during its first two minutes.

4.5 Checksum and compression

1. Record a checksum, so that a damaged or altered file is recognised before it is written to a controller:

```
> certutil -hashfile D:\Zeus-Images\zeus-cm4-master-0.1.0.13-2026-09-30.img SHA256
```

Save the result in a text file next to the image with the same name and `.sha256`.

2. Optionally compress the image with 7-Zip (`.7z` or `.xz`). Raspberry Pi Imager can write a `.xz` file directly; Win32 Disk Imager needs the plain `.img`.

4.6 File the image

Keep each master image with a short record:

Field	Example
File name and SHA-256	zeus-cm4-master-0.1.0.13-2026-09-30.img , 3f1c...
Board family	CM4 (IPC-CM400200)
Zeus runtime version	0.1.0.13
Operating system and date of last update	Debian 13 (trixie), updated 2026-09-30
Extras installed (2.3)	Power-fail monitor, NTP to 10.0.0.1
Network settings inside the image	eth0 10.10.0.10/24 static; eth1 DHCP
Host name inside the image	CM4002000

Field	Example
Made by, date	

Keep at least the previous master image as well: if a new one turns out to have a problem, you can go back.

5. Writing the image to a new controller

5.1 Before you write

- Check the image against its checksum:

```
> certutil -hashfile D:\Zeus-Images\zeus-cm4-master-0.1.0.13-2026-09-30.img SHA256
```

The result must match the `.sha256` file. If not, do not use the file.

- Use the image for the **same board family**. A CM4 image does not boot on a CM5, nor the other way round.
- The card or eMMC must be at least as large as the image (or the original card, if the image was not shrunk).

5.2 SD-card controllers

With Raspberry Pi Imager

1. Put the new SD card in the card reader.
2. Start **Raspberry Pi Imager**.
3. **Choose Device**: the board family (Raspberry Pi 4 for CM4, Raspberry Pi 5 for CM5).
4. **Choose OS** → **Use custom** → select the `.img` (or `.img.xz`).
5. **Choose Storage** → the card. Check its size.
6. **Next**. When asked *Would you like to apply OS customisation settings?* click **No**.

Important: Always answer **No**. OS customisation would change the user, password, SSH keys and host name that the image carries, and the IDE could not log in.

7. Confirm the warning that the card will be erased, and wait for *Write Successful* (Imager verifies the card after writing).

With Win32 Disk Imager

1. Put the card in the reader; cancel any format prompt.
2. **Image File**: the `.img`. **Device**: the card's drive letter.
3. Click **Write**, confirm, and wait for *Write Successful*.
4. Optionally click **Verify Only** to compare card and file.

5.3 eMMC controllers

1. Set the carrier to USB-boot mode (nRPIBOOT jumper or BOOT button), connect the USB-boot port to the PC, power on and run **rpiboot** (as in 4.3).
2. Write to the drive that appears, with either tool from 5.2.
3. Power off, remove the jumper and USB cable.

5.4 Label the controller

Label each controller as it leaves this step — at least the board family, the image name and a unit number. The unit number is what you will give it as its host name in 6.3.

6. Commissioning a controller made from the image

6.1 One at a time, on the bench

Every controller made from the image starts with the **same network addresses and the same host name** as the reference unit had (4.6). Two of them on the same network at the same time answer on the same address, and neither works properly.

Important: Power on and commission **one new controller at a time**, on the bench, and give it its own address (6.3) before you power on the next one.

6.2 First start

1. Fit the written card (or the controller with its written eMMC), connect the network port that carries the image's plant-side address, and power on.
2. Wait two minutes. On the first start the controller:
 - generates new SSH host keys and a new machine ID (`zeus-firstboot`);
 - if the image was shrunk, expands the file system to the whole card and reboots once;
 - starts the runtime with no program, monitor on loopback only.
3. From the PC, check it answers:

```
> ping 10.10.0.10
```

6.3 Give it its own address and name

1. **Clear the old key on the PC.** The PC remembers the SSH key of the last controller it saw at this address. The new controller has a new key, so SSH — and the IDE — refuse to connect with *REMOTE HOST IDENTIFICATION HAS CHANGED*. That is the protection working. Remove the stale entry:

```
> ssh-keygen -R 10.10.0.10
```

Do this in the Windows account that runs the IDE. Repeat it for every address a controller made from this image has been reached on from this PC.

2. Log in once to accept the new key:

```
> ssh bio@10.10.0.10
```

3. Set its host name to its unit number (5.4), for example `ZEUS-0042` :

```
$ sudo hostnamectl set-hostname ZEUS-0042
```

4. In the IDE, **Target...** → add the controller at its current address and **Connect**.
5. In the device tree, open **Network Interfaces** → the plant-side port → **Configure addressing...**, and give it its final address (Static or DHCP), then **Apply**. Tick *I understand this will disconnect me*: the IDE connection drops.
6. **Target...** → edit the controller's **Host** to the new address → **Connect**.
7. Set the fieldbus port's address the same way if the machine needs it.

Tip: If every controller will run on its own isolated machine network, the image's addresses may be exactly right and step 5 is not needed. The host name still should be set, so that logs and audit trails say which unit wrote them.

6.4 Give it its own access

Target... → **Deploy runtime...** on the new controller. This re-installs the runtime, writes the controller's configuration with the correct file ownership, and sets up how the IDE is allowed in. Every step must show **OK**.

The machine's project...	Access to the controller is protected by
has at least one engineer account (Users & Roles) — the normal case	The accounts, installed by Download hardware (6.5). No shared token is used.
has no accounts	A new random token written by Deploy runtime... , known only to this IDE entry and this controller

Important: Add each new controller as its **own** entry under **Target...**. An entry that was used for another controller carries that controller's token, and Deploy would give it to the new one as well.

Important: Do not skip this step even though the runtime is already installed. Until it runs, the monitor answers only on the controller itself.

License the controller

Every Zeus controller leaves the bench licensed (Operation Manual, chapter 37). Without a license, a runtime with real IO stops its program after 2 hours. The license names the controller's CPU serial number, so it is made for this unit and no other.

On the Zeus Automation license PC (the one holding the license key), run **Zeus PLC License Generator** from the desktop, choose **1**, and give the controller as `user@address` and the customer. Or from a command prompt:

```
node tools\license\zeus-license.mjs controller bio@192.168.1.50 --customer "Customer name"
```

It reads the CPU serial over SSH, signs the license, installs it as `/var/lib/zeus-plc/license.json` and keeps a copy and a log line in `%USERPROFILE%\zeus-license\`. The runtime reads it at its next start; to adopt it at once, use **Online** → **Controller** → **License...** → **Install license file...** with the copy.

Important: The license key (`%USERPROFILE%\zeus-license\zeus-license-private.pem`) never leaves the license PC and is never put in a package or an image. Keep a backup of it in a safe place: without it no further license can be issued for the controllers in the field.

6.5 Download the machine

1. Open the machine's project in the IDE.
2. **Download hardware** — the IO configuration and the user accounts. The runtime restarts.
3. Log in with an engineer account.
4. **Download program**, then go online. The top bar must show **IN SYNC**.
5. If the machine has operator screens: **Download screens**.

6.6 Acceptance

Check each controller before it leaves the bench:

Check	How	Expected
Own SSH identity	<code>ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub</code>	A fingerprint no other controller has
Own machine ID	<code>cat /etc/machine-id</code>	A value no other controller has
Own host name	<code>hostname</code>	The unit number
First boot ran once	<code>journalctl -t zeus-firstboot</code>	regenerated SSH host keys , regenerated machine-id
Own access	IDE connects and logs in with an engineer account; <code>ls -l /opt/zeus-plc/config/active.json</code>	Logged in; file is <code>-rw-r----- root zeusplc</code>
Program	IDE top bar	RUNNING and IN SYNC
License	Online → Controller → License...	Licensed - controller IPC-CM400200 / IPC-CM500200; no DEMO pill
IO	IO Devices in the tree	Each device OPERATIONAL

Record the results with the controller's unit number — Appendix A has a sheet.

6.7 Where required: hardware root of trust

Secure-boot fuses and TPM keys are per-controller and cannot come from any image. Where the project requires them, provision them on the bench after 6.6, following the Zeus Automation secure-boot and TPM procedures. Both are irreversible; try them on a spare first.

7. Keeping the image current

7.1 When to make a new master image

Event	New image?
New Zeus runtime version	Yes — or update each controller with Deploy runtime...
Operating-system security updates	Yes, at least every six months
A new standard item for every controller (2.3)	Yes
New carrier board or board revision	Yes, and test it on a spare first
A new plant program or screens	No — those are downloaded per machine

Controllers already in the field are not changed by a new image. Update them in place with **Deploy runtime...** and the operating system's updates, or re-flash them in a planned stop (and then commission them as in Chapter 6).

7.2 Making the next image

The reference unit's identity was removed in Chapter 3, and after imaging it is simply a controller with no identity. To make the next master image:

1. Write the **current** master image to the reference unit (Chapter 5) and let it start once (6.2) — it becomes a normal controller again.
2. Clear its old key on the PC (6.3 step 1) and connect.
3. Apply the changes: **Deploy runtime...** with the new version, operating-system updates, new standard items.
4. Continue with 2.5, Chapter 3 and Chapter 4, with a new file name.

7.3 Re-using the reference unit as a normal controller

After Chapter 4, the reference unit can go into service like any other: power it on and commission it exactly as in Chapter 6. Its first start gives it an identity like every controller made from the image.

8. Troubleshooting

Symptom	Cause	What to do
SSH or the IDE says <i>REMOTE HOST IDENTIFICATION HAS CHANGED</i>	The PC remembers the previous controller's key at this address	<code>ssh-keygen -R <address></code> in the Windows account that runs the IDE (6.3)
SSH refuses the connection; <code>ping</code> works	The image had no SSH host keys and the first-boot service did not run	The image was made without 2.4. Make a new image. On this unit, connect a keyboard and screen and run <code>sudo ssh-keygen -A && sudo systemctl restart ssh</code>
SSH asks for a password	The engineering PC's key was not in <code>authorized_keys</code> on the reference unit	Log in with the password once and add the key (<code>ssh-copy-id</code>), then fix the reference unit before the next image
Two controllers keep dropping off the network	Both started with the image's address	Power one off, give the other its own address (6.3), then the second
The controller does not start at all (no activity LED)	CM4 image written to a CM5, or the other way round; or a damaged write	Check the image label; write again and verify
The first start takes several minutes and reboots	The image was shrunk; it is expanding to the card	Normal once. Do not remove power during the first two minutes
<code>golden-image-prep.sh</code> prints <code>\$'\r': command not found</code>	The script was saved with Windows line endings	<code>sed -i 's/\r\$//' /tmp/*.sh /tmp/*.service</code> , then run it again
<code>golden-image-prep.sh</code> prints <code>zeus-firstboot.service not installed</code>	2.4 was skipped	Do 2.4, then run the script again
The IDE connects but every command says <i>not logged in</i> or is refused	Deploy runtime... (6.4) or Download hardware (6.5) not done yet	Do 6.4 and 6.5
Two controllers show the same <code>machine-id</code>	The reference unit was started between Chapter 3 and Chapter 4, or a controller was copied rather than written from the master	Repeat Chapter 3 on the reference unit; on the affected controller run <code>sudo rm /etc/machine-id && sudo systemd-machine-id-setup</code>
Windows asks to format the card when it is inserted	Windows cannot read the Linux partition	Always Cancel
Win32 Disk Imager: <i>not enough space</i> when writing	The target card is smaller than the image	Use a larger card, or shrink the image (4.4)

ZEUS PLC · CONTROLLER MASTER IMAGE GUIDE

Appendices

A. Checklists

Copy or print these sheets and keep them with the image and with each controller's record.

A.1 Making a master image

#	Step	Done
1	Spare reference controller of the right family on the bench, nothing on its IO	☐
2	SSH key login from the engineering PC works (2.1)	☐
3	Operating system updated (2.1)	☐
4	Deploy runtime... — all steps OK; version: _____ (2.2)	☐
5	Standard items installed (2.3): _____	☐
6	No plant program, screens or recipes left on the unit (2.3)	☐
7	First-boot service installed and enabled (2.4)	☐
8	golden-image-prep.sh run, YES typed, no !! lines (3.1)	☐
9	sudo poweroff ; unit NOT started again (3.3)	☐
10	Storage read with Read into: _____ .img (4.2 / 4.3)	☐
11	Shrunk (optional) (4.4)	☐
12	SHA-256 recorded: _____ (4.5)	☐
13	Image record filled in and filed with the image (4.6)	☐

Made by: _____ Date: _____

A.2 Each new controller

Unit number / host name: _____ Image: _____

#	Step	Done
1	Image checksum matches (5.1)	☐
2	Written and verified (5.2 / 5.3); controller labelled (5.4)	☐
3	First start on the bench, alone on the network (6.1, 6.2)	☐
4	Old key cleared on the PC; SSH login accepted (6.3)	☐
5	Host name set: _____ (6.3)	☐
6	Addresses set — plant: _____ fieldbus: _____ (6.3)	☐
7	Own Target... entry; Deploy runtime... — all steps OK (6.4)	☐
7a	License generated and installed; License... shows Licensed (6.4)	☐

#	Step	Done
8	Download hardware , log in, Download program, IN SYNC (6.5)	☐
9	Screens downloaded (if any) (6.5)	☐
10	SSH fingerprint: _____ (unique) (6.6)	☐
11	machine-id: _____ (unique) (6.6)	☐
12	IO devices OPERATIONAL (6.6)	☐

Commissioned by: _____ Date: _____

B. The files in controller-image

The Zeus PLC release package contains a folder `controller-image\` with the files this guide uses. They are plain text; read them before running them.

File	Installed to	What it does
<code>golden-image-prep.sh</code>	Run from <code>/tmp</code> , not installed	Chapter 3: asks for <code>YES</code> , stops the runtime, deletes the SSH host keys, empties <code>/etc/machine-id</code> , deletes <code>retain.bin</code> , the boot program and its signature files, the runtime logs and audit log, removes the monitor token and binds the monitor to loopback, clears shell history and <code>/tmp</code> , and enables the first-boot service
<code>zeus-firstboot.sh</code>	<code>/opt/zeus-plc/bin/zeus-firstboot.sh</code>	On a new controller: generates SSH host keys (<code>ssh-keygen -A</code>) and restarts SSH, and generates a new <code>machine-id</code> . Logs to the system journal under <code>zeus-firstboot</code>
<code>zeus-firstboot.service</code>	<code>/etc/systemd/system/zeus-firstboot.service</code>	Runs <code>zeus-firstboot.sh</code> only when <code>/etc/ssh/ssh_host_ed25519_key</code> does not exist — exactly once, on the first start of a new controller — and before SSH and the runtime start

B.1 Files on the controller the guide refers to

Path	Meaning
<code>/opt/zeus-plc/bin/zeus-plcd</code>	The Zeus runtime
<code>/opt/zeus-plc/config/active.json</code>	The runtime's configuration, <code>640 root:zeusplc</code> ; holds the monitor token when the project has no engineer accounts (6.4)
<code>/var/lib/zeus-plc/retain.bin</code>	Retained variable values
<code>/var/lib/zeus-plc/boot.st</code> (+ <code>.sig</code> , <code>.ver</code> , <code>.manifest</code>)	The downloaded control program and its signature
<code>/var/lib/zeus-plc/hmi/</code>	Operator screens served by the controller
<code>/var/log/zeus-plc/</code>	Runtime log and audit log
<code>/etc/ssh/ssh_host_*</code>	SSH host keys — the controller's SSH identity
<code>/etc/machine-id</code>	The operating system's unique machine ID